IN THE UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF TEXAS
MIDLAND-ODESSA DIVISION

ADVANCED CLUSTER SYSTEMS, INC., *Plaintiff*, v. INTEL CORPORATION, *Defendant*.	Case No. 7:24-cv-00245-ADA JURY TRIAL DEMANDED

INTEL'S FIRST AMENDED PRELIMINARY INVALIDITY AND SUBJECT-MATTER ELIGIBILITY CONTENTIONS

TABLE OF CONTENTS

I. Introduction 5

A. Plaintiff's Infringement Contentions 5

B. Overview of Invalidity Grounds 6

II. Identification of Relevant Prior Art 9

A. Priority Dates 9

1. The Asserted Patents Are Not Entitled to Claim Priority to the Filing Dates of the '738 and '908 Provisional Applications 9

2. The '768 Patent Is Not Entitled to a Priority Date Any Earlier than February 14, 2014. 12

3. The '621 Patent Is Not Entitled to a Priority Date Any Earlier than July 13, 2021. 12

4. The '034 Patent Is Not Entitled to a Priority Date Any Earlier than July 13, 2021. 13

5. The '582 Patent Is Not Entitled to a Priority Date Any Earlier than January 23, 2023. 13

6. The '679 Patent Is Not Entitled to a Priority Date Any Earlier than October 2, 2023. 14

B. Prior Art for the Asserted Patents 14

1. Table 1: Prior Art Patents / Patent Publications 16

2. Table 2: Non-Patent Prior Art Publications 27

3. Table 3: Prior Art Products and Systems 117

C. Admitted Prior Art 120

1. Admitted Products 120

2. Admitted Claim Elements 125

III. Anticipation and Obviousness Under 35 U.S.C. §§ 102 and 103 137

A. Intel's Claim Chart Exhibits 137

B. Exemplary Obviousness Combinations 143

1. Exemplary Obviousness Combinations for the '768 Patent 144

2. Exemplary Obviousness Combinations for the '621 Patent 145

3. Exemplary Obviousness Combinations for the '034 Patent 146

4. Exemplary Obviousness Combinations for the '582 Patent 147

5. Exemplary Obviousness Combinations for the '679 Patent 148

C. Motivations to Combine 149

D. Obviousness In Light of the State of the Art and Motivations to Combine 155
1. Background 156
2. Development of Parallelized Computing 160
3. Discussion of Particular Claim Limitations 163
(a) Peer-to-Peer Architecture 163
(b) Additional MPI Functionality 183
(c) Kernels 189
(d) Claimed Three-Node Sequence of Operations 194
(e) Fourier Transforms and Other Computational Libraries 219
(f) Special Purpose and Multicore Processors 230
(g) Cluster Initialization and Establishing Communication Among Nodes 235
E. Lack of Secondary Indicia of Nonobviousness 239
IV. Double Patenting 246
V. Prior Art Under 35 U.S.C. §§ 102(f) and (g) 267
A. Prior Art Under 35 U.S.C. § 102(f) 267
B. Prior Art Under 35 U.S.C. § 102(g) 269
VI. Invalidity Under 35 U.S.C. § 112 269
A. Lack of Enablement/Written Description 270
1. Enablement/Written Description Contentions for the ’768 Patent 271
2. Enablement/Written Description Contentions for the ’034 Patent 277
3. Enablement/Written Description Contentions for the ’679 Patent 282
4. Enablement/Written Description Contentions for the ’621 Patent 286
5. Enablement/Written Description Contentions for the ’582 Patent 289
B. Indefiniteness 293
VII. Subject-Matter Eligibility 295
A. The Asserted Claims Are Directed to an Abstract Idea 295
B. The Asserted Claims Lack an Inventive Step 299
VIII. Document production 322

Pursuant to the Scheduling Order in the above-captioned matter (Dkt. 54), on April 9, 2025, Defendant Intel Corporation ("Intel") served Preliminary Invalidity and Subject-Matter Eligibility Contentions for the claims asserted by Plaintiff Advanced Cluster Systems, Inc. ("Plaintiff"), from U.S. Patent Nos. 10,333,768 (the "'768 patent"); 11,563,621 (the "'621 patent"); 11,570,034 (the "034 patent"); 11,811,582 (the "'582 patent"); and 12,021,679 (the "'679 patent") (collectively, the "Asserted Patents") in Plaintiff's Preliminary Infringement Contentions served January 22, 2025 ("Infringement Contentions").

Intel provides the following First Amended Preliminary Invalidity and Subject-Matter Eligibility Contentions ("Invalidity Contentions"). These Invalidity Contentions incorporate (1) new Claim Chart Exhibits A-16–A-20, B-16–B-21, C-16–C20, D-16–D-21, and E-16–E-20, which are substantively identical to the same-numbered Claim Chart Exhibits served by Defendant Advanced Micro Devices, Inc. ("AMD"), in *Advanced Cluster Systems, Inc. v. Advanced Micro Devices, Inc.*, No. 7:24-cv-244-ADA (W.D. Tex.), on April 22, 2025; and (2) the additional prior art references identified by AMD on April 22, 2025, which have been incorporated into Tables 1–3 in Section II.B (Prior Art for the Asserted Patents).[1] Pursuant to the Scheduling Order, Intel certifies through its undersigned counsel that it undertook reasonable efforts in preparing its original contentions, and that this amendment is based on material identified after the original contentions were served.

These Invalidity Contentions address the invalidity of the Asserted Claims, identify additional reservations and explanations, and describe the accompanying document production.

[1] Table 2 also incorporates the corrections provided to Plaintiff on April 21, 2025—specifically, the addition of native files (e.g., source code and video files) that were produced by Intel and cited in its Claim Chart Exhibits served on April 9, 2025, but were inadvertently omitted from the original Table 2.

Intel reserves the right to amend these contentions as provided by Federal Rule of Civil Procedure 26(e), the Scheduling Order, and any other applicable law, rule, or precedent. Intel also reserves the right to amend and/or supplement these Invalidity Contentions based on information subsequently identified and/or should Plaintiff be permitted to correct, clarify, amend, and/or supplement its Infringement Contentions.

I. INTRODUCTION

A. Plaintiff's Infringement Contentions

On January 22, 2025, Plaintiff served Intel with its Infringement Contentions, asserting 74 claims (the "Asserted Claims") across five patents. The Asserted Claims are:

Patent	Asserted Claims
10,333,768	1, 4, 20, 21, 26, 27, 29, 30, 31, 33, 34, 35, 36, 37, 39
11,563,621	1, 3, 4, 6, 7, 8, 9, 10, 11, 13, 15, 16, 17, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
11,570,034	1, 2, 3, 8, 10, 24, 25, 27, 28, 30
11,811,582	1, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
12,021,679	1, 2, 3, 8, 10, 17, 18

The Infringement Contentions are deficient in that they fail to properly and completely disclose how the Accused Instrumentalities allegedly infringe the Asserted Claims. Infringement contentions much be sufficient to provide reasonable notice to the defendant as to why they plaintiff believes it has a "reasonable chance of proving infringement." *View Engineering, Inc. v. Robotic Vision Systems, Inc.*, 208 F.3d 981, 986 (Fed. Cir. 2000). Reasonable notice requires the patentee to specifically tie the asserted claim language to some feature within the accused product. Yet Plaintiff has failed to identify specifically how the Accused Products allegedly infringe numerous limitations of the Asserted Claims. For example, each of the Asserted Claims requires numerous specific elements arranged in a particular manner and configured to perform certain functions. Instead of meaningfully identifying which aspects of the Accused Products satisfy each

claim limitation, Plaintiff's infringement contentions merely parrot the claim language and recite boilerplate, vague, and conclusory accusations that the products practice the claim limitations—utterly failing to describe where and how the claim limitation is found in the Accused Products.

Plaintiff's failure to provide more detail is particularly egregious given that it has access to the *publicly available source code* for the Intel oneCCL, oneMKL, and HCCL Demo software cited in its contentions. When source code is available to the patentee, the patentee must provide pinpoint citations to the code identifying the location of each limitation. *See, e.g.*, *MasterObjects, Inc. v. Amazon.com, Inc.*, No. C 20-08103 WHA, 2021 WL 5987101, at *2 (N.D. Cal. Dec. 17, 2021) ("Where the accused instrumentality includes computer software based upon source code made available to the patentee, the patentee must provide 'pinpoint citations' to the code identifying the location of each limitation." (quoting *Finjan, Inc. v. Check Point Software Techs.*, Inc., No. C 18-02621 WHO, 2019 WL 955000, at *5 (N.D. Cal. Feb. 27, 2019))); *Treehouse Avatar LLC v. Valve Corp.*, No. C17-1860-RAJ, 2019 WL 917403, at *3 (W.D. Wash. Feb. 25, 2019) (ordering plaintiff to serve amended infringement contentions containing source code citations); *Zix Corp. v. Echoworx Corp.*, 2:15-CV-1272-JRG, 2016 WL 3410367, at *2 (E.D. Tex. May 13, 2016) (after accused infringer had produced its source code, striking patentee's amended infringement contentions for not providing identification of the specific source code of the accused product and ordering amended infringement contentions). Accordingly, Intel reserves the right to amend and/or supplement these Invalidity Contentions should the Court permit Plaintiff to correct, clarify, amend, and/or supplement its Infringement Contentions.

B. Overview of Invalidity Grounds

The Asserted Claims are invalid under 35 U.S.C. §§ 102 and 103 because all the claims are anticipated by, or would have been obvious to a skilled artisan in view of, the prior art. *See*

Sections II (Identification of Relevant Prior Art) and III, *infra.* The patents also are invalid for lack of inventorship by the named inventors under 35 U.S.C. § 102(f), and at least the '768 and '621 patents also are invalid for double patenting.[2] *See* Section IV (Prior Art Under §§ 102(f) and (g)) and Section V (Double Patenting), *infra*. Furthermore, the Asserted Claims are invalid under 35 U.S.C. § 112 for lack of written description, lack of enablement, indefiniteness, and/or because they do not set forth what the patentee regarded as the invention. *See* Section VI (Invalidity Under 35 U.S.C. § 112), *infra*. All the Asserted Claims of the Asserted Patents are further invalid for failure to meet the eligibility requirements of 35 U.S.C. § 101. *See* Section VII (Subject-Matter Eligibility). Intel also incorporates by reference and reserves the right to rely on the invalidity and subject-matter eligibility contentions served on Plaintiff by the defendant(s) in other cases, including *Advanced Cluster Systems, Inc., v. Nvidia Corporation et al.,* No. 1:19-cv-2032-MN-CJB (D. Del.), and *Advanced Cluster Systems, Inc., v. Advanced Micro Devices, Inc.,* No. 7:24-cv-244-ADA (W.D. Tex.) (collectively, with any other cases filed by Plaintiff or its privies, the "Other Plaintiff Cases").

These disclosures are made based on prior art presently known to Intel. Discovery in this case has not yet commenced. The Court has not construed the terms of the Asserted Claims in this case, and the Court has not determined the level of education or experience of one of ordinary skill in the art. Intel reserves the right to modify, amend, and/or supplement these Invalidity Contentions in view of, without limitation, information provided by Plaintiff concerning its infringement allegations; to the extent relevant, additional discovery concerning the alleged priority, conception,

[2] Unless otherwise noted, citations to Title 35 of the United States Code in these contentions refer to the pre-AIA version of the applicable statutes. However, to the extent the Asserted Patents are not entitled to priority dates earlier than March 16, 2013, Intel reserves the right to rely on corresponding provisions of the America Invents Act.

and reduction to practice of any of the Asserted Claims; additional prior art obtained through discovery or further investigation, including without limitation discovery from Plaintiff or third parties; claim constructions by the Court; positions taken by Plaintiff in this action or any other proceeding; the parties' expert reports; and/or as other circumstances warrant.

Intel's Invalidity Contentions are based on information currently available to Intel and its current understanding of the Asserted Claims as stated in Plaintiff's deficient Infringement Contentions, without the benefit of a claim construction order in this case, fact discovery, or expert testimony. Intel objects to any attempt to infer claim construction positions from any identification of prior art or invalidity contention contained herein. Certain positions put forth in these Invalidity Contentions are predicated (to the extent that Plaintiff's positions can be ascertained) on Plaintiff's incorrect interpretation of the Asserted Claims as evidenced by its Infringement Contentions provided to Intel and its claim construction positions. Intel's Invalidity Contentions are not, and should not be construed as, admissions or adoptions of a particular claim scope or construction, or as an admission that any particular element of an Asserted Claim is found in any accused instrumentality. Intel reserves the right to adopt claim construction positions that differ from positions put forth in this document. Intel disagrees with Plaintiff's application of the Asserted Claims to the accused instrumentalities (to the extent that Plaintiff's positions can be ascertained), and Intel expressly denies infringement.

Intel expressly incorporates by reference, as if fully set forth herein, the Invalidity Contentions and claim charts served on Plaintiff by the defendants in the Other Plaintiff Cases. Intel also incorporates by reference all prior art and Invalidity Contentions identified in any post-issuance review by the U.S. Patent and Trademark Office of any Asserted Patent or patent owned by Plaintiff and covering a subject matter that overlaps even in part with the Asserted Patents,

including *Nvidia Corporation v. Advanced Cluster Systems, Inc.*, IPR2020-01608, filed September 18, 2020 (U.S. Patent No. 8,082,289); *Nvidia Corporation v. Advanced Cluster Systems, Inc.*, IPR2021-00075, filed October 16, 2020 (U.S. Patent No. 8,140,612); *Nvidia Corporation v. Advanced Cluster Systems, Inc.*, IPR2021-00108, filed October 23, 2020 (U.S. Patent No. 8,676,877); *Nvidia Corporation v. Advanced Cluster Systems, Inc.*, IPR2021-0019, filed October 6, 2020 (U.S. Patent No. 10,333,768); and *Nvidia Corporation v. Advanced Cluster Systems, Inc.*, IPR2021-0020, filed October 10, 2020 (U.S. Patent No. 10,333,768) (collectively, the "NVIDIA IPRs").

II. IDENTIFICATION OF RELEVANT PRIOR ART

A. Priority Dates

1. The Asserted Patents Are Not Entitled to Claim Priority to the Filing Dates of the '738 and '908 Provisional Applications.

In its Infringement Contentions, Plaintiff contends that the Asserted Claims are entitled to the filing dates of two provisional applications: U.S. Provisional Application Nos. 60/813,738 (the "'738 provisional application"), filed on June 13, 2006, and 60/850,908 (the "'908 provisional application"), filed on October 11, 2006. *See* Infringement Contentions at 13. Plaintiff provides no contention nor any evidence in its infringement contentions demonstrating that the alleged inventions claimed in the Asserted Patents were invented earlier than June 13, 2006. Thus, the Asserted Patents are not entitled to any date of invention earlier than the effective filing date of their earliest application dates.

A patent's claims are not entitled to an earlier priority date merely because the patent owner claims priority. *See Dynamic Drinkware, LLC v. National Graphics*, *Inc.*, 800 F.3d 1375, 1379-1380 (Fed. Cir. 2015). Rather, Plaintiff must show the earlier application provides written description support under 35 U.S.C. § 112(a). 35 U.S.C. § 120; *Dynamic Drinkware*, 800 F.3d at

1379–80; *Paice LLC v. Ford Motor Co.*, 881 F.3d 894, 906 (Fed. Cir. 2018); *PowerOasis, Inc. v. T-Mobile USA, Inc.*, 522 F.3d 1299, 1306–11 (Fed. Cir. 2008). The test for written description sufficiency is whether the prior application's disclosure "reasonably conveys to those skilled in the art that the inventor had possession of the claimed subject matter as of the filing date." *Ariad Pharms., Inc. v. Eli Lilly & Co.*, 598 F.3d 1336, 1351 (Fed. Cir. 2010) (en banc); *Augustine Medical, Inc. v. Gaymar Indus., Inc.*, 181 F.3d 1291, 1302–03 (Fed. Cir. 1999).

Plaintiff cannot meet this burden. None of the Asserted Claims are entitled to the filing dates of the '738 and '908 provisional applications because those applications lack disclosure sufficient to support the possession by the alleged inventors of each claim element of the Asserted Claims. "[T]he specification of the provisional must 'contain a written description of the invention and the manner and process of making and using it, in such full, clear, concise, and exact terms,' 35 U.S.C. § 112 ¶ 1, to enable an ordinarily skilled artisan to practice the invention claimed in the non-provisional application." *New Railhead Mfg., L.L.C. v. Vermeer Mfg. Co.*, 298 F.3d 1290, 1294 (Fed. Cir. 2002). "[T]he disclosure must show he had invented each feature that is included as a claim limitation. The adequacy of the written description (*i.e.*, the disclosure) is measured from the face of the application; the requirement is not satisfied if one of ordinary skill in the art must first make the patented invention before he can ascertain the claimed features of that invention." *Id.* at 1295. The '738 and '908 provisional applications do not satisfy this standard in that they fail to disclose the claimed structure, functions, arrangement and intercommunications of the claimed cluster nodes in the Asserted Claims.

For example, the '738 and '908 provisional applications fail to disclose the claimed "computer cluster," "kernel," "single-node kernel," "cluster node module," "cluster initialization process," or the claimed arrangements, functions, and communications of these claim elements in

each asserted claim. The ’738 and ’908 provisional applications also do not disclose the claimed arrangement, functions, and communication of the computer clusters and computer cluster node recited in the Asserted Claims.

As just one example, the provisional applications do not disclose “a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” of ’768 patent claim 1 and similar limitations in ’768 patent claims 26, 29 and 35. As another example, the provisional applications do not disclose the “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels” limitation of ’768 patent claim 1 and the similar limitations of ’768 patent claims 26, 29 and 35. As another example, the provisional applications do not disclose the “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” limitation of ’768 patent claim 1 and the similar limitations of ’768 patent claims 26, 29, and 35. As another example, the provisional applications do not disclose the “evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node”

limitation of ’768 patent claim 36. Other non-limiting examples can be found below in Section VI.A (Lack of Enablement/Written Description).

2. The ’768 Patent Is Not Entitled to a Priority Date Any Earlier than February 14, 2014.

Intel also contends that the ’768 patent claims are not entitled to a priority date any earlier than February 14, 2014. As set forth below in Section VI.A (Lack of Enablement/Written Description), there is no support in the written description or the ’768 patent’s parent applications for the Asserted Claims of the ’768 patent, including the claimed arrangements, functions, and communications of the recited “computer cluster,” “nodes,” “single-node kernels,” “cluster node module,” “cluster initialization process,” or other elements in each Asserted Claim. Each of the Asserted Claims of the ’768 patent therefore is entitled to a priority date no earlier than the ’768 patent’s filing date of February 14, 2014. *See Anascape, Ltd. v. Nintendo of Am. Inc.*, 601 F.3d 1333, 1340 (Fed. Cir. 2010) (claims only entitled to priority on application filing date). Due to a lack of priority, all Asserted Claims are invalid over intervening ACS patents and publications, such as U.S. Patent No. 8,140,612, which issued on March 20, 2012.

3. The ’621 Patent Is Not Entitled to a Priority Date Any Earlier than July 13, 2021.

Intel also contends that the ’621 patent claims are not entitled to a priority date any earlier than July 13, 2021. As set forth below in Section VI.A (Lack of Enablement/Written Description), there is no support in the written description or the ’621 patent’s parent applications for the Asserted Claims of the ’621 patent, including the claimed arrangements, functions, and communications of the recited “computer cluster,” “processors,” “cluster node modules,” “kernels,” “first commands from a user interface,” “second commands,” or other elements in each Asserted Claim. Each of the Asserted Claims of the ’621 patent therefore is entitled to a priority

date no earlier than the ’621 patent’s filing date of July 13, 2021. *See Anascape*, 601 F.3d at 1340. Due to a lack of priority, all Asserted Claims are invalid over intervening ACS patents and publications, such as U.S. Patent No. 8,140,612, which issued on March 20, 2012, and the ’768 patent, which issued on June 25, 2019.

4. The ’034 Patent Is Not Entitled to a Priority Date Any Earlier than July 13, 2021.

Intel also contends that the ’034 patent claims are not entitled to a priority date any earlier than July 13, 2021. As set forth below in Section VI.A (Lack of Enablement/Written Description), there is no support in the written description or the ’034 patent’s parent applications for the Asserted Claims of the ’034 patent, including the claimed arrangements, functions, and communications of the recited “computer cluster,” “nodes,” “single-node kernels,” “cluster node module,” “cluster initialization process,” “communications network interface,” or other elements in each Asserted Claim. Each of the Asserted Claims of the ’034 patent therefore is entitled to a priority date no earlier than the ’034 patent’s filing date of July 13, 2021. *See Anascape*, 601 F.3d at 1340. Due to a lack of priority, all Asserted Claims are invalid over intervening ACS patents and publications, such as U.S. Patent No. 8,140,612, which issued on March 20, 2012, and the ’768 patent, which issued on June 25, 2019.

5. The ’582 Patent Is Not Entitled to a Priority Date Any Earlier than January 23, 2023.

Intel also contends that the ’582 patent claims are not entitled to a priority date any earlier than January 23, 2023. As set forth below in Section VI.A (Lack of Enablement/Written Description), there is no support in the written description or the ’582 patent’s parent applications for the Asserted Claims of the ’582 patent, including the claimed arrangements, functions, and communications of the recited “computer cluster,” “processors,” “cluster node modules,”

"kernels," the translation of "user commands into calls for execution of matrix operations on an array of data elements," or other elements in each Asserted Claim. Each of the Asserted Claims of the '582 patent therefore is entitled to a priority date no earlier than the '582 patent's filing date of January 23, 2023. *See Anascape*, 601 F.3d at 1340. Due to a lack of priority, all Asserted Claims are invalid over intervening ACS patents and publications, such as U.S. Patent No. 8,140,612, which issued on March 20, 2012, and the '768 patent, which issued on June 25, 2019.

6. The '679 Patent Is Not Entitled to a Priority Date Any Earlier than October 2, 2023.

Intel also contends that the '679 patent claims are not entitled to a priority date any earlier than October 2, 2023. As set forth below in Section VI.A (Lack of Enablement/Written Description), there is no support in the written description or the '679 patent's parent applications for the Asserted Claims of the '679 patent, including the claimed arrangements, functions, and communications of the recited "computer cluster," "nodes," "communications network interface," "program code configured to interpret user instructions, identify an array of data elements, and distribute calls to at least one of a plurality of other nodes for execution of matrix operations on the array of data elements," or other elements in each Asserted Claim. Each of the Asserted Claims of the '679 patent therefore is entitled to a priority date no earlier than the '679 patent's filing date of October 2, 2023. *See Anascape*, 601 F.3d at 1340. Due to a lack of priority, all Asserted Claims are invalid over intervening ACS patents and publications, such as U.S. Patent No. 8,140,612, which issued on March 20, 2012; the '768 patent, which issued on June 25, 2019; and the '621 patent, which issued on January 24, 2023.

B. Prior Art for the Asserted Patents

Subject to its reservation of rights, Intel identifies below (in Tables 1, 2, and 3) each item of prior art that anticipates or renders obvious one or more of the Asserted Claims. Table 1 provides

the identity of each prior art patent publication by its number, filing or priority date, and date of issuance or publication. Table 2 provides the identity of each non-patent prior art publication including, where possible, its title, date of publication, author, and publisher. Table 3 provides the identity of each system or product that was in public use, on sale, or otherwise available to the public before the effective filing dates of the claimed inventions of all of the Asserted Patents, including (where possible): the name of the item; the identity of the person or entity that made the use or sale or that made the information known; and date the system or product was in public use, on sale, or otherwise available to the public. In addition to the systems or products listed in Table 3, Intel also relies on the systems and products discussed, disclosed, or referred to the prior art references listed in Tables 1 and 2. Tables 1, 2, and 3 further include references that may be relied upon to show the state of the art at the time and/or inherency regardless of whether the references themselves qualify as prior art. Intel reserves the right to rely on each reference with respect to any and all of the Asserted Patents.

Prior-art patents or publications included in these Contentions may be related (such as a divisional, continuation, continuation-in-part, parent, or child) to earlier or later-filed patents or publications, may have counterparts filed in other jurisdictions, or may incorporate (or be incorporated by) other patents or publications by reference. The listed patents or publications are intended to be representative of these other patents or publications to the extent they exist. Intel accordingly reserves the right to modify, amend, or supplement these Contentions with these related patents or publications, as well as other prior art references, upon further investigation. Additionally, any reference in these Contentions, including the appendices and exhibits thereto, to a specific subsection or subsections of 35 U.S.C. § 102, is merely exemplary, and Intel expressly reserves the right to rely on additional or other sections of 35 U.S.C. § 102, as appropriate. If

Plaintiff asserts that one or more of these references or systems fails to disclose one or more elements of a claim, Intel reserves the right to also use those references to invalidate the claim under 35 U.S.C. § 103.

As noted above, Intel incorporates by reference all prior art identified in the Other Plaintiff Cases and the NVIDIA IPRs. Intel also incorporates by reference all prior art identified in any other post-issuance review by the U.S. Patent and Trademark Office of any Asserted Patent or patent owned by Plaintiff and covering a subject matter that overlaps in whole or in part with the Asserted Patents.

Discovery is ongoing, and Intel expects relevant discovery to be obtained from Plaintiff and third parties. Intel's prior art investigation, therefore, is not yet complete. As a result, Intel reserves the right to present additional items of prior art under 35 U.S.C. § 102 or § 103 located through discovery. Intel specifically reserves the right to supplement these Invalidity Contentions with information obtained through discovery of Plaintiff and third parties.

1. Table 1: Prior Art Patents / Patent Publications[3]

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US4739476	8/1/1985	4/19/1988	Fiduccia 476	98800DOC00041406
US5423046	12/17/1992	6/6/1995	Nunnelley 046	98800DOC00038194
US5546023	6/26/1995	8/13/1996	Borkar 023	98800DOC00042414
US5829052	12/10/1996	10/27/1998	Pawlowski 052	98800DOC00038064
US5881315	8/18/1995	3/9/1999	Cohen 315	98800DOC00041973
US5930768	2/6/1996	7/27/1999	Hooban 768	98800DOC00040643
US6006259	11/20/1998	12/21/1999	Adelman 259	98800DOC00042606
US6023753	6/30/1997	2/8/2000	Pechanek 753	98800DOC00037975

[3] Intel may rely on earlier priority dates claimed by these patent publications, including to establish that the patent publication is prior art under pre-AIA 35 U.S.C. § 102(e).

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US6047323	6/27/1996	4/4/2000	Krause 323	98800DOC00039333
US6067609	4/9/1998	5/23/2000	Meeker 609	98800DOC00038751
US6074427	8/30/1997	6/13/2000	Fought 427	98800DOC00041386
US6092098	7/24/1998	7/18/2000	Araki 098	98800DOC00042650
US6108699	6/27/1997	8/22/2000	Moiin 699	98800DOC00038656
US6151663	1/2/1998	11/21/2000	Pawlowski 663	98800DOC00038056
US6167502	10/10/1997	12/26/2000	Pechanek 502	98800DOC00037914
US6195680	7/23/1998	2/27/2001	Goldszmidt 680	98800DOC00041172
US6202080	12/11/1997	3/13/2001	Lu 080	98800DOC00039229
US6338129	6/1/1999	1/8/2002	Pechanek 129	98800DOC00037871
US6418462	1/7/1999	7/9/2002	Xu 462	98800DOC00034806
US6438737	2/15/2000	8/20/2002	Morelli 737	98800DOC00038599
US6449700	9/4/1998	9/10/2002	Hagersten 700	98800DOC00040966
US6470441	11/6/2000	10/22/2002	Pechanek 441	98800DOC00037811
US6546403	1/19/2000	4/8/2003	Carlson 403	98800DOC00042083
US6578068	8/31/1999	6/10/2003	Bowman-Amuah 068	98800DOC00042132
US6604134	4/5/2002	8/5/2003	Haury 134	98800DOC00040909
US6691216	10/24/2001	2/10/2004	Kelly 216	98800DOC00039433
US6697773	3/24/1999	2/24/2004	Karchmer 773	98800DOC00039690
US6728961	3/31/1999	4/27/2004	Velasco 961	98800DOC00035147
US6751698	9/29/1999	6/15/2004	Deneroff 698	98800DOC00041742
US6769056	9/24/2002	7/27/2004	Barry 056	98800DOC00043059
US6782537	9/23/1999	8/24/2004	Blackmore 537	98800DOC00042457
US6856621	10/10/2000	2/15/2005	Artes 621	98800DOC00042640
US6859869	4/12/1999	2/22/2005	Vorbach	98800DOC00034838
US6865591	6/30/2000	3/8/2005	Karchmer 591	98800DOC00039583
US6892291	12/21/2001	5/10/2005	Pechanek 291	98800DOC00037769
US6904335	8/21/2003	6/7/2005	Solomon 335	98800DOC00036967

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US6968335	11/14/2002	11/22/2005	Bayliss 335	98800DOC00042924
US6968359	8/14/2000	11/22/2005	Miller 359	98800DOC00038696
US7003777	6/20/2001	2/21/2006		98800DOC00040820
US7015915	8/12/2003	3/21/2006	Diard 915	98800DOC00041657
US7031944	5/15/2001	4/18/2006	Tanioka 944	98800DOC00036830
US7065579	1/22/2002	6/20/2006	Traversat 579	98800DOC00036036
US7093004	11/27/2002	8/15/2006	Bernardin 004	98800DOC00042707
US7124404	5/21/2002	10/17/2006	Bebout 404	98800DOC00042903
US7130305	7/2/2001	10/31/2006	Kuukankorpi 305	98800DOC00039300
US7136924	4/16/2002	11/14/2006	Dauger 924	98800DOC00041915
US7136927	1/22/2002	11/14/2006	Traversat 927	98800DOC00035970
US7139882	2/21/2003	11/21/2006	Suzuoki 882	98800DOC00036866
US7146421	10/19/2001	12/5/2006	Syvanne 421	98800DOC00036845
US7162590	3/1/2004	1/9/2007	Pruvost 590	98800DOC00037580
US7167920	1/22/2002	1/23/2007	Traversat 920	98800DOC00035897
US7174381	12/4/2002	2/6/2007	Gulko 381	98800DOC00040995
US7177874	3/23/2004	2/13/2007	Jardin 874	98800DOC00040162
US7197624	2/9/2004	3/27/2007	Pechanek 624	98800DOC00037727
US7206841	1/22/2002	4/17/2007	Traversat 841	98800DOC00035819
US7222339	6/13/2003	5/22/2007	Rothman 339	98800DOC00037466
US7231337	2/10/2004	6/12/2007	Karchmer 337	98800DOC00039543
US7249357	8/20/2001	7/24/2007	Landman 357	98800DOC00039284
US7275123	12/6/2004	9/25/2007	Duncan 123	98800DOC00041616
US7280540	10/19/2001	10/9/2007	Halme 540	98800DOC00040940
US7289125	2/27/2004	10/30/2007	Diard 125	98800DOC00041641
US7313619	3/21/2003	12/25/2007	Torrant 619	98800DOC00036460
US7315877	2/25/2002	1/1/2008	Bhanot 877	98800DOC00042551
US7325123	3/7/2003	1/29/2008	Master 123	98800DOC00038960
US7334232	7/7/2005	2/19/2008	Jacobs 232	98800DOC00040198

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US7340500	1/22/2002	3/4/2008	Traversat 500	98800DOC00035742
US7350036	8/1/2005	3/25/2008	Kim 036	98800DOC00039410
US7401152	1/22/2002	7/15/2008	Traversat 152	98800DOC00035676
US7401153	1/22/2002	7/15/2008	Traversat 153	98800DOC00035606
US7437460	8/12/2004	10/14/2008	Chidambaran 460	98800DOC00042053
US7469309	12/12/2005	12/23/2008	Duncan 309	98800DOC00041597
US7472193	11/13/2006	12/30/2008	Dauger 193	98800DOC00041892
US7475225	12/30/2005	1/6/2009	Jourdan 225	98800DOC00040130
US7502823	2/2/2005	3/10/2009	Garg 823	98800DOC00041289
US7502915	9/29/2003	3/10/2009	Jacob 915	98800DOC00040227
US7519652	4/24/2003	4/14/2009	Page 652	98800DOC00038072
US7533141	1/24/2003	5/12/2009	Nadgir 141	98800DOC00038406
US7533168	9/11/2003	5/12/2009	Pabla 168	98800DOC00038085
US7533172	1/22/2002	5/12/2009	Traversat 172	98800DOC00035537
US7533389	6/20/2003	5/12/2009	Verbeke 389	98800DOC00034896
US7542981	10/28/2005	6/2/2009	Choy 981	98800DOC00042006
US7554949	11/7/2003	6/30/2009	Chen 949	98800DOC00042070
US7562174	6/15/2006	7/14/2009	Danilak 174	98800DOC00041954
US7568131	1/20/2006	7/28/2009	Vertes 131	98800DOC00034866
US7574523	1/22/2002	8/11/2009	Traversat 523	98800DOC00035469
US7596790	8/29/2003	9/29/2009	Moakley 790	98800DOC00038677
US7627744	5/10/2007	12/1/2009	Maher 744	98800DOC00039044
US7631165	3/7/2007	12/8/2009	Pechanek 165	98800DOC00037685
US7634530	11/10/2005	12/15/2009	Dean 530	98800DOC00041854
US7644130	12/30/2005	1/5/2010	Magro 130	98800DOC00039078
US7663633	6/25/2004	2/16/2010	Diamond 633	98800DOC00041700
US7694158	4/18/2006	4/6/2010	Melpignano 158	98800DOC00038740
US7698390	3/29/2005	4/13/2010	Harkness 390	98800DOC00040921

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US7716323	7/18/2003	5/11/2010	Gole 323	98800DOC00041115
US7757236	6/28/2004	7/13/2010	Singh 236	98800DOC00037122
US7788314	10/25/2005	8/31/2010	Holt 314	98800DOC00040659
US7788465	12/4/2003	8/31/2010	Garg 465	98800DOC00041275
US7814462	8/31/2005	10/12/2010	Husbands 462	98800DOC00040354
US7831713	12/4/2007	11/9/2010	Torrant 713	98800DOC00036443
US7835022	6/6/2006	11/16/2010	Matsumoto 022	98800DOC00038809
US7861060	12/15/2005	12/28/2010	Nickolls 060	98800DOC00038320
US7870081	12/31/2004	1/11/2011	Li 081	98800DOC00039241
US7937455	7/28/2004	5/3/2011	Saha 455	98800DOC00037424
US7962716	11/17/2004	6/14/2011	Master 716	98800DOC00038780
US7996592	5/2/2001	8/9/2011	Kim 592	98800DOC00039395
US8010593	12/17/2007	8/30/2011	Garg 593	98800DOC00041237
US8074224	12/19/2005	12/6/2011	Nordquist 224	98800DOC00038275
US8082289	5/4/2007	12/20/2011	Tannenbaum 289	98800DOC00036623
US8117288	10/12/2004	2/14/2012	Bhanot 288	98800DOC00042531
US8140612	2/29/2008	3/20/2012	Tannenbaum 612	98800DOC00036600
US8160077	12/18/2006	4/17/2012	Traversat 077	98800DOC00035405
US8176189	1/22/2002	5/8/2012	Traversat 189	98800DOC00035343
US8271605	12/21/2007	9/18/2012	Torrant 605	98800DOC00036427
US8321849	1/26/2007	11/27/2012	Nickolls 849	98800DOC00038291
US8341381	7/30/2007	12/25/2012	Pechanek 381	98800DOC00037643
US8356161	10/15/2008	1/15/2013	Master 161	98800DOC00038927
US8359397	1/22/2002	1/22/2013	Traversat 397	98800DOC00035270
US8402080	5/16/2007	3/19/2013	Tannenbaum 080	98800DOC00036578
US8402083	5/13/2009	3/19/2013	Tannenbaum 083	98800DOC00036556
US8533431	10/15/2008	9/10/2013	Garg 431	98800DOC00041205

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US8539207	11/3/2006	9/17/2013	LeGrand 207	98800DOC00039265
US8543794	1/19/2012	9/24/2013	Master 794	98800DOC00038894
US8543795	1/19/2012	9/24/2013	Master 795	98800DOC00038860
US8601101	6/28/2004	12/3/2013	Singh 101	98800DOC00037091
US8601112	3/14/2006	12/3/2013	Nordstrom 112	98800DOC00038243
US8601112	3/14/2006	12/3/2013	Nordstrom 112	98800DOC00038211
US8676877	3/16/2012	3/18/2014	Tannenbaum 877	98800DOC00036533
US8849889	3/18/2013	9/30/2014	Tannenbaum 889	98800DOC00036509
US8941668	6/25/2004	1/27/2015	Diamond 668	98800DOC00041671
US9164952	9/24/2013	10/20/2015	Master 952	98800DOC00038826
US9390057	9/14/2012	7/12/2016	Pechanek 057	98800DOC00037600
US9405564	7/31/2006	8/2/2016	Muellers 564	98800DOC00038512
US10313260	9/14/2012	6/4/2019	Torrant 260	98800DOC00036492
US11128519	6/21/2019	9/21/2021	Tannenbaum 519	98800DOC00036774
US20020013832	4/13/2001	1/31/2002	Hubbard 832	98800DOC00040376
US20020049859	8/22/2001	4/25/2002	Bruckert 859	98800DOC00042090
US20020091752	1/9/2002	7/11/2002	Firlie 752	98800DOC00041397
US20020143944	1/22/2002	1/22/2002	Traversat 944	98800DOC00036174
US20020143944	1/22/2002	1/22/2002	Traversat 944	98800DOC00036102
US20020161768	4/30/2001	4/30/2001	Miller 768	98800DOC00038716
US20020198932	6/7/2001	6/7/2001	Wagner 932	98800DOC00034835
US20030005068	9/12/2001	9/12/2001	Nickel 068	98800DOC00038349
US20030005266	6/28/2001	6/28/2001	Akkary 266	98800DOC00042628
US20030051062	9/10/2001	9/10/2001	Circenis 062	98800DOC00041988
US20030074393	10/2/2001	10/2/2001	Peart 393	98800DOC00038018
US20030121027	6/19/2001	6/19/2001	Karchmer 027	98800DOC00039924
US20030135621	12/6/2002	12/6/2002	Romagnoli 621	98800DOC00037494

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US20030177240	12/4/2002	12/4/2002	Gulko 240	98800DOC00041041
US20030195931	4/16/2002	4/16/2002	Dauger 931	98800DOC00041934
US20030195938	1/10/2003	1/10/2003	Howard 938	98800DOC00040552
US20030217112	5/16/2002	5/16/2002	Emmot 112	98800DOC00041489
US20030220960	5/21/2002	5/21/2002	Demoff 960	98800DOC00041844
US20040015968	6/20/2002	6/20/2002	Neiman 968	98800DOC00038377
US20040098359	11/14/2002	11/14/2002	Bayliss 359	98800DOC00043013
US20040098373	11/14/2002	11/14/2002	Bayliss 373	98800DOC00042971
US20040098447	11/14/2002	11/14/2002	Verbeke 447	98800DOC00035038
US20040110209	9/26/2003	9/26/2003	Yokota 209	98800DOC00034644
US20040157203	2/10/2004	2/10/2004	Dunk 203	98800DOC00041581
US20040195572	2/11/2004	10/7/2004	Kato 572	98800DOC00039482
US20040225443	5/8/2003	11/11/2004	Kamps 443	98800DOC00040111
US20040252710	6/11/2003	12/16/2004	Jeter 710	98800DOC00040144
US20040254984	6/12/2003	12/16/2004	Dinker 984	98800DOC00041626
US20050015460	7/18/2003	1/20/2005	Gole 460	98800DOC00041134
US20050021751	7/24/2003	1/27/2005	Block 751	98800DOC00042421
US20050033818	1/16/2003	2/10/2005	Jardin 818	98800DOC00040192
US20050038852	9/9/2004	2/17/2005	Howard 852	98800DOC00040435
US20050060237	9/11/2003	3/17/2005	Barsness 237	98800DOC00043120
US20050076105	9/18/2003	4/7/2005	Keohane 105	98800DOC00039420
US20050097300	10/30/2003	5/5/2005	Gildea 300	98800DOC00041195
US20050108394	11/5/2003	5/19/2005	Braun 394	98800DOC00042120
US20050132379	12/11/2003	6/16/2005	Sankaran 379	98800DOC00037330
US20050154789	1/13/2004	7/14/2005	Fellenstein 789	98800DOC00041451
US20050160398	11/22/2004	7/21/2005	Bjornson 398	98800DOC00042469
US20050165932	1/22/2004	7/28/2005	Banerjee 932	98800DOC00042892
US20050180095	10/15/2003	8/18/2005	Ellis 095	98800DOC00041517
US20050185662	2/25/2004	8/25/2005	Sheehan 662	98800DOC00037300

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US20050188055	12/31/2003	8/25/2005	Saletore 055	98800DOC00037362
US20050188088	1/13/2004	8/25/2005	Fellenstein 088	98800DOC00041433
US20050198469	11/12/2004	9/8/2005	Mitchell 469	98800DOC00038686
US20050201272	6/29/2004	9/15/2005		98800DOC00034814
US20050207421	5/20/2005	9/22/2005	Yasue 421	98800DOC00036921
US20050207421	5/20/2005	9/22/2005	Suzuki 421	98800DOC00036937
US20050235055	4/15/2004	10/20/2005	Davidson 055	98800DOC00041870
US20050235264	3/31/2005	10/20/2005	Karchmer 264	98800DOC00039827
US20050246682	7/15/2005	11/3/2005	Karchmer 682	98800DOC00039730
US20050251580	8/10/2004	11/10/2005	Tu 580	98800DOC00035160
US20050251644	5/6/2004	11/10/2005	Maher 644	98800DOC00039064
US20060026161	7/16/2004	2/2/2006	Henseler 161	98800DOC00040882
US20060026601	7/29/2004	2/2/2006	Solt 601	98800DOC00036953
US20060037016	7/28/2004	2/16/2006	Saha 016	98800DOC00037446
US20060053216	3/18/2005	3/9/2006	Deokar 216	98800DOC00041722
US20060059473	9/13/2004	3/16/2006	Moler 473	98800DOC00038628
US20060069811	8/27/2004	3/30/2006	Kelly 811	98800DOC00039441
US20060101050	10/28/2005	5/11/2006	Choy 050	98800DOC00042029
US20060106931	11/17/2004	5/18/2006	Richoux 931	98800DOC00037518
US20060107266	2/17/2005	5/18/2006	Martin 266	98800DOC00039026
US20060153201	1/11/2006	7/13/2006	Hepper 201	98800DOC00040866
US20060179436	10/28/2005	8/10/2006	Yasue 436	98800DOC00034704
US20070044099	8/17/2005	2/22/2007	Rajput 099	98800DOC00037569
US20070073705	9/23/2005	3/29/2007	Gray 705	98800DOC00041085
US20070094214	7/15/2005	4/26/2007	Li 214	98800DOC00039253
US20070094532	10/18/2006	4/26/2007	Sengupta 532	98800DOC00037316
US20070124363	12/12/2005	5/31/2007	Lurie 363	98800DOC00039087
US20070124684	11/30/2005	5/31/2007	Riel 684	98800DOC00037510
US20080250347	4/9/2007	10/9/2008	Gray 347	98800DOC00041068

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US20080281997	5/9/2007	11/13/2008	Archer 997	98800DOC00042687
US20090037691	10/15/2008	2/5/2009	Master 691	98800DOC00039000
US20200007458	6/4/2019	1/2/2020	Torrant 458	98800DOC00036477
US20200153677	6/21/2019	5/14/2020	Tannenbaum 677	98800DOC00036717
US20210344551	7/13/2021	11/4/2021	Tannenbaum 551	98800DOC00036693
US20210344552	7/13/2021	11/4/2021	Tannenbaum 552	98800DOC00036669
US20230308339	1/23/2023	9/28/2023	Tannenbaum 339	98800DOC00036647
U.S. Prov. App. No. 60799474	5/10/2006		Muellers 474	98800DOC00038538
U.S. Prov. App. No. 60813738	6/13/2006		Tannenbaum 738	98800DOC00036824
U.S. Prov. App. No. 60850908	10/11/2006		Tannenbaum 908	98800DOC00036799
WO2002033564	10/18/2001	4/25/2002	Osborn 564	98800DOC00038153
WO2002035799	10/26/2001	5/2/2002	Sim 948	98800DOC00037153
WO2002057917	1/18/2002	7/25/2002	Traversat 917	98800DOC00035180
WO2004086270	3/18/2004	10/7/2004	Katz 270	98800DOC00039452
WO2005091129	1/19/2005	9/29/2005	Ferri 129	98800DOC00041416
WO2006050404	10/28/2005	5/11/2006	Husbands 404	98800DOC00040300
EP1229442A2	1/22/2002	8/7/2002	Traversat 442	98800DOC00036335
EP1229443A2	1/22/2002	8/7/2002	Traversat 443	98800DOC00036246
EP1501013A2	7/5/2004	1/26/2005	Kanai 013	98800DOC00040056
JP2000315160A	4/20/2000	11/14/200	Krishnaswamy 160	98800DOC00039318
JP2001229145A	2/16/2000	8/24/2001	Ashikawa 145	98800DOC00042914
JP2002117010A	8/3/2001	4/19/2002	Miller 010	98800DOC00038726
JP2002223214A	11/1/2001	8/9/2002	Block 214	98800DOC00042440
JP2004061359A	7/30/2002	7/30/2002	Yamazaki 359	98800DOC00034767

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
JP2004247405A	2/12/2003	9/2/2004	Kato 405	98800DOC00039513
JP2005063033A	8/8/2003	3/10/2005	Endo 033	98800DOC00041470
JP2005071333A	7/20/2004	3/17/2005	Gole 333	98800DOC00041152
JP2006031699A	7/8/2005	2/2/2006	Romero 699	98800DOC00037484
JP2009519548A	12/12/2006	5/14/2009	Lurie 548	98800DOC00039146
JP3495010B2	6/5/2001	6/5/2001	Oshima 010	98800DOC00038112
JP3601955B2	10/23/1997	12/15/2004	Yasuda 955	98800DOC00034742
JP3628514B2	5/14/1998	3/16/2005	Imaki 514	98800DOC00040269
JPH0887473A	9/16/1994	4/2/1996	Kanai 473	98800DOC00040022
JPH10275125A	3/31/1997	10/13/1998	Saito 125	98800DOC00037400
JPH10301875A	4/28/1997	11/13/1998	Yamasato 875	98800DOC00034798
JPH11126196A	10/23/1997	5/11/1999	Yasuda 196	98800DOC00034723
JPH11224245A	2/6/1998	8/17/1999	Abe 245	98800DOC00043133
JPH11272622A	3/19/1998	10/8/1999	Saito 622	98800DOC00037382
JPH11328134A	5/14/1998	11/30/1999	Imaki 134	98800DOC00040246
US7908313	12/12/2005	3/15/2011	Lurie 313	DEF_00055985
JP2000285081A	3/31/1999	10/13/2000	Kuroishi	DEF_00054797
JPH11175485A	12/16/1997	7/2/1999	Ishii	DEF_00054433 & DEF_00054438
US20040078405A1	2/25/2002	4/22/2004	Bhanot 405	DEF_00054520
US20040081155A1	2/25/2002	8/29/2004	Bhanot 155	DEF_00053656
US20050131865A1	11/15/2004	6/16/2005	Jones	DEF_00054285
US20050240931A1	6/12/2005	10/27/2005	Padisetty	DEF_00054538
US20060088047A1	10/26/2004	4/27/2006	Dimitrov	DEF_00054303
US20060117208A1	11/17/2004	6/6/2006	Davidson	DEF_00053942
US20070233440A1	3/29/2006	10/4/2007	Fitch	DEF_00053687
US20070255782A1	5/16/2007	11/1/2007	Tannenbaum 782	DEF_00056439
US5471622	4/20/1995	11/28/1995	Eadline	DEF_00057957
US6192401B1	3/10/1999	2/20/2001	Modiri	DEF_00054586

Patent / Application	Filing Date	Publication / Issue Date	Short Name	Starting Bates Number
US6327607B1	6/25/1999	12/4/2001	Fant	DEF_00054599
US7047179B2	6/11/2003	5/16/2006	Beausoleil	DEF_00053993
US7313582	2/25/2002	12/25/2007	Bhanot 582	DEF_00053700
US7552232B2	10/24/2003	6/23/2009	Helmer	DEF_00055452
US7555566	2/25/2002	6/30/2009	Blumrich 566	DEF_00053733
US7587516	2/25/2002	9/8/2009	Bhanot 516	DEF_00053671
Provisional App. No. 60271124	2/24/2001	2002	'124 Provisional	DEF_00053531
Provisional App. No. 60720442	9/26/2005		442 Provisional	DEF_00056392
US7650434	2/25/2002	1/19/2010	Blumrich 434	DEF_00053782
US7796527	4/13/2006	9/14/2010	Archer	DEF_00053794
US7958513B2	11/17/2005	6/7/2011	Jia 513	DEF_00055469
US8001280	7/18/2005	8/16/2011	Blumrich 280	DEF_00053815
US8065503B2	12/15/2006	11/22/2011	Jia 503	DEF_00055488
US8230426B2	9/13/2007	7/24/2012	Powers	DEF_00054642
US8726278B1	7/21/2004	5/13/2014	Shawver	DEF_00057973
WO2005111843A1	5/11/2005	11/24/2005	Howard 843	DEF_00054003 & DEF_00055518
US8533402	6/22/2005	9/10/2013	Moler 402	DEF_00058873
US8055878B2	2/8/2005	11/8/2011	Chatterjee	DEF_00058858
US20050071404A1	9/25/2003	3/31/2005	Nutter	DEF_00058788

2. Table 2: Non-Patent Prior Art Publications[4]

Description	Publication Date	Short Name	Starting Bates Number
"Digital Signal Processing with Short Term Fourier Transforms" (2004). Maplesoft.	2004	MapleSoft Application Demo	98800DOC00053884
"Getting started with Star-P, Taking Your First Test Drive" (2006). Interactive Supercomputing, Inc.	2006	Star-P Getting Started	98800DOC00046679
"gridMathematica 1.1: Grid Computing Gets a Speed Boost from Mathematica 5" (2003). The Mathematica Journal, vol. 9, No. 2 .https://web.archive.org/web/20040618053429/https://www.mathematica-journal.com/issue/v9i2/newproducts/gridmathematica.html	2003		98800DOC00059991
"Home" (2006). Interactive Supercomputing https://web.archive.org/web/20060427034450/http://www.interactivesupercomputing.com, as archived Apr. 27, 2006, in 1 page.	2006		98800DOC00046672
"Install" (June 16 2000). Distributed Maple Application Software.	2000	Install2	98800DOC00072594
"Install" (March 28 2003). Distributed Maple.	2003	Install1	98800DOC00059981
"Parallel Matlab Survey" (2006). Interactive Supercomputing https://web.archive.org/web/20061018110626/http://www.interactivesupercomputing.com/reference/ParallelMatlabsurvey.htm, as archived Oct. 18, 2006, in 8 pages.	2006		98800DOC00048352

[4] References in Table 2 without Bates numbers indicate that, although NVIDIA cited the reference in its invalidity contentions, Plaintiff has not yet produced it to Intel, and Intel has not been able to locate it.

Description	Publication Date	Short Name	Starting Bates Number
"Products"(2006). Interactive Supercomputing. https://web.archive.org/web/20061018104836/http://www.interactivesupercomputing.com/product.php, as archived Oct. 18, 2006, in 3 pages.	2006		98800DOC00046657
"Resources" (2006). Interactive Supercomputing. https://web.archive.org/web/20061018105452/http://www.interactivesupercomputing.com/resources.php, as archived Oct. 18, 2006, in 3 pages.	2006		98800DOC00046654
(2001). IEEE Cluster 2001 Conference Award https://web.archive.org/web/20041221235527/http://www.cacr.caltech.edu:80/cluster2001/awards.html	2001		98800DOC00066181
(2002). MPICH2 License, https://web.archive.org/web/20050312051250/http://www-unix.mcs.anl.gov/mpi/mpich2/license.htm as archived Mar. 12, 2005 in 1 page.	2002		98800DOC00061400
(2004). MPICH—A Portable Implementation of MPI, https://web.archive.org/web/20050318092617/http://www-unix.mcs.anl.gov/mpi/mpich/ as archived Mar. 18, 2005 in 3 pages.	2005		98800DOC00052352
(2005). MPICH2 Home Page, https://web.archive.org/web/20050401035643/http://www-unix.mcs.anl.gov/mpi/mpich2/index.htm#docs as archived Apr. 1, 2005 in 2 pages.	2005		98800DOC00048674

Description	Publication Date	Short Name	Starting Bates Number
(2005). Partial MPI Library based on Sockets in Macintosh OS X, using TCP/IP Protocol, https://web.archive.org/web/20051225215129/http:/exodus.physics.ucla.edu/appleseed/dev/MacMPI_S.c, as archived Dec. 25, 2005, pp. 1.	2005		98800DOC00048673
(2005). Running Star-P on Beowulf http://courses.csail.mit.edu/18.337/2005/QuickStartGuide.pdf	2005		98800DOC00106115
(2007). Wolfram gridMathematica https://web.archive.org/web/20071004045611/http://www.wolfram.com/products/gridmathematica/	2007		98800DOC00045031
"Chapter 4, Merrimac: The Streaming Supercomputer", 2003, pp. 155-209.	2003	Merrimac Report 2003	98800DOC00059499
"Microsoft Windows Server 2003 Deployment Kit, Automating and Customizing Installations, A Resource Kit Publication", 2003, pp. 286.	2003		98800DOC00059213
"Omneta Technology: Pooch", Omneta PLC, Oct. 30, 2005, pp. 1.	2005		98800DOC00049948
"QuickSilver Software Architecture", Software, Data Sheet, Nov. 2005, pp. 4.	2005		98800DOC00048745
"The Power of Distributed Computing in Cancer Research", Supercomputing 2005 Keynote Demonstration, 2005, pp. 2.	2005		98800DOC00049966
"Topics", Cornell University, Cornell Theory Center, 2005, pp. 2.	2005		98800DOC00052224
128-xserve G5 Cluster Achieves 1.21 Teraflop. HPCWire. (2005, January 14). https://www.hpcwire.com/2005/01/14/128-xserve-g5-cluster-achieves-1-21-teraflop/	2005		98800DOC00051424

Description	Publication Date	Short Name	Starting Bates Number
1999.02.19 - Launch Den Mother and Launch Puppy (1999).			*[see footnote 3]*
1999.05.04 - Viktor Decyk's UCLA Webpage https://web.archive.org/web/19990504001939/http://exodus.physics.ucla.edu:80/Viktor/viktor.html.	1999		98800DOC00072762
1999.05.06 - UCLA Plasma Physics Group Homepage (1999).			*[see footnote 3]*
2001.04.29 - Apple - Science & Technology - Scientific Computing - Part 2 (2001).			*[see footnote 3]*
2001.04.29 - Apple - Science & Technology - Scientific Computing (2001).			*[see footnote 3]*
2002 Technical Information (2002).			*[see footnote 3]*
2003 Xserve Datasheet	2003		98800DOC00061347
2003.06.15 - Apple - Server - Solutions - Clustering Resources (2003).			*[see footnote 3]*
A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data clustering: a review. ACM Comput. Surv. 31, 3 (Sept. 1999), 264–323. https://doi.org/10.1145/331499.331504	1999		98800DOC00073268
A. Oppenheimer (1993), AppleTalk Update-Based Routing Protocol - Enhanced AppleTalk Routing	1993	Oppenheimer	98800DOC00093221
Abdennadher, N., & Boesch, R. (2005). A large scale distributed platform for high performance computing. In H. Zhuge & G. C. Fox (Eds.), Grid and Cooperative Computing - GCC 2005 (Vol. 3795, pp. 926-931). Springer.	2005	Abdennadher	98800DOC00052346

Description	Publication Date	Short Name	Starting Bates Number
Afeyan B. (2002) Choice of Optimal Wavelet Families for the Adaptive Solution of Nonlinear PDEs	2002		98800DOC00066093
Afeyan, B. (2001). Kinetic theory in laser plasma interactions.	2001		98800DOC00070733
Affidavit of Elizabeth Rosenberg, dated Sep. 1, 2020 in 7 pages.	2020	Rosenberg Affidavit	98800DOC00044004
Affidavit of Mina Ching (2025)	2025		98800DOC00044000
Agarwal, R C, et al. “A high performance parallel algorithm for 1-D FFT.” , Dec. 1994.	1994	Agarwal	98800DOC00092688
Agarwal, R. C., Gustavson, F. G., & Zubair, M. (1994, November). A high performance parallel algorithm for 1-D FFT. In Supercomputing'94: Proceedings of the 1994 ACM/IEEE Conference on Supercomputing (pp. 34-40). IEEE.	1994	Agarwal	98800DOC00092688
Ahn et al, Scatter-Add in Data Parallel Architectures, published in 11th International Symposium on High-Performance Computer Architecture, February 2005 (“Scatter-Add”).	2005	Scatter-Add	98800DOC00052334
Al Zain, A., Berthold, J., Hammond, K., & Trinder, P. (2008, May). Orchestrating production computer algebra components into portable parallel programs. In 2008 Open Source Grid and Cluster Conference, Oakland, CA, USA (pp. 257-266).	2008		98800DOC00044950

Description	Publication Date	Short Name	Starting Bates Number
Algorithms and Theory of Computation Handbook, CRC Press LLC, 1999, “single program multiple data”, in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed. 17 December 2004. (accessed TODAY) Available from: https://www.nist.gov/dads/HTML/singleprogrm.html	1999		98800DOC00072783
AMD to Unveil Dual-Core PC Chips (2005, May 31). Los Angeles Times.	2005	AMD Article	98800DOC00051419
Annaratone, M., Arnould, E., Gross, T., Kung, H. T., Lam, M., Menzilcioglu, O., & Webb, J. A. (1987). The Warp computer: Architecture, implementation, and performance. IEEE transactions on Computers, 100(12), 1523-1538.	1987		98800DOC00106690
Apple (1987). Inside Macintosh - Designing Cards and Drivers for the Macintosh II and SE .	1987		98800DOC00106421
Apple (1990). Inside AppleTalk 2nd Ed.	1990		98800DOC00105108
Apple (1990). Inside Macintosh - Designing Cards and Drivers (2nd Ed)	1990		98800DOC00104635
Apple (1990). Inside Macintosh - Guide to Macintosh Family Hardware 2nd Ed.	1990		98800DOC00104076
Apple (1991). AppleTalk Remote Access Users Guide	1991		98800DOC00103228
Apple (1991). Inside The Macintosh Communications Toolbox.	1991		98800DOC00101117
Apple (1991). Planning and Managing AppleTalk Networks.	1991		98800DOC00100839

Description	Publication Date	Short Name	Starting Bates Number
Apple (1991). Programming With AppleTalk.	1991		98800DOC00100561
Apple (1992) Inside Macintosh - Overview,	1992		98800DOC00097862
Apple (1992). Inside Macintosh - Designing Cards and Drivers (3rd Ed).	1992		98800DOC00099753
Apple (1992). Inside Macintosh - Human Interface Guidelines	1992		98800DOC00099343
Apple (1992). Inside Macintosh - Macintosh Toolbox Essentials.	1992		98800DOC00098434
Apple (1992). Inside Macintosh - Memory	1992		98800DOC00098130
Apple (1992). Inside Macintosh- Files	1992		98800DOC00097326
Apple (1993), Inside Macintosh - Text	1993		98800DOC00093742
Apple (1993). Inside Macintosh - More Macintosh Toolbox	1993		98800DOC00094858
Apple (1993). Inside Macintosh: Interapplication Communication	1993		98800DOC00095771
Apple (1993). Networking the Macintosh a step-by-step guide to using AppleTalk in Business Environments.	1993		98800DOC00093373
Apple (1994), Inside Macintosh - PowerPC System Software	1994		98800DOC00091168
Apple (1994), Inside Macintosh - Processes	1994		98800DOC00090962
Apple (1994), Inside Macintosh - Sound	1994		98800DOC00090428
Apple (1994). Inside Macintosh - Imaging With QuickDraw.	1994		98800DOC00088236
Apple (1994). Inside Macintosh - Networking	1994		98800DOC00092110

Description	Publication Date	Short Name	Starting Bates Number
Apple (1994). Inside Macintosh - Operating System Utilities.	1994		98800DOC00091718
Apple (1994). Inside Macintosh - PowerPC Numerics.	1994		98800DOC00091386
Apple (1994). Inside Macintosh: AOCE Application Interfaces	1994		98800DOC00089058
Apple (1994). Inside Macintosh: Service Access Modules	1994		98800DOC00087762
Apple (1994). Inside Macintosh-Devices	1994		98800DOC00087218
Apple (1997). Inside Macintosh - Networking with Open Transport (1997)	1997		98800DOC00081942
Apple (1999) Products - Power Mac G4 Expansion https://web.archive.org/web/19991010062925/http://www.apple.com/powermac/expansion.html.	1999		98800DOC00076688
Apple (1999) Products - Power Mac G4 Graphics Acceleration https://web.archive.org/web/19991013013450/http://apple.com/powermac/graphics.html.	1999		98800DOC00076686
Apple (1999) Products - Power Mac G4 https://web.archive.org/web/19991003085843/http://apple.com/powermac/.	1999		98800DOC00076676
Apple (1999) Products - Power Mac G4 Processor https://web.archive.org/web/19991013065614/http://apple.com/powermac/processor.html.	1999		98800DOC00076682
Apple (1999) Products - Power Mac G4Tech Specs https://web.archive.org/web/19991011004726/http://www.apple.com/powermac/specs.html.	1999		98800DOC00076679

Description	Publication Date	Short Name	Starting Bates Number
Apple (1999). Inside Macintosh - Thread Manager	1999		98800DOC00076690
Apple (2001 June 13). Apple - Science & Technology - High Performance Computing https://web.archive.org/web/20010613002734/http://apple.com/scitech/research/hiperformance/.	2001		98800DOC00070651
Apple (2001 June 25). Apple - Science & Technology - Cluster Construction https://web.archive.org/web/20010625212639/http://apple.com/scitech/research/hiperformance/cluster/.	2001		98800DOC00070653
Apple (2001) Products - Power Mac G4 https://web.archive.org/web/20010124065400/http://apple.com/powermac/.	2001		98800DOC00070655
Apple (2002 Aug 2)- Xserve https://web.archive.org/web/20020802122110/http://www.apple.com/xserve/	2002		98800DOC00065991
Apple (2002 June 4). Xserve - Technical Specifications https://web.archive.org/web/20020604023247/http://www.apple.com/xserve/specs.html	2002		98800DOC00065994
Apple (2002 Sep 3) - Xserve - Architecture https://web.archive.org/web/20020903234943/http://www.apple.com/xserve/architecture.html	2002		98800DOC00066001
Apple (2002 Sep 3)- Xserve - Storage https://web.archive.org/web/20020903234946/http://www.apple.com/xserve/storage.html	2002		98800DOC00065997

Description	Publication Date	Short Name	Starting Bates Number
Apple (2002 Sep 4) - Xserve - Design https://web.archive.org/web/20020904000510/http://www.apple.com/xserve/design.html	2002		98800DOC00065999
Apple (2003 April 1). Apple - Xserve https://web.archive.org/web/20030401205753/http://www.apple.com/xserve/.	2003		98800DOC00061322
Apple (2003 April 1). Apple - Xserve Management https://web.archive.org/web/20030401082612/http://www.apple.com/xserve/management.html.	2003		98800DOC00061344
Apple (2003 April 1). Apple - Xserve Storage https://web.archive.org/web/20030401210006/http://www.apple.com/xserve/storage.html.	2003		98800DOC00061329
Apple (2003 April 1). Apple - Xserve Technical Specifications https://web.archive.org/web/20030401205624/http://www.apple.com/xserve/specs.html.	2003		98800DOC00061325
Apple (2003 February 2). Xserve Performance https://web.archive.org/web/20030202065752/http://www.apple.com/xserve/performance.html.	2003		98800DOC00061332
Apple (2003 October 09). Apple - Server - Solutions - Compute Clustering https://web.archive.org/web/20031009111022/http://www.apple.com/server/clustering.html.	2003		98800DOC00061360

Description	Publication Date	Short Name	Starting Bates Number
Apple (2003) ADC Developer Documentation https://web.archive.org/web/20030409071058/http://developer.apple.com/techpubs/.	2003		98800DOC00061398
Apple (2003). Power Mac G4-Xserve Technology Overview.	2003		98800DOC00061363
Apple (2003). Xserve Cluster Node Quick Start Guide	2003		98800DOC00061352
Apple (2003). Xserve Performance Data	2003		98800DOC00061338
Apple Computer Inc. (1988). Apple Numerics Manual (2nd ed.). Addison-Wesley Pub. Co.	1988		98800DOC00105802
Apple WWDC 2004 Session 642			*[see footnote 3]*
Apple. (1985). Inside Macintosh: Volume I.	1985		98800DOC00108394
Apple. (1985). Inside Macintosh: Volume II.	1985		98800DOC00107954
Apple. (1985). Inside Macintosh: Volume III.	1985		98800DOC00107664
Apple. (1986). Inside Macintosh: Volume IV.	1986		98800DOC00107330
Apple. (1986). Inside Macintosh: Volume V.	1986		98800DOC00106706
Apple. (1991). Inside Macintosh: Volume VI.	1991		98800DOC00101479
Apple. (2001 Aug. 31) - Apple - Science & Technology - UCLA Project AppleSeed Page 2 https://web.archive.org/web/20010831135648/http://www.apple.com/scitech/research/appleseed/index2.html	2001		98800DOC00070647

Description	Publication Date	Short Name	Starting Bates Number
Apple. (2001 June 25) - Apple - Science & Technology - UCLA Project AppleSeed Page 1 https://web.archive.org/web/20010625212304/http://www.apple.com:80/scitech/research/appleseed/	2001		98800DOC00070649
Appleseed (1999 April 28). Appleseed Development Page https://web.archive.org/web/19990428014109/http://exodus.physics.ucla.edu:80/appleseed/appleseed.html	1999		98800DOC00076672
AppleSeed (1999). High School Mac Cluster Projects.	1999		98800DOC00076671
Appleseed (2000 August 16). Appleseed Recipe https://web.archive.org/web/20000816011958/http://exodus.physics.ucla.edu:80/appleseed/appleseedrecipe.html	2000		98800DOC00072672
Appleseed (2000 June 22). Appleseed Development Page https://web.archive.org/web/20000622031549/http://exodus.physics.ucla.edu:80/appleseed/appleseed.html	2000		98800DOC00070824
Appleseed (2000). Brent's Appleseed Cluster Page-The Lovercraft Cluster.	2000		98800DOC00072671
AppleSeed (2000). Email from Jan de Leeuw re Appleseed Development.	2000		98800DOC00072670
Appleseed (2001 April 29). Appleseed Development Page https://web.archive.org/web/20010429235433/http://exodus.physics.ucla.edu/AppleSeed/AppleSeed.html	2001		98800DOC00070639

Description	Publication Date	Short Name	Starting Bates Number
Appleseed (2001 February 1). Appleseed Development Page https://web.archive.org/web/20010201075800/http://exodus.physics.ucla.edu/appleseed/appleseed.html	2001		98800DOC00070643
Appleseed (2001 May 1). Appleseed Recipe https://web.archive.org/web/20010501201550/http://exodus.physics.ucla.edu/AppleSeed/appleseedrecipe.html	2001		98800DOC00070638
Appleseed (2002). Frequently Asked Questions about AppleSeed.	2002		98800DOC00065987
Appleseed (2003). Appleseed Multiprocessing Development Page.	2003		98800DOC00061321
AppleSeed (2004). AppleSeed Multiprocessing Development Page.	2004		98800DOC00056029
AppleSeed (2004). AppleSeed Technical Report Archive.	2004		98800DOC00056028
Appleseed (2004). AppleSeed:Recipe for Building a Simple Cluster	2004		98800DOC00056027
Appleseed (2004). Benchmarks of AppleSeed Performance.	2004		98800DOC00056025
AppleSeed (2004). Header File for MacMP.	2004		98800DOC00056024
AppleSeed (2004). Multitasking library for Apple Macintosh.	2004		98800DOC00056023
AppleSeed (2004). Other AppleSeed Clusters, at https://web.archive.org/web/20041012015003/http://exodus.physics.ucla.edu:80/appleseed/appleseedsites.html.	2004		98800DOC00056020
AppleSeed (2004). Publicity About AppleSeed.	2004		98800DOC00056019

Description	Publication Date	Short Name	Starting Bates Number
Appleseed (2005). AppleSeed Technical Report Archive.	2005		98800DOC00052333
AppleSeed Development Page, (Apr. 12, 2004) https://web.archive.org/web/20040505012419/https://exodus.physics.ucla.edu/appleseed/dev/developer.html	2004		98800DOC00056030
Association for Computing Machinery. (n.d.). ACM Digital Library. ACM. Retrieved March 26, 2025, from https://www.acm.org/publications/digital-library	2025		98800DOC00044003
Atkin, P. (1987). Performance maximisation. INMOS.	1987		98800DOC00106402
Atkin, P., & Ghee, S. (1991). A transputer based multi-user flight simulator. Microprocessors and Microsystems, 15(5), 271-278.	1991		98800DOC00100536
Atkin, P., & Packer, J. (1988). High performance graphics with the IMS T800. INMOS Technical Node, 37.	1988		98800DOC00105769
Awan, A., Ferreira, R. A., Jagannathan, S., & Grama, A. Y. (2004). Unstructured peer-to-peer networks for sharing processor cycles (Department of Computer Science Technical Report No. 1616). Purdue University.	2005		98800DOC00052317
Background (2006). Interactive Supercomputing. https://web.archive.org/web/20060427034450/http://www.interactivesupercomputing.com/index.php?option=com_content&task=view&id=30&Itemid=73, as archived Apr. 27, 2006, pp. 2.	2006		98800DOC00048671

Description	Publication Date	Short Name	Starting Bates Number
Bailey, D., Barton, J., Lasinski, T., & Simon, H. (1993). The NAS parallel benchmarks. NASA Technical Memorandum.	1993		98800DOC00093303
Bailey, D., Harris, T., Saphir, W., Van Der Wijngaart, R., Woo, A., & Yarrow, M. (1995). The NAS parallel benchmarks 2.0 (Vol. 156). Technical Report NAS-95-020, NASA Ames Research Center.	1995	NAS Benchmark	98800DOC00086617
Bajaj, C. L., Pascucci, V., Thompson, D., & Zhang, X. Y. (1999). Parallel accelerated isocontouring for out-of-core visualization. In Proceedings of the 1999 IEEE Parallel Visualization and Graphics Symposium (pp. 97-122).	1999		98800DOC00076568
Bajaj, C., & Cutchin, S. (1999). Web-based collaborative visualization of distributed and parallel simulation. In Proceedings of the 1999 IEEE Parallel Visualization and Graphics Symposium (pp. 47-54). IEEE.	1999		98800DOC00076552
Bajaj, C., Ihm, I., Koo, Gb., Park, S. (1999). Parallel Ray Casting of Visible Human on Distributed Memory Architectures. In: Gröller, E., Löffelmann, H., Ribarsky, W. (eds) Data Visualization ’99. Eurographics. Springer, Vienna. https://doi.org/10.1007/978-3-7091-6803-5_26	1999		98800DOC00076560

Description	Publication Date	Short Name	Starting Bates Number
Bajaj, C., Ihm, I., Min, J., & Oh, J. (2004, December). SIMD optimization of linear expressions for programmable graphics hardware. In Computer Graphics Forum (Vol. 23, No. 4, pp. 697-714). 9600 Garsington Road, Oxford, OX4 2DQ, UK.: Blackwell Publishing Ltd..	2004		98800DOC00055981
Bajaj, C., Park, S., & Thane, A. (2002). Parallel multi-PC volume rendering system. CS & ICES Technical Report, University of Texas at Austin, 2, 2.	2002		98800DOC00065970
Bajaj, C., Zhang, X., & Blanke, W. (2001). Scalable isosurface visualization of massive datasets on COTS clusters. In Proceedings of the IEEE Symposium on Parallel and Large-Data Visualization and Graphics (pp. 51-58). IEEE.	2001		98800DOC00070628
Balay et al., "PETSc Users Manual", ANL-95/11—Revision 2.3.2, Argonne National Laboratory, Sep. 1, 2006, pp. 184.	2006	PETSc User Manual	98800DOC00048487
Banerjee, P., et. al (2000, April). Heterogeneous platform. In Proceedings of the Symposium on FPGA Custom Computing Machines (FCCM-2000). Napa Valley, CA.	2000		98800DOC00072660
Barnell, M. D., & Rahn, B. J. (2006, May). Migrating modeling and simulation applications on to high performance computers. In Enabling Technologies for Simulation Science X (Vol. 6227, pp. 198-205). SPIE.	2006	Barnell	98800DOC00048479

Description	Publication Date	Short Name	Starting Bates Number
Bassi, M., Becciani, U., Lombardo, U., Russo, G., Sandulescu, N., & Zuo, W. (1999). A parallel code for the kinetic Landau—Vlasov transport equation. Computer physics communications, 118(2-3), 110-118.	1999	Bassi	98800DOC00076543
Bauer, M. A., Coburn, N., Erickson, D. L., Finnigan, P. J., Hong, J. W., Larson, P. A., ... & Teorey, T. J. (1994). A distributed system architecture for a distributed application environment. IBM Systems Journal, 33(3), 399-425.	1994	Bauer	98800DOC00111749
Becker, J. C., Nitzberg, B., & Van Der Wijngaart, R. F. (1998, January). Predicting cost/performance trade-offs for Whitney: a commodity computing cluster. In Proceedings of the Thirty-First Hawaii International Conference on System Sciences (Vol. 7, pp. 504-513). IEEE.	1998	Whitney Tradeoff	98800DOC00080795
Behling, S. et. al, The POWER4 Processor Introduction and Tuning Guide (Nov 2001). Redbooks	2001		98800DOC00068136
Bekker, S. (2004). HPC Edition of Windows Server 2003 on Tap for 2005. ENT News, 3.	2004		98800DOC00055970
Bekker, S. J. (2004, December 10). Muglia updates Windows Server Roadmap. Redmond Channel Partner. https://rcpmag.com/articles/2004/12/10/muglia-updates-windows-server-roadmap.aspx	2004		98800DOC00055968

Description	Publication Date	Short Name	Starting Bates Number
Bekker, S. J. (2005, January 10). Microsoft Enterprise Product Roadmap for 2005. Redmond Channel Partner. https://rcpmag.com/articles/2005/01/10/microsoft-enterprise-product-roadmap-for-2005.aspx	2005		98800DOC00052306
Bergeron, S., & Vincent, A. (1999). Implementation strategies for real-time particle transport solver. Computer physics communications, 120(2-3), 177-184.	1999	Bergeron	98800DOC00076535
Bernardin, L. (1996). Maple on the Intel Paragon. Internal report/Swiss Federal Institute of Technology, Computer Science Departement, 251.	1996	Bernardin	98800DOC00085835
Bernardin, L. (1997). Maple on a massively parallel, distributed memory machine. In Proceedings of the second international symposium on Parallel symbolic computation (PASCO '97) (pp. 217–222). Association for Computing Machinery. https://doi.org/10.1145/266670.266732	1997		98800DOC00081936
Bosa, K., & Schreiner, W. (2002). Task logging, rescheduling, and peer checking in Distributed Maple. Technical Report 02-10, RISC-Linz, Johannes Kepler University, Linz, Austria.	2002	Schreiner3	98800DOC00065929
Bridges, P., Doss, N., Gropp, W., Karrels, E., Lusk, E., & Skjellum, A. (1995). User's Guide to MPICH, a Portable Implementation of MPI. Argonne National Laboratory, 9700, 60439-4801.	1995	Bridges	98800DOC00086582

Description	Publication Date	Short Name	Starting Bates Number
Brook Files, at https://sourceforge.net/projects/brook/files/Official%20Releases/BrookGPU%20v0.4/.	2004		98800DOC00055967
Buchberger, Bruno & Schreiner, Wolfgang. (1997). CONCERT - A Software Architecture for Coordinating Education Sessions in Distributed Environments.	1997	Buchberger	98800DOC00081875
Buck et al., "Brook for GPUs", Presentation, Feb. 10, 2003, pp. 19.	2003		98800DOC00061302
Buck, I. (2001). Brook: A streaming programming language. Oct, 8, 12.	2001		98800DOC00070616
Buck, I. (2007). Gpu computing with nvidia cuda. In ACM SIGGRAPH 2007 courses (pp. 6-es).	2007		98800DOC00046310
Buck, I. (2009). From Brook to CUDA. In Gpu Technology Conference (p. 30).	2009		98800DOC00044877
Buck, I. (2012). Tesla, GPU Computing, Past, Present, Future.	2012		98800DOC00044634
Buck, I. (September 2006). Stream Computing on Graphics Hardware. Dissertation, Stanford University, 106.	2006	Buck Thesis	98800DOC00048373
Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., & Hanrahan, P. (2004). Brook for GPUS: Stream Computing on Graphics Hardware" . ACM SIGGRAPH 2004 Papers, 777–786.	2004	Brook for GPU 2004	98800DOC00055921

Description	Publication Date	Short Name	Starting Bates Number
Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., & Hanrahan, P. (2004). Brook for GPUS: Stream Computing on Graphics Hardware, Presentation. ACM SIGGRAPH 2004 Papers, 36.	2004	SIGGRAPH 2004 Presentation	98800DOC00055931
Burns et al., "LAM: An Open Cluster Environment for MPI", Ohio Supercomputer Center, Colombus, OH, 2001, pp. 8.	2001		98800DOC00070608
Butt, A. R., Zhang, R., & Hu, Y. C. (2003). A self-organizing flock of Condors. In SC '03: Proceedings of the 2003 ACM/IEEE Conference on Supercomputing (pp. 42-42). IEEE. https://doi.org/10.1145/1048935.1050192	2003	Butt	98800DOC00061291
Carns, P. H., Ligon, W. B., McMillan, S. P., & Ross, R. B. (1999, March). An evaluation of message passing implementations on Beowulf workstations. In 1999 IEEE Aerospace Conference. Proceedings (Cat. No. 99TH8403) (Vol. 5, pp. 41-54). IEEE.	1999		98800DOC00075798
Cesari, G., & Maeder, R. (1996). Performance analysis of the parallel Karatsuba multiplication algorithm for distributed memory architectures. Journal of Symbolic Computation, 21(4-6), 467-473.	1996	Cesari	98800DOC00085828

Description	Publication Date	Short Name	Starting Bates Number
Cha, D., Zhang, Q., Tithi, J. J., Rand, A., Chowdhury, R. A., & Bajaj, C. (2015, September). Accelerated molecular mechanical and solvation energetics on multicore CPUs and manycore GPUs. In Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics (pp. 222-231).	2015		98800DOC00044240
Chen, J. Y. C. (2006). Fast Fourier Transform with BrookGPU. Apr, 25, 16.	2006	FFT with Brook GPU Presentation	98800DOC00048357
Cheng, David & Shah, Viral & Gilbert, John & Edelman, Alan. (2004). Fast Sorting on a Distributed-Memory Architecture.	2004	Cheng	98800DOC00055912
Chinchilla, F., Gamblin, T., Sommervoll, M., & Prins, J. F. (2004). Parallel n-body simulation using GPUs. Department of Computer Science, University of North Carolina at Chapel Hill, http://gamma. cs. unc. edu/GPGP, Technical Report TR04-032.	2004	Chinchilla	98800DOC00055904
Choi, J., Dongarra, J. J., & Walker, D. W. (1995). Parallel matrix transpose algorithms on distributed memory concurrent computers. *Parallel Computing, 21*(10), 1387–1405.	1994		98800DOC00087210
Chowdhury, R., & Bajaj, C. (2010, September). Multi-level grid algorithms for faster molecular energetics. In Proceedings of the 14th ACM Symposium on Solid and Physical Modeling (pp. 147-152).	2010		98800DOC00044711

Description	Publication Date	Short Name	Starting Bates Number
Choy et al., "Star-P High {Productivity Parallel Computing", Presentation Agenda, Jun. 9, 2004, pp. 33.	2004		98800DOC00055869
Choy, L. Y. (2002). MATLAB* P 2.0: Interactive supercomputing made practical (Doctoral dissertation, Massachusetts Institute of Technology).	2002	Choy Thesis	98800DOC00065846
Choy, Long Yin, (2002). Chapter 1, Draft, MIT.	2002		98800DOC00065913
Choy, R. (2004). Lecture Slides. MIT. https://web.archive.org/web/2005090320572/http://beowulf.Ics.mit.edu/18.337-2004/lectslides.html, pp. 2.	2004		98800DOC00055902
Choy, R. et al., (2005). Parallel MATLAB: Doing it right. Presentation. MIT, pp 55	2005	Choy 3	98800DOC00052240
Choy, R. et al., (2005). Parallel MATLAB: Doing it right. Proceedings of the IEEE. vol. 93, No. 2, Feb. 2005, pp. 331-341	2005	Choy 2	98800DOC00052295
Choy, R., & Edelman, A. (2003). Matlab* P 2.0: a unified parallel MATLAB.	2003	Choy 1	98800DOC00061285
Choy, R., & Edelman, A. (2005). Parallel MATLAB: Doing it right. Proceedings of the IEEE, 93(2), 331-341.	2005	Choy 2	98800DOC00052229
Choy, R., Edelman, A., Gilbert, J. R., Shah, V., & Cheng, D. (2004, June). Star-P: High productivity parallel computing. In In 8th Annual Workshop on High-Performance Embedded Computing (HPEC 04).	2004	Choy Star-P	98800DOC00055899

Description	Publication Date	Short Name	Starting Bates Number
Chu, E., & George, A. (1998). FFT algorithms and their adaptation to parallel processing. Linear algebra and its applications, 284(1-3), 95-124.	1998	Chu	98800DOC00080765
Conference Program. SC 2009 Portland, Oregon. (2009 November 20,).	2009		98800DOC00044719
Cornell University. (2005, November 15). Theory Center offers new options, seeks input from researchers. Cornell Chronicle. Retrieved March 26, 2025, from https://news.cornell.edu/stories/2005/11/theory-center-offers-new-options-seeks-input-researchers	2005		98800DOC00052226
Corp, X., & Corp, D. (1981). The ethernet: a local area network: data link layer and physical layer specifications. Comput. Commun. Rev., 11, 20-66.	1981		98800DOC00108958
Cotronis, Y., & European PVM MPI Users' Group Meeting Thira 2001. (2001). Recent advances in parallel virtual machine and message passing interface proceedings. In Recent Advances in Parallel Virtual Machine and Message Passing Interface. Springer.	2001		98800DOC00070155
Coulaud, O., Sonnendrücker, E., Dillon, E., Bertrand, P., & Ghizzo, A. (1999). Parallelization of semi-Lagrangian Vlasov codes. Journal of plasma physics, 61(3), 435-448.	1999	Coulaud	98800DOC00075784

Description	Publication Date	Short Name	Starting Bates Number
Cownie, J., & Gropp, W. (1999, September). A standard interface for debugger access to message queue information in MPI. In European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting (pp. 51-58). Berlin, Heidelberg: Springer Berlin Heidelberg.	1999	Cownie	98800DOC00075775
Cramer, C. E., & Board, J. A. (2000). The Development and Integration of a Distributed 3D {FFT} for a Cluster of Workstations. In 4th Annual Linux Showcase & Conference (ALS 2000).	2000	Cramer	98800DOC00072651
Crandall, R. (2000). PowerPC G4 for Engineering, Science, and Education. Apple Distinguished Scientist Advanced Computation Group.	2000	Crandall	98800DOC00072639
Culler, D., Singh, J. P., & Gupta, A. (1998). Parallel computer architecture: a hardware/software approach. Gulf Professional Publishing.	1998	Culler	98800DOC00079709
Culler, D., Singh, J. P., & Gupta, A. (1999). Parallel computer architecture: a hardware/software approach. Gulf Professional Publishing.	1999		98800DOC00074716
D. Jeffrey & S. Ghemawat(2004). MapReduce: simplified data processing on large clusters. In OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December.	2004		98800DOC00055598

Description	Publication Date	Short Name	Starting Bates Number
Dally, W. J., Labonte, F., Das, A., Hanrahan, P., Ahn, J. H., Gummaraju, J., ... & Kapasi, U. J. (2003, November). Merrimac: Supercomputing with streams. In Proceedings of the 2003 ACM/IEEE Conference on Supercomputing (p. 35).	2003	Merrimac 2003	98800DOC00061277
Dally, W., Hanrahan, P., & Fedkiw, R. (2001). A streaming supercomputer. Whitepaper, Sep.	2001		98800DOC00070138
Dana, P., & Diana, D. (2005). An extension of Maple for grid and cluster computing. *Studies in Informatics and Control, 14*(1), 31–36.	2005		98800DOC00052210
Dauger (2001). Pooch Quick Start. Dauger Research Inc.	2001		98800DOC00070129
Dauger (2003). Pooch Application-Dauger Research at FOSE 2003. Dauger Research Inc.	2003		98800DOC00061270
Dauger (2004). 33-XServe Cluster Running the AltiVec Fractal Benchmark. Dauger Research Inc.	2004		98800DOC00055864
Dauger (2004). Fresnel Diffraction Webpage, Dauger Research Inc.	2004		98800DOC00055858
Dauger (2004). High-performance parallel software utilizing. Dauger Research Inc.	2004		98800DOC00055857
Dauger (2004). mpich with Pooch https://web.archive.org/web/20041216175303/http://daugerresearch.com/pooch/mpich/	2004		98800DOC00055854
Dauger (2004). Pooch Application-Command Line Control. Dauger Research Inc. https://web.archive.org/web/20041216175653/http://daugerresearch.com/pooch/cli.html.	2004		98800DOC00055852

Description	Publication Date	Short Name	Starting Bates Number
Dauger (2004). Pooch Application-Getting Started. Dauger Research Inc.	2004		98800DOC00055850
Dauger (2004). Pooch Application-Mac Cluster Deployments. Dauger Research Inc. https://web.archive.org/web/20041220140942/http://daugerresearch.com/pooch/deployment.html	2004		98800DOC00055849
Dauger (2004). Pooch Application-Parallel Applications. Dauger Research Inc. https://web.archive.org/web/20041216073536/http://daugerresearch.com/pooch/parallelapps.html.	2004		98800DOC00055868
Dauger (2004). Pooch Application-Pooch in use. Dauger Research Inc. https://web.archive.org/web/20041205134719/http://www.daugerresearch.com:80/pooch/users.html.	2004		98800DOC00055847
Dauger (2004). Pooch Application-Pooch Manual. Dauger Research Inc.	2004		98800DOC00055846
Dauger (2004). Pooch Application-Pooch Software Development Kit. Dauger Research Inc.	2004		98800DOC00055845
Dauger (2004). Pooch Application-Publications. Dauger Research Inc.	2004		98800DOC00055844
Dauger (2004). Pooch Application-Sample AppleScripts. Dauger Research Inc.	2004		98800DOC00055843
Dauger (2004). Pooch Application-Supported Implementation of MPI. Dauger Research Inc.	2004		98800DOC00055842

Description	Publication Date	Short Name	Starting Bates Number
Dauger (2004). Pooch Application-Visualization Monitor. (MacMPI X Visualization) Dauger Research Inc. https://web.archive.org/web/20041213185909/http://daugerresearch.com/pooch/visualization.html	2004		98800DOC00055841
Dauger (2004). Pooch in Use. Dauger Research Inc.	12/5/2004	Pooch in Use	98800DOC00055831
Dauger (2004). Pooch Quick Start. Dauger Research Inc.	2004		98800DOC00055796
Dauger (2004). Pooch Quick Start. Pooch Application, Parallel Operation and Control Heuristic Application, Version 1.4. Dauger Research, Inc.	1/6/2004	Pooch Manual v1.4	98800DOC00055709
Dauger (2004). Pooch Quick Start. Pooch Application, Parallel Operation and Control Heuristic Application, Version 1.5. Dauger Research, Inc.	6/28/2004	Pooch Manual v1.5	98800DOC00055611
Dauger (2004). Pooch v1.5.5 Supports LAM/MPI and mpich-gm. Dauger Reasearch Inc. https://web.archive.org/web/20041027031211/http://daugerresearch.com:80/pr/poochv1.5.5.html	2004		98800DOC00055795
Dauger (2004). USC Macintosh Cluster Running the AltiVec Fractal Benchmark. Dauger Research Inc.	2001		98800DOC00055791
Dauger (2005). Altive Fractal Carbon Demonstration Software. Dauger Research Inc.	2005		98800DOC00052205

Description	Publication Date	Short Name	Starting Bates Number
Dauger (2005). Pooch - MPI Implementation. Dauger Research Inc. https://web.archive.org/web/20051228111917/http://www.daugerresearch.com/pooch/mpi.shtml.	2005		98800DOC00052107
Dauger (2006). Pooch Scripting and Automation. Dauger Research Inc. https://web.archive.org/web/20060217183552/http://daugerresearch.com/pooch/scripting.shtml.	2006		98800DOC00048286
Dauger Research (2004). "Pooch Application- Build Your Personal Computing Cluster in minutes!" Dauger Research Inc.	2004		98800DOC00055853
Dauger Research Store (Oct. 15, 2004) https://web.archive.org/web/20041015041402/http://daugerresearch.com:80/store/.	2004		98800DOC00055862
Dauger Research, Inc., - Pooch v1.4 Announcement https://web.archive.org/web/20040202010019/http://daugerresearch.com:80/pr/poochv1.4.html	2004		98800DOC00055866
Dauger Research. (2005). Visualizing Message Passing with MacMPI. Retrieved February 5, 2025, from https://daugerresearch.com/vault/macmpitutorial.shtml	2005		98800DOC00052086
Dauger Research. (2005, March/April). Computing in Science & Engineering, excerpt (p. 2). IEEE Computer Society.	2005		98800DOC00052197

Description	Publication Date	Short Name	Starting Bates Number
Dauger Research. (2006). PoochMPI Math. Dauger Research. Retrieved January 31, 2025, from https://daugerresearch.com/pr/poochmpimath.shtml	2006		98800DOC00046343
Dauger, “Plug-and-Play” Clustering: Build Your Cluster in Minutes!, Apple HPC Seminar Series (2005)	7/1/2005	Plug-and-Play Presentation	98800DOC00052110
Dauger, D et al. (2005) Pooch Application-What's New. Dauger Research Inc.	2005		98800DOC00052099
Dauger, D. (1999). The Launch Den Mother and Launch Puppy. Read Me.	1999	Launch Read Me	98800DOC00074706
Dauger, D. (2002). Pooch application: Pooch quick start, parallel operation and control heuristic application (Version 1.2.1, p. 61). Dauger Research.	2002		98800DOC00065774
Dauger, D. (2003). Pooch application: Parallel operation and control heuristic application (Version 1.3.5, p. 67). Dauger Research.	2003		98800DOC00061203
Dauger, D. (2004). Pooch Application - The Mac Cluster Recipe. Dauger Research Inc. https://web.archive.org/web/20041214054400/http://daugerresearch.com/pooch/recipe.html	2004		98800DOC00055793
Dauger, D. (2004). Pooch FAQ Page. Dauger Research, Inc. https://web.archive.org/web/20050404033441/http://daugerresearch.com/pooch/faq.html	2005		98800DOC00055834

Description	Publication Date	Short Name	Starting Bates Number
Dauger, D. (2005). DR's Pooch Cluster Software For Mac OS X v10.4 "Tiger." Dauger Research, Inc., - pooch rides the "Tiger." https://www.daugerresearch.com/pr/PoochRidesTiger.shtml	2005		98800DOC00052203
Dauger, D. (2005). Pooch-Related Documentation. Dauger Research, Inc.	2005		98800DOC00052092
Dauger, D. (2006). Pooch Application - The Mac Cluster Recipe. Dauger Research Inc. https://web.archive.org/web/20060118083804/http://daugerresearch.com/pooch/recipe.shtml	2006		98800DOC00047966
Dauger, D. E. (1996). Simulation and study of Fresnel diffraction for arbitrary two-dimensional apertures. Computers in Physics, 10(6), 591-604.	1996		98800DOC00085813
Dauger, D. E. (2001). Semiclassical modeling of quantum-mechanical multiparticle systems using parallel particle-in-cell methods. University of California, Los Angeles.	2001		98800DOC00070105
Dauger, D. E. (2004). Parallelization- Knock - knock.f sample code https://web.archive.org/web/20041027034254/http://daugerresearch.com/pooch/knock.f			98800DOC00048318
Dauger, D. E. (2004). Parallelization- Parallel Pascal Triangle - parallelpascaltriangle.f90 sample code https://web.archive.org/web/20041103044816/http://daugerresearch.com/pooch/parallelpascalstriangle.f90			98800DOC00055806

Description	Publication Date	Short Name	Starting Bates Number
Dauger, D. E. (2004). Parallelization- Pascal Triangle - pascaltriangle.f90 sample code https://web.archive.org/web/20041103044632/http://daugerresearch.com/pooch/pascalstriangle.f90			98800DOC00048292
Dauger, D. E. (2005). Keeping America Strong Award-Dauger Research Honored with "Keeping America Strong" Award. Dauger Research, Inc. https://daugerresearch.com/media/kasfull.shtml	2005		98800DOC00052199
Dauger, D. E. (2005).Pooch-Purchase. Dauger Research https://www.daugerresearch.com/pooch/purchase.shtml https://web.archive.org/web/20050404073345/http://daugerresearch.com/pooch/purchase.html.	2005		98800DOC00052093
Dauger, D. E. (2005, September 21). "Pooch Selected for Omneta Supercomputer-Omneta Selects Pooch Cluster Technology for Several Thousand Node System." Dauger Research, inc. https://www.daugerresearch.com/pr/omneta.shtml	2005		98800DOC00052095
Dauger, D. E. (2006) "Supercomputing Engine for Mathematica". Advanced Cluster Systems, LLC and Dauger Research inc.	2006		98800DOC00047968
Dauger, D. E. (2006) "The PoochMPI Toolkit", High-Performance Computing with Mathematica, Wolfram Technology Conference 2006, Champaign, IL. Advanced Cluster Systems, LLC and Dauger Research Inc.	2006	PoochMPI Presentation 2006	98800DOC00047921

Description	Publication Date	Short Name	Starting Bates Number
Dauger, D. E. (2006). "Supercomputer-Style Cluster Computing Support for Mathematica." Advanced Cluster Systems, LLC and Dauger Research Inc. https://daugerresearch.com/pooch/mathematica.shtml	2006		98800DOC00047991
Dauger, D. E. (2006). "Visualizing Message Passing with MacMPI", Dauger Research Inc. https://daugerresearch.com/vault/macmpitutorial.shtml	2006		98800DOC00047915
Dauger, D. E. (2006). Parallel Zoology, Dauger Research Vault. https://daugerresearch.com/vault/parallelzoology.shtml https://web.archive.org/web/20041028071907/http://www.daugerresearch.com/pooch/parallelzoology.html	2006		98800DOC00048336
Dauger, D. E. (2006). Parallelization, Dauger Research Vault. https://daugerresearch.com/vault/parallelization.shtml https://web.archive.org/web/20041216073536/http://daugerresearch.com/pooch/parallelapps.html.	2006		98800DOC00048288
Dauger, D. E. (2006). Parallelization-Parallel Adder. Dauger Research Vault. https://daugerresearch.com/vault/paralleladder.shtml https://web.archive.org/web/20041019044831/http://daugerresearch.com/pooch/paralleladder.html.	2006		98800DOC00055822

Description	Publication Date	Short Name	Starting Bates Number
Dauger, D. E. (2006). Parallelization-Parallel Circle Pi. Dauger Research Vault. https://daugerresearch.com/vault/parallelcirclepi.shtml https://web.archive.org/web/20050404034821/http://daugerresearch.com/pooch/parallelcirclepi.html.	2006		98800DOC00048322
Dauger, D. E. (2006). Parallelization-Parallel Knock. Dauger Research Vault. https://www.daugerresearch.org/vault/parallelknock.shtml https://web.archive.org/web/20041019045911/http://daugerresearch.com/pooch/parallelknock.html.	2006		98800DOC00048318
Dauger, D. E. (2006). Parallelization-Parallel Life. Dauger Research Vault. https://www.daugerresearch.org/vault/parallellife.shtml https://web.archive.org/web/20050404035143/http://daugerresearch.com/pooch/parallellife.html.	2006		98800DOC00048303
Dauger, D. E. (2006). Parallelization-Parallel Pascal's Triangle. Dauger Research Vault. https://www.daugerresearch.org/vault/parallelpascalstriangle.shtml https://web.archive.org/web/20041028061207/http://daugerresearch.com/pooch/parallelpascalstriangle.html.	2006		98800DOC00048292
Dauger, D. E. (2007). Plug-and-Play" Cluster Computing Brings HPC to the Mainstream. Dauger Research, Inc.	2007		98800DOC00046309

Description	Publication Date	Short Name	Starting Bates Number
Dauger, D. E. (2008, December). Supercomputing Engine for Mathematica. Machine Evaluation Workshop 19-2 Dec 2008. Runcorn, Daresbury, United Kingdom.	2008		98800DOC00044992
Dauger, D. E. (2012). What's New, SET and SET-Scilab Debut at Supercomputing 2012. Dauger Research Inc. https://daugerresearch.com/whatsnew.shtml	2012		98800DOC00044609
Dauger, D. E., & Decyk, V. K. (2001, October). Numerically-intensive" plug-and-play" parallel computing. In Proceedings 2001 IEEE International Conference on Cluster Computing (pp. 75-82). IEEE.	2001	Dauger 4	98800DOC00070130
Dauger, D. E., & Decyk, V. K. (2005 March/April). Plug-and-play cluster computing: High-performance computing for the mainstream. Computing in science & engineering, 7(2), 27-33.	2005	Dauger 1	98800DOC00052176
Dauger, D. E., & Decyk, V. K. (2005, May). "Plug-and-Play" Cluster Computing HPC Designed for the Mainstream Scientist. In International Conference on Computational Science (pp. 84-90). Berlin, Heidelberg: Springer Berlin Heidelberg.	2005	Dauger 2	98800DOC00052176
Dauger, D. E., & Decyk, V. K. (2006). Clúster de Cómputo "Enchufar y Listo""Plug-and-Play" Cluster Computing.	2006		98800DOC00048344

Description	Publication Date	Short Name	Starting Bates Number
Dauger, D. E., Decyk, V. K., & Dawson, J. M. (2005). Using semiclassical trajectories for the time-evolution of interacting quantum-mechanical systems. Journal of Computational Physics, 209(2), 559-581.	2005	Dauger 5	98800DOC00052063
Dauger, D. E., Decyk, V. K., & Tonge, J. W. (2006). Semiclassical Modeling of Multiparticle Quantum Systems Using High-Performance Clustering and Visualization on the Macintosh. Dauger Research, Inc.	2006	Dauger Dissertation	98800DOC00047995
Dauger, D. et al. (2005) Pooch Application-What is Pooch. Dauger Research Inc.	2005		98800DOC00052106
Dauger, D. et al., (2005). "Plug-and-Play" Parallel Computing: Build Your Own Parallel Computing Cluster in Under an Hour. Dauger Research Inc.	2005		98800DOC00052108
Dauger, D.E., & Decyk, V.K. (2003). "Plug-and-play" cluster computing using Mac OS X. 2003 Proceedings IEEE International Conference on Cluster Computing, 430-435.	2003	Dauger 3	98800DOC00061271
Dauger. (2006). Update to Pooch Software Enhances the Cluster Experience. Dauger Research, Inc.	2006		98800DOC00043920
Dauger. D. (2005). High-Performance and Scientific Computing and Visualization. Dauger Research Inc.	2005		98800DOC00052202

Description	Publication Date	Short Name	Starting Bates Number
David Ascher et al., "Numerical Python," No. UCRL-MA-128569, Lawrence Livermore National Lab. (1999), at https://w3.pppl.gov/~hammett/comp/python/LLNLDoc/numericalpython.pdf	1999	Numerical Python	98800DOC00076576
David Barkai, Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net (2001)		Barkai	*[see footnote 3]*
Declaration of Sylvia Hall-Ellis submitted in IPR2020-01608	2020	Hall-Ellis Declaration	98800DOC00044095
Declaration of Wolfgang Schreiner submitted in IPR2020-01608	2020	Schreiner Declaration	98800DOC00044011
Decyk et al., (2001). AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing.	2001		98800DOC00070099
Decyk et al., (2002). Appleseed-Personal Parallel Computing, Plug and Play Parallel Processing, as archived Jun. 12, 2005. https://web.archive.org/web/20050612083102/http://exodus.physics.ucla.edu:80/appleseed/appleseed.html	2002		98800DOC00065771
Decyk et al., "How to Build an AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing", Jan. 2001, pp. 11.	2001		98800DOC00072610
Decyk et al., "Two Dimensional Particle-in-Cell Code for Simulation of Plasma", Bulletin of the American Physical Society, Nov. 2002, vol. 47, No. 9, pp. 52 (2 pages total). NVIDIA-ACS-0166306.	2002		98800DOC00064853

Description	Publication Date	Short Name	Starting Bates Number
Decyk, V. K. & Dauger, D., (2000). How to build an AppleSeed: A parallel Macintosh cluster for numerically intensive computing. Physica Scripta, 84, 85.	2000	Decyk 2	98800DOC00072621
Decyk, V. K. (2005). Advanced PIC techniques: Parallelization and object-oriented programming. Proceedings of ISSS-7, 3.	2005		98800DOC00052207
Decyk, V. K. (n.d.). Simple C Multitasking Library for the Apple Macintosh Computer. Department of Physics and Astronomy UCLA.	n.d.		98800DOC00043914
Decyk, V. K. (n.d.). Simple Fortran Multitasking Library for the Apple Macintosh Computer. Department of Physics and Astronomy UCLA.	n.d.		98800DOC00043908
Decyk, V. K., & Norton, C. D. (2004). UCLA Parallel PIC Framework. Computer Physics Communications, 164(1–3), 80-85.	2004		98800DOC00055583
Decyk, V. K., Dauger, D. E., & Kokelaar, P. R. (1999). Appleseed: a parallel Macintosh cluster for numerically intensive computing. Computer Physics Communications, 121, 627-627.	1999		98800DOC00074705
Decyk, V. K., Dauger, D., & Kokelaar, P, How to Build an AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing (2000)	2000	Decyk 6	98800DOC00072610
Delivering Parallel Computing Power to the Desktop (2004). Interactive Supercomputing, presentation.	2004		98800DOC00054131

Description	Publication Date	Short Name	Starting Bates Number
Deng, Y. (2004). GSHELL: A command interpreter for a public computing utility (Master's thesis, McGill University). McGill University.	2004		98800DOC00054219
dist.Scheduler source code for Distributed Maple			*[see footnote 3]*
Distributed Maple	2004		98800DOC00111195
Distributed Maple Papers	2004		98800DOC00111192
Distributed Mathematica Source Code (2000), at https://www3.risc.jku.at/software/distmath/.	2000		98800DOC00070828
Dongarra et al., (1995). High Performance Computing: Technology, Methods and Applications. Netherlands: Elsevier Science.	1995	Dongarra	98800DOC00086123
Dongarra, J. (2006). Applied Parallel Computing: State of the Art in Scientific Computing.	2006		98800DOC00046714
Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., & White, A. (2003). Sourcebook of parallel computing (Vol. 3003). San Francisco: Morgan Kaufmann Publishers.	2003		98800DOC00060311
Dowd, K., & Severance, C. (2005). High performance computing (2nd ed.).	2005		98800DOC00051769
Edelman, A. (2004). SIAM Conference on Parallel Processing for Scientific Computing. San Francisco, CA, 3.	2004		98800DOC00054216

Description	Publication Date	Short Name	Starting Bates Number
Edelman, A., Husbands, P., & Leibman, S. (2006). Interactive Supercomputing's Star-P Platform (No. DOE/FE/25631-1). Massachusetts Institute of Technology, Cambridge, MA.	2006		98800DOC00046712
Edelman, A., McCorquodale, P., & Toledo, S. (1999). The future fast Fourier transform?. SIAM Journal on Scientific Computing, 20(3), 1094-1114.	1999	Edelman FFT	98800DOC00074684
EDN. (2003, October 15). SUN Debuts UltraSPARC IV.	2003	SPARC IV Article	98800DOC00060309
EETimes (Oct. 15, 2003) SUN Debuts UltraSPARC IV	2003	SPARC IV Article	98800DOC00060307
Elsevier. (1991). High performance servers. Elsevier Science. https://doi.org/10.1016/C2009-0-11764-6	1991		98800DOC00100426
Epema, D. H., Livny, M., van Dantzig, R., Evers, X., & Pruyne, J. (1996). A worldwide flock of condors: Load sharing among workstation clusters. Future Generation Computer Systems, 12(1), 53-65.	1996	Epema	98800DOC00085800
Erez et al., Analysis and Performance Results of a Molecular Modeling Application on Merrimac, SC 2004	2004	Merrimac Molecular Modeling	98800DOC00054197
Erez et al., Fault Tolerance Techniques for the Merrimac Streaming Supercomputer, SCI 05, November, 2005	2005	Merrimac Fault Tolerance	98800DOC00051471
Erez, M. (2007). Merrimac: high-performance and highly-efficient scientific computing with streams (Vol. 67, No. 11).	2007	Erez	98800DOC00046080

Description	Publication Date	Short Name	Starting Bates Number
Erez, M., Ahn, J. H., Jayasena, N., Knight, T. J., Das, A., Labonté, F., & Rosenblum, M. (2004, June). Merrimac-supercomputing with streams. In Proceedings of the 2004 SIGGRAPH GP2 Workshop on General Purpose Computing on Graphics Processors.	2004	Merrimac 2003	98800DOC00054207
Evers, J. (2005). Windows Server for Supercomputers Delayed. Network World, 2.	2005		98800DOC00051469
Exhibit 1134—Maple 5—PC Magazine (1992)	1992	PC Magazine1	98800DOC00096775
Exhibit 1135—Maple V Looks Better with Improved Graphics—PC Magazine (Jul. 1993)	1993	PC Magazine2	98800DOC00110605
Fagg, G. E., & Dongarra, J. J. (1996). PVMPI: An Integration of the PVM and MPI Systems. Calculateurs Paralleles, 2.	1996	Fagg	98800DOC00085786
Fan, Z., Qiu, F., Kaufman, A., & Yoakum-Stover, S. (2004, November). GPU cluster for high performance computing. In SC'04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing (pp. 47-47). IEEE.	2004	Fan 1	98800DOC00054185
FCIT. (2005). An Educator's Guide to School Networks. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051466

Description	Publication Date	Short Name	Starting Bates Number
FCIT. (2005). Chapter 1. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051463
FCIT. (2005). Chapter 2. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051459
FCIT. (2005). Chapter 3. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051453
FCIT. (2005). Chapter 4. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051447
FCIT. (2005). Chapter 5. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051441

Description	Publication Date	Short Name	Starting Bates Number
FCIT. (2005). Chapter 6. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051437
FCIT. (2005). Glossary. Florida Center for Instructional Technology. Retrieved January 31, 2025, from https://web.archive.org/web/20050527073004/http://fcit.usf.edu:80/Network/default.htm	2005		98800DOC00051431
Ferrari, A., Filipi-Martin, A., & Viswanathan, S. (1995). The NAS parallel benchmark kernels in MPL.	1995	MPL	98800DOC00086109
Ferron, J. (2003). Microsoft Compute Cluster Server Partner Training. Windows Compute Cluster Server, 2005, 161.	2003		98800DOC00060146
Fijalkow, E. (1999). A numerical solution to the Vlasov equation. Computer physics communications, 116(2-3), 319-328.	1999	Fijalkow 1	98800DOC00074672
Fijalkow, E. (1999). Numerical solution to the Vlasov equation: the 1D code. Computer physics communications, 116(2-3), 329-335.	1999	Fijalkow 2	98800DOC00074665
Fijalkow, E. (1999). Numerical solution to the Vlasov equation: the 2D code. Computer physics communications, 116(2-3), 336-344.	1999	Fijalkow 3	98800DOC00074656

Description	Publication Date	Short Name	Starting Bates Number
Fineberg, S. A., & Kutler, P. (1997). A Scalable Software Architecture Booting and Configuring Nodes in the Whitney Commodity Computing Testbed (No. NAS-97-024).	1997	Whitney Software	98800DOC00081863
Fineberg, S. A., & Pedretti, K. T. (1997, October). Analysis of 100 Mb/s Ethernet for the Whitney commodity computing testbed. In Proceedings. Frontiers' 99. Seventh Symposium on the Frontiers of Massively Parallel Computation (pp. 276-283). IEEE.	1999	Whitney Analysis	98800DOC00081839
Fosdick, L. D. (1996). An introduction to high-performance scientific computing. MIT Press.	1996	Fosdick	98800DOC00084982
Foster, I. T., & Worley, P. H. (1997). Parallel algorithms for the spectral transform method. SIAM Journal on Scientific Computing, 18(3), 806–837. https://doi.org/10.1137/S1064827594266891	1997		98800DOC00081807
Fried, I. (2005). Gates Touts Microsoft's Supercomputing Move. ZD Net, 3	2005		98800DOC00051426
Frigo et al., (2003). FFTW User's Manual. Version 2.1.5. pp. 74.	2003	FFTW Manual	98800DOC00059998
Frigo, M. et al., (1997). The Fastest Fourier Transform in the West, Technical Report, MIT.	1997	Frigo	98800DOC00081787
Frigo, M., & Johnson, S. G. (2003). FFTW user's manual. Massachusetts Institute of Technology.	2003	FFTW Manual	98800DOC00059998

Description	Publication Date	Short Name	Starting Bates Number
Geist, A. (1994). PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing. MIT press.	1994	Geist	98800DOC00086912
George, A. D., & Cook Jr, S. W. (1996). Distributed simulation of parallel DSP architectures on workstation clusters. Simulation, 67(2), 94-105.	1996	George	98800DOC00084970
Gilbert, J. R., Reinhardt, S., & Shah, V. B. (2006, June). High-performance graph algorithms from parallel sparse matrices. In International Workshop on Applied Parallel Computing (pp. 260-269). Berlin, Heidelberg: Springer Berlin Heidelberg.	2006	Gilbert	98800DOC00046674
Goedecker, S., & Hoisie, A. (2001). Performance optimization of numerically intensive codes. Society for Industrial and Applied Mathematics.	2001	Goedecker Book	98800DOC00069913
Goedecker, S., Boulet, M., & Deutsch, T. (2003). An efficient 3-dim FFT for plane wave electronic structure calculations on massively parallel machines composed of multiprocessor nodes. Computer Physics Communications, 154(2), 105-110.	2003	Goedecker	98800DOC00059992
Grigoras, D. (Ed.). (2001). Advanced Environments, Tools, and Applications for Cluster Computing: NATO Advanced Research Workshop, IWCC 2001, Mangalia, Romania, September 1-6, 2001. Revised Papers (Vol. 2326). Springer Science & Business Media.	2001		98800DOC00069908

Description	Publication Date	Short Name	Starting Bates Number
Gropp et al., "MPICH2 Installer's Guide", Version 0.4, Mathematics and Computer Science Division, Argonne National Laboratory, Sep. 9, 2004, pp. 21.	2004	MPICH2 Installer Guide	*[see footnote 3]*
Gropp et al., "MPICH2 User's Guide", Version 0.4, Mathematics and Computer Science Division, Argonne National Laboratory, Sep. 9, 2004, pp. 19.	2004	MPICH2 User Guide	98800DOC00054153
Gropp, W., & Lusk, E. (1996). Installation Guide to mpich, a Portable Implementation of MPI Version 1.2. 2. Technical repon. Argonne National Laboratory, University of Chicago, 2m.	1996		98800DOC00084903
Gropp, W., Lusk, E., & Skjellum, A. (1999). Using MPI: Portable parallel programming (2nd ed.). MIT Press.	1999	Gropp 1	98800DOC00073996
Gropp, W., Lusk, E., & Thakur, R. (1999). Using MPI-2: Advanced features of the message-passing interface. MIT Press.	1999	Gropp 2	98800DOC00074381
Gropp, W., Lusk, E., Doss, N., & Skjellum, A. (1996). A high-performance, portable implementation of the MPI message passing interface standard. Parallel computing, 22(6), 789-828.	1996	MPI Implementatio n	98800DOC00084698
Grueber et al., (2003). "Introduction" Microsoft Windows Server 2003 Deployment Kit, Automating and Customizing Installations.	2003		98800DOC00059984
H. M. Deitel et al., Operating Systems, 3d ed., Prentice Hall (2004)	2004	Deitel	98800DOC00054313

Description	Publication Date	Short Name	Starting Bates Number
Hamscher, V., Schwiegelshohn, U., Streit, A., & Yahyapour, R. (2000). Evaluation of job-scheduling strategies for grid computing. In Grid Computing—GRID 2000: First IEEE/ACM International Workshop Bangalore, India, December 17, 2000 Proceedings 1 (pp. 191-202). Springer Berlin Heidelberg.	2000		98800DOC00072598
Hanselman, D., Littlefield B., Mastering MatLab 5: A comprehensive Tutorial and Reference (1998).	1998		98800DOC00079704
Haynos, M. (2004). Perspectives on grid: Grid computing--next-generation distributed computing. IBM developerWorks, 27.	2004		98800DOC00054146
Heal et al., (1996). Maple V Learning Guide. Springer, Waterloo Maple Inc.	1996		98800DOC00084738
Hemsoth, N. (2011, November 16). Startup aims to bring parallel applications to the masses. HPC Wire.https://www.hpcwire.com/2011/11/16/startup_aims_to_bring_parallel_applications_to_the_masses/	2011	Hemsoth	98800DOC00044706
Hemsoth, Nicole, (2011). Startup Aims to Bring Parallel Applications to the Masses, HPC Wire.	2011		98800DOC00044702
Hennessy, J. L., & Patterson, D. A. (1990). Computer architecture: a quantitative approach. Elsevier.	1990		98800DOC00103288
Hennessy, J. L., & Patterson, D. A. (1996 2d ed.). Computer architecture: a quantitative approach. Elsevier.	1996		98800DOC00083679

Description	Publication Date	Short Name	Starting Bates Number
Hennessy, J. L., & Patterson, D. A. (1998). Computer Organization and Design: The Hardware/Software Interface	1998		98800DOC00078203
Hennessy, J. L., & Patterson, D. A. (2005 3 ed.). Computer Organization and Design: The Hardware/Software Interface	2005		98800DOC00049287
Hennessy, J. L., & Patterson, D. A. (2007 4th ed.). Computer architecture: a quantitative approach. Elsevier.	2007		98800DOC00045216
Hesham El-Rewini & Mostafa Abd-EI-Barr, Advanced Computer Architecture And Parallel Processing (2005)	2005	El-Rewini	98800DOC00051482
High Performance Computing Systems and Applications. (1998). United Kingdom: Kluwer Academic Publishers.	1998	Schaeffer	98800DOC00077571
High-Performance Computing with Mathematica: The PoochMPI Toolkit (2006). Wolfram Library Archive.	2006		98800DOC00046345
Hillis W. D. (1989). Richard Feynman and the Connection Machine. For Physics Today http://longnow.org/essays/richard-feynman-connection-machine/, printed Oct. 1, 2019, pp. 8.	1989		98800DOC00105717
Honan, M. (2006). WWDC: Dauger announces PoochMPI Toolkit for Mathematica. Macworld. https://www.macworld.com/article/181089/pooch-4.html	2006		98800DOC00046668

Description	Publication Date	Short Name	Starting Bates Number
HPC Wire 2011, Advanced Cluster Systems Unveils Software that Parallelizes Sequential Apps.	2011		98800DOC00044701
HPC Wire, VISUAL SUPERCOMPUTING TO BE DEMONSTRATED AT SC97 (1997). https://www.hpcwire.com/1997/11/14/visual-supercomputing-demonstrated-sc97/	1997		98800DOC00081771
HPC, SC1997 Flyer (1997).pdf	1997		98800DOC00081786
Huang, J.-Y., & Gau, C.-Y. (2002). A PC cluster high-fidelity mobile crane simulator. *Tamkang Journal of Science and Engineering, 5*(1), 7–20.	2002		98800DOC00064839
Husbands, P. (1999). Interactive Supercomputing. Thesis, MIT,. http://dspace.mit.edu/bitstream/handle/1721.1/79973/42430246-MIT.pdf;sequence=2	1999	Husbands Thesis	98800DOC00073900
Husbands, P. (2001). Interactive Supercomputing with MITMatlab.	2001	Husbands 1	98800DOC00069898
Huss-Lederman, S., Gropp, B., Skjellum, A., Lumsdaine, A., Saphir, B., & Squyres, J. (1997). MPI-2: Extensions to the message passing interface. University of Tennessee, available online at http://www. mpiforum. org/docs/docs. html. https://web.archive.org/web/20050403213517/http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html as archived Apr. 3, 2005 in 11 pages.	1997		98800DOC00081379

Description	Publication Date	Short Name	Starting Bates Number
IBM, IBM Parallel Environment for AIX: Operation and Use, Volume 1, Using the Parallel Operating Environment, October 1998	1998	POEref	98800DOC00078021
IBM, IBM Parallel Environment for AIX: Operation and Use, Volume 2, Using the Parallel Operating Environment, December 2001	2001		98800DOC00069712
IBM, Operation and Use, Volume 1 Using the Parallel Operating Environment (1998). Ver 2. Release 4	1998		98800DOC00079522
IBM, The RS/6000 SP Inside Out (1999).	1999		98800DOC00073328
IEEE. (n.d.). About IEEE Xplore. IEEE Xplore. Retrieved March 26, 2025, from https://ieeexplore.ieee.org/Xplorehelp/overview-of-ieee-xplore/about-ieee-xplore	2025		98800DOC00043998
Inmos 1987, Reference Manual, Transputer, IMS.	1987		98800DOC00106240
Inmos Technical Note 2	1988	Inmos Technical Note 2	98800DOC00105763
Inmos Technical Note 36	1988	Inmos Technical Note 36	98800DOC00105745
Install file for Distributed Maple	2003	Install1	98800DOC00111204
Install File for Distsoft Library	2000	Install2	98800DOC00111199
Intel (1996), Paragon System User's Guide.	1996	Paragon Guide	98800DOC00083217
Interactive Supercomputing (2003). Delivering Parallel Computing Power to the Desktop, Presentation	2003		98800DOC00059964

Description	Publication Date	Short Name	Starting Bates Number
Iowa State University. (2019). Using MATLAB Parallel Server. Iowa State University High Performance Computing. Retrieved January 31, 2025, from https://www.hpc.iastate.edu/guides/using-matlab-parallel-server	2019		98800DOC00044233
Jeremy Kepner, Parallel Programming with MatlabMPI, at https://archive.ll.mit.edu/mission/cybersec/softwaretools/matlabmpi/matlabmpi.html.	2025		98800DOC00052395
Jia, Q. (2004). A Programming System for Peer-to-peer Computing (Doctoral dissertation, University of New Brunswick, Faculty of Computer Science).	2004	Jia	98800DOC00054005
Johnson, S. G. (2004, May). FFTs, Portability, & Performance. MIT.	2004		98800DOC00053958
Johnston, S. J. (2005, November 16). Gates: Microsoft Ships beta 2 of HPC cluster version of Windows. Redmondmag. https://redmondmag.com/articles/2005/11/16/gates-microsoft-ships-beta-2-of-hpc-cluster-version-of-windows.aspx	2005		98800DOC00051422
Kahney, That's a Whole Lot of Power, Mac, Wired (Jan. 29, 2002), at https://www.wired.com/2002/01/thats-a-whole-lot-of-power-mac/	2002	Kahney	98800DOC00064833
Kane, G. (1988). mips RISC Architecture. Prentice-Hall, Inc.	1988		98800DOC00096782

Description	Publication Date	Short Name	Starting Bates Number
Kapasi, Ujval (2002).The Imagine Stream Processor. In Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02) (ICCD '02). IEEE Computer Society, USA, 282.	2002		98800DOC00064826
Karonis, N.T., Toonen, B.R., & Foster, I.T. (2002). MPICH-G2: A Grid-enabled implementation of the Message Passing Interface. ArXiv, cs.DC/0206040.	2002		98800DOC00064804
Kaufman et al., Scalable Scientific GPU Computing, DOE Proposal, March 12, 2004	2004	Kaufman 1	*[see footnote 3]*
Kepner et al., MatlabMPI, 64 J. Parallel & Distrib. Comput. 997–1005 (2004)	2004	Kepner 2	98800DOC00053949
Kepner et al., MatlabMPI, MIT Lincoln Laboratory (Jan. 3, 2003)	2003	Kepner 1	98800DOC00059926
Kepner, J. (2004). Parallel Matlab: The Next Generation. MIT Lincoln Laboratory, Lexington, MA.	2004	Kepner 3	98800DOC00053947
Kepner, J., & Ahalt, S. (2004). MatlabMPI. Journal of parallel and distributed computing, 64(8), 997-1005.	2004		98800DOC00053949
Khot, et al., (2004). A Parallel Data Mining Toolbox Using MatlabMPI. 44.	2004	Khot	98800DOC00053902
Kim, H., Mullen, J., & Kepner, J. (2011). Introduction to Parallel Programming and pMatlab v2.0. https://doi.org/10.21236/ADA549346	2011		98800DOC00044672
Kofler, M. (1997). Maple: an introduction and reference. Addison-Wesley Longman Publishing Co., Inc.	1997		98800DOC00080829

Description	Publication Date	Short Name	Starting Bates Number
Koontz, K. W. (1989). Embedded parallel architectures in real-time applications. Johns Hopkins APL Technical Digest;(USA), 10.	1989		98800DOC00105705
Korpela, E., Werthimer, D., Anderson, D., Cobb, J., & Leboisky, M. (2001). SETI@ home-massively distributed computing for SETI. Computing in science & engineering, 3(1), 78-83.	2001		98800DOC00069706
Krishnan, A. (2006). Windows Compute Cluster Server and deskside clusters. Windows Server HPC Microsoft Corporation.	2006		98800DOC00046490
Labonte, F., Mattson, P., Thies, W., Buck, I., Kozyrakis, C., & Horowitz, M. (2004, October). The stream virtual machine. In Proceedings. 13th International Conference on Parallel Architecture and Compilation Techniques, 2004. PACT 2004. (pp. 267-277). IEEE.	2004		98800DOC00053891
Lai, E. (2005). Microsoft Offers Beta 2 of Compute Cluster Server 2003. Computerworld, 4.https://www.computerworld.com/article/2559328/microsoft-offers-beta-2-of-compute-cluster-server-2003.html	2005		98800DOC00051414
Lantz, E. (2005, August). Introduction to Windows 2003 Compute Cluster Edition. In Datacenter Fabric Workshop Windows IB (p. 28).	2003	CCS Introduction	98800DOC00051386
Lefohn et al., “GPGPU”, Presentation, VIS 05, 2005, pp. 528.	2005		98800DOC00050858
Lenoski, D. E., & Weber, W. D. (2014). Scalable shared-memory multiprocessing. Elsevier.	2014		98800DOC00044269

Description	Publication Date	Short Name	Starting Bates Number
Lindholm, E., Nickolls, J., Oberman, S., & Montrym, J. (2008). NVIDIA Tesla: A unified graphics and computing architecture. IEEE micro, 28(2), 39-55.	2008		98800DOC00044965
Lisandro Dalcín et al., "MPI for Python," Journal of Parallel and Distributed Computing 65(9) at 1108-1115 (2005)	2005	MPI for Python	98800DOC00052216
Lusk, E. (2001). Programming with MPI on clusters (No. ANL/MCS/CP-105480). Argonne National Lab., IL (US), at https://www.osti.gov/servlets/purl/797872.	2001		98800DOC00069691
Mac OS X Server v10.2 Administrator Guide (2002)	2002		98800DOC00064182
Maeder, R. (2005). Mathematica Parallel Computing Toolkit: Unleash the Power of Parallel Computing. Wolfram Research, Jan.	2005		98800DOC00050758
Maple - An Intro & Reference	1997	Maple Reference	98800DOC00110647
Maple 5 - PC Magazine	1992	PC Magazine1	98800DOC00110616
Maple 7: Getting started guide. (2001). Waterloo Maple, Inc.	2001		98800DOC00066482
Maple 7: Learning Guide (2001). Waterloo Maple, Inc., 2001.	2001		98800DOC00066182
MAPLE V Learning Guide	1998	Maple Guide	98800DOC00111341
Martens, C. (2005). Platform Steps Up Partnership with Microsoft. InfoWorld, 3.https://www.infoworld.com/article/2673914/platform-steps-up-partnership-with-microsoft.html	2005		98800DOC00050755

Description	Publication Date	Short Name	Starting Bates Number
Math Supercomputer-In-A-BoxTM. Advanced Cluster System, LLC. (2007). https://paperzz.com/doc/8886625/product-flyer	2007		98800DOC00046339
Matlab - CDMA Reference Blockset (2004).			*[see footnote 3]*
Matlab - Communications Toolbox User Guide (2002).			*[see footnote 3]*
Matlab - Control System Toolbox User Guide (2005).			*[see footnote 3]*
Matlab - Data Acquisition Toolbox Quick Reference Guide (2004)	2004		98800DOC00053871
Matlab - Data Acquisition Toolbox User Guide (2004).			*[see footnote 3]*
Matlab - Dial and Gauges Blockset User Guide (2004).			*[see footnote 3]*
Matlab - Getting Started with Control System Toolbox (2002).			*[see footnote 3]*
Matlab - MAT-File Format User Guide (1999).			*[see footnote 3]*
Matlab - Neural Network Toolbox (2003).			*[see footnote 3]*
Matlab - Optimization Toolbox (2003).			*[see footnote 3]*
Matlab - Report Generator User Guide (2004).			*[see footnote 3]*
Matlab - Requirements Management Interface (2000).			*[see footnote 3]*
Matlab - Robust Control Toolbox User Guide (2004).			*[see footnote 3]*
Matlab - Signal Processing Blockset Users Guide (2004).			*[see footnote 3]*
Matlab - Signal Processing Toolbox User Guide (2004).			*[see footnote 3]*
Matlab - Spline Toolbox (2003).			*[see footnote 3]*

Description	Publication Date	Short Name	Starting Bates Number
Matlab - Symbolic Math Toolbox (2005).			*[see footnote 3]*
Matlab - Wavelet Toolbox (2003).			*[see footnote 3]*
Matlab - xPC Target I-O Reference Guide (2004).			*[see footnote 3]*
MATLAB: The Langauge of Technical Computing - External Interfaces (2003)	2003	MATLAB External Interfaces	98800DOC00110105
Matsumura, et al. (1998). Construction of Distributed Computing System for Large-Scale Matrices Intended or Reduction of Communication Blocks. vol. 98, No. 18, pp. 9.	1998		98800DOC00079264
Mattos, P. (1987). The Transputer Based Navigation System: An Example of Testing Embedded Systems. INMOS.	1987		98800DOC00106234
Mattson, T. G., Sanders, B., & Massingill, B. (2004). Patterns for parallel programming. Pearson Education.	2004	Mattson	98800DOC00049978
McIntosh-Smith, Simon (2006). Accelerating Mathematica: Vectors for All. ClearSpeed Technology. Wolfram Technology Conference.	2006		98800DOC00046463
Menon et al., MultiMATLAB: Integrating MATLAB With High-Performance Parallel Computing, Proceedings of the 1997 ACM/IEEE conference on Supercomputing, November 1997	1997	Menon	98800DOC00080811
Menon, Trefethen, RS6000, & POEref, Full Internet Archive	2025		98800DOC00043922

Description	Publication Date	Short Name	Starting Bates Number
Menon, V. and Trefethen, A.E. (1996) MultiMATLAB: MATLAB on Multiple Processors cited by, ACM Digital Library. Available at: https://dl.acm.org/doi/10.5555/866863#cited-by-sec	1996		98800DOC00083210
Merrimac - Stanford Streaming Supercomputer Project (2004), at https://web.archive.org/web/20051226203012/http://merrimac.stanford.edu/.	2004	Merrimac Website	98800DOC00052403
Message Passing Interface Forum. (2003). MPI: A Message-Passing Interface Standard (Version 1.1). University of Tennessee, Knoxville.	11/15/2003	MPI Standard 2003	98800DOC00058449
Message Passing Interface Forum. (May 5th, 1994). MPI: A Message-Passing Interface Standard (Version 1.0). University of Tennessee, Knoxville.	1994	MPI Standard 1994	98800DOC00086676
Michael Woodacre et al., The SGI Altix 3000 Global Shared-Memory Architecture, SGI White Paper (2003), at https://www.cs.umd.edu/class/spring2021/cmsc714/readings/Woodacre-Altix-3000.pdf	2003	Woodacre	98800DOC00056034
Microsoft (2001). Parallel Programming Software Microsoft Windows 2000.	2001		98800DOC00068375

Description	Publication Date	Short Name	Starting Bates Number
Microsoft (2003). Accelerating Scientific Discovery with Microsoft Windows Compute Cluster Server 2003. http://download.microsoft.com/download/8/6/1/86149563-03ec-427e-80f1 d683bd715b2b/Bio%20Team%20and%20Life%20Sciences%20Solution%20Brief.pdf	2003		98800DOC00059131
Microsoft (2003). MPICH for Microsoft Windows Download Page, https://web.archive.org/web/20050309131254/http://www-unix.mcs.anl.gov/mpi/mpich/mpich-nt/ as archived Mar. 9, 2005 in 2 pages.	2003		98800DOC00059049
Microsoft (2005). Bill Gates Shares Microsoft's Vision for Technical Computing. https://news.microsoft.com/2005/11/15/bill-gates-shares-microsofts-vision-for-technical-computing/	2005		98800DOC00049974
Microsoft (2005). Windows Server 2003 Deployment Kit, Automating and Customizing Installations.	2003		98800DOC00049963
Microsoft (2006). Deploying and Managing Microsoft Windows Compute Cluster Server 2003.	2006		98800DOC00046449
Microsoft (2007). Increasing the Value of Simulation While Shrinking the Costs. Microsoft Windows Compute Cluster Server 2003 Partner Solution Brief.	2007		98800DOC00045208
Microsoft (2007). On-Demand Supercomputing Multiplies the Possibilities. Microsoft Windows Compute Cluster Server 2003 Partner Solution Brief.	2007	CCS gridMathematica	98800DOC00045201

Description	Publication Date	Short Name	Starting Bates Number
Microsoft (2007). Technical Computing Tools Combined with Cluster Computing Deliver High-Performance Solutions. Microsoft Windows Compute Cluster Server 2003 Partner Solution Brief.	2007	CCS Matlab	98800DOC00045195
Microsoft Chairman and Chief Software Architect Bill Gates to Deliver Keynote at SC|05 Conference. SC|05 Gateway to Discovery. (2005, July). https://www.sc05.supercomputing.org/news/press_releases_07252005.php.html	2005		98800DOC00048719
Microsoft Corporation. (2003) Chapter 2-Designing Unattended Installations. Microsoft Windows Server 2003 Deployment Kit, Automating and Customizing Installations, A Resource Kit Publication https://download.microsoft.com/download/e/2/b/e2bfb017-8525-4991-bbd5-7d7081f3d228/04_CHAPTER_2_Designing_Unattended_Installations.doc	2003		98800DOC00059141
Microsoft, (2006). Microsoft Releases Windows Compute Cluster Server 2003, Bringing High-Performance Computing to the Mainstream. Microsoft.com.	2003		98800DOC00046445
Microsoft, "Cluster Components and Interconnects", Microsoft Windows 2000, Jan. 5, 2001, pp. 3.	2000		98800DOC00072591
Microsoft, Getting Started with Compute Cluster Server 2003 (June 6, 2006)	2006	CCS Getting Started	98800DOC00046462

Description	Publication Date	Short Name	Starting Bates Number
Microsoft. (2003). Compute cluster job aids. In Windows Server 2003 Deployment Kit (p. 80). Microsoft.	2003		98800DOC00059051
Microsoft. (2005, December 15). Compute Cluster Server: Glossary (p. 6). Microsoft.	2005		98800DOC00049968
Microsoft. (2006). Compute Cluster Server: Getting started (p. 1). Microsoft.	2006	CCS Getting Started	98800DOC00046459
Microsoft. (2006, June 6). Compute Cluster Server setup command-line interface reference (p. 2). Microsoft.	2006		98800DOC00046460
Microsoft. (2007). Compute Cluster Server: Technical reference (p. 2). Microsoft.	2007		98800DOC00045214
Mirman, I. (2006). Going parallel the new way. Desktop Engineering, 11(10), 22.	2006	Mirman	98800DOC00046437
MIT, MPI Demo, at https://courses.csail.mit.edu/18.337/2004/demos.pdf.	2004		98800DOC00043818
MIT. (2003). MATLAB*P Instructions. https://courses.csail.mit.edu/18.337/2005/matlabp.html	2003		98800DOC00058687
Moore, C. (2000). The AppleSeed Project: Clustered Power Macs Outperform Cray Supercompter. LowEndMac. com, 3.	4/19/2000	Moore	98800DOC00072588
Moore, C. (2005). Apple Ends DIY Repairs, Does Mac mini Kill the Used Mac Market?, USB Drive Kit Matches Mac mini, and More. https://lowendmac.com/newsrev/05/0513.html	2005		98800DOC00049950

Description	Publication Date	Short Name	Starting Bates Number
Morrow, G., & van de Geijn, R. (1998, November). A parallel linear algebra server for MATLAB-like environments. In SC'98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing (pp. 13-13). IEEE.	1998	Morrow	98800DOC00079251
MPI (1994) MPI: A Message-Passing Interface Standard cited by, ACM Digital Library. Available at: https://dl.acm.org/doi/10.5555/898758#cited-by-sec	1994		98800DOC00086657
MPI: A Message-Passing Interface Standard (June 12, 1995)	6/12/1995	MPI Standard 1995	98800DOC00085870
NAS Benchmarks source code, at https://www.nas.nasa.gov/software/npb.html	1997		98800DOC00080805
Nayak, A. (1999). Design and Characterization of MATLAB Functions on a Distributed and Heterogeneous System. Center for Parallel and Distributed Computing, Technical Report No. CPDC-TR-9912-021, 98.	1999		98800DOC00072784
Nayak, A., Haldar, M., Kanhere, A., Joisha, P., Shenoy, N., Choudhary, A., & Banerjee, P. (2000, August). A library based compiler to execute MATLAB programs on a heterogeneous platform. In Proc. of Conf. on Parallel and Distributed Computing Systems.	2000	Nayak	98800DOC00072582
Ning Kang & Leroy A. Drummond, “A First Prototype of PyACTS,” No. LBNL-53849, Lawrence Livermore National Laboratories (2003), at https://www.osti.gov/biblio/816793	2003	PyACTS	98800DOC00059938

Description	Publication Date	Short Name	Starting Bates Number
NVIDIA Corp. v. Advanced Cluster Systems, IPR2021-00019-2009	2021	SEM Poster	98800DOC00043907
NVIDIA Corp. v. Advanced Cluster Systems, IPR2021-00019-2010	2021	SEM Benchmark	98800DOC00043893
NVIDIA Corp. v. Advanced Cluster Systems, IPR2021-00019-2011	2021	SEM Manual	98800DOC00043873
NVIDIA Corp. v. Advanced Cluster Systems, IPR2021-00019-2012	2021	SEM White Paper	98800DOC00043853
NVIDIA Corp. v. Advanced Cluster Systems, IPR2021-00019-2023	2021	SEM Report	98800DOC00043851
NVIDIA Corp. v. Advanced Cluster Systems, IPR2021-00019-2028	2021	SEM Claim Chart 1	98800DOC00043894
NVIDIA Corp. v. Advanced Cluster Systems, IPR2021-00020-2031	2021	SEM Claim Chart 2	98800DOC00043822
Nvidia, C. U. D. A. (2007). Nvidia cuda compute unified device architecture programming guide. NVIDIA corporation.	2007		98800DOC00045070
Other MacMPI software from Project Appleseed. Appleseed. (2001, April 17). https://web.archive.org/web/20050301180634/http://exodus.physics.ucla.edu:80/appleseed/other.html	2001		98800DOC00070637
Othmer, C., & Schüle, J. (2002). Dynamic load balancing of plasma particle-in-cell simulations: The taskfarm alternative. Computer physics communications, 147(1-2), 741-744.	2002	Othmer	98800DOC00062237

Description	Publication Date	Short Name	Starting Bates Number
Pacheco, P. S. (1998). A User's Guide to MPI. University of San Francisco, 56. https://arcb.csc.ncsu.edu/~mueller/cluster/mpi.guide.pdf	1998	Pacheco	98800DOC00079205
Parallel versions of CASA Functions	2000	CASA Code	98800DOC00111208
Parallel versions of Maple Functions	2000	Parallel Maple Code	98800DOC00111255
Park, S., Bajaj, C., & Ihm, I. (2004). Visualization of very large oceanography time-varying volume datasets. In Computational Science-ICCS 2004: 4th International Conference, Kraków, Poland, June 6-9, 2004, Proceedings, Part II 4 (pp. 419-426). Springer Berlin Heidelberg.	2004		98800DOC00055973
Partial MPI Library based on the program-to-Program Communications ToolBox in the Macintosh OS, Nov. 8, 2000, pp. 65.	2000		98800DOC00070759
Patrick Miller, "Parallel, Distributed Scripting with Python," No. UCRL-JC-148522, Lawrence Livermore National Laboratories(2002), at https://www.osti.gov/servlets/purl/15013331	2002	Parallel Scripting	98800DOC00062255
Patrick Miller, "pyMPI—An Introduction to Parallel Python using MPI," Lawrence Livermore National Laboratories (2002), at https://hpc.llnl.gov/sites/default/files/pyMPI.pdf	2002	pyMPI Introduction	98800DOC00062241

Description	Publication Date	Short Name	Starting Bates Number
Pau, C., & Schreiner, W. (2000). Distributed Mathematica-User and Reference Manual. RISC Report 00–25, RISC, JK University, Linz, Australia.	2000	Schreiner7	98800DOC00072542
PC Magazine2 - Maple V Looks Better with Improved Graphics	1993	PC Magazine2	98800DOC00110605
PCI Express Base Specification, Revision 1.1 (2005). PCI Express	2005		98800DOC00048779
Pelikán, J. (2003). Interconnect Efficiency of Tyan PSC T-630 with Microsoft Compute Cluster Server 2003.	2003		98800DOC00058436
Petcu, D. (2000, August). PVMaple: A distributed approach to cooperative work of Maple processes. In European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting (pp. 216-224). Berlin, Heidelberg: Springer Berlin Heidelberg.	2000	Petcu2	98800DOC00072533
Petcu, D. (2001). Solving initial value problems with parallel maple processes. In Euro-Par 2001 Parallel Processing: 7th International Euro-Par Conference Manchester, UK, August 28–31, 2001 Proceedings 7 (pp. 926-934). Springer Berlin Heidelberg.	2001	Petcu4	98800DOC00068366
Petcu, D. (2001). Working with multiple Maple kernels connected by Distributed Maple or PVMaple. Preprint RISC, 18-01.	2001	Petcu5	98800DOC00068348
PETCU, D., & GHEORGHIU, D. (2003). PAVIS: a parallel virtual environment for solving large mathematical problems. In Parallel Computing: Advances and Current Issues (pp. 490-497).	2003	Petcu3	98800DOC00058441

Description	Publication Date	Short Name	Starting Bates Number
Petcu, D., Dubu, D., & Paprzycki, M. (2004, July). Extending Maple to the Grid: design and implementation. In Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (pp. 209-216). IEEE.	2004		98800DOC00052387
Petcu, D., Paprzycki, M., & Dubu, D. (2005). Design and implementation of a grid extension for Maple. Scientific Programming, 13(2), 137-149.	2005		98800DOC00048766
Petcu, D., Paprzycki, M., & Ganzha, M. (2005, September). Clustering multiple and cooperative instances of computational intensive software tools. In International Conference on Parallel Computing Technologies (pp. 452-456). Berlin, Heidelberg: Springer Berlin Heidelberg.	2005	Petcu 6	98800DOC00048760
Petcu, D., Popa, H., & Ţepeneu, D. (2001, September). A user-level interface for clustering mathematical software kernels. In NATO Advanced Research Workshop on Cluster Computing (pp. 172-178). Berlin, Heidelberg: Springer Berlin Heidelberg.	2001	Petcu1	98800DOC00068359

Description	Publication Date	Short Name	Starting Bates Number
Petcu, D., Tepeneu, D., Paprzycki, M., Mizutani, T., & Ida, T. (2005, March). Survey of symbolic computations on the Grid. In Proc. of the 3rd Int. Conference Sciences of Electronic, Technologies of Information and Telecommunications SETIT 2005 (pp. 27-31).	2005		98800DOC00048749
Pfister, G. F. (1995). In search of clusters. Prentice Hall PTR.	1995	Pfister 1	98800DOC00109052
Pfister, G. F. (1998). In search of clusters (2nd ed.). Prentice-Hall.	1998	Pfister 2	98800DOC00109494
Pingpong MPI Benchmark—SEM vs 'grid (2009). Dauger Research, Inc. https://www.daugerresearch.org/vault/sc09semposter2.pdf	2009		98800DOC00044718
Pooch - Other Links (Dec. 14, 2004).			*[see footnote 3]*
Pooch Application (Apr. 2001)		2001 Pooch App	*[see footnote 3]*
Pooch Pro 1.6 Download (2005). https://web.archive.org/web/20051228104911/http://www.daugerresearch.com/pooch/download.shtml.	2005		98800DOC00052098
Pooch Pro v1.6 Release (2005). https://web.archive.org/web/20060217183350/http://daugerresearch.com/pr/poochv1.6.shtml.	2005		98800DOC00052096
Pountain, D., & May, D. (1987). A tutorial introduction to OCCAM programming. McGraw-Hill, Inc..	1987		98800DOC00106120
Power Book G4 Developer Note (2002)	2002		98800DOC00066003
Power Mac G4 Developer Note (2000)	2000		98800DOC00072673

Description	Publication Date	Short Name	Starting Bates Number
Power Mac G4 Developer Notes (2001)	2001		98800DOC00070657
PowerBookG3Series (2001).			*[see footnote 3]*
PowerMac G3 (2001).			*[see footnote 3]*
PowerMacintosh G3 (2001).			*[see footnote 3]*
PVMaple publicly available source code			*[see footnote 3]*
Quan Ming Lu, Dong Sheng Cai, Implementation of parallel plasma particle-in-cell codes on PC cluster, 135 COMPUTER PHYSICS COMMUNICATIONS 93–104 (2001)	2001	Lu	98800DOC00069694
Quinn, M. J., Miller, R., Miller, R., & Quinn, M. J. (2003). Parallel processing. In Encyclopedia of Computer Science (pp. 1349-1365).	2003		98800DOC00058419
Quinn, Michael J. (2003). Parallel Programming in C with MPI and OpenMP, McGraw Hill.	2003	Quinn Book	98800DOC00057903
Rajkumar Buyya, High Performance Cluster Computing: Programming and Applications, Volume 2 (1999)	1999	Buyya	98800DOC00075812
Ratowsky, R., Afeyan, B., & Feit, M. (1999). Propagation modeling for multimode photonics. Proceedings of SPIE - The International Society for Optical Engineering, 3625, 10.1117/12.356869.	1999		98800DOC00072775
Read Me File for Distsoft Library	2000	Readme2	98800DOC00111196
ReadMe file for Distributed Maple	2003	Readme1	98800DOC00110611
Reuters (2005). Microsoft Aims to Enter Supercomputer Market.	2005		98800DOC00048743

Description	Publication Date	Short Name	Starting Bates Number
RS 6000 SP BOOKS: IBM Parallel Environment for AIX (PE) (1998). https://web.archive.org/web/19990203132150/http://www.rs6000.ibm.com/resource/aix_resource/sp_books/pe/index.html	1998		98800DOC00079520
RS/6000 SP: Planning vol. 1, hardware and physical environment (2001). RS/6000 SP: Planning Vol. 1, Hardware and Physical Environment - 375 MHz POWER3 SMP High Node (F/C 2058). (n.d.). https://web.archive.org/web/20010726183314/http://www.rs6000.ibm.com/resource/aix_resource/sp_books/hardware/planguide1/da709mst08.html#HDRHIGH-PW3-2	2001	RS6000-2	98800DOC00068126
RS/6000 SP: Planning vol. 1, hardware and physical environment (2001). RS/6000 SP: Planning Vol. 1, Hardware and Physical Environment - Introducing the RS/6000 SP https://web.archive.org/web/20010726184202/http:/www.rs6000.ibm.com/resource/aix_resource/sp_books/hardware/planguide1/da709mst07.html	2001	RS6000	98800DOC00068118
RS/6000 SP: Planning vol. 1, hardware and physical environment (2002) 12th Edition.	2002	RS6000 2002	98800DOC00061939
Russel, C. (2005). Overview of Microsoft Windows Compute Cluster Server 2003. Microsoft Windows.	2003		98800DOC00048731
Sameh, A. (2005, February). The Schlumberger High Performance Cluster at AUC. In Proceedings of the 13th International Conference on Artificial Intelligence Applications, Cairo (p. 9).	2005		98800DOC00048722

Description	Publication Date	Short Name	Starting Bates Number
SC05 (2005), Exhibitor Freight Move in Schedule, Level 4, Exhibit Halls.	2005		98800DOC00048721
Schreiner et al., (2003). Distributed Maple: Lessons Learned on Parallel Computer Algebra in Distributed Environments, Presentation.	2003	Schreiner6	98800DOC00057856
Schreiner, W. (2000). Manager-Worker Parallelism versus Dataflow in Distributed Maple. Research Institute for Symbolic Computation (RISC-Linz), Johannes Kepler University, A-4040 Linz, Austria, 23.	2000	Schreiner4	98800DOC00072510
Schreiner, W. (2001, August). Manager-worker parallelism versus dataflow in a distributed computer algebra system. In International Conference on Parallel Computing Technologies (pp. 329-343). Berlin, Heidelberg: Springer Berlin Heidelberg.	2001	Schreiner5	98800DOC00068066
Schreiner, W. (2003). "Distributed Maple Papers", https://web.archive.org/web/20040304035455/http://www.risc.uni-linz.ac.at/software/distmaple/index_1.html as archived Mar. 4, 2004 in 3 pages.	2003		98800DOC00057852
Schreiner, W. (2003). Distributed Mapel. https://web.archive.org/web/20040216100904/http:/www.risc.uni-linz.ac.at/software/distmaple/ as archived Feb. 16, 2004 in 1 page.	2003		98800DOC00057855

Description	Publication Date	Short Name	Starting Bates Number
Schreiner, W., Mittermaier, C., & Bosa, K. (2003). Distributed Maple: parallel computer algebra in networked environments. Journal of Symbolic Computation, 35(3), 305-347.	2003	Schreiner1	98800DOC00057809
Schreiner, Wolfgang., (2001). Distributed Maple—User and Reference Manual (V 1.1.12), Research Institute for Symbolic Computation (RISC-Linz), Johannes Kepler University, A-4040 Linz, Austria.	2001	Schreiner2	98800DOC00068081
Schreiner2 - Distributed Maple 2001	2001	Schreiner2	98800DOC00111641
Schreiner3 - Task Logging 2002	2002	Schreiner3	98800DOC00110623
Schriener, Wolfgang (2001). README File, Distributed Maple Application Software.	2001	Readme2	98800DOC00068063
Schriener, Wolfgang (2003). README File. Distributed Maple 1.1.15.	2003	Readme1	98800DOC00057804
Seidel, Brent, "Sample Software", May 24, 2000, at https://web.archive.org/web/20050220000225/http://www.serv.net/~bseidel/AppleSeed/Software.html.	2000	Seidel	98800DOC00072509
SGI Altix 3000 User's Guide (2003), at https://www.infania.net/misc1/sgi_techpubs/techpubs/007-4579-001.pdf	2003	SGI Altix 3000 User Guide	98800DOC00057544
SGI NUMAlink, White Paper (2005), at https://www.graphica.com.au/files/numalink.pdf?ref=just.graphica.com.au	2005	SGI Numalink White Paper	98800DOC00048706

Description	Publication Date	Short Name	Starting Bates Number
Shah, V., & Gilbert, J. R. (2004, December). Sparse matrices in Matlab* P: design and implementation. In Proceedings of the 11th international conference on High Performance Computing (pp. 144-155).	2004	Shah	98800DOC00052375
Sherwani, J., Ali, N., Lotia, N., Hayat, Z., & Buyya, R. (2004). Libra: a computational economy-based job scheduling system for clusters. Software: Practice and Experience, 34(6), 573-590.	2004		98800DOC00052357
Silberschatz, A., Galvin, P. B., & Gagne, G. (2000). Applied operating system concepts. John Wiley & Sons, Inc..	2000	Silberschatz	98800DOC00070829
Smyth, E. S., Parker, J. S., & Taylor, K. T. (1998). Numerical integration of the time-dependent Schrödinger equation for laser-driven helium. Computer physics communications, 114(1-3), 1-14.	1998	Smyth	98800DOC00077557
Snir, M., Otto, S., Huss-Lederman, S., Walker, D., & Dongarra, J. (1996). MPI: The complete reference (1st ed.). MIT Press.	1996		98800DOC00082860
Snir, M., Otto, S., Huss-Lederman, S., Walker, D., & Dongarra, J. (1998). MPI: The complete reference (Vol. 1, 2nd ed.). MIT Press.	1998		98800DOC00077207
Snir, M., Otto, S., Huss-Lederman, S., Walker, D., & Dongarra, J. (1998). MPI: The complete reference (Vol. 2, 2nd ed.). MIT Press.	1998		98800DOC00076798
Sonneveld, N. (2007). Star-P Client Installation. Jan, 18, 30.	2007		98800DOC00045040

Description	Publication Date	Short Name	Starting Bates Number
Sonneveld, N. (2007). Star-P/MATLAB Installation Notes. Jan, 18, 8.	2007		98800DOC00045032
Source Code (2001) CASA NeighbGraph", Distributed Maple Application Software from the distsoft directory, v. 1.6.	2001	CASA Code	98800DOC00070606
Source Code (2001). Berlekamp Distinct Degree Factorization. Distributed Maple Application Software from the "distsoft" directory, 1, 58.	2001	Parallel Maple Code	98800DOC00068005
Source Code, "matlabpv2" 2003, pp. 4931. [NVIDIA-ACS-0233752 - NVIDIA-ACS-0234467]	2003		*[see footnote 3]*
Source Code, "Maple Interface to Java Scheduler", Distributed Maple, dist.maple5, v 1.63, Mar. 28, 2003.	2003	Dist.Maple Code	98800DOC00111313
SourceForge.net, MPI Python Files: Old Files (2025), at https://sourceforge.net/projects/pympi/files/OldFiles/.	2025		98800DOC00049949
Star-P® User Guide. (2006). Interactive Supercomputing,	2006	Star-P User Guide	98800DOC00046520
Sterling, T. L. (2002). Beowulf Cluster Computing With Windows (Scientific and Engineering Computation Series). MIT Press.	2002	Sterling	98800DOC00061406
Stone, H. S. (1993). High-performance computer architecture (3rd ed.). Addison-Wesley.	1993	Stone	98800DOC00092695
Sunderam, V. S., Geist, G. A., Dongarra, J., & Manchek, R. (1994). The PVM concurrent computing system: Evolution, experiences, and trends. Parallel computing, 20(4), 531-545.	1994		98800DOC00086641

Description	Publication Date	Short Name	Starting Bates Number
Swarztrauber, P. N. (1987). Multiprocessor ffts. Parallel computing, 5(1-2), 197-210.	1987	Swarztrauber	98800DOC00106101
Tanenbaum, A., Modern Operating Systems (2nd Edition) (2001). Prentice Hall	2001		98800DOC00066864
Tatebe, O., Kodama, Y., Sekiguchi, S., & Yamaguchi, Y. (1999). Efficient implementation of MPI using remote memory write in Japanese. In Proc. of Joint Symp. on Parallel Processing (Vol. 98, p. 199).	1999		98800DOC00072765
Technical Program - Technical Papers. SC|05 Gateway to Discovery. (2005, November 12-18). https://www.cresco.enea.it/SC05/programs/technical_papers.html	2005		98800DOC00048714
Technical Program - Tutorials. SC|05 Gateway to Discovery. (2005, November 12-18). https://www.cresco.enea.it/SC05/programs/tutorials.html	2005		98800DOC00048714
Tepeneu, D., & Ida, T. (2003). MathGridLink-A bridge between Mathematica and" the Grid".	2003		98800DOC00057540
Tezduyar, T., Aliabadi, S., Behr, M., Johnson, A., Kalro, V., & Litke, M. (1996). Flow simulation and high performance computing. Computational Mechanics, 18, 397-412.	1996		98800DOC00082844
Thakur, R., Rabenseifner, R., & Gropp, W. (2005). Optimization of collective communication operations in MPICH. The International Journal of High Performance Computing Applications, 19(1), 49-66.	2005		98800DOC00048689

Description	Publication Date	Short Name	Starting Bates Number
The MathWorks, Inc. (1998). Model Control Toolbox.	1998		98800DOC00079270
The Mathworks, Inc. (1999). MATLAB Compiler User's Guide (Version 2.0). The MathWorks, Inc.	1999		98800DOC00072882
The MathWorks, Inc. (2001). External Interface-API V6.	2001		98800DOC00069283
The MathWorks, Inc. (2001). Using Matlab.	2001		98800DOC00068379
The MathWorks, Inc. (2002). Getting Started with MATLAB.	2002		98800DOC00064044
The MathWorks, Inc. (2002). Nonlinear Control Design Blockset User Guide.	2002		98800DOC00063936
The MathWorks, Inc. (2002). Partial Differential Equation Toolbox User Guide.	2002		98800DOC00063619
The MathWorks, Inc. (2002). Real-Time Windows Target User Guide. The MathWorks, Inc.	2002		98800DOC00063441
The MathWorks, Inc. (2002). Using MATLAB	2002		98800DOC00062261
The MathWorks, Inc. (2003). xPC Target API User’s and Reference Guide (Version 2). The MathWorks, Inc.	2003		98800DOC00059554
The MathWorks, Inc. (2004). MAT-File Format User Guide.	2004		98800DOC00053826
The MathWorks, Inc. (2004). Signal Processing Toolbox User Guide. The MathWorks, Inc.	2004		98800DOC00052811
The MathWorks, Inc. (2004). System Identification Toolbox	2004		98800DOC00052404
The MathWorks, Inc. (2005). Using MATLAB Graphics V7.	2005		98800DOC00050039

Description	Publication Date	Short Name	Starting Bates Number
The Message Passing Interface (MPI) Standard (2005). Argonne National Libary. https://web.archive.org/web/20050403085859/http://www-unix.mcs.anl.gov.mpl/	2005		98800DOC00048687
Thiruvathukal, G. K. (2005). Guest editors' introduction: Cluster computing. Computing in Science and Engineering, 7(2), 11-13.	2005		98800DOC00048684
Trefethen, A. E., Menon, V. S., Chang, C. C., Czajkowski, G. J., Myers, C., & Trefethen, L. N. (1996). MultiMATLAB: MATLAB on multiple processors. Cornell University.	1996	Trefethen	98800DOC00082828
University of Tennessee. (2003, June 19). SP2 Guide. Internet Archive. Retrieved March 26, 2025, from https://web.archive.org/web/20030619070712/http:/www-jics.cs.utk.edu:80/SP2/SP2_guide/SP2_guide.html	2003		98800DOC00059948
V. K. Decyk & D. E. Dauger, Supercomputing for the Masses: A Parallel Macintosh Cluster, Parallel Processing and Applied Mathematics, Springer (2002), pp. 10-22	2002	Decyk 3	98800DOC00065755
V. K. Decyk, D. E. Dauger, & Pieter R. Kokelaar, Appleseed: A Parallel Macintosh Cluster for Numerically Intensive Computing, COMPUTER PHYSICS COMMUNICATIONS, 121: 627 (1999)	1999	Decyk 5	98800DOC00074705

Description	Publication Date	Short Name	Starting Bates Number
V. K. Decyk, Skeleton PIC code for parallel computers, Computer Physics Communications 87(1-2) (1995), pp. 87-94	1995	Decyk 4	98800DOC00086560
V.K. Decyk & D.E. Dauger, AppleSeed: A Parallel Macintosh Cluster for Scientific Computing, J. Plasma Fusion Rsch., Vol. 79, No. 8, pp. 722–79 (Aug. 2003), at https://www.jstage.jst.go.jp/article/jspf/79/8/79_8_772/_pdf/-char/ja	2003	Decyk 1	98800DOC00061179
Vlasov Textbooks (pre-2005).			*[see footnote 3]*
Voges, K. E., & Pope, N. K. L. (2001, September). Computational Marketing using an AppleSeed cluster. In Apple University Consortium Academic and Developers Conference (p. 15).	2001		98800DOC00066614
White, S., Alund, A., & Sunderam, V. S. (1995). Performance of the NAS parallel benchmarks on PVM-Based networks. Journal of Parallel and Distributed Computing, 26(1), 61-71.	1995	PVM	98800DOC00085843
Windows in Financial Services (2004). Microsoft's Leadership in Distributed Enterprise Computing Builds Case for HPC	2004		98800DOC00052355
Wired (2002). Slideshow: That's a Whole Lot of Power, Mac. Wired.com. pp. 10.	2002		98800DOC00061402
Wolfgana Schreiner, Distributed Maple website (2010), at https://www3.risc.jku.at/software/distmaple/.	2010		98800DOC00044710

Description	Publication Date	Short Name	Starting Bates Number
Wolfgang Schreiner et al., Distributed Maple: Parallel Computer Algebra in Networked Environments, 35 Journal of Symbolic Computation 305 (2003)	2003	Schreiner1	98800DOC00111678
Wolfram Library Archive (2006). 2006 Wolfram Technology Conference Proceedings.	2021		98800DOC00046340
Wolfram Library Archive, "Microsoft Windows Compute Cluster Server 2003 Partner Solution Brief", library.wolfram.com/infocenter/TechNotes/6723, pp. 1.	2003	CCS gridMathematica	98800DOC00057533
Wolfram Research, Inc. (2003). Wolfram Research Offers Support for Microsoft Windows Compute Cluster Server 2003", Jun. 9, 2006, pp. 1.	2003		98800DOC00056045
Wolfram Research, Inc. (2006). gridMathematica 2: Grid Mathematica Extras. https://web.archive.org/web/20061109144357/http://www.wolfram.com/products/gridmathematica/extras.html	2006		98800DOC00046436
Wolfram, S. (2003). The Mathematica Book. 5th Edition Wolfram Media.	2003		98800DOC00056047
Z. Fan, F. Qiu, and A. Kaufman. ZippyGPU: Programming Toolkit for General-Purpose Computation on GPU Clusters. In GPGPU Workshop at Supercomputing, 2006.	2006	Fan 6	*[see footnote 3]*

Description	Publication Date	Short Name	Starting Bates Number
Z. Fan, F.Qiu, A.Kaufman, and S.Yoakum-Stover. GPU Cluster for Scientific Computing and Large-Scale Simulation, Conference Poster, ACM Workshop on General-Purpose Computing on Graphics Processors, pages C–32, 2004.	2004	Fan 3	*[see footnote 3]*
Zhang, D. H., Zhang, J. B., Zhao, Y. Z., & Wong, M. M. (2005, September). Event-based communications for equipment supervisory control. In 2005 IEEE Conference on Emerging Technologies and Factory Automation (Vol. 1, pp. 7-pp). IEEE.	2005		98800DOC00048676
Zhang, X., & Bajaj, C. (2009). Scalable isosurface visualization of massive datasets on commodity off-the-shelf clusters. Journal of parallel and distributed computing, 69(1), 39-53.	2009		98800DOC00044907
Zhang, X., Bajaj, C., & Ramachandran, V. (2002). Parallel and out-of-core view-dependent isocontour visualization using random data distribution. In D. Ebert, P. Brunet, & I. Navazo (Eds.), Eurographics / IEEE VGTC Symposium on Visualization (pp. 9-18). The Eurographics Association.	2002		98800DOC00065976
Zhe Fan et al. Lattice-Based Flow Field Modeling, IEEE Transactions on Visualization and Computer Graphics, Volume 10 (2004), Issue 6	2004	Fan 8	98800DOC00054174

Description	Publication Date	Short Name	Starting Bates Number
Zhe Fan et al., A General Framework for GPU-Cluster Based Visualization and Visual Simulation, submission to IEEE Visualization Journal, 2006 ("Fan 7")	2006	Fan 7	*[see footnote 3]*
Zhe Fan et al., Lattice-Based Flow Simulation on GPU, Conference Poster, 2004	2004	Fan 5	*[see footnote 3]*
Zhe Fan et al., Zippy: A Framework for Computation and Visualization on a GPU Cluster, EUROGRAPHICS 2008, Volume 27 (2008), Number 2	2008	Fan 4	98800DOC00044982
Zhe Fan, Flow Simulation and Visualization on GPU Clusters, Thesis Proposal, Stony Brook University, May 5, 2007	2007	Fan 2	98800DOC00045924
brookgpu-0.4.tar.gz (Brook source code, Oct. 15, 2004), at https://sourceforge.net/projects/brook/files/Official%20Releases/BrookGPU%20v0.4/.	2003		98800DOC00111739
brookgpu-0.4.tar.gz (Brook source code, Oct. 15, 2004), at https://sourceforge.net/projects/brook/files/Official%20Releases/BrookGPU%20v0.4/.	1997		98800DOC00111746
Dauger Research Profiled on National Television Part 1 (Mar. 24, 2006), at https://www.youtube.com/watch?app=desktop&v=jyFM5BWybB8.	2005	Heartbeat of America 1	98800DOC00111727
Dauger Research Profiled on National Television Part 2 (Mar. 24, 2006), at https://www.youtube.com/watch?v=WzCHgIuQIUU.	2005	Heartbeat of America 2	98800DOC00111726

Description	Publication Date	Short Name	Starting Bates Number
Dauger Research Profiled on National Television Part 3 (Mar. 24, 2006), at https://www.youtube.com/watch?v=o0UMsnZ9kn8.	2005	Heartbeat of America 3	98800DOC00111725
Dauger Research, Apple Canada Video – Demo Pt 1 (2002), at https://daugerresearch.com/media/applecanadavideodemopart1.shtml.	2002		98800DOC00111736
Dauger Research, Apple Canada Video – Demo Pt 2 (2002), at https://daugerresearch.com/media/applecanadavideodemopart2.shtml.	2002		98800DOC00111735
Dauger Research, Apple Canada Video – Demo Pt 3 (2002), at https://daugerresearch.com/media/applecanadavideodemopart3.shtml.	2002		98800DOC00111734
Dauger Research, Apple Canada Video – Demo Pt 4 (2002) available at https://daugerresearch.com/media/applecanadavideodemopart4.shtml.	2002		98800DOC00111733
Dauger Research, Apple Canada Video – Demo Pt 5 (2002), at https://daugerresearch.com/media/applecanadavideodemopart5.shtml.	2002		98800DOC00111732
Dauger Research, Apple Canada Video – Introduction (2002), at https://daugerresearch.com/media/applecanadavideointroduction.shtml.	2002		98800DOC00111730
Dauger Research, Apple Canada Video – Mac Cluster Recipe (2002), at https://daugerresearch.com/media/applecanadavideomacrecipe.shtml.	2002		98800DOC00111731

Description	Publication Date	Short Name	Starting Bates Number
Dauger Research, Apple Canada Video – Pooch's Four User Interfaces (2002), at https://daugerresearch.com/media/applecanadavideofouruis.shtml.	2002		98800DOC00111737
Dauger Research, Apple Canada Video – Slides (2002), at https://daugerresearch.com/media/applecanadavideo.shtml.	2002		98800DOC00111738
Dauger Research, Apple Canada Video – Why Macs (2002), at https://daugerresearch.com/media/applecanadavideowhymacs.shtml.	2002		98800DOC00111729
Dauger Research, Apple Canada Video – Why Parallel (2002), at https://daugerresearch.com/media/applecanadavideowhyparallel.shtml.	2002		98800DOC00111728
distmaple-1.1.16.zip (Distributed Maple, version 1.1.16) (2003), at https://www3.risc.jku.at/software/distmaple/distmaple-1.1.16.zip.	2003		98800DOC00111740
distmaple-1.1.17.zip (Distributed Maple, version 1.1.17) (2004), at https://www3.risc.jku.at/software/distmaple/.	2004		98800DOC00111741
distmath.zip, (Distributed Mathematica, June 27, 2000), at https://www3.risc.jku.at/software/distmath/.	2000		98800DOC00111742
FFTW-2.1.5, Copyright (c) 2003, Massachusetts Institute of Technology, source code available from http://www.fftw.org.	2003		98800DOC00111743
MatlabMPI_v0.95.tar.gz, (MatlabMPI source code, version 0.95) (2002), at https://archive.ll.mit.edu/mission/cybersec/softwaretools/matlabmpi/matlabmpi.html.	2002		98800DOC00111744

Description	Publication Date	Short Name	Starting Bates Number
MatlabMPI_v1.2br2.tar (MatlabMPI source code, version 1.2) (2004), at https://archive.ll.mit.edu/mission/cybersec/softwaretools/matlabmpi/matlabmpi.html.	2004		98800DOC00111745
PDE Solver, at https://web.archive.org/web/20040719214221/http://www.serv.net/~bseidel/AppleSeed/pdes.tar.	2000	PDE Solver Code	98800DOC00111747
PoochMPI Toolkit for Mathematica - Part 3 (Nov. 11, 2006), at https://www.youtube.com/watch?v=tJ4x7tPxPxY.	2006	PoochMPI Video 3	98800DOC00111724
PoochMPI Toolkit for Mathematica - Part 4 (Nov. 11, 2006), at https://www.youtube.com/watch?v=cQZ_y20P0BQ.	2006	PoochMPI Video 4	98800DOC00111723
pyMPI-2.4b1.tar.gz (pyMPI source code, version 2.4) (July 12, 2005), at https://sourceforge.net/projects/pympi/files/OldFiles/pyMPI-2.4b1.tar.gz/download.	2005		98800DOC00111748
Architecture and Implementation of Vulcan	1994		DEF_00058693
Image Understanding Architecture Final Report	1991	IUA Final Report	DEF_00057362
"Distributed Pixel Method to Speed-Up the RS data assimilation of SWAP Model" by Akhter et al., 2005, link	2005	Akhter	DEF_00054702
A Case for Now Network of Workstations	1995		DEF_00056503

Description	Publication Date	Short Name	Starting Bates Number
"A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard," Parallel Computing Volume 22, Issue 6, September 1996, Pages 789-828	1996		DEF_00054710
A Peer-to-Peer Network Based on Multi-Mesh Architecture - Murthy	2003	Murthy	DEF_00053459
A Scalable Content-Addressable Network	2001		DEF_00056514
"AMD Announces World's First 64-Bit, x86 Multi-Core Processors For Servers And Workstations At Second-Anniversary Celebration Of AMD Opteron™ Processor," AMD, April 21, 2005, https://web.archive.org/web/20050421160559/http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_543~97108,00.html	April 21, 2005		DEF_00056342
"An Extension of Maple for Grid and Cluster Computing," Studies in Informatics and Control, vol. 14, No. 1	March 2004		DEF_00053464
G. Almasi et al., "An Overview of the Blue Gene/L System Software Organization"	2003		DEF_00056527
N. R. Adiga et al., "An Overview of the Blue Gene/L Supercomputer," IBM J. RES. & DEV.	2002	Adiga1	DEF_00053470
Applications in Parallel MATLAB	2000		DEF_00056537

Description	Publication Date	Short Name	Starting Bates Number
Becker, Donald J., Thomas Sterling, Daniel Savarese, John E. Dorband, Udaya A. Ranawak, Charles V. Packer, "Beowulf: A Parallel Workstation for Scientific Computation," Proceedings, International Conference on Parallel Processing.	1995		DEF_00054751
N. R. Adiga et al., "Blue Gene/L torus interconnection network," IBM J. RES. & DEV.	March/May 2005	Adiga2	DEF_00053492
C. Yang, "Using PC Cluster For High Performance Computing"	July 2002		DEF_00053864
Using Multivariate Clustering to Characterize Ecoregion Borders, Computing in Science & Engineering 1(4):18-25	July 1999		DEF_00054783
Chord A Scalable Peer-to-peer Lookup Service for Internet Applications - Stoica	2001	Stoica	DEF_00053887
"Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," Ion Stoica et al.	Jan. 2002		DEF_00057425 & DEF_00057470
Communication in iWarp Systems, T. Gross	1989		DEF_00056555
"Cplant Runtime System Support for Multi-Processor and Heterogenous Compute Nodes," Kevin Pedretti et al.	2002	Pedretti	DEF_00056565
Scalability and Performance of a Large Linux Cluster	2000		DEF_00056573
CRAY T3E and SGI Origin2000: Merging Architectures from the User's Point of View	February 1999		DEF_00056541

Description	Publication Date	Short Name	Starting Bates Number
Design and Implementation of the iWarp Protocol in Software, Dalessandro et al., Conference: International Conference on Parallel and Distributed Computing Systems, PDCS 2005, Phoenix, AZ, USA	November 2005		DEF_00057482
Designing a Scalable Parallel System- The IBM SP2	1995		DEF_00056595
Distributed Maple–parallel computer algebra in networked environments, W. Schreiner et al. / Journal of Symbolic Computation 35 (2003) 305–347	2003		DEF_00053899
"Dual Core Era Begins, PC Makers Start Selling Intel-Based PCs," Intel News Release, https://www.intel.com/pressroom/archive/releases/2005/20050418comp.htm	April 18, 2005	Intel News Release	DEF_00056346
Dynamic Load Balancing with MatlabMPI, ICCS 2006, Part II, LNCS 3992, pp. 430–437, 2006	2006		DEF_00056619
R. S. Germain et al., "Early performance data on the Blue Matter molecular simulation framework," IBM J. RES. & DEV.	March/May 2005	Germain	DEF_00053504
"Message Passing Under MATLAB," J. Baldomero	2001		DEF_00056631
"GridInfo: a graphical Grid monitoring tool in Java" by Osborne et al., Journal of Computing Sciences in Colleges, Volume 21, Issue 1, Pages 104 – 113, Published:, https://dl.acm.org/doi/abs/10.5555/1088791.1088810	2005	Osborne	DEF_00056348
Hargrove_SciAm_20010801 The Do-It-Yourself Supercomputer	2001	SA Article	DEF_00054755 & DEF_00057954

Description	Publication Date	Short Name	Starting Bates Number
"High Performance Computing: An Introduction to Parallel Programming With Beowulf," Forrest M. Hoffman & William W. Hargrove, https://web.archive.org/web/20030418102558/http:/climate.ornl.gov/~forrest/osdj-2000-11/	April 2003	Hoffman	DEF_00054763
Internet Archive Affidavit re Hoffman et al.	N/A		DEF_00058529
High Performance Tools and Technologies_2004	December 2004		DEF_00056641
High-Performance Virtual Machines Cluster with Supercomputing APIs and Performance	1996		DEF_00056727
Fourmanoit et al., "Towards Best Achievable Floating Point Performance for Linear Algebra Computations on COTS Beowulf Clusters, Applications of High Performance," Computing in Engineering VII, ISBN 1-85312-924-0 (WIT Press 2002)	2002	Fourmanoit	DEF_00055800
Mechanistic-Based Genetic Algorithm Search on a Beowulf Cluster of Linux PCs, Conference: Proceedings of the High Performance Computing 2000 (HPC2000) Conference	April 2000		DEF_00054791
"Peer-to-Peer Computing," Dejan S. Milojicic et al., HP Laboratories	7/3/2003	Milojicic	DEF_00055810
"Development of Parallel Methods For a 1024-Processor Hypercube," John L. Gustafson, Gary R. Montry, and Robert E. Benner, Sandia National Laboratories	1988	Gustafson	DEF_00058491

Description	Publication Date	Short Name	Starting Bates Number
IBM Blue Gene Supercomputers go green, https://www.ibm.com/history/blue-gene	N/A	IBM Blue Gene History	DEF_00058520
Inside the Transputer	1990		DEF_00056735
Interactive Supercomputing's Star-P Platform: Parallel MATLAB and MPI Homework Classroom Study on High Level Language Productivity, Edelman, A., Husbands, P., & Leibman, S. (2006).	2006		DEF_00058024
Introduction to Parallel Programing and pMatlab v2.0	2011		DEF_00056975
B.M. Wise and N.B. Gallagher, "An Introduction to Linear Algebra," Critical Reviews in Analytical Chemistry, 28(1), pps 1-19, 1998	1998		DEF_00055958
Massively parallel computing using commodity components	2000		DEF_00057086
MathGridLink - A bridge between Mathematica and "the Grid"	January 2003		DEF_00056337
MATLAB*G: A Grid Based Parallel MATLAB	2004		DEF_00057004
MATLAB*P 2.0: A unified parallel MATLAB	2003		DEF_00057011
Choy, Long Yin, "MATLAB*P: Interactive Supercomputing Made Practical," MIT	September 2002	Choy Thesis	DEF_00057017
MatlabMPI, J. Kepner and S. Ahalt	2004		DEF_00057110
MATmarks: A shared memory environment for MATLAB programming	1999		DEF_00057084
Message Passing Evaluation and Analysis on Cray T3E and SGI Origin 2000 Systems	1999		DEF_00057119

Description	Publication Date	Short Name	Starting Bates Number
mpi_ibm			DEF_00058026
MPI-standard 1994	May 5, 1994		DEF_00056101
Experiments On Distributed Remote Sensing Data (Modis And Aster) Processing Using Optima Cluster, May 2005 Conference: NECSEC, https://www.researchgate.net/publication/295256794_Experiments_On_Distributed_Remote_Sensing_Data_Modis_And_Aster_Processing_Using_Optima_Cluster	May 2005		DEF_00054775
On Grid Programming and MATLAB*G, Teo et al.	2004		DEF_00057129
A. Gara et al., "Overview of the Blue Gene/L system architecture," IBM J. RES. & DEV.	March/May 2005	Gara	DEF_00053513
Parallel MATLAB: Doing It Right, Ron Choy and Alan Edelman	February 2005		DEF_00056492
Parallel MATLAB: Doing It Right, Ron Choy and Alan Edelman	November 2003		DEF_00057145
Experiments On Distributed Remote Sensing Data (Modis And Aster) Processing Using Optima Cluster, May 2005 Conference: NECSEC, https://www.researchgate.net/publication/295256794_Experiments_On_Distributed_Remote_Sensing_Data_Modis_And_Aster_Processing_Using_Optima_Cluster	May 2005		DEF_00056358
Parallel Programing with MatlabMPI *	October 31, 2018		DEF_00057172

Description	Publication Date	Short Name	Starting Bates Number
Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, Ant Rowstron and Peter Druschel, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)	November 2001		DEF_00057520
Peer to Peer Messaging System for Distributed and Parallel Applications	November 2005	Geoffery	DEF_00054446
PeerCluster:_A_Cluster Based Peer-to-Peer System	October 2006		DEF_00058563
Peer-to-Peer Support for MATLAB-Style Computing	May 2004	Agarwal	DEF_00054319
Performance Evaluation of MatlabMPI and the MPI Toolbox for MATLAB	April 2005		DEF_00057175
Performance of Message-Passing MPI Toolboxes	2003		DEF_00057183
Nadya Travinin and Jeremy Kepner, "pMatlab Parallel Matlab Library"	June 2006	Travinin	DEF_00057488
The Portals 3.0 Message Passing Interface, Revision 1.0	December 1999		DEF_00057198
Profiling pMatlab and MatlabMPI Applications Using the MATLAB 7 Profiler, http://www.ll.mit.edu/pMatlab/files/Profiling_pMatlab_MatlabMPI.pdf	Before 2006		DEF_00057242
PVMaple: A Distributed Approach to Cooperative Work of Maple Processes, Dana Petcu	2000		DEF_00056483
Carson et al., Computational Gains Using RPVM on a Beowulf Cluster, R News, ISSN 1609-3631, Vol. 3/1	2003	Carson	DEF_00057684

Description	Publication Date	Short Name	Starting Bates Number
Programming Future Architectures: Dusty Decks, Memory Walls, and the Speed of Light, Arun Francis Rodrigues	October 5, 2006		DEF_00057690
Scaling up parallel scientific applications on the IBM SP	January 8, 2004		DEF_00057260
George Almasi et al., "Scaling Physics and Material Science Applications on a Massively Parallel Blue Gene/L System"	June 2005	Almasi2	DEF_00053648
SCE Toolboxes for the Development of High Level Parallel Applications	2006		DEF_00057248
The POWER4 Processor Introduction and Tuning Guide, IBM Redbooks	November 2001		DEF_00058264
Sonzogni et al., "A Parallel Finite Element Program on a Beowulf Cluster," Advances in Engineering Software, 33, 427-443	2002	Sonzogni	DEF_00058712
High Performance Computing - HiPC 2004 Conference paper, "Sparse Matrices in Matlab*P: Design and Implementation"	2004		DEF_00057542
M. Krishnan and J. Nieplocha, "Optimizing Parallel Multiplication Operation for Rectangular and Transposed Matrices," in Proceedings of 10th IEEE International Conference on Parallel and Distributed Systems (ICPADS'04)	2004		DEF_00057554
SRUMMA: A Matrix Multiplication Algorithm Suitable for Clusters and Scalable Shared Memory Systems, Manojkumar Krishnan and Jarek Nieplocha	May 2004		DEF_00058476
PASTRY: the third generation of peer-to-peer networks	January 2006		DEF_00057579

Description	Publication Date	Short Name	Starting Bates Number
STARAN parallel processor system software, Edward W. Davis, Goodyear Aerospace Corporation	1974		DEF_00058583
Super-peer architectures for distributed computing, Atul Saini	2002		DEF_00058700
"Supporting Systolic and Memory Communication in iWarp," S. Borkar et al.	1990	Borkar1	DEF_00057300
Survey of Symbolic Computations on the Grid	March 2005		DEF_00057926
Symbolic Computations on Grids (Petcu et al.)	2005	Petcu6	DEF_00057937
Tapestry: A Resilient Global-Scale Overlay for Service Deployment, Ben Y. Zhao et al.	January 2004		DEF_00057671
The Berkeley Networks of Workstations (NOW) Project	1995		DEF_00057312
"The IBM Victor V256 partitionable multiprocessor," D. G. Shea et al.	1991	Shea	DEF_00058589
The Performance of MPI Derived Types on a SGI Origin 2000, a Cray T3E-900, a Myrinet Linux Cluster and an Ethernet Linux Cluster	October 2001		DEF_00057317
The transputer, Colin Whitby-Strevens, INMOS Limited,	1985		DEF_00057342
"The NYU Ultracomputer – designing an MIMD Shared Memory Parallel Computer," Allan Gottlieb et al.	1983	Gottlieb	DEF_00058607
Analysis and Simulation of the Transputer Microprocessor, Winston Chang Liu, University of Oxford			DEF_00058622
Transputers-Past, Present and Future	1990		DEF_00057351

Description	Publication Date	Short Name	Starting Bates Number
George Almasi et al., "Unlocking the Performance of the BlueGene/L Supercomputer"	2004	Almasi1	DEF_00053850

3. Table 3: Prior Art Products and Systems[5]

Name	Developer, Provider, or Manufacturer[6]	Short Name	Date in Public Use, on Sale, or Otherwise Available to the Public[7]
PVMaple	Western University of Timișoara	PVMaple	Sept. 2005
Distributed Maple	Johannes Kepler University	Distributed Maple	Mar. 2003
Matlab*P	Massachusetts Institute of Technology	Matlab*P	Aug. 2002
Fastest Fourier Transform in the West with Message-Passing Interface	Massachusetts Institute of Technology	FFTW	March 2003
Intel Paragon Maple	ETH Zürich	Intel Paragon Maple	Oct. 1996
Stony Brook GPU Cluster for High Performance Computing	Stony Brook University	Stony Brook	Nov. 2004
AppleSeed-Pooch System	University of California, Los	AppleSeed-Pooch	Jan. 2000

[5] Documents and things relied on as evidence of the prior art products and systems are included in Table 2 above.

[6] Including all relevant affiliates and/or successors in interest. In addition, in some cases, the prior art systems were combined and used together with software components developed by others, as noted in the relevant charts and/or supporting documents.

[7] These products were in public use, on sale, offered for sale, or otherwise known to the public at least as of this date. Intel reserves the right to revise these dates in light of additional discovery from the parties or third parties.

Name	Developer, Provider, or Manufacturer[6]	Short Name	Date in Public Use, on Sale, or Otherwise Available to the Public[7]
	Angeles / Dauger Research		
MultiMATLAB	Cornell University	MultiMATLAB	Nov. 1997
Microsoft Windows Compute Cluster Server 2003 System	Microsoft Corporation	CCS	June 2006
Supercomputing Engine for Mathematica System	Advanced Cluster Systems / Dauger Research	SEM	July 2006
Brook and Merrimac System	Stanford University	Merrimac	Apr. 2006
Whitney Commodity Computing Testbed	NASA Ames Research Center	Whitney	Oct. 1997
MatlabMPI from MIT	Massachusetts Institute of Technology	MatlabMPI	Jan. 2003
pyMPI	Lawrence Livermore National Laboratories	pyMPI	July 2005
IBM Blue Gene system	IBM and Lawrence Livermore National Laboratory	IBM Blue Gene	Sept. 2004
Beowulf-style cluster	Forrest M. Hoffman and William W. Hargrove at Oak Ridge National Laboratory	Hoffman Beowulf	1999
The Beowulf project (http://www.beowulf.org/)	NASA	NASA Beowulf	1995
Transputer	INMOS	Transputer	Oct. 1985
iWarp	Intel / Carnegie Mellon University	iWarp	May 1990
pMatlab	Massachusetts Institute of Technology	pMatlab	June 2006

Name	Developer, Provider, or Manufacturer[6]	Short Name	Date in Public Use, on Sale, or Otherwise Available to the Public[7]
Cornell Multitask Toolbox for Matlab	Cornell University	CMTM	July 2000
PVMTB/MPITB	Universidad de Granada	PVMTB/MPITB	June 2002
MATmarks	University of Illinois Urbana-Champaign	MATmarks	Aug. 1999
MATLAB*G	National University of Singapore	MATLAB*G	Jan. 2004
Scalable POWERParallel	IBM	IBM SP	Feb. 1993
Cray T3E	Cray Research / SiliconGraphics	T3E	Nov. 1995
Network of Workstations	University of California, Berkeley	NOW	Mar. 1995
High Performance Virtual Machines	University of Illinois Urbana-Champaign	HPVM	July 2002
ASCI Red	Intel / Sandia National Laboratories	ASCI Red	June 1997
Computational Plant	Sandia National Laboratories	Cplant	June 2001
Pastry Distributed Hash Table	Microsoft Research	Pastry	Nov. 2001
Tapestry Distributed Hash Table	University of California, Berkeley	Tapestry	Jan. 2004
Chord Distributed Hash Table	Massachusetts Institute of Technology	Chord	Aug. 2001
Content-Addressable Network Distributed Hash Table	University of California, Berkeley	CAN	Aug. 2001
Shared and Remote memory access based Universal Matrix	Pacific Northwest National Laboratory	SRUMMA	Apr. 2004

Name	Developer, Provider, or Manufacturer[6]	Short Name	Date in Public Use, on Sale, or Otherwise Available to the Public[7]
Multiplication Algorithm			
Image Understanding Architecture	University of Massachusetts	IUA	September 1991
IBM Victor System	IBM	IBM Victor	1987
The NYU Ultracomputer	New York University	NYU Ultracomputer	1983

Any citation to one or more of these prior art references, or other prior art references regarding any method or system, should be construed to constitute not only a citation to the prior art reference itself but also a reference to the system itself.

C. Admitted Prior Art

The Asserted Patents, in their file histories or specifications, contain statements admitting that certain claim limitations were already known in the art at the time of the purported invention or cite references as prior art. Such admissions ("admitted prior art") "are binding on the patentee for purposes of a later inquiry into obviousness." *PharmaStem Therapeutics, Inc. v. ViaCell, Inc.*, 491 F.3d 1342, 1362 (Fed. Cir. 2007); *see also Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 1570 (Fed. Cir. 1988) (same as to anticipation). Exemplary admissions are discussed below. Intel reserves the right to rely on all admitted prior art for the Asserted Patents. References cited as admitted prior art in the Asserted Patents are incorporated herein by reference. All specification citations are to the '768 patent as a representative example, but the admissions therein bind Plaintiff on all Asserted Patents, as they share a common specification.

1. Admitted Products

- Mathematica

- "An example of a software package that includes a kernel that is designed to communicate with a single node is Mathematica® from Wolfram Research, Inc. ('Mathematica')." '768 patent, 1:38–41.
- "One embodiment provides access to such high-performance computing through a Mathematica Front End, a command line interface, one or more high-level commands, or a programming language such as C or FORTRAN." '768 patent, 2:13–17.

- gridMathematica
  - "A product known as gridMathematica, also from Wolfram Research, Inc., gives Mathematica the capability to perform a form of grid computing known as 'distributed computing.'" '768 patent, 1:44–46.
- C
  - "One embodiment provides access to such high-performance computing through a Mathematica Front End, a command line interface, one or more high-level commands, or a programming language such as C or FORTRAN." '768 patent, 2:13–17.
- FORTRAN
  - "One embodiment provides access to such high-performance computing through a Mathematica Front End, a command line interface, one or more high-level commands, or a programming language such as C or FORTRAN." '768 patent, 2:13–17.
- Message Passing Interface
  - "One embodiment adds and supports Message Passing Interface ('MPI') calls directly from within a user interface, such as, for example, the Mathematica programming environment." '768 patent, 2:24–27.
- Maple
  - "The present disclosure is not limited to a single software program; the systems and methods can be used with other application software such as, for example, Maple®, MATLAB®, MathCAD®, Apple Shake®, Apple® Compressor, IDL®, other applications employing an interpreter or a kernel, Microsoft Excel®, Adobe After Effects®, Adobe Premiere®, Adobe Photoshop®, Apple Final Cut Pro®, and Apple iMovie®." '768 patent, 4:16–23.
- MATLAB
  - *Id.*
- MathCAD
  - *Id.*
- Apple Shake
  - *Id.*
- Apple Compressor
  - *Id.*
- IDL
  - *Id.*
- Microsoft Excel
  - *Id.*
- Adobe After Effects
  - *Id.*
- Adobe Premiere
  - *Id.*

- Adobe Photoshop
  - *Id.*
- Apple Final Cut Pro
  - *Id.*
- Apple iMovie
  - *Id.*
- Pentium processor
  - "The processors 112a-b can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® 20 processor, etc." '768 patent, 7:12–21.
- Pentium II processor
  - *Id.*
- Pentium Pro processor
  - *Id.*
- Pentium III processor
  - *Id.*
- Pentium 4 processor
  - *Id.*
- Core Duo processor
  - *Id.*
- Core 2 Duo processor
  - *Id.*
- Xeon processor
  - *Id.*
- Itanium processor
  - *Id.*
- Pentium M processor
  - *Id.*
- x86 processor
  - *Id.*
- Athlon processor
  - *Id.*
- 8051 processor
  - *Id.*
- MIPS processor
  - *Id.*
- PowerPC processor
  - *Id.*
- ALPHA processor
  - *Id.*
- Point-to-Point Protocol

- "However, it is to be appreciated that other types of network communication protocols such as, for example, Point-to-Point Protocol ('PPP'), Server Message Block ('SMB'), Serial Line Internet Protocol ('SLIP'), tunneling PPP, AppleTalk, etc., may also be used." '768 patent, 7:40–44.

- Server Message Block
  - *Id.*
- Serial Line Internet Protocol
  - *Id.*
- Tunnelling PPP
  - *Id.*
- AppleTalk
  - *Id.*
- Microsoft Windows 3.X
  - "The computer system 110 may be used in connection with various operating systems such as: Microsoft® Windows® 3.X, Windows 95®, Windows 98®, Windows NT®, Windows 2000®, Windows XP®, Windows CE®, Palm Pilot OS, OS/2, Apple® MacOS®, MacOS X®, MacOS X Server®, Disk Operating System (DOS), UNIX, Linux®, VxWorks, or IBM® OS/2®, Sun OS, Solaris OS, IRIX OS operating systems, etc." '768 patent, 7:60–67.
- Windows 95
  - *Id.*
- Windows 98
  - *Id.*
- Windows NT
  - *Id.*
- Windows 2000
  - *Id.*
- Windows XP
  - *Id.*
- Windows CE
  - *Id.*
- Palm Pilot OS
  - *Id.*
- OS/2
  - *Id.*
- Apple MacOS
  - *Id.*
- Apple Mac OS X
  - *Id.*
- MacOS X Server
  - *Id.*
- Disk Operating System
  - *Id.*
- UNIX
  - *Id.*

- Linux
  - *Id.*
- VxWorks
  - *Id.*
- IBM OS/2
  - *Id.*
- Sun OS
  - *Id.*
- Solaris OS
  - *Id.*
- Irix OS
  - *Id.*
- Ethernet (IEEE 802.3)
  - "In addition, the connectivity to the network can be, for example, a modem, Ethernet (IEEE 802.3), Gigabit Ethernet, 10-Gigabit Ethernet, Token Ring (IEEE 802.5), Fiber Distributed Datalink Interface (FDDI), Frame Relay, InfiniBand, Myrinet, Asynchronous Transfer Mode (ATM), or another interface." '768 patent, 10:64–11:2.
- Gigabit Ethernet
  - *Id.*
- 10-Gigabit Ethernet
  - *Id.*
- Token Ring (IEEE 802.5)
  - *Id.*
- Fiber Distributed Datalink Interface
  - *Id.*
- Frame Relay
  - *Id.*
- InfiniBand
  - *Id.*
- Myrinet
  - *Id.*
- Asynchronous Transfer Mode
  - *Id.*
- Mathematica MathLink
  - "A cluster node module 204 can use an application-specific toolkit or interface such as, for example, Mathematica's MathLink, Add-Ons, or packets, to interact with an application. Normally used to connect a Mathematica kernel to a user interface known as the Mathematica Front End or other Mathematica kernels, MathLink is a bidirectional protocol to sends 'packets' containing messages, commands, or data between any of these entities." '768 patent, 11:55–58.
- Parallel Computing Toolkit
  - "In contrast, a parallel computing toolkit (PCT) that is provided as part of the gridMathematica software package does not provide a means for instances of the same code running on different nodes to communicate, collaborate, or coordinate work among the instances." '768 patent, 12:26–30.

- Beowulf Computer Cluster
  - "As personal computers led the hardware and software evolution to where substantial computing power became attainable by the individual, systems known as 'Clusters' became prevalent. Computer clusters are a, type of parallel computation system where a network of computing devices' processors are tightly coupled to share computation tasks in parallel fashion. An early and fairly prevalent version of a cluster computer is the 'Beowulf' system first assembled at a NASA site to solve Earth Sciences problems (NASA Goddard Space Flight Center). The Beowulf cluster is characterized by a set of personal computers connected by specialized network hardware and running a specialized version of the open-source Linux operating system." U.S. Patent No. 7,136,924 (incorporated by reference at '768 patent, 22:29–31.) at 1:35–47.
- Cray T3E
  - "Large parallel computers are typically located at major supercomputing centers, the hardware consists of large parallel computers (e.g. Cray T3E, IBM SP, Fujitsu, etc.), the Software is commonly proprietary vendor Software (many time a Unix variant)." '924 patent at 1:27–31.
- IBM SP
  - *Id.*
- Fujitsu
  - *Id.*

2. Admitted Claim Elements

- Computer Clusters
  - "Computer clusters include a group of two or more computers, microprocessors, and/or processor cores ('nodes') that intercommunicate so that the nodes can accomplish a task as though they were a single computer." '768 patent, 1:19–22.
  - "A product known as gridMathematica, also from Wolfram Research, Inc., gives Mathematica the capability to perform a form of grid computing known as 'distributed computing.'" '768 patent, 1:44–46.
  - "In gridMathematica, each of a plurality of kernels runs on a single node. One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels)." '768 patent, 1:53–57.
  - "As personal computers led the hardware and software evolution to where substantial computing power became attainable by the individual, systems known as 'Clusters' became prevalent. Computer clusters are a, type of parallel computation system where a network of computing devices' processors are tightly coupled to share computation tasks in parallel fashion. An early and fairly prevalent version of a cluster computer is the 'Beowulf' system first assembled at a NASA site to solve Earth Sciences problems (NASA Goddard Space Flight Center). The Beowulf cluster is characterized by a set of personal computers connected by specialized network hardware and running a specialized version of the open-source Linux operating system." U.S. Patent No. 7,136,924 (incorporated by reference at '768 patent, 22:29–31.) at 1:35–47.
- Plurality of Nodes

- "A product known as gridMathematica, also from Wolfram Research, Inc., gives Mathematica the capability to perform a form of grid computing known as 'distributed computing.'" '768 patent, 1:44–46.
- "In gridMathematica, each of a plurality of kernels runs on a single node. One kernel is designated the master kernel, which handles all input, output, and scheduling of the other kernels (the computational kernels or slave kernels)." '768 patent, 1:53–57.
- "For example, if the processors 112a-b were each Core 2 Duo® processors having two processing cores, computer system 110 would have four nodes in all." '768 patent, 7:27–30.

- Hardware Processors with a Plurality of Processing Cores
  - "The processors 112a-b can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® processor, etc." '768 patent, 7:12–21.
  - "For example, if the processors 112a-b were each Core 2 Duo® processors having two processing cores, computer system 110 would have four nodes in all." '768 patent, 7:27–30.
- Command to Start a Cluster Initialization Process for the Computer Cluster, Wherein the Cluster Initialization Process Comprises Establishing Communication Among Two or More of the Nodes
  - "One embodiment adds and supports Message Passing Interface ('MPI') calls directly from within a user interface, such as, for example, the Mathematica programming environment." '768 patent, 2:24–27.
  - "A cluster node module 204 can include at least a portion of an application programming interface ('API') known as the Message Passing Interface ('MPI'), which is used in several supercomputer and cluster installations." '768 patent, 11:45–49.
  - "Cluster configuration module 208 includes program code for initializing a plurality of cluster node modules to add cluster computing support to computer systems 110, 120, 130. U.S. Pat. No. 7,136,924, issued to Dauger (the ''924 patent'), the entirety of which is hereby incorporated by reference and made a part of this specification, discloses a method and system for parallel operation and control of computer clusters." '768 patent, 22:26–33.
  - "The cluster node modules 204a-e establish communication with one another at 404. In one embodiment, each of the cluster node modules 204a-e establish direct connections using the MPI_Init command with other cluster node modules 204a-e launched on the computer cluster 100 by the cluster configuration module 208." '768 patent, 23:51–56.
  - "Message passing support is initialized on the kernel modules 206a-e at 412. In one embodiment, each cluster node module 204 supplies initialization code (for example, Mathematica initialization code) to the local kernel module 206 to support message passing." '768 patent, 24:12–16.
- Program Code for a Single-Node Kernel That, When Executed, is Capable of Causing the Hardware Processor to Evaluate Mathematical Expressions

- “Some application programs include an interpreter that executes instructions provided to the program by a user, a script, or another source. Such an interpreter is sometimes called a ‘kernel’ because, for example, the interpreter can manage at least some hardware resources of a computer system and/or can manage communications between those resources and software (for example, the provided instructions, which can include a high-level programming language). Some software programs include a kernel that is designed to communicate with a single node. An example of a software package that includes a kernel that is designed to communicate with a single node is Mathematica® from Wolfram Research, Inc. (‘Mathematica’).” ’768 patent, 1:30–41.
- “The present disclosure is not limited to a single software program; the systems and methods can be used with other application software such as, for example, Maple®, MATLAB®, MathCAD®, Apple Shake®, Apple® Compressor, IDL®, other applications employing an interpreter or a kernel, Microsoft Excel®, Adobe After Effects®, Adobe Premiere®, Adobe Photoshop ®, Apple Final Cut Pro®, and Apple iMovie®.” ’768 patent, 4:16–23.
- “One embodiment adapts a software module designed to run on a single node, such as, for example, the Mathematica kernel, to support cluster computing, even when the software module is not designed to provide such support.” ’768 patent, 2:18–21.
- “In one embodiment, the single-node kernel includes a Mathematical kernel.” ’768 patent, 3:1–2.
- “In one embodiment, the cluster node modules 204a-e provide a way for many kernel modules 206a-e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another.” ’768 patent, 11:42–45.
- “The mpilSend[ ] command can be called from within a kernel module 206 (for example, a Mathematica kernel). It creates a packet containing the Mathematica expression to be sent as payload and where the expression should be sent.” ’768 patent, 14:1–4.

TABLE I

Call	Description
ParallelDo[expr, loopspec]	Like Do[] except that it evaluates expr across the cluster, rather than on just one processor. The rules for how expr is evaluated is specified in loopspec, like in Do[].
ParallelFunctionToList[f, count] ParallelFunctionToList[f, count, root]	Evaluates the function f[i] from 1 to count, but across the cluster, and returns these results in a list. The third argument has it gather this list into the processor whose ID is root.
ParallelTable[expr, loopspec] ParallelTable[expr, loopspec, root]	Like Table[] except that it evaluates expr across the cluster, rather than on just one processor, returning the locally evalulated portion. The third argument has it gather this table in to the processor whose ID is root.
ParallelFunction[f, inputs, root]	Like f[inputs] except that it evaluates f on a subset of inputs scattered across the cluster from processor root and gathered back to root.
ParallelNintegrate[expr, loopspec]	Like Nintegrate[] except that it evaluates a numerical integration

TABLE I-continued

Call	Description
ParallelNintegrate[expr, loopspec, digits]	of expr over domains partitioned into the number of processors in the cluster, then returns the sum. The third argument has each numerical integration execute with at least that many digits of precision.

TABLE K

Call	Description
ParallelTranspose[matrix]	Like Transpose[] except that it transposes matrix that is in fact represented across the cluster, rather than on just one processor. It returns the portion of the transposed matrix meant for that processor.
ParallelProduct[matrix, vector]	Evaluates the product of matrix and vector, as it would on one processor, except that matrix is represented across the cluster.
ParallelDimensions[matrix]	Like Dimensions[] except that matrix is represented across the cluster, rather than on just one processor. It returns a list of each dimension.
ParallelTr[matrix]	Like Tr[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the trace of this matrix.
ParallelIdentity[rank]	Like Identity[], it generates a new identity matrix, except that the matrix is represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelOuter[f, vector1, vector2]	Like Outer[f, vector1, vector2] except that the answer becomes a matrix represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelInverse[matrix]	Like Inverse[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the inverse of the matrix.

TABLE M	
Call	Description
ParallelFourier[list]	Like Fourier[] except that list is a two- or three-dimensional list represented across the cluster, like for matrices, above. It returns the portion of the Fourier-transformed array meant for that processor.

-
- "The user commands can be simple (for example, '1+1'), but can also be entire subroutines and sequences of code (such as, for example, Mathematica code), including calls to MPI from within the user interface module 202 (for example, the Mathematica Front End) to perform message passing between kernel modules 206 a-e (for example, Mathematica kernels)." '768 patent, 24:58–64.

- Mechanism for the Nodes to Communicate Results of Mathematical Expression Evaluation With Each Other Using a Peer-to-Peer Architecture/Asynchronous Calls
  - "One embodiment adds and supports Message Passing Interface ('MPI') calls directly from within a user interface, such as, for example, the Mathematica programming environment. In one embodiment, MPI calls are added to or made available from an interactive programming environment, such as the Mathematica Front End." '768 patent, 2:24–29.
  - "In one embodiment, the cluster node modules 204 a-e provide a way for many kernel modules 206 a-e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another. A cluster node module 204 can include at least a portion of an application programming interface ('API') known as the Message-Passing Interface ('MPI'), which is used in several supercomputer and cluster installations." '768 patent, 11:42–49.
  - "In one embodiment, the MPI module 302 can include basic MPI calls such as, for example, relatively low-level routines that map MPI calls that are commonly used in other languages (such as C and Fortran), so that such calls can be available directly from the Mathematica user interface 204. In some embodiments, basic MPI calls include calls that send data, equations, formulas, and/or other expressions." '768 patent, 13:10–16.

- TABLE B

Call	Description
mpiSend[expr, target, comm, tag]	Sends an expression expr to a node with the ID target in the communicator would comm, waiting until that expression has left this kernel
mpiRecv [expr, target, comm, tag]	Receives an expression into expr from a node with the ID target in the communicator world comm, waiting until the expression has arrived
mpiSendRecv[sendexpr, dest, recvexpr, source, comm]	Simultaneously sends the expression sendexpr to the node with the ID target and receives an expression into recvexpr from the node with the ID source in the communicator world comm, waiting until both operations have returned.

- Asynchronous MPI Calls

  Asynchronous calls make it possible for the kernel to do work while communications are proceeding simultaneously. It is also possible that another node may not be able to send or receive data yet, allowing one kernel to continue working while waiting.

  TABLE C

Call	Description
mpiISend[expr, target, comm, tag, req]	Sends an expression expr to a processor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to mpiTest[req] until mpiTest[req] returns True.
mpiIRecv[expr, target, comm, tag, req]	Receives an expression expr from a proessor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to mpiTest[req] until mpiTest[req] returns True. The expr is not safe to access until mpiTest[req] returns True.
mpiTest[req]	Completes asynchronous behavior of mpiISend and mpiIRecv
mpWait[req]	Calls mpiTest until it returns True.
mpiWaitall[reglist]	Calls mpiWait all on every element of reqlist
mpiWaitany[reqlist]	Calls mpiTest on each element of realist until one of them returns True

- ’768 patent, 13:41–65.
- “Peer-to-Peer MPI[:] The mpiSend and mpiRecv commands make possible basic message passing, but one needs to define which processor to target. The following defines a new variable, targetProc, so that each pair of processors will point to each other.” ’768 patent, 26:65–27:3.

In this example, the even processors target its "right" processor, while the odd ones point its "left." For example, if the processors were lined up in a row and numbered in order, every even-numbered processor would pair with the processor following it in the line, and every odd-numbered processor would pair with the processor preceding it. Then a message can be sent:

```
In[4]:= If [ 1==Mod[ $IdProc , 2],mpiSend[N[Pi,22],targetProc,
    mpiCommWorld,d], mpiRecv[a,targetProc,mpiCommWorld,d]]
```

The If [ ] statement causes the processors to evaluate different code: the odd processor sends 22 digits of Pi, while the even receives that message. Note that these MPI calls return nothing. The received message is in the variable a:

```
In[5]:= a
Out[5]:= 3.1415926535897932384626
In[6]:= Clear[a]
```

The variable a on the odd processors would have no definition. Moreover, if $NProc is 8, processor 3 sent Pi to processor 2, processor 5 sent Pi to processor 4, and so on. These messages were not sent through processor 0, but they communicated on their own.

- ’768 patent, 27:5–37.

- User Interfaces/User Interface Modules/User Connection Interface
  - "One embodiment adds and supports Message Passing Interface (‘MPI’) calls directly from within a user interface, such as, for example, the Mathematica programming environment." ’768 patent, 2:24–27.
  - "In some embodiments, the user interface can include at least one of a Mathematica front end or a command line." ’768 patent, 3:2–4.
  - "As shown in FIG. 2, an optional user interface module 202 such as, for example, a Mathematica front end and/or a command line interface, can connect to a cluster node module 204a." ’768 patent, 5:56–59.
  - "A cluster node module 204 can use an application-specific toolkit or interface such as, for example, Mathematica's MathLink, Add-Ons, or packets, to interact with an application. Normally used to connect a Mathematica kernel to a user interface known as the Mathematica Front End or other Mathematica kernels, MathLink is a bidirectional protocol to sends ‘packets’ containing messages, commands, or data between any of these entities." ’768 patent, 11:55–62.
  - "In one embodiment, the MPI module 302 can include basic MPI calls such as, for example, relatively low-level routines that map MPI calls that are commonly used in other languages (such as C and Fortran), so that such calls can be available directly from the Mathematica user interface 204." ’768 patent, 13:10–14.
  - "In some embodiments, computer cluster 100 includes a user interface module 202, such as, for example a Mathematica Front End or a command line interface, that includes program code for a kernel module 206 to provide graphical output, accept graphical

input, and provide other methods of user communication that a graphical user interface or a command-line interface provides." '768 patent, 22:49–55.

  - "The user commands can be simple (for example, '1+1'), but can also be entire subroutines and sequences of code (such as, for example, Mathematica code), including calls to MPI from within the user interface module 202 (for example, the Mathematica Front End) to perform message passing between kernel modules 206 a-e (for example, Mathematica kernels)." '768 patent, 24:58–64.
- Cluster Node Modules
  - "One embodiment adds and supports Message Passing Interface ('MPI') calls directly from within a user interface, such as, for example, the Mathematica programming environment. In one embodiment, MPI calls are added to or made available from an interactive programming environment, such as the Mathematica Front End." '768 patent, 2:24–29.
  - "In one embodiment, the cluster node modules 204 a-e provide a way for many kernel modules 206 a-e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another. A cluster node module 204 can include at least a portion of an application programming interface ('API') known as the Message-Passing Interface ('MPI'), which is used in several supercomputer and cluster installations." '768 patent, 11:42–49.
  - "A cluster node module 204 can use an application-specific toolkit or interface such as, for example, Mathematica's MathLink, Add-Ons, or packets, to interact with an application. Normally used to connect a Mathematica kernel to a user interface known as the Mathematica Front End or other Mathematica kernels, MathLink is a bidirectional protocol to sends 'packets' containing messages, commands, or data between any of these entities." '768 patent, 11:55–62.
  - "In one embodiment, the MPI module 302 can include basic MPI calls such as, for example, relatively low-level routines that map MPI calls that are commonly used in other languages (such as C and Fortran), so that such calls can be available directly from the Mathematica user interface 204." '768 patent, 13:10–14.
- First Command to Create a First Packet Containing: An Expression to be Sent as Payload; and a Target Node Where the Expression Should be Sent
  - Asynchronous MPI Calls

    Asynchronous calls make it possible for the kernel to do work while communications are proceeding simultaneously. It is also possible that another node may not be able to send or receive data yet, allowing one kernel to continue working while waiting.

    TABLE C

Call	Description
mpiISend[expr, target, comm, tag, req]	Sends an expression expr to a processor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to mpiTest[req] until mpiTest[req] returns True.

    '768 patent, 13:41–55.
- Fourier Transform

-
  TABLE M

Call	Description
ParallelFourier[list]	Like Fourier[] except that list is a two- or three-dimensional list represented across the cluster, like for matrices, above. It returns the portion of the Fourier-transformed array meant for that processor.

- Special Purpose Microprocessor
  - "In addition, one or more of the processors 112 a-b can be a special purpose microprocessor such as a digital signal processor." '768 patent, 7:21–23.
- Cluster Configuration Module that Initializes Cluster Node Modules
  - "Cluster configuration module 208 includes program code for initializing a plurality of cluster node modules to add cluster computing support to computer systems 110, 120, 130. U.S. Pat. No. 7,136,924, issued to Dauger (the ''924 patent'), the entirety of which is hereby incorporated by reference and made a part of this specification, discloses a method and system for parallel operation and control of computer clusters." '768 patent, 22:26–33.
- Calls for Matrix Operations/Matrix Operations on Arrays of Data Elements
  -
    TABLE K

Call	Description
ParallelTranspose[matrix]	Like Transpose[] except that it transposes matrix that is in fact represented across the cluster, rather than on just one processor. It returns the portion of the transposed matrix meant for that processor.
ParallelProduct[matrix, vector]	Evaluates the product of matrix and vector, as it would on one processor, except that matrix is represented across the cluster.
ParallelDimensions[matrix]	Like Dimensions[] except that matrix is represented across the cluster, rather than on just one processor. It returns a list of each dimension.
ParallelTr[matrix]	Like Tr[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the trace of this matrix.
ParallelIdentity[rank]	Like Identity[], it generates a new identity matrix, except that the matrix is represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelOuter[f, vector1, vector2]	Like Outer[f, vector1, vector2] except that the answer becomes a matrix represented across the cluster, rather than on just one processor. It returns the portion of the new matrix for this processor.
ParallelInverse[matrix]	Like Inverse[] except that the matrix is represented across the cluster, rather than on just one processor. It returns the inverse of the matrix.

- Communications Networks
  - "The cluster computing system described herein generally includes one or more computer systems connected to one another via a communications network or networks. The communications network can include one or more of a local area network ('LAN'), a wide area network ('WAN'), an intranet, the Internet, etc. . . . Communications among the processors and such other devices may occur, for example, via one or more local buses of a computer system or via a LAN, a WAN, a storage area network ("SAN"),

and/or any other communications network capable of carrying signals among computer system components. . . . FIG. 1 is a block diagram of one embodiment of a computer cluster 100 wherein computer systems 110, 120, 130 communicate with one another via a communications network 102. Network 102 includes one or more of a LAN, a WAN, a wireless network, an intranet, or the Internet." '768 patent, 4:37–42, 50–56, 63–67.

  - "The communications network 102 may include one or more of any type of electronically connected group of computers including, for instance, the following networks: a virtual private network, a public Internet, a private Internet, a secure Internet, a private network, a public network, a value-added network, a wired network, a wireless network, an intranet, etc. In addition, the connectivity to the network can be, for example, a modem, Ethernet (IEEE 802.3), Gigabit Ethernet, 10-Gigabit Ethernet, Token Ring (IEEE 802.5), Fiber Distributed Datalink Interface (FDDI), Frame Relay, InfiniBand, Myrinet, Asynchronous Transfer Mode (ATM), or another interface." '768 patent, 10:58–11:2.

- First and Second Processors on Same Die
  - "The processors 112a-b can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® 20 processor, etc. . . . For example, if the processors 112a-b were each Core 2 Duo® processors having two processing cores, computer system 110 would have four nodes in all." '768 patent, 7:12–21,27–30.

- Communications Network Interface for Nodes to Communicate Using a Peer-to-Peer Architecture/Asynchronous Calls
  - "One embodiment adds and supports Message Passing Interface ('MPI') calls directly from within a user interface, such as, for example, the Mathematica programming environment. In one embodiment, MPI calls are added to or made available from an interactive programming environment, such as the Mathematica Front End." '768 patent, 2:24–29.
  - "In one embodiment, the cluster node modules 204 a-e provide a way for many kernel modules 206 a-e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another. A cluster node module 204 can include at least a portion of an application programming interface ('API') known as the Message-Passing Interface ('MPI'), which is used in several supercomputer and cluster installations." '768 patent, 11:42–49.
  - "In one embodiment, the MPI module 302 can include basic MPI calls such as, for example, relatively low-level routines that map MPI calls that are commonly used in other languages (such as C and Fortran), so that such calls can be available directly from the Mathematica user interface 204. In some embodiments, basic MPI calls include calls that send data, equations, formulas, and/or other expressions." '768 patent, 13:10–16.

- TABLE B

Call	Description
mpiSend[expr, target, comm, tag]	Sends an expression expr to a node with the ID target in the communicator would comm, waiting until that expression has left this kernel
mpiRecv [expr, target, comm, tag]	Receives an expression into expr from a node with the ID target in the communicator world comm, waiting until the expression has arrived
mpiSendRecv[sendexpr, dest, recvexpr, source, comm]	Simultaneously sends the expression sendexpr to the node with the ID target and receives an expression into recvexpr from the node with the ID source in the communicator world comm, waiting until both operations have returned.

- Asynchronous MPI Calls

  Asynchronous calls make it possible for the kernel to do work while communications are proceeding simultaneously. It is also possible that another node may not be able to send or receive data yet, allowing one kernel to continue working while waiting.

  TABLE C

Call	Description
mpiISend[expr, target, comm, tag, req]	Sends an expression expr to a processor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to mpiTest[req] until mpiTest[req] returns True.
mpiIRecv[expr, target, comm, tag, req]	Receives an expression expr from a proessor with the ID target in the communicator world comm, returning immediately. It can be balanced with calls to mpiTest[req] until mpiTest[req] returns True. The expr is not safe to access until mpiTest[req] returns True.
mpiTest[req]	Completes asynchronous behavior of mpiISend and mpiIRecv
mpWait[req]	Calls mpiTest until it returns True.
mpiWaitall[reglist]	Calls mpiWait all on every element of reqlist
mpiWaitany[reqlist]	Calls mpiTest on each element of realist until one of them returns True

  ’768 patent, 13:41–65.
- “Peer-to-Peer MPI[:] The mpiSend and mpiRecv commands make possible basic message passing, but one needs to define which processor to target. The following defines a new variable, targetProc, so that each pair of processors will point to each other.” ’768 patent, 26:65–27:3.

In this example, the even processors target its "right" processor, while the odd ones point its "left." For example, if the processors were lined up in a row and numbered in order, every even-numbered processor would pair with the processor following it in the line, and every odd-numbered processor would pair with the processor preceding it. Then a message can be sent:

```
In[4]:= If [ 1==Mod[ $IdProc , 2],mpiSend[N[Pi,22],targetProc,
    mpiCommWorld,d], mpiRecv[a,targetProc,mpiCommWorld,d]]
```

The If [ ] statement causes the processors to evaluate different code: the odd processor sends 22 digits of Pi, while the even receives that message. Note that these MPI calls return nothing. The received message is in the variable a:

```
In[5]:= a
Out[5]:= 3.1415926535897932384626
In[6]:= Clear[a]
```

The variable a on the odd processors would have no definition. Moreover, if $NProc is 8, processor 3 sent Pi to processor 2, processor 5 sent Pi to processor 4, and so on. These messages were not sent through processor 0, but they communicated on their own.

- '768 patent, 27:5–37.
- "The communications network can include one or more of a local area network ('LAN'), a wide area network ('WAN'), an intranet, the Internet, etc. . . . Communications among the processors and such other devices may occur, for example, via one or more local buses of a computer system or via a LAN, a WAN, a storage area network ("SAN"), and/or any other communications net- work capable of carrying signals among computer system components. . . . Network 102 includes one or more of a LAN, a WAN, a wireless network, an intranet, or the Internet." '768 patent, 4:39–42, 50–56, 66–67.
- "The communications network 102 may include one or more of any type of electronically connected group of computers including, for instance, the following networks: a virtual private network, a public Internet, a private Internet, a secure Internet, a private network, a public network, a value-added network, a wired network, a wireless network, an intranet, etc. In addition, the connectivity to the network can be, for example, a modem, Ethernet (IEEE 802.3), Gigabit Ethernet, 10-Gigabit Ethernet, Token Ring (IEEE 802.5), Fiber Distributed Datalink Interface (FDDI), Frame Relay, InfiniBand, Myrinet, Asynchronous Transfer Mode (ATM), or another interface." '768 patent, 10:58–11:2.
- "The network interface module can use a variety of network protocols. In one embodiment, the network interface module includes TCP/IP." '768 patent, 7:38–40.
- "A network of connections (for example, the arrows shown in FIG. 2) between the cluster node modules 204a-e can be implemented using a communications network 102, such as,

for example, TCP/IP over Ethernet, but the connections could also occur over any other type of network or local computer bus." '768 patent, 11:49–54.

III. ANTICIPATION AND OBVIOUSNESS UNDER 35 U.S.C. §§ 102 AND 103

Intel contends that all the Asserted Claims are invalid as anticipated under 35 U.S.C. § 102 and/or obvious under 35 U.S.C. § 103. Intel relies on and incorporates all prior art references cited on the cover of the Asserted Patents and in their prosecution histories, including for supporting Intel's arguments regarding the obviousness of the Asserted Claims. Intel further relies on and incorporates by reference, as if originally set forth herein, all invalidity positions, and all associated prior art and claim charts, disclosed to Plaintiff in the Other Plaintiff Cases or by potential or actual licensees to any of the Asserted Claims. Intel hereby discloses and identifies as if originally set forth herein, all prior art references listed and/or asserted in the above as invalidating prior art against each of the Asserted Claims. In addition, Intel identifies prior art patents, publications, products, and systems that disclose the elements of the Asserted Claims explicitly, inherently, or as part of an obvious combination, and also may be relied on to show the state of the art in the relevant timeframes and/or reasons or motivations to combine prior art references.

In those instances in which Intel asserts that an Asserted Claim is invalid under 35 U.S.C. § 112, Intel has applied the prior art in accordance with its assumption that Plaintiff contends such Asserted Claim (1) has sufficient supporting written description, (2) is enabled, (3) is not indefinite, and (4) sets forth what the inventor(s) regarded as the invention. Intel's Invalidity Contentions should not be construed as an admission that any Asserted Claim meets the requirements of § 112.

A. Intel's Claim Chart Exhibits

Intel contends that the prior art references listed in Tables 1–3 above anticipate and/or render obvious each of the Asserted Claims of those patents under the proper construction of those

claims and/or under Plaintiff's apparent interpretation of the claims as set forth in its First Amended Complaint and Infringement Contentions.

In addition, Intel has prepared Claim Chart Exhibits of prior art references that render the Asserted Claims anticipated and/or obvious under the proper construction of those claims and/or under Plaintiff's apparent interpretation of the claims as set forth in its First Amended Complaint and Preliminary Infringement Contentions. Charts identifying where each element of each Asserted Claim is found in each of these prior art references are attached as Exhibits A-1 to A-20 for the '768 patent, Exhibits B-1 to B-21 for the '621 patent, Exhibits C-1 to C-20 for the '034 patent, Exhibits D-1 to D-21 for the '582 patent, and Exhibits E-1 to E-20 for the '679 patent(collectively, the "Claim Chart Exhibits").

Claim Charts for the '768 Patent

Exhibit	Primary Reference and/or System Prior Art for the '768 Patent
A-1	PVMaple
A-2	Distributed Maple
A-3	Matlab*P
A-4	Fastest Fourier Transform in the West with Message-Passing Interface ("FFTW")
A-5	Intel Paragon Maple
A-6	Stony Brook GPU Cluster for High Performance Computing ("Stony Brook")
A-7	AppleSeed-Pooch System ("AppleSeed-Pooch")
A-8	MultiMATLAB
A-9	Microsoft Windows Compute Cluster Server 2003 System ("CCS")
A-10	Supercomputing Engine for Mathematica System ("SEM")
A-11	Brook and Merrimac System ("Merrimac")
A-12	Whitney Commodity Computing Testbed ("Whitney")
A-13	MatlabMPI from MIT ("MatlabMPI")
A-14	Abdennadher, N., & Boesch, R. (2005). A large scale distributed platform for high performance computing. In H. Zhuge & G. C. Fox (Eds.), Grid and Cooperative Computing - GCC 2005 (Vol. 3795, pp. 926–931). Springer. ("Abdennadher")
A-15	pyMPI
A-16	IBM Blue Gene
A-17	U.S. Patent No. 8,726,278 ("Shawver")
A-18	"High Performance Computing: An Introduction to Parallel Programming With Beowulf," Forrest M. Hoffman & William W. Hargrove, https://web.archive.org/web/20030418102558/http:/climate.ornl.gov/~forrest/osdj-2000-11/ ("Hoffman")

Exhibit	Primary Reference and/or System Prior Art for the ’768 Patent
A-19	U.S. Patent No. 8,065,503 (“Jia 503”)
A-20	Omnibus obviousness chart

Claim Charts for the ’621 Patent

Exhibit	Primary Reference and/or System Prior Art for the ’621 Patent
B-1	PVMaple
B-2	Distributed Maple
B-3	Matlab*P
B-4	FFTW
B-5	Intel Paragon Maple
B-6	Stony Brook
B-7	AppleSeed-Pooch
B-8	MultiMATLAB
B-9	CCS
B-10	SEM
B-11	Merrimac
B-12	Whitney
B-13	MatlabMPI
B-14	Abdennadher
B-15	pyMPI
B-16	IBM Blue Gene
B-17	U.S. Patent No. 8,533,402 (“Moler 402”)
B-18	International Application WO 2006/050404 A1 (“Husbands 404”)
B-19	Peer-to-Peer Support for MATLAB-Style Computing, Thesis by Rajeev Agrawal, Texas A&M University, 2004 (“Agrawal”)
B-20	U.S. Patent No. 5,471,622 (“Eadline”)
B-21	Omnibus obviousness chart

Claim Charts for the ’034 Patent

Exhibit	Primary Reference and/or System Prior Art for the ’034 Patent
C-1	PVMaple
C-2	Distributed Maple
C-3	Matlab*P
C-4	FFTW
C-5	Intel Paragon Maple
C-6	Stony Brook
C-7	AppleSeed-Pooch
C-8	MultiMATLAB
C-9	CCS
C-10	SEM
C-11	Merrimac
C-12	Whitney

Exhibit	Primary Reference and/or System Prior Art for the ’034 Patent
C-13	MatlabMPI
C-14	Abdennadher
C-15	pyMPI
C-16	IBM Blue Gene
C-17	Shawver
C-18	Hoffman
C-19	Jia 503
C-20	Omnibus obviousness chart

Claim Charts for the ’582 Patent

Exhibit	Primary Reference and/or System Prior Art for the ’582 Patent
D-1	PVMaple
D-2	Distributed Maple
D-3	Matlab*P
D-4	FFTW
D-5	Intel Paragon Maple
D-6	Stony Brook
D-7	AppleSeed-Pooch
D-8	MultiMATLAB
D-9	CCS
D-10	SEM
D-11	Merrimac
D-12	Whitney
D-13	MatlabMPI
D-14	Abdennadher
D-15	pyMPI
D-16	IBM Blue Gene
D-17	Moler 402
D-18	International Application WO 2006/050404 A1 (“Husbands”)
D-19	Peer-to-Peer Support for MATLAB-Style Computing, Thesis by Rajeev Agrawal, Texas A&M University, 2004 (“Agrawal”)
D-20	U.S. Patent No. 5,471,622 (“Eadline”)
D-21	Omnibus obviousness chart

Claim Charts for the ’679 Patent

Exhibit	Primary Reference and/or System Prior Art for the ’679 Patent
E-1	PVMaple
E-2	Distributed Maple
E-3	Matlab*P
E-4	FFTW
E-5	Intel Paragon Maple
E-6	Stony Brook

Exhibit	Primary Reference and/or System Prior Art for the ’679 Patent
E-7	AppleSeed-Pooch
E-8	MultiMATLAB
E-9	CCS
E-10	SEM
E-11	Merrimac
E-12	Whitney
E-13	MatlabMPI
E-14	Abdennadher
E-15	pyMPI
E-16	IBM Blue Gene
E-17	Shawver
E-18	Hoffman
E-19	Jia 503
E-20	Omnibus obviousness chart

Intel’s claim charts provide exemplary citations to the prior art references that teach or suggest elements of the Asserted Claims of the Asserted Patent. The references discussed in the Claim Chart Exhibits may disclose the elements of the Asserted Claims explicitly or inherently, or they may be relied upon to show the state of the art in the relevant time frame, including elements that were well-understood, routine, or conventional. To the extent that an element of an Asserted Claim is not shown in a chart, the Asserted Claims would have been obvious based on a combination of one or more other prior art references, as set forth below and in Exhibits A–E.

Moreover, nothing in the Claim Chart Exhibits should be construed as an admission that the identified references do not disclose any claim limitation. The identified references may contain additional disclosures for a particular claim limitation, and Intel reserves all rights to rely on other portions of the identified references to support its contentions. Persons of ordinary skill in the art generally read a prior art reference as a whole and in the context of other publications and literature. Intel may rely on uncited portions of the prior art references, other prior art references incorporated into such references, other publications, and percipient and/or expert testimony to provide context and as aids to understanding and interpreting the portions of the cited

prior art references. Disclosures relating to elements of dependent claims are disclosed in connection with the claims from which they depend, even though they are not repeated for each dependent claim.

Intel may also rely on uncited portions of the prior art references, other publications, and expert testimony to establish that a person of ordinary skill in the art ("POSA") would have been motivated to modify or combine certain cited references and/or would have applied its or her general knowledge and common sense so as to render the claims obvious. Where Intel cites to a particular figure, table, chart, annex, or appendix in a prior art reference, the citation should be understood to encompass the caption and description of the figure, table, chart, annex, or appendix; any text relating to the figure, table, chart, annex, or appendix; and the figure, table, chart, annex, or appendix itself. Similarly, where a portion of text refers to a figure, chart, annex, appendix, or other part of a reference, the citation should be understood to include the figure, chart, annex, appendix, or other part of the reference as well.

The U.S. and foreign patents, publications, products and systems disclosed in the Claim Chart Exhibits qualify as prior art under various provisions of the Patent Act, including 35 U.S.C. §§ 102 and/or 103. Intel may also rely on persons identified as named inventors of the patents and authors of the publications cited in the Claim Chart Exhibits, as well as individuals identified in any of the parties' initial disclosures and discovery responses, and others as they become identified through further discovery, as prior inventors, as persons of ordinary skill in the art, or as persons having knowledge of, or having previously used, publicly disclosed, published, offered for sale, or sold, the alleged inventions claimed in the Asserted Patents. Discovery is ongoing, and Intel continues to investigate and analyze the prior art. Intel reserves the right to supplement and/or amend these contentions based on the results of such investigation and analysis. Intel further

reserves the right to rely on the system described in any prior art publication(s) cited herein in addition to, or in place of, relying on the publications themselves.

Some of the disclosures cited in certain of the Claim Chart Exhibits are based on Intel's understanding of how Plaintiff is interpreting the Asserted Claims for the purpose of its Infringement Contentions (to the extent Plaintiff's contentions can be understood). Intel does not intend to and does not adopt Plaintiff's claim interpretations or concede that Plaintiff's claim interpretations are correct. Nothing stated herein or in any Claim Chart Exhibit should be treated as an admission or suggestion that Intel agrees with Plaintiff regarding either the scope of any of the Asserted Claims, or any express or implied claim constructions advanced by Plaintiff in its Infringement Contentions or anywhere else, or that any of Intel's accused instrumentalities meet any limitation of the Asserted Claims. Nothing stated herein or in any Claim Chart Exhibit should be construed as an admission or waiver of any particular construction of any claim term. Nothing stated herein or in any Claim Chart Exhibit should be construed as a waiver of any argument available under 35 U.S.C. § 112. Intel reserves all rights to challenge any of the Asserted Claims under 35 U.S.C. § 112, including by arguing that they are not supported by the written description, not enabled, indefinite, and/or do not set forth what the inventors regarded as the invention, notwithstanding the disclosure in any Claim Chart Exhibit corresponding to that claim. *See* Section VI (Invalidity Under 35 U.S.C. § 112), *infra.* Nothing stated herein or in any Claim Chart Exhibit should be construed as an admission or waiver with respect to the effective filing date of any of the Asserted Claims.

B. Exemplary Obviousness Combinations

The following charts provide exemplary combinations of prior art that renders the Asserted Claims obvious. However, all the Asserted Claims of the Asserted Patents are invalid as

anticipated under 35 U.S.C. § 102 in view of each of the prior art references identified herein as specified in the Claim Chart Exhibits, which identify specific examples of where each limitation of the Asserted Claims is found in the prior art reference, and/or obvious under 35 U.S.C. § 103 in view of the state of the prior art alone, the admitted prior art described in the Invalidity Contentions, and/or in combination with the references and disclosures detailed in Tables 1–3 of the Invalidity Contentions as well as the prior art references identified herein as specified in the Claim Chart Exhibits. As such, Intel's inclusion of exemplary prior art references and combinations listed below does not preclude Intel from identifying other invalidating combinations as appropriate.

1. Exemplary Obviousness Combinations for the '768 Patent

Exemplary Obviousness Combinations for the '768 Patent
PVMaple alone or in combination with one or more of the references in Exhibits A-2–A-20.
Distributed Maple alone or in combination with one or more of the references in Exhibits A-1 and A-3–A-20.
Matlab*P alone or in combination with one or more of the references in Exhibits A-1–A-2 and A-4–A-20.
FFTW alone or in combination with one or more of the references in Exhibits A-1–A-3 and A-5–A-20.
Intel Paragon Maple alone or in combination with one or more of the references in Exhibits A-1–A-4 and A-6–A-20.
Stony Brook alone or in combination with one or more of the references in Exhibits A-1–A-5 and A-7–A-20.
AppleSeed Pooch alone or in combination with one or more of the references in Exhibits A-1–A-6 and A-8–A-20.
MultiMATLAB alone or in combination with one or more of the references in Exhibits A-1–A-7 and A-9–A-20.
CCS alone or in combination with one or more of the references in Exhibits A-1–A-8 and A-10–A-20.
SEM alone or in combination with one or more of the references in Exhibits A-1–A-9 and A-11–A-20.
Merrimac alone or in combination with one or more of the references in Exhibits A-1–A-10 and A-12–A-20.
Whitney alone or in combination with one or more of the references in Exhibits A-1–A-11 and A-13–A-20.
MatlabMPI alone or in combination with one or more of the references in Exhibits A-1–A-12 and A-14–A-20.

Exemplary Obviousness Combinations for the ’768 Patent
Abdennadher alone or in combination with one or more of the references in Exhibits A-1–A-13 and A-15–A-20.
pyMPI alone or in combination with one or more of the references in Exhibits A-1–A-14 and and A-16–A-20.
IBM Blue Gene alone or in combination with one or more of the references in Exhibits A-1–A-15 and A-17–A-20.
Shawver alone or in combination with one or more of the references in Exhibits A-1–A-16 and A-18–A-20.
Hoffman alone or in combination with one or more of the references in Exhibits A-1–A-17 and A-19–A-20.
Jia 503 alone or in combination with one or more of the references in Exhibits A-1–A-18 and A-20.

2. Exemplary Obviousness Combinations for the ’621 Patent

Exemplary Obviousness Combinations for the ’621 Patent
PVMaple alone or in combination with one or more of the references in Exhibits B-2–B-21.
Distributed Maple alone or in combination with one or more of the references in Exhibits B-1 and B-3–B-21.
Matlab*P alone or in combination with one or more of the references in Exhibits B-1–B-2 and B-4–B-21.
FFTW alone or in combination with one or more of the references in Exhibits B-1–B-3 and B-5–B-21.
Intel Paragon Maple alone or in combination with one or more of the references in Exhibits B-1–B-4 and B-6–B-21.
Stony Brook alone or in combination with one or more of the references in Exhibits B-1–B-5 and B-7–B-21.
AppleSeed Pooch alone or in combination with one or more of the references in Exhibits B-1–B-6 and B-8–B-21.
MultiMATLAB alone or in combination with one or more of the references in Exhibits B-1–B-7 and B-9–B-21.
CCS alone or in combination with one or more of the references in Exhibits B-1–B-8 and B-10–B-21.
SEM alone or in combination with one or more of the references in Exhibits B-1–B-9 and B-11–B-21.
Merrimac alone or in combination with one or more of the references in Exhibits B-1–B-10 and B-12–B-21.
Whitney alone or in combination with one or more of the references in Exhibits B-1–B-11 and B-13–B-21.
MatlabMPI alone or in combination with one or more of the references in Exhibits B-1–B-12 and B-14–B-21.
Abdennadher alone or in combination with one or more of the references in Exhibits B-1–B-13 and B-15–B-21.

Exemplary Obviousness Combinations for the ’621 Patent
pyMPI alone or in combination with one or more of the references in Exhibits B-1–B-14 and B-16–B-21.
IBM Blue Gene alone or in combination with one or more of the references in Exhibits B-1–B-15 and B-17–B-21.
Moler 402 alone or in combination with one or more of the references in Exhibits B-1–B-16 and B-18–B-21.
Husbands alone or in combination with one or more of the references in Exhibits B-1–B-17 and B-19–B-21.
Eadline alone or in combination with one or more of the references in Exhibits B-1–B-18 and B-20–B-21.
Agarwal alone or in combination with one or more of the references in Exhibits B-1–B-19 and B-21.

3. Exemplary Obviousness Combinations for the ’034 Patent

Exemplary Obviousness Combinations for the ’034 Patent
PVMaple alone or in combination with one or more of the references in Exhibits C-2–C-20.
Distributed Maple alone or in combination with one or more of the references in Exhibits C-1 and C-3–C-20.
Matlab*P alone or in combination with one or more of the references in Exhibits C-1–C-2 and C-4–C-20.
FFTW alone or in combination with one or more of the references in Exhibits C-1–C-3 and C-5–C-20.
Intel Paragon Maple alone or in combination with one or more of the references in Exhibits C-1–C-4 and C-6–C-20.
Stony Brook alone or in combination with one or more of the references in Exhibits C-1–C-5 and C-7–C-20.
AppleSeed Pooch alone or in combination with one or more of the references in Exhibits C-1–C-6 and C-8–C-20.
MultiMATLAB alone or in combination with one or more of the references in Exhibits C-1–C-7 and C-9–C-20.
CCS alone or in combination with one or more of the references in Exhibits C-1–C-8 and C-10–C-20.
SEM alone or in combination with one or more of the references in Exhibits C-1–C-9 and C-11–C-20.
Merrimac alone or in combination with one or more of the references in Exhibits C-1–C-10 and C-12–C-20.
Whitney alone or in combination with one or more of the references in Exhibits C-1–C-11 and C-13–C-20.
MatlabMPI alone or in combination with one or more of the references in Exhibits C-1–C-12 and C-14–C-20.
Abdennadher alone or in combination with one or more of the references in Exhibits C-1–C-13 and C-15–C-20.

Exemplary Obviousness Combinations for the ’034 Patent
pyMPI alone or in combination with one or more of the references in Exhibits C-1–C-14 and C-16–C-20.
IBM Blue Gene alone or in combination with one or more of the references in Exhibits C-1–C-15 and C-17–C-20.
Shawver alone or in combination with one or more of the references in Exhibits C-1–C-16 and C-18–C-20.
Hoffman alone or in combination with one or more of the references in Exhibits C-1–C-17 and C-19–C-20.
Jia 503 alone or in combination with one or more of the references in Exhibits C-1–C-18 and C-20.

4. Exemplary Obviousness Combinations for the ’582 Patent

Exemplary Obviousness Combinations for the ’582 Patent
PVMaple alone or in combination with one or more of the references in Exhibits D-2–D-21.
Distributed Maple alone or in combination with one or more of the references in Exhibits D-1 and D-3–D-21.
Matlab*P alone or in combination with one or more of the references in Exhibits D-1–D-2 and D-4–D-21.
FFTW alone or in combination with one or more of the references in Exhibits D-1–D-3 and D-5–D-21.
Intel Paragon Maple alone or in combination with one or more of the references in Exhibits D-1–D-4 and D-6–D-21.
Stony Brook alone or in combination with one or more of the references in Exhibits D-1–D-5 and D-7–D-21.
AppleSeed Pooch alone or in combination with one or more of the references in Exhibits D-1–D-6 and D-8–D-21.
MultiMATLAB alone or in combination with one or more of the references in Exhibits D-1–D-7 and D-9–D-21.
CCS alone or in combination with one or more of the references in Exhibits D-1–D-8 and D-10–D-21.
SEM alone or in combination with one or more of the references in Exhibits D-1–D-9 and D-11–D-21.
Merrimac alone or in combination with one or more of the references in Exhibits D-1–D-10 and D-12–D-21.
Whitney alone or in combination with one or more of the references in Exhibits D-1–D-11 and D-13–D-21.
MatlabMPI alone or in combination with one or more of the references in Exhibits D-1–D-12 and D-14–D-21.
Abdennadher alone or in combination with one or more of the references in Exhibits D-1–D-13 and D-15–D-21.
pyMPI alone or in combination with one or more of the references in Exhibits D-1–D-14 and D-16–D-21.

Exemplary Obviousness Combinations for the ’582 Patent
IBM Blue Gene alone or in combination with one or more of the references in Exhibits D-1–D-15 and D-17–D-21.
Moler 402 alone or in combination with one or more of the references in Exhibits D-1–D-16 and D-18–D-21.
Husbands alone or in combination with one or more of the references in Exhibits D-1–D-17 and D-19–D-21.
Eadline alone or in combination with one or more of the references in Exhibits D-1–D-18 and D-20–D-21.
Agarwal alone or in combination with one or more of the references in Exhibits D-1–D-19 and D-21.

5. Exemplary Obviousness Combinations for the ’679 Patent

Exemplary Obviousness Combinations for the ’679 Patent
PVMaple alone or in combination with one or more of the references in Exhibits E-2–E-20.
Distributed Maple alone or in combination with one or more of the references in Exhibits E-1 and E-3–E-20.
Matlab*P alone or in combination with one or more of the references in Exhibits E-1–E-2 and E-4–E-20.
FFTW alone or in combination with one or more of the references in Exhibits E-1–E-3 and E-5–E-20.
Intel Paragon Maple alone or in combination with one or more of the references in Exhibits E-1–E-4 and E-6–E-20.
Stony Brook alone or in combination with one or more of the references in Exhibits E-1–E-5 and E-7–E-20.
AppleSeed Pooch alone or in combination with one or more of the references in Exhibits E-1–E-6 and E-8–E-20.
MultiMATLAB alone or in combination with one or more of the references in Exhibits E-1–E-7 and E-9–E-20.
CCS alone or in combination with one or more of the references in Exhibits E-1–E-8 and E-10–E-20.
SEM alone or in combination with one or more of the references in Exhibits E-1–E-9 and E-11–E-20.
Merrimac alone or in combination with one or more of the references in Exhibits E-1–E-10 and E-12–E-20.
Whitney alone or in combination with one or more of the references in Exhibits E-1–E-11 and E-13–E-20.
MatlabMPI alone or in combination with one or more of the references in Exhibits E-1–E-12 and E-14–E-20.
Abdennadher alone or in combination with one or more of the references in Exhibits E-1–E-13 and E-15–E-20.
pyMPI alone or in combination with one or more of the references in Exhibits E-1–E-14 and E-16–E-20.

Exemplary Obviousness Combinations for the ’679 Patent
IBM Blue Gene alone or in combination with one or more of the references in Exhibits E-1–E-15 and E-17–E-20.
Shawver alone or in combination with one or more of the references in Exhibits E-1–E-16 and E-18–E-20.
Hoffman alone or in combination with one or more of the references in Exhibits E-1–E-17 and E-19–E-20.
Jia 503 alone or in combination with one or more of the references in Exhibits E-1–E-18 and E-20.

C. Motivations to Combine

To the extent a finder of fact finds that any primary prior art reference does not disclose one or more limitations of an asserted claim, the asserted claim is nevertheless obvious because the allegedly missing limitations contain nothing beyond ordinary improvements. In other words, the asserted claim combines known elements to achieve predictable results or chooses between clear alternatives known to those of skill in the art, particularly in view of the state of the art as reflected in the relevant prior art.

Moreover, as explained above, it would have been obvious to a person of skill in the art at the time of the alleged invention of the Asserted Claims to combine any primary reference with any combination of other primary references or secondary references so as to practice the Asserted Claims. To the extent that Plaintiff argues that any concept claimed in the Asserted Claims is not disclosed in a primary reference, it would, at a minimum, have been obvious to adapt the primary reference to include the concept or combine it with other primary references or secondary references that disclose the concept. Each concept described and claimed in the Asserted Patents was known to those of skill in the art as available design choices for parallelizing or distributing mathematical evaluations or other processing tasks across computers or nodes in a cluster.

The Supreme Court has held that “[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results.” *KSR*

Int'l Co. v. Teleflex Inc., 550 U.S. 398, 416 (2007). "When a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one." *Id.* at 417. As the Supreme Court made clear, "[f]or the same reason, if a technique has been used to improve one device, and a POSA would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill." *Id.*

To determine whether there is an apparent reason to combine the known elements in the fashion claimed by the patent at issue, a court can "look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art." *Id.* at 418. For example, obviousness can be demonstrated by showing "there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims." *Id.* at 420. "[A]ny need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id.* Common sense also teaches that "familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle." *Id.*

The Supreme Court in *KSR* held that a claimed invention can be obvious even if there is no explicit teaching, suggestion, or motivation for combining the prior art to produce that invention. In summary, *KSR* holds that patents that are based on new combinations of elements or components already known in a technical field may be found to be obvious. *See, generally*, *KSR*, 127 S.Ct. 1727. Specifically, the Court in *KSR* rejected a rigid application of the "teaching, suggestion, or motivation [to combine]" test. *Id.* at 1741. "In determining whether the subject

matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. What matters is the objective reach of the claim." *Id.* at 1741–42. "Under the correct analysis, any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id.* at 1742. A key inquiry is whether the "improvement is more than the predictable use of prior art elements according to their established functions." *Id.* at 1740.

The rationale to combine or modify prior art references is significantly stronger when, as here, the references seek to solve the same problem, come from the same field, and correspond well to each other. *In re Inland Steel Co*., 265 F.3d 1354, 1362 (Fed. Cir. 2001). The Federal Circuit has held that two references may be combined as invalidating art under similar circumstances, namely "[the prior art] focus[es] on the same problem that the . . . patent addresses: enhancing the magnetic properties of . . . steel. Moreover, both [prior art references] come from the same field. Finally, the solutions to the identified problems found in the two references correspond well." *Id.* at 1364 (concerning patents and prior art relating to improving the magnetic and electrical properties of steel).

In view of the Supreme Court's *KSR* decision, the PTO issued a set of Examination Guidelines. Examination Guidelines for Determining Obviousness Under 35 U.S.C. §103 in view of the Supreme Court Decision in *KSR International Co. v. Teleflex, Inc*., 72 Fed. Reg. 57526 (October 10, 2007). Those Guidelines summarized the *KSR* decision and identified various rationales for finding a claim obvious, including those based on other precedents. Those rationales include:

(A) Combining prior art elements according to known methods to yield predictable results;

(B) Simple substitution of one known element for another to obtain predictable results;

(C) Use of known technique to improve similar devices (methods, or products) in the same way;

(D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;

(E) "Obvious to try" – choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;

(F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art;

(G) Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

Id. at 57529. The above rationales likewise apply in rendering obvious the Asserted Claims of the Asserted Patents.

Further, the Federal Circuit has held that a motivation to combine references may be implicit for patents based on improvements:

> [A]n implicit motivation to combine exists not only when a suggestion may be gleaned from the prior art as a whole, but when the "improvement" is technology-independent and the combination of references results in a product or process that is more desirable, for example because it is stronger, cheaper, cleaner, faster, lighter, smaller, more durable, or more efficient. Because the desire to enhance commercial opportunities by improving a product or process is universal—and even common-sensical—we have held that there exists in these situations a motivation to combine prior art references even absent any hint of suggestion in the references themselves. In such situations, the proper question is whether the ordinary artisan possesses knowledge and skills rendering him capable of combining the prior art references.

DyStar Textilfarben GmbH & Co. Deutschland KG v. C.H. Patrick Co., 464 F.3d 1356, 1368 (Fed. Cir. 2006).

Additional evidence establishing that there would have been a motivation to modify and/or combine the prior art references identified above and herein includes the interrelated teachings of multiple prior art references; the effects of demands known to the design community or present in

the marketplace; the existence of a known problem for which there was an obvious solution encompassed by the Asserted Claims; the existence of a known need or problem in the field of the endeavor at the time of the invention; and the background knowledge that would have been possessed by a person having ordinary skill in the art.

Furthermore, the prior art references identified in these contentions explicitly or implicitly refer to other prior art references, share common authors or inventors, were published in the same journals, presented at the same conferences or meetings, submitted or proposed to standards development study groups or task groups with overlapping contributors, members, or attendees, and/or were developed at common companies, schools, or organizations, all of which would motivate one of skill in the art to combine them. These references are also within the field(s) of the Asserted Patents and are directed to similar subject matter or problems within the field(s). Additionally, any products, devices, or processes described in the references existed and/or were invented before or during the period in which the claimed inventions were developed, providing further motivation to combine them.

Thus, the motivations to combine the teachings of the prior art references disclosed herein are found in the references themselves and: (1) the nature of the problem being solved, (2) the express, implied, and inherent teachings of the prior art, (3) the knowledge of persons of ordinary skill in the art, (4) the fact that the prior art is generally directed toward similar subject matter, problems, or goals, and (5) the predictable results obtained in combining the different elements of the prior art—e.g., using known techniques for parallelizing or distributing mathematical evaluations or other processing tasks across computers or nodes in a cluster.

As an example of those reasons and motivations to combine, each cited prior art reference generally relates to parallel or cluster computing and thus constitutes analogous art within the same

field of endeavor. The prior art references depict, disclose, and discuss similar components, techniques, and systems for initializing nodes, communicating between and among nodes, and interacting with a user interface, as claimed in the Asserted Patents. Thus, a POSA would have understood the teachings of the references to be applicable to one another. A POSA also would have found it obvious to implement (i.e., obvious to try) such combinations to utilize these well-known parallel processing and communication techniques in computer clusters. *Id.* A POSA would have been motivated to make such combinations based on, for example, the below. The below list of motivations to combine is exemplary and representative, and is not an exhaustive list of motivations to combine, nor potential combinations.

For example, a POSA would look to the primary and secondary references discussed above to improve or tailor the disclosures thereof to help add cluster-computing functionality to computer programs, to improve throughput, reduce redundancy, and achieve benefits based on the well-known "advantages that computer clusters can offer." '768 patent, 1:22–25. A POSA would have understood and been aware of motivations to distribute processing tasks across nodes, including for programs designed to execute user instructions on a single node, using known methods. Accordingly, a POSA would seek to combine or modify the disclosure of any given primary and secondary references to achieve those goals, and would have readily understood that doing so could increase program performance and/or reliability, improve user interactions and/or satisfaction, and reduce cost. *See also* Exs. A–E and Tables 1–3 of these contentions.

One of skill in the art would also have been motivated to combine the different publications and patents that were authored by employees of a given company or assigned to the same assignee and/or related to the same subject matter. Additionally, one of skill in the art would have been motivated to combine different references that were authored, developed, or invented by the same

individual(s) related to the same subject matter. The common inventors/authors/architects of the references demonstrate that they relate to continued work in a common field of effort and continued related developments in that field. One of skill in the art would, therefore, be motivated to combine the references. Additionally, based on the teachings of the references and/or the knowledge of one of ordinary skill, a POSA would have been motivated to combine different references from the same company. For example, a POSA would have been motivated to combine prior art systems or products (e.g., AppleSeed, FFTW, MacMPI, MPI, Mathematica, MATLAB, MatlabMPI, Maple, Matlab*P, NumPy, Pooch, SEM, Whitney, etc.) with any related or applicable patent or non-patent documentation or literature relating to that system or owned by the same entity, including for the reason that these materials are related.

Further, below are additional motivations to combine prior art for particular claim limitations. The following discussions of specific claim limitations are merely examples and are not limiting. For example, where a POSA would have been motivated to combine references which together render obvious limitations from the independent claims, a POSA would have also been motivated to combine said references in such a way as to render obvious various asserted dependent claims. The motivations identified with respect to any one Asserted Patent apply with equal force to any of the other Asserted Patents by virtue of their relationship and similarities. Moreover, the background and state of the art discussed below with respect to any one Asserted Patent apply with equal force, where applicable, to any of the other Asserted Patents by virtue of their relationship and similarities.

D. Obviousness In Light of the State of the Art and Motivations to Combine

A POSA at the time of the claimed invention(s) had reason to modify each reference listed and charted in Intel's Claim Chart Exhibits or combine each reference with one or more of the

other references and disclosures detailed in Tables 1–3 of these Invalidity Contentions as well as the other prior art references charted in the Claim Chart Exhibits in light of the knowledge of a person of ordinary skill at the time, including information in the prior art cited in these Invalidity Contentions.

The following discussion of prior-art computer clustering and parallelization solutions provides further support for Intel's contentions. The motivations to combine discussed herein are exemplary and subject to further investigation and discovery, including third-party and expert discovery. Further rationale and support for motivations to combine various types of references can be found in at least the common specification of the Asserted Patents, their prosecution histories, the related European and Japanese patents claiming priority to the same underlying applications and their prosecution histories, and the prior art identified above and in Tables 1–3 of the Invalidity Contentions.

1. Background

Many computational problems are small enough to be solved on a single computer. Some, however, are not. These tasks may be so large that it would take an impractical amount of time and/or space on just one computer. One way to address this problem is to use a "supercomputer," which is a particularly powerful computer. *See* Buyya at xxi, 654. Another approach for handling such tasks is to coordinate multiple ordinary computers to act as a single, more powerful system, a computer cluster, that can practically execute large tasks. *See id.* at 5. This approach is sometimes called "cluster computing," and it was conventional and well-known before the earliest-claimed priority date of the Asserted Patents.

"A cluster is a type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers working together as a single, integrated

computing resource." Buyya at 5; *see also id.* at 6 ("A cluster generally refers to two or more computers (nodes) connected together."). "A computer node can be a single or multiprocessor system (PCs, workstations, or SMPs) with memory, I/O facilities, and an operating system." *Id.* at 5–6. "The cluster nodes can work collectively as an integrated computing resource, or they can operate as individual computers." *Id.* at 7. Typically, the plurality of computers in a cluster will be physically connected by a network or switch, and each computer will have "network interface hardware [that] acts as a communication processor and is responsible for transmitting and receiving packets of data between cluster nodes via a network/switch." *See id.* at 6–7. The computers will also have a layer of software called "cluster middleware," which "is responsible for offering an illusion of a unified system image (single system image) and availability out of a collection on independent but interconnected computers." *Id.*; *see also* Gulko 240 at [0010] ("Cluster middleware is used to interface between the server requesting distributed processing of an application and a computer cluster, to achieve distributed processing."). A cluster can run both sequential and parallel applications. *See* Buyya at 6. To facilitate the execution of the latter, the cluster will have various parallel programming environments and tools, such as PVM and MPI (which will be discussed below). *See id.* at 6–7 ("Communication software offers a means of fast and reliable data communication among cluster nodes and to the outside world."). The figure below illustrates the typical cluster architecture described above:

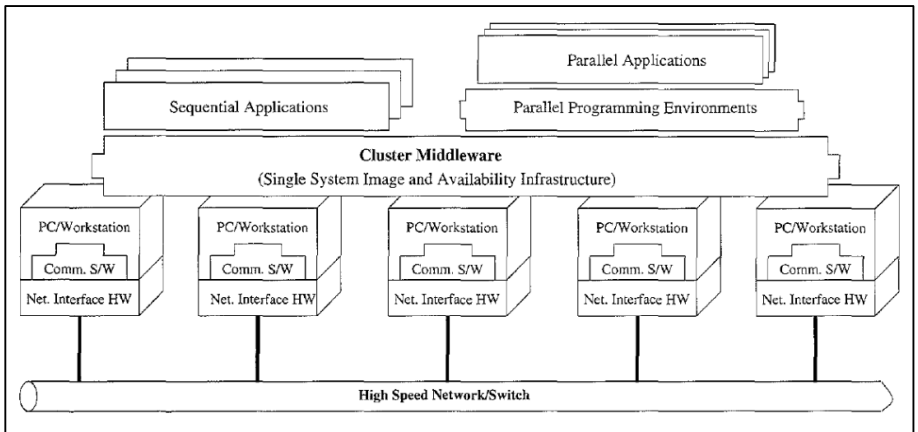


Figure 1.1 Cluster computer architecture.

Id. at 6.

The nodes in a cluster can communicate and coordinate with each other in a variety of different ways. One simple communication pattern is to designate one computer as a "master" and the rest as "slaves." *See id.* at 19. The master node distributes tasks to the slave nodes, which execute them and return the results. *Id.* In some cases, the slave nodes cannot directly communicate with each other—the sending slave node must send the message to the master node, which passes the message on to the receiving slave node. *See id.* at 19, 53–54. However, in contrast, other algorithms allow the slave nodes to communicate amongst themselves while computing results. *See id.* at 53–54. For example, "peer-to-peer computing" is an architecture implemented in software in which each node communicates tasks and data directly with another node without transmitting to a central server or master node. *See id.* Both communication patterns (master-slave and peer-to-peer) were conventional and well-known in the prior art before the earliest-claimed priority date of the Asserted Patents.

Even after a communication pattern has been selected, software still needs to be written to actually implement inter-node communication. Commonly, when the cluster follows a "distributed memory" model where each CPU can access only its own local memory, the various nodes will

use a "message passing" paradigm, where they communicate information with each other by sending and receiving messages. *See* Buyya at 13, 33, 39. Multiple such message passing libraries were implemented before the priority dates of the Asserted Patents, such that they were conventional and not an inventive feature of the Asserted Claims. One popular library is the Message Passing Interface ("MPI"). *Id.* at 13. Another is the Parallel Virtual Machine ("PVM"). *Id.* As early as 1999, these libraries were described as "the two most popular distributed memory message-passing environments that have become the standards of message-passing programming today." *Id.* at 2; *see also id.* at 48–86. These libraries typically include asynchronous calls, which allow the computers to keep working while awaiting the completion of the call, as opposed to having to block until the result of the call is returned to the calling computer. *See id.* at 59–60.

One major context in which cluster computing is often used is scientific computing, including computer algebra systems. In contrast to "[c]onventional mathematical programs" that "require numerical values for all variables," a computer algebra system has the "ability to manipulate information in a symbolic or algebraic manner," which includes "maintain[ing] and manipulat[ing] the underlying symbols and expressions." Maple Guide at 1. Various computer algebra programs, such as Mathematica, Maple, and MATLAB, were well-known and conventional in the prior art before the earliest-claimed priority date of the Asserted Patents. Some of these programs used interpreters ("kernels") to translate high-level user commands into low-level code executable by the processor. *See, e.g.*, *id.* at 88. Additionally, as discussed below, multiple efforts to parallelize these kernels had successfully been made before the earliest-claimed priority date of the Asserted Patents.

2. Development of Parallelized Computing

Cluster computing had its genesis in the 1960s at IBM. *See* Buyya at xxi. In the 1980s and 1990s, the idea gained traction over the competing notions of improving performance via better processors or expensive supercomputers. *See id.* at xxi, 4. This is at least because improvements in microprocessors, networks, and tools improved the performance of even "cheap and commodity components," such that "[c]lusters[] built using commodity-off-the-shelf (COTS) hardware components" were "an appealing vehicle for cost-effective parallel computing." *See id.* at xxi. Even in 1999, before the turn of the millennium, it was reported that "[s]calable computing clusters, ranging from a cluster of (homogeneous or heterogeneous) PCs or workstations, to SMPs, are rapidly becoming the standard platforms for high-performance and large-scale computing." *Id.* at 4.

Focusing on the more specific context of parallel scientific computing/computer algebra systems, the prior art included a plethora of software libraries that parallelize Maple, MATLAB, and Mathematica. *See* Petcu3 at 1–3. Starting with Maple, one early attempt at parallelization was the 1990 Sugarbush system, which used the Linda language to coordinate Maple kernels across multiple computers. *See* Schreiner1 at 307–08. Another similar system was ||Maple|| in 1993, which relied on the Strand language instead. *See id.* at 308. In 1996 and 1997, the Intel Paragon Maple system extended the Maple kernel to take advantage of the NX message passing library of the Intel Paragon supercomputer. *See id.*; Bernardin at 1. The 1998 FoxBox system used Maple as an interface to C++ and MPI parallel implementations of polynomial factorization algorithms. *See* Schreiner1 at 308. The Distributed Maple system, which started development at least by 1999, clustered ordinary Maple kernels on multiple machines through a Java scheduler and a Maple interface file read by each kernel to enable communication with the scheduler. *See* Petcu3 at 2–3,

8; Schreiner1 at 310–12. Distributed Maple was further extended to use Mathematica kernels instead. *See* Schreiner7 at 3–5. Finally, the PVMaple system, in development at least by 2000, was inspired by Distributed Maple to integrate the PVM message passing library with Maple. *See* Petcu2 at 216–17.

Mathematica and MATLAB also have prior art clustering systems. As alluded to above, by 2000 the Distributed Mathematica system had adapted Distributed Maple's design to use Mathematica kernels instead of Maple kernels. *See* Schreiner7. And the Asserted Patents admit that gridMathematica, which includes a Parallel Computing Toolkit, connected a plurality of Mathematica kernels. *See, e.g.*, Section II.C (Admitted Prior Art), *supra*; '768 patent, 1:44–62, 12:26–33; *see also* Petcu3 at 2–3; Dauger 4 at 82; SEM Poster; SEM Whitepaper at 9 ("That triggers the cluster to launch its MPI processes, which in turn launches *Mathematica* kernels.").

As for MATLAB, parallelization efforts date back to at least the 1992 Conlab system. *See* Petcu3 at 2, 8. One paper found at least 27 parallel MATLAB projects as of February 2005. Choy 2 at 332. One notable effort is the MultiMATLAB system, dating back at least as far as 1997, which uses MPI to connect MATLAB processes on multiple machines so that they can pass "commands and data" between each other and "solve course-grained [*sic*] large-scale problems using Matlab." *See* Petcu3 at 2–3; *see also, e.g.*, Menon. MultiMATLAB, "one of the most well known parallel MATLABs," "is a general extension of the MATLAB environment to any distributed memory multiprocessors." Choy 2 at 333; Menon at 1. Specifically, it is an "extension[] of the Matlab interpreter that essentially run on each node of the parallel machine," *i.e.*, the MultiMATLAB architecture is a computer cluster where MATLAB runs on each node of the computer cluster. Morrow at 1. MultiMATLAB's applicability is not limited to MATLAB only, as the designers "present[ed] the MultiMATLAB architecture as a promising design for a parallel

numerical computing environment." Menon at 16. This follows from the designers' decision to "to build upon MATLAB itself" while avoiding "any changes in the MATLAB architecture; indeed, we have not had access to the MATLAB source code." Trefethen at 13–14.

A table to summarize many of these systems (not including PVMaple) can be found below:

Table 1. Examples of SCEs with parallel and distributed facilities

SCE	Multiproces. version	Built upon	Type	Parallel	Cluster
Maple	Distributed[13]	Java	Extended	No	Yes
	FoxBox[3]	MPI	Extended	Yes	Yes
	For Paragon[1]	Kernel	New	Yes	No
	‖Maple‖[14]	Strand	Extended	Yes	No
	Sugarbush[2]	Linda	Extended	Yes	No
Matlab	AlphaBridge[6]	Mex	New	Yes	No
	Conlab[5]	PICL	Extended	Yes	Yes
	DPToolbox[7]	PVM	Extended	Yes	Yes
	Falcon[12]	F90	Translation	Yes	Yes
	MultiMatlab[15]	MPI	Extended	Yes	Yes
Mathematica	Distributed[13]	Java	Extended	No	Yes
	Parallel Toolkit[16]	RSH	Extended	Yes	Yes

Petcu3 at 2. PVMaple is described in various papers by Dana Petcu and others. *See generally, e.g.*, Petcu2; Petcu3. *See also, e.g.*, Dauger 4 at 82; Pooch - MacMPI X Visualization (Dec. 13, 2004); Pooch - MPI Implementation (Dec. 28, 2005) (listing "Supported Implementations of MPI"); Choy 2 at 3 ("In MultiMATLAB, a MATLAB user can start MATLAB processes on other machines and then pass commands and data between these various processes."); Menon at 3 ("The key component in this architecture is the MultiMatlab interface module. The interface module . . . is responsible for initializing an underlying communication layer such as MPI [2], PVM [3], BLACS [4], or any other package available on the platform, and exposing it to the rest of the system."); Kepner 1 at 1 ("MatlabMPI consists of a set of Matlab scripts that implements a subset of MPI and allows any Matlab program to be run on a parallel computer.").

3. Discussion of Particular Claim Limitations

(a) Peer-to-Peer Architecture

Cluster nodes each having single-node programs were known in the art prior to the Asserted Patents' earliest-claimed priority date, as Buyya explains that each process executing on a node of a cluster "executes basically the same piece of code but on a different part of the data" in the most common parallel programming paradigms, including the SPMD cluster computing paradigm. Buyya at 19. The SPMD cluster computing paradigm "involves the splitting of application data among the available processors," where "[p]rocessors communicate with neighbouring processors," "some global synchronization [may be performed] periodically among all the processes," and "data . . . may be read from the disk during the initialization stage." *Id.* at 19–20. FIG. 1.5 of Buyya below shows a schematic representation of the SPMD cluster computing paradigm.

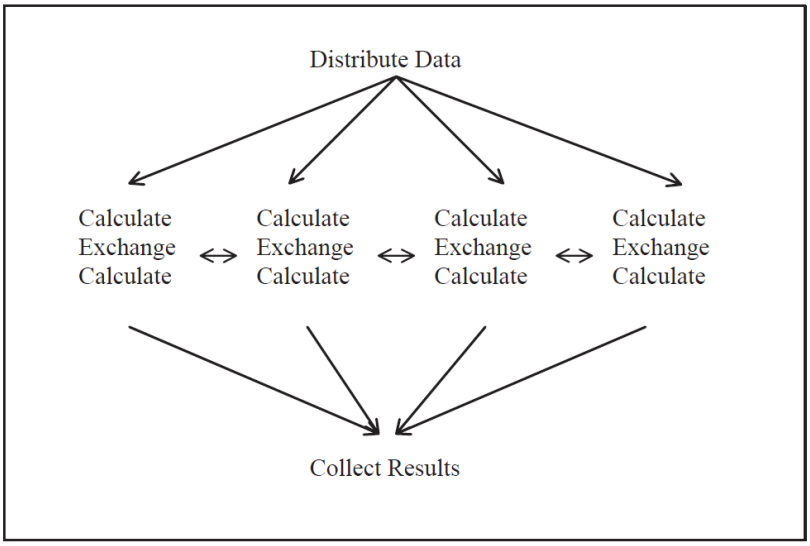


Figure 1.5 Basic structure of a SPMD program.

Buyya at Fig. 1.5.

The ’768 patent claims MPI as the “mechanism for the nodes [of the computer cluster] to communicate . . . with each other using a peer-to-peer architecture.” ’768 patent, claims 1 and 35; *see also* ’768 patent, 12:41–53. The use of MPI to enable processes on cluster nodes in a peer-to-peer network to communicate with each other, however, was known before the earliest-claimed priority date of the Asserted Patents, as exemplified by Buyya’s explanation that “MPI consists of a specification of a set of library functions that tasks of a parallel program can use to communicate among themselves” and “[t]he MPI environment supports [] the all-peers . . . programming model[] very well.” Buyya at 56, 58. MPI enables “both point-to-point and collective [communications] within a group of tasks,” where point-to-point communication refers to “direct communication between a source and a destination task” using “point-to-point send and receive functions,” and collective communication refers to “communication between a group of MPI tasks, where all tasks belonging to that group participate in the communication” using a “set of collective communication primitives.” *Id.* at 57–58. Example send and receive functions include MPI_Send() and MPI_Recv(), respectively. *Id.* at 60.

Buyya provides an example of implementing MPI using the SPMD cluster computing paradigm in a peer-to-peer network. Buyya at 53, 55, 64–68. The implementation is with reference to the parallel Floyd algorithmic solution to “the all pairs shortest path problem”:

> [E]ach task is a worker, with the *task identifier* (*taskid*) assigned by MPI serving as its instance identifier. One of the tasks is designated as the one responsible for reading in the input matrix and distributing the relevant portions to its peers. This task is the one with the taskid of 0 []. At the end, it collects the results from its peers, which together yield the solution to the all pairs problem.

Id. at 53, 55–56. During “the main computation (interspersed with communication) phase,” “the peers work on their respective stripes of the input data matrix,” communicating “among themselves in every iteration of updates to the matrix to get the most up-to-date value.” *Id.* at 64.

"At the end of this main phase of the algorithm, the peers send their results to the pseudo-master. Since all the tasks are peers, and the pseudo-master is arbitrarily chosen to be the task with rank MASTERTID, all the peers know a priori who to send the results back to." *Id.* Excerpts of Buyya's Listing 3.3, which is "a code excerpt from the Parallel Floyd's MPI program, with the MPI calls [] **highlighted**," are shown below. Buyya at 64. The two excerpts of Listing 3.3 show taskid = 0 (the pseudo-master or cluster computer coordinating task) distributing calls (i.e., matrix stripes) to the other tasks (using MPI command MPI_Send) and the other tasks receiving the calls sent by taskid = 0 (using MPI command MPI_Recv). *Id.* at 65–66.

```
/* Get our task ID (our rank in the basic group). */
MPI_Comm_rank(MPI_COMM_WORLD, &myTid);

/* Get the number of tasks. */
MPI_Comm_size(MPI_COMM_WORLD, &nTasks);

/* The master reads the input, and distributes it to the workers. */
if (myTid == MASTERTID) {
    /*
     * Read in the number of vertices and the input matrix itself from a
     * specified file.
     */
    :
    /* Assign stripes to everyone, including ourselves. */
    sendWork();
}
else {
    collectWork();
}

/* Calculate shortest paths for the stripe assigned to us. */
calcShort();
```

Buyya at Listing 3.3 (excerpted)

```
void sendWork()
{
    for (i = 0; i < nTasks; i++) {
        if (i == MASTERTID) {
            /* Read in my stripeMatrix entries directly from inMatrix. */
            :
        }
        else {
            /* Calculate the stripe for each peer task and send it to them. */
            bufPos = 0;
            MPI_Pack(&nVert, 1, MPI_INT, buf, BUFSIZE, &bufPos,
                     MPI_COMM_WORLD);
            for (j = 0; j < nVert; j++) {
                MPI_Pack(&(inMatrix[j][begCol]), width, MPI_INT,
                         buf, BUFSIZE, &bufPos, MPI_COMM_WORLD);
            }
            MPI_Send(buf, bufPos, MPI_PACKED, i, WORK_MSG,
                     MPI_COMM_WORLD);
        }
    }
} /* end of "sendWork" */


void collectWork()
{
    bufPos = 0;

    /* Get stripe matrix from master. */
    MPI_Recv(buf, BUFSIZE, MPI_PACKED, MPI_ANY_SOURCE, WORK_MSG,
             MPI_COMM_WORLD, &status);

    /* Unpack data. */
    MPI_Unpack(buf, BUFSIZE, &bufPos, &nVert, 1, MPI_INT,
               MPI_COMM_WORLD);

    /* Calculate my stripe width and allocate stripeMatrix. */
    :
    for (j = 0; j < nVert; j++) {
        MPI_Unpack(buf, BUFSIZE, &bufPos, stripeMatrix[j],
                   stripeWidth, MPI_INT, MPI_COMM_WORLD);
    }
} /* end of "collectWork" */
```

Buyya at Listing 3.3 (excerpted)

The parallel MATLAB architecture discussed above, MultiMATLAB, also employs the MPI communication standard for communication between MATLAB processes running on the processor nodes of the MultiMATLAB architecture. Menon at 3. The MultiMATLAB architecture comprises multiple processors where "each processor in [the] parallel platform individually runs a MATLAB process" (*id.*), as shown below. "MATLAB processes [are run] on multiple processors, with full access to all the usual capabilities such as Toolboxes. These processes

communicate via simple MATLAB-style commands built on MPI . . . Both master/slave and SPMD paradigms are implemented" (Trefethen at 14).

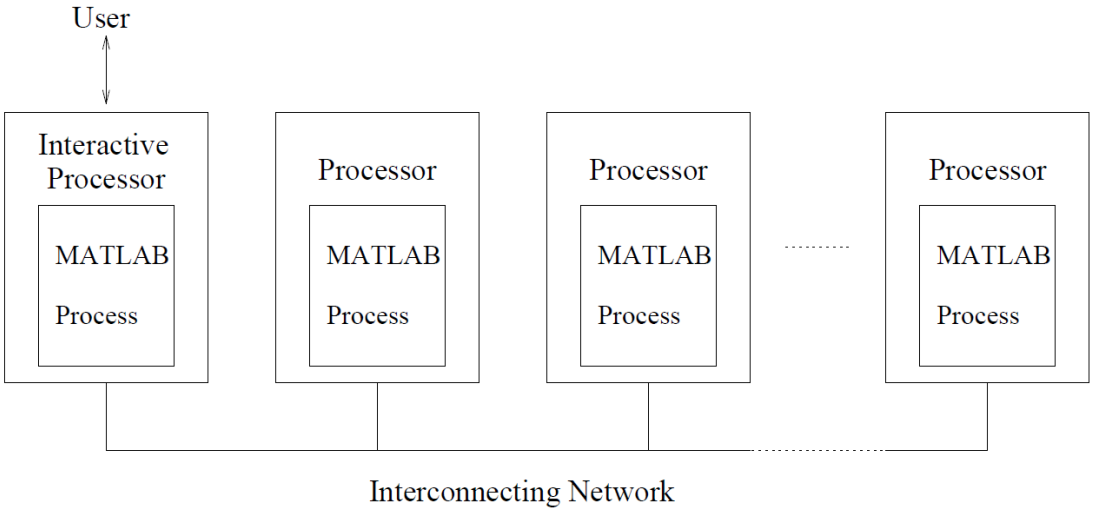


Figure 1: MultiMATLAB Architecture

Menon at FIG. 1

As shown in the figure above, the MultiMATLAB architecture has an interconnection network that connects the processors of the MultiMATLAB architecture to each other. Menon at 3, FIG. 1. The communication layer MPI "runs over the parallel platform's interconnection network," providing each MATLAB process "with the ability to communicate with other [MATLAB] processes," i.e., the interconnection network is a peer-to-peer network. *Id.* at 3.

Other literature also illustrates a POSA's understanding of MPI. For example, "MPI provides explicit point-to-point messaging operations." pyMPI Introduction at 6. MPI implementations use the "send, recv" functions, as well as queues to send data among processes. *Id.* at 6, 11.

Additionally, each of the other systems and references in the Claim Chart Exhibits teaches a cluster of kernels, such as Mathematica, Maple, or MATLAB, or scientific computing libraries such as FFTW or Numerical Python ("NumPy"). As set forth in detail in these Invalidity Contentions, each of the systems and references in these charts teaches peer-to-peer communications between the cluster nodes, including communication of commands, instructions, and computation results between the cluster nodes. Like the preferred embodiment of the Asserted Patents, many of these prior art systems utilized existing message passing functionalities, such as the MPI or PVM message passing libraries used by Apple-Seed, FFTW, Intel Paragon Maple, Matlab*P, MatlabMPI, MultiMATLAB, Pooch, PVMaple, pyMPI, SEM, Whitney, and the Stony Brook GPU cluster. Other kernel clusters, such as Distributed Maple, used peer-to-peer messaging that was similar in principle to MPI and PVM. In fact, Distributed Maple publications taught that "a native C implementation on the basis of a message passing library such as MPI or PVM will always yield a more efficient implementation." Schreiner1 at 344, motivating a combination with MPI.

The prior art thus reflects a clear motivation to implement efficient peer-to-peer communication using MPI and similar protocols. Given the widespread use and performance benefits of such protocols, a POSA would have been motivated to combine teachings from systems and references employing MPI to enhance cluster communications. Accordingly, a POSA would have found it obvious to modify or combine one or more of the systems and references disclosed in the Claim Chart Exhibits to implement peer-to-peer communication consistent with the Asserted Patents.

As one example, it would have been obvious to implement Distributed Maple with an MPI or PVM protocol, as expressly motivated by the Distributed Maple references. A POSA seeking

to make such a modification would naturally have looked to other parallel cluster systems, such as Matlab*P or PVMaple. For instance, in reviewing Matlab*P, a POSA would have found it obvious to incorporate convenient higher-level MultiMATLAB commands (MultiMATLAB formed a component of Matlab*P) to the advanced functions already present in Distributed Maple:

> **2.5. Higher-level MultiMATLAB Commands**
>
> The MultiMATLAB commands described so far represent communication primitives as they are used in the message-passing paradigm of programming. One of the aims of this project, however, is to provide also an interface at a higher level by building on these routines, hiding as much of the message passing as possible.
>
> We can do this by taking a data-parallel approach in a simplistic fashion. We have developed a number of routines such as `Distribute` and `Collect` that allow a user to distribute a matrix or to collect a set of matrices into one large matrix. These functions operate using a mask that indicates which processors hold which portions of the matrix. This allows us also to develop routines such as `Shift` and `Copy` that are useful in data-parallel computing, keeping the communication to a more abstract level.
>
> Additional geometry routines such as `Grid` and `Coord` have also been constructed that allow the user to create a grid of processors in 1,2 or 3 dimensions. These provide a powerful tool for more sophisticated parallel coding. An optional argument on the communication routines allows communication within a given set of nodes, for example along a column or row of the grid. We do not give further details, as these facilities are under development.

Trefethen at 8.

The MultiMATLAB documentation excerpt above illustrates another common motivation to combine systems and references using MPI, PVM, or similar protocols: the ease with which convenient functionality could be implemented by adding higher-level functions on top of standard protocols. In this example, the reference teaches that standard MPI send and receive functions (used in the preferred embodiment of the Asserted Patents for peer-to-peer communication) can be used to implement additional functions. Thus, a POSA would have been motivated to add higher level or advanced functions available in one system to another system, knowing that they could do so without undue experimentation by leveraging standard protocols and following the teachings of the references.

These examples show that the similarity of peer-to-peer communication protocols in the systems and references cited in the Claim Chart Exhibits made it obvious to incorporate features across systems. This was not inventive—it was a predictable application of known techniques using well-understood, standardized protocols that were widely documented and consistently implemented.

The motivation to combine one or more of the systems and references in the Claim Chart Exhibits is further underscored by extensive cross-referencing of the prior artisans' work. For example, in his publications describing Distributed Maple, Dr. Schreiner referenced work by other prior artists, including PVMaple and Intel Paragon Maple systems and references, noting the similarities between these systems. *See, e.g.*, Schreiner1 at 307–09. Likewise, the PVMaple documentation references Distributed Maple and Intel Paragon Maple as similar systems, observing that "[t]he design principles are very similar to that of Distributed Maple." *See, e.g.*, Petcu2 at 216–17. These citations show that the prior art community viewed these efforts as part of a common body of work directed to parallelizing computer algebra system kernel computations, and that prior artists were actively drawing on and incorporating each other's contributions.

In addition, it would have been obvious to combine one or more of the clusters described in the Claim Chart Exhibits and/or the references describing these clusters, with additional publications disclosing MPI-, PVM-, or similarly based cluster systems. For example, the well-known PETSc cluster computing environment parallelized scientific computation libraries using the MPI messaging protocol and implemented peer-to-peer communication among processing nodes:

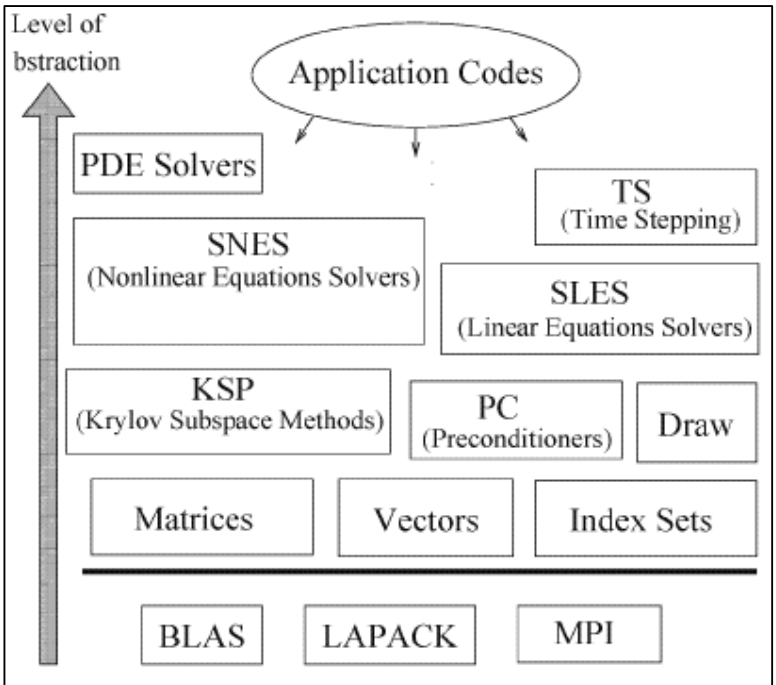


Figure 1: Organization of the PETSc Libraries

PETSc User Manual at 18.

PETSc also provided an extensive array of MPI-based parallel programming tools, including advanced functionality built atop basic messaging primitives:

> **Parallel Programming**
>
> Since PETSc uses the message-passing model for parallel programming and employs MPI for all interprocessor communication, the user is free to employ MPI routines as needed throughout an application code. However, by default the user is shielded from many of the details of message passing within PETSc, since these are hidden within parallel objects, such as vectors, matrices, and solvers. In addition, PETSc provides tools such as generalized vector scatters/gathers and distributed arrays to assist in the management of parallel data.

Id. at 26.

A POSA would have been motivated to apply these teachings to the prior art clusters disclosed in these Invalidity Contentions, including the Claim Chart Exhibits, as those systems—like PETSc—used MPI or similar protocols to parallelize kernels or scientific computing workloads. Given the functional similarities among these systems, and the common use of standardized message-passing protocols, it would have been a straightforward and predictable

design choice to incorporate PETSc's advanced MPI-based features into other known cluster environments.

Peer-to-peer communication between nodes using standard protocols such as MPI was also implemented in prior art GPU clusters, including the Stony Brook cluster comprising NVIDIA GPUs:

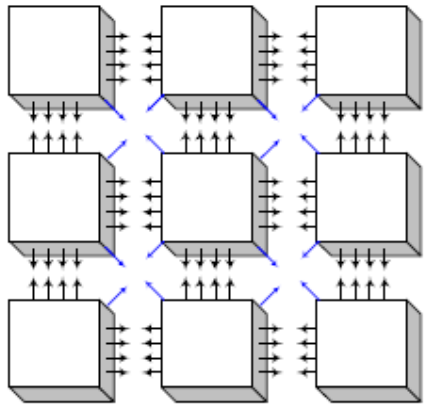


Figure 6: Each block represents a sub-domain of the LBM lattice processed by one GPU. Velocity distributions at border sites stream to adjacent nodes at every computation step. Black arrows indicate velocity distributions that stream axially to nearest neighbor nodes while blue arrows indicate velocity distributions that stream diagonally to second-nearest neighbor nodes.

Fan 1 at 5.

Similarly, researchers at UNC Chapel Hill constructed a GPU cluster using NVIDIA GPUs and MPI-based peer-to-peer communication:

Node(s)	CPU Configuration	GPU Configuration
0-2	3GHz Pentium 4 with Hyperthreading	nVidia GeForce 6800 GT
3	3.4 GHz Pentium 4 with Hyperthreading	nVidia GeForce 6800 Ultra
4	2.8 GHz AMD Athlon 64 FX-53	nVidia GeForce 6800 Ultra
5	Dual 2.8GHz Pentium Xeon	nVidia GeForce 6800
6	2.4GHz Pentium Xeon	nVidia GeForce 6800

Table 1: Configurations of Cluster Nodes

Chinchilla at 3.

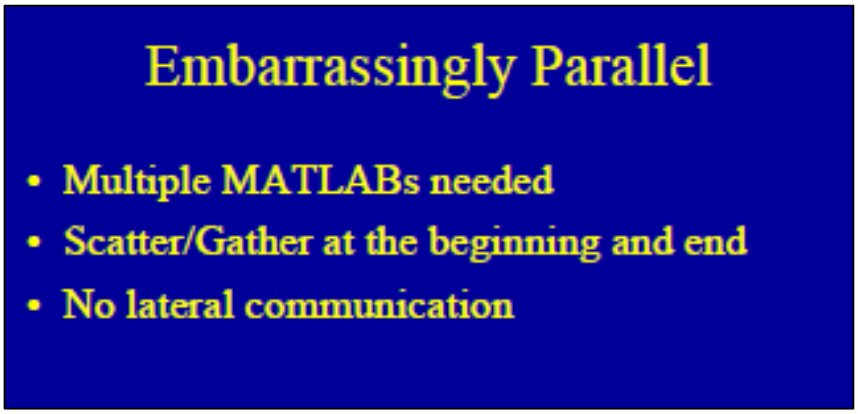
> The second optimization we performed with the new algorithm was changing the mechanism by which bodies not local to the GPU were transferred there. The algorithm described in §5 has one texture for bodies belonging to other nodes, and it swaps these in and out of the GPU during each iteration of force accumulation. We noted that on the GPU,

Id. at 4.

These prior art systems relied on the MPI library to implement peer-to-peer communication among nodes. Chinchilla at 3–4. It would have been obvious to combine these MPI-based scientific computing GPU clusters with one or more of the other references cited in these Invalidity Contentions for substantially the same reasons described above. The use of MPI provided a clear motivation to consult and incorporate teachings from other systems using the same or similar protocols, particularly where they addressed parallelization of computational workloads. Moreover, both the Fan and Chinchilla references describe the performance benefits of MPI-enabled GPU clusters, including increased computational throughput and efficiency—further motivating a POSA to combine their teachings with those of the clusters disclosed in the Claim Chart Exhibits. Because these systems all used MPI for peer-to-peer communication, a POSA would have recognized that such combinations could be made without undue experimentation, leveraging a shared and well-understood messaging framework across systems.

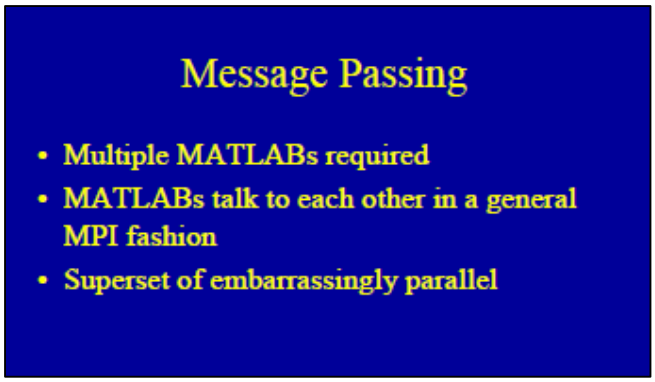
In addition, it would have been obvious to combine features of the Stony Brook and Chapel Hill clusters, as both used NVIDIA GPUs and implemented peer-to-peer communication via the MPI protocol. The Chinchilla reference discussing the Chapel Hill cluster explicitly references and describes its similarities with the Stony Brook cluster, further motivating a POSA to combine them.

As yet another example, Choy and others at MIT, Cornell, and other institutions, recognized as early as 2003 that the alleged problem "solved" by the Asserted Patents—namely, how to manage parallel kernel computation on a cluster—was already well-understood, and some computing problems simply do not require lateral node communications. In a public presentation, Choy and his co-authors, including MIT professor Alan Edelman, called these problems "embarrassingly parallel," meaning they require no lateral node communication at all:



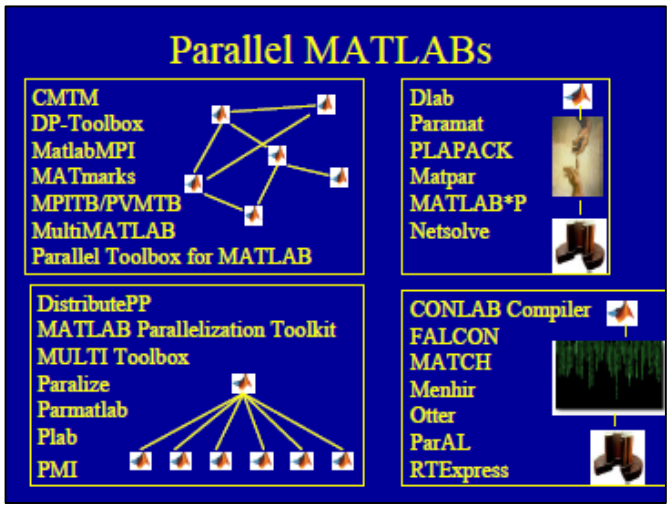
Choy 3 at slide 24.

Choy also described the more general class of problems that *do* benefit from peer-to-peer communication, which were routinely addressed at the time using MPI and other conventional message-passing techniques:



Choy 3 at 25.

Thus, by 2003, the art already understood both when lateral communication was needed and how to implement it using standard libraries. Choy's presentation summarized the state of the art, describing a conventional solution—message passing between multiple instances of MATLAB kernels using MPI—that either anticipates or renders obvious the Asserted Claims. These known approaches operated exactly like the preferred embodiment described in the Asserted Patents, by parallelizing kernels on a computer cluster using the standard MPI library. Unlike the alleged inventors of the Asserted Patents, Choy did not claim that any such solution was novel. To the contrary, he expressly recognized that *multiple existing systems* already implemented peer-to-peer communication among kernel instances:



Choy 3 at 20.

In the slide reproduced above, Choy listed seven exemplar systems supporting peer-to-peer communication among cluster nodes—depicted in the upper-left quadrant of Choy's chart using a diagram that closely resembles Fig. 3 of the Asserted Patents. Choy then described his own Matlab*P software, which built on these prior efforts and further confirms that such designs were not only known, but routinely combined by persons of ordinary skill in the art.

This is unsurprising. Parallelizing kernels such as Mathematica, Maple, and MATLAB was a natural and convenient way to accelerate computation for certain problems, and was easily achieved using conventional protocols and architectures such as MPI and PVM. This well-understood design landscape provided a clear motivation for prior artisans to develop and combine such clustering software—including the very systems listed by Choy.

Yet another example is the work by Dr. Wolfgang Schreiner, described in publications and implemented in publicly available source code and executable files. Well before the earliest-claimed priority date of the Asserted Patents, Dr. Schreiner developed a system for parallelizing Maple and Mathematica kernels—called Distributed Maple and Distributed Mathematica—that disclosed nodes with kernels and messaging software enabling lateral peer-to-peer communications between nodes. This system anticipated and rendered obvious the Asserted Claims:

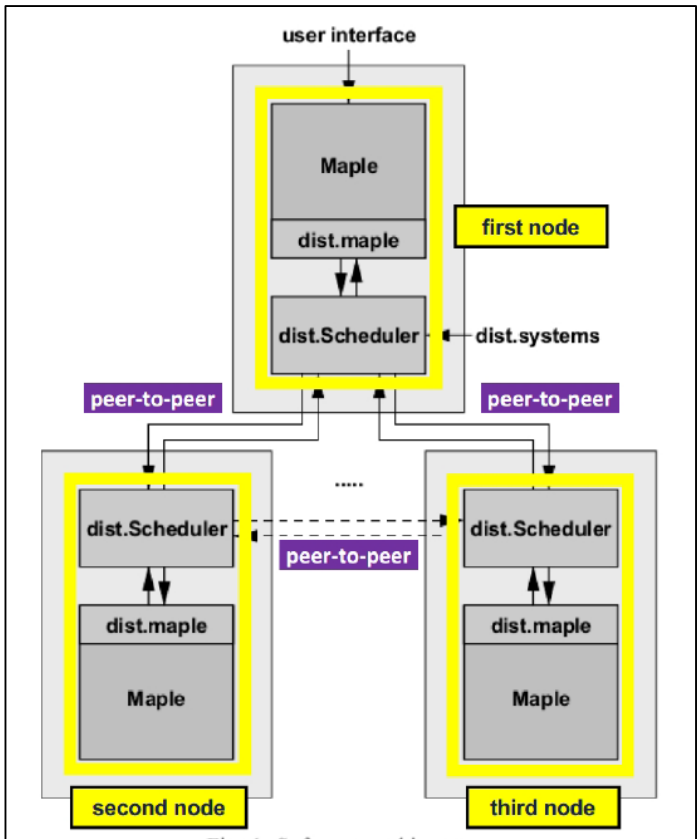


Fig. 1. Software architecture.

Schreiner1 at 311 (annotated).

The annotated figure above depicts a representative cluster with three nodes (and an ellipses indicating additional nodes). Arrows between the nodes indicate that each node is directly connected to all others, enabling direct, peer-to-peer communications across the cluster. As Schreiner explained,

> all nodes know of each other, i.e. a node knows the address of a machine and the number of a port on which (a thread of) the remote scheduler is listening for connection requests. ***When a node needs to send a message to one of its peers, it can thus establish a direct connection for message transfers.*** The connection remains persistent through the rest of the session such that no more startup overhead is involved.

Schreiner1 at 315 (emphasis added); *id*. at 317–18.

As shown in Schreiner's annotated figure above, each node comprised a Maple kernel along with *dist.maple* and *dist.Scheduler* software. These enabled direct message passing between the kernels using a peer-to-peer mechanism, rather than routing messages through a centralized controller. In another embodiment, Dr. Schreiner's system implemented the same architecture to parallelize Mathematica kernels:

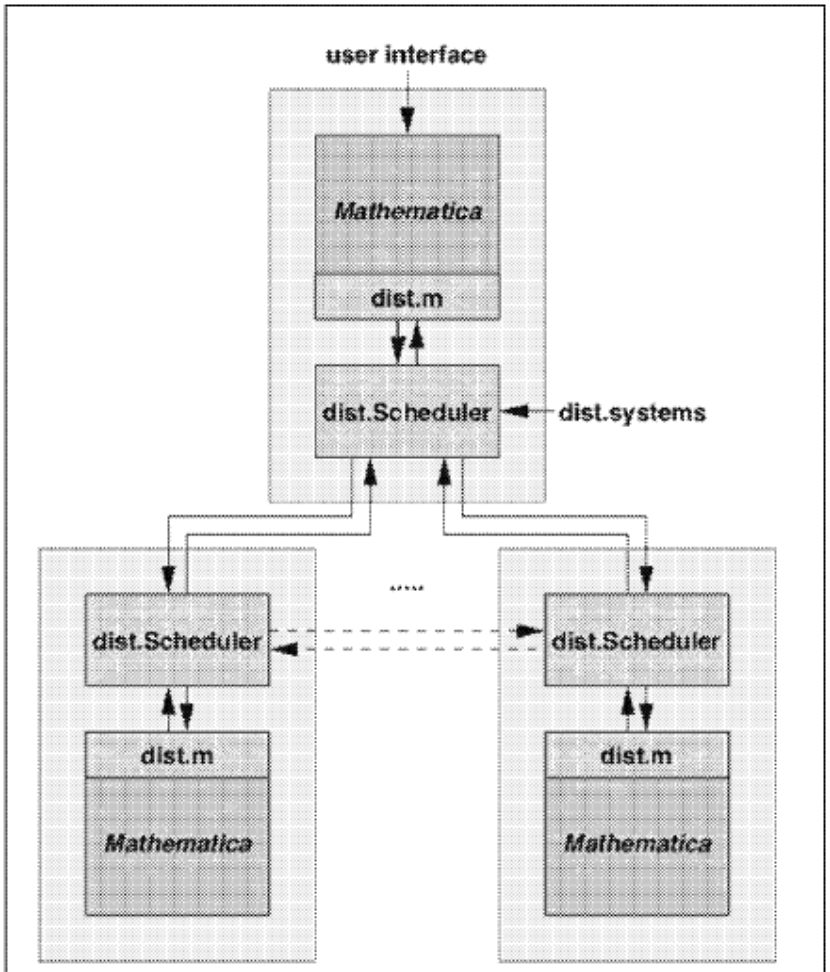


Figure 1: The architecture of a distributed *Mathematica* session

Schreiner7 at 3.

These prior art systems and references did exactly what the Asserted Patents later claimed as their alleged contribution: "[i]ntercommunication provided by cluster computing [that] permits exchange of information between nodes during the course of a parallel computation," as opposed to "communication between only one master node and many slave nodes." '768 patent, 6:26–38. Dr. Schreiner's work expressly contradicts the Asserted Patents' alleged point of novelty. His Distributed Maple and Distributed Mathematica system provided the very same cluster-based, peer-to-peer communication functionality described in the Asserted Patents, and did so using conventional message-passing techniques that were widely understood and in public use well before the patents' earliest alleged priority date.

As another example, the PVMaple system and related publications disclose parallelized Maple, Mathematica, and other kernel clusters using the PVM messaging protocol—a protocol that, for purposes of the Asserted Claims, is functionally interchangeable with MPI. PVMaple was described in publications and embodied in source code and executable files well before the earliest-claimed priority date of the Asserted Patents.

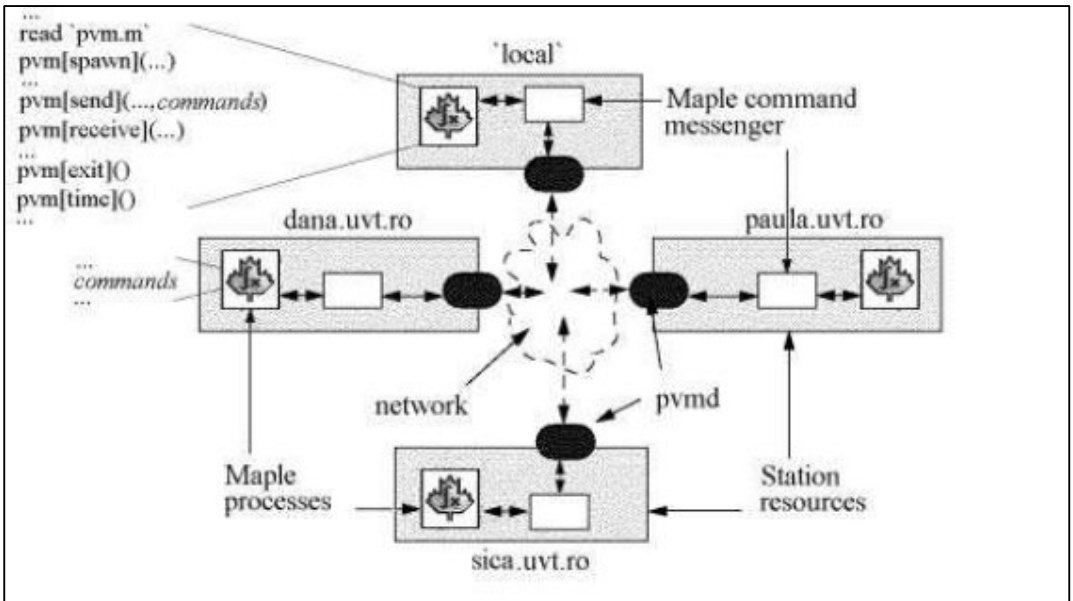


Fig. 1. Processes and inter-communication routes

Petcu2 at 218.

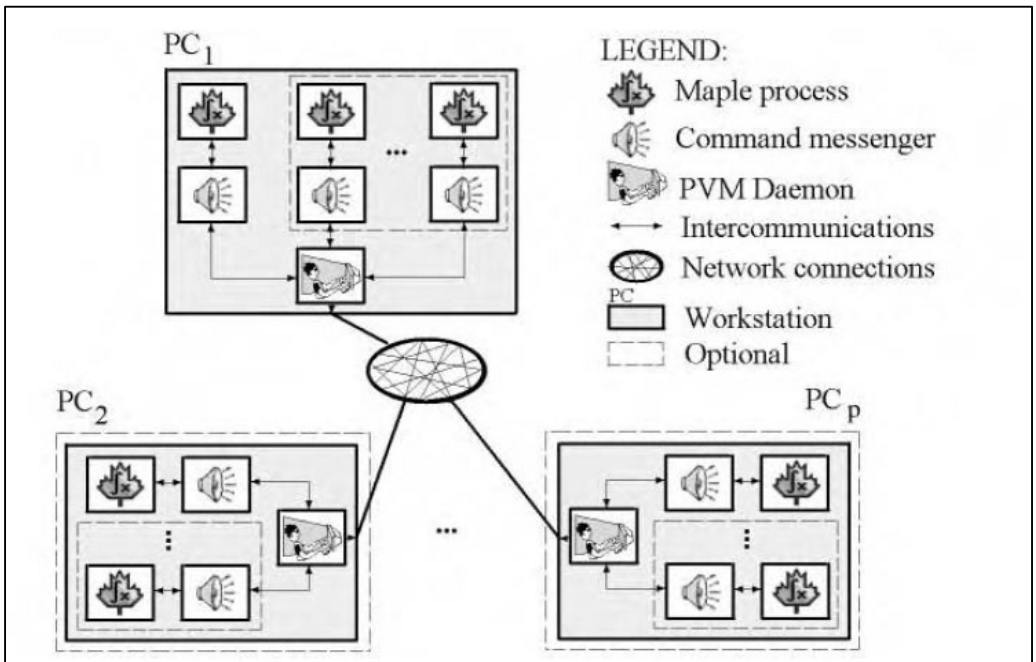
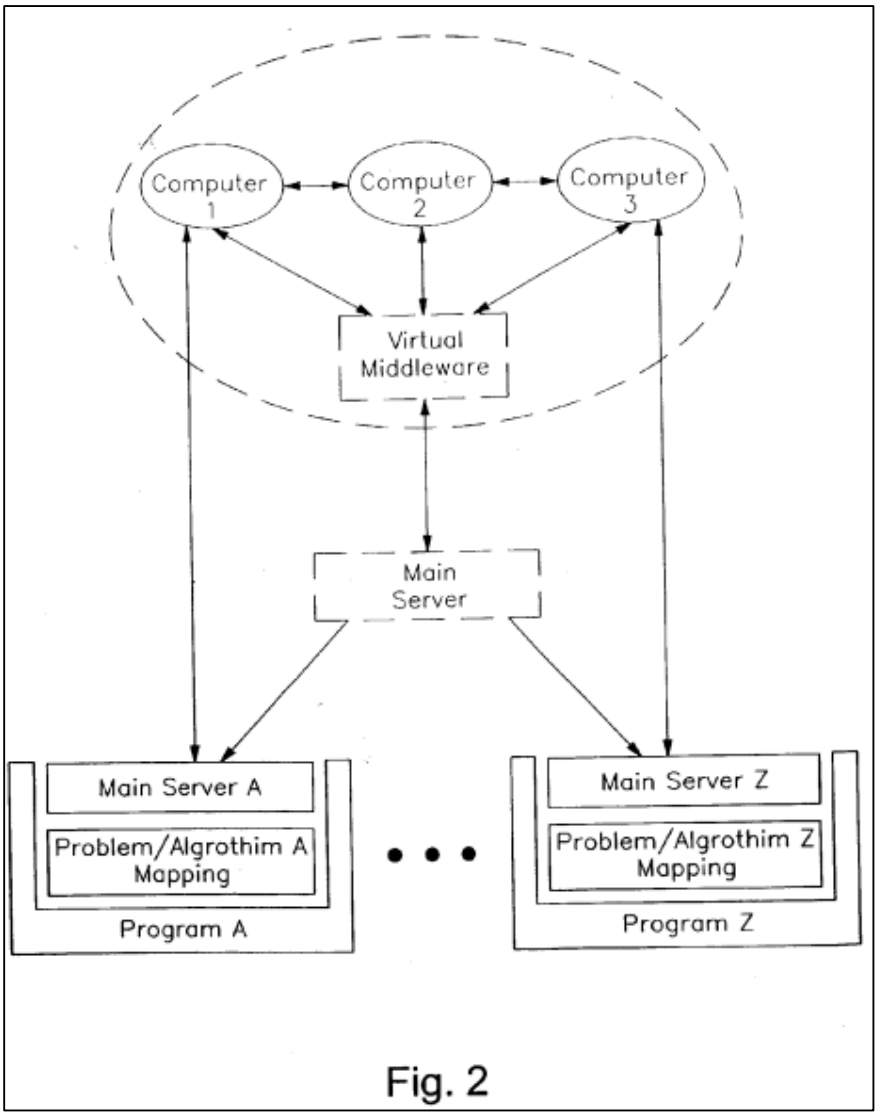


Fig. 1. Active processes and intercommunication routes

Petcu4 at 927.

Another illustrative reference is Gulko 240, which discloses a system and method for parallelizing existing software applications. *See* Gulko 240 at [0014]–[0019]. Gulko 240 explains that "[c]luster middleware is used to interface between the server requesting distributed processing of an application and a computer cluster, to achieve distributed processing." *See id.* at [0010]. However, Gulko 240 recognizes a key limitation of conventional middleware: it must be specifically "incorporated into an application adapted for distributed processing" and often requires users to "write a substantial amount of code" to make use of its features. *Id.* To address this, Gulko 240 discloses a universal "virtual middleware" layer, referred to as "DNET," that simplifies application development by abstracting the underlying messaging protocols. *Id.* at [0042]. Gulko 240 further discloses that DNET "has enabled peer-to-peer communication among the computers 1-3," that is, direct messaging between lateral nodes. *Id.* DNET includes a messaging interface "based on the PVM standard," and its "implementation maps the DNET API onto (parallel virtual machine) PVM or MPI." *Id.* at [0147]. As Gulko 240 explains, "PVM and MPI allow adapters to be constructed that make inter-slave communications available to their applications." *Id.* at [0147].

Figure 2 of Gulko 240, reproduced below, confirms this architecture, showing a distributed cluster in which lateral node communications are enabled by the DNET messaging layer. Gulko 240 therefore reinforces that, well before the priority date of the Asserted Patents, it was known in the art to enable peer-to-peer messaging across a cluster using MPI or PVM, and to abstract those communications through a middleware interface. These teachings further demonstrate that the Asserted Claims reflect a predictable application of known cluster messaging techniques, not any inventive advance.



Gulko 240 at Fig. 2.

For all of these reasons—and the additional reasons set forth in the Claim Chart Exhibits and these Invalidity Contentions—to the extent Plaintiff contends that any limitation related to peer-to-peer networking or lateral communication between cluster nodes is not expressly disclosed in a particular system or reference in the Claim Chart Exhibits, it would have been obvious to a POSA to incorporate that functionality based on the teachings of the other references. The

widespread use of MPI, PVM, and similar protocols in the prior art provided a clear motivation to combine, and doing so would have yielded predictable results without undue experimentation.

(b) Additional MPI Functionality

Several Asserted Claims further recite the use of asynchronous message-passing functionality. *See, e.g.*, '768 patent, claims 26, 27, 30, 31, 34; '034 patent, claim 25. This type of functionality was well-known and widely used in parallel systems that implemented peer-to-peer communication via MPI, PVM, or similar protocols. As discussed in Section III.D.3.a (Peer-to-Peer Architecture), the use of MPI in parallel clusters was conventional and not inventive. The Asserted Patents simply adopt functionality already specified and standardized by MPI—particularly asynchronous messaging primitives that were routinely available in parallel cluster environments long before the alleged invention.

The MPI standard expressly discloses the exact asynchronous behavior described in the Asserted Patents. It states, for example, that "MPI calls may initiate operations that continue asynchronously after the call returned." MPI Standard 1995 at 13. It further explains that "[w]ith suitable hardware, the transfer of data out of the sender memory may proceed concurrently with computations done at the sender after the send was initiated and before it completed." *Id.* at 36.

Indeed, the MPI standard explicitly defines nonblocking asynchronous variants of the MPI send and receive primitives, including MPI_SEND, MPI_RECV, and MPI_TEST. These calls initiate asynchronous message passing and include arguments such as the source or destination rank and the address of the buffer to be transmitted or received. *See* MPI Standard 1995 at 38 (MPI_ISEND), 40 (MPI_IRECV), 42 (MPI_TEST).

MPI_ISEND(buf, count, datatype, dest, tag, comm, request)

IN	buf	initial address of send buffer (choice)
IN	count	number of elements in send buffer (integer)
IN	datatype	datatype of each send buffer element (handle)
IN	dest	rank of destination (integer)
IN	tag	message tag (integer)
IN	comm	communicator (handle)
OUT	request	communication request (handle)

```
int MPI_Isend(void* buf, int count, MPI_Datatype datatype, int dest,
              int tag, MPI_Comm comm, MPI_Request *request)
```

```
MPI_ISEND(BUF, COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, DEST, TAG, COMM, REQUEST, IERROR
```

Start a standard mode, nonblocking send.

Id. at 38.

MPI_IRECV (buf, count, datatype, source, tag, comm, request)

OUT	buf	initial address of receive buffer (choice)
IN	count	number of elements in receive buffer (integer)
IN	datatype	datatype of each receive buffer element (handle)
IN	source	rank of source (integer)
IN	tag	message tag (integer)
IN	comm	communicator (handle)
OUT	request	communication request (handle)

```
int MPI_Irecv(void* buf, int count, MPI_Datatype datatype, int source,
              int tag, MPI_Comm comm, MPI_Request *request)
```

```
MPI_IRECV(BUF, COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR)
    <type> BUF(*)
    INTEGER COUNT, DATATYPE, SOURCE, TAG, COMM, REQUEST, IERROR
```

Start a nonblocking receive.

These calls allocate a communication request object and associate it with the request handle (the argument request). The request can be used later to query the status of the communication or wait for its completion.

A nonblocking send call indicates that the system may start copying data out of the send buffer. The sender should not access any part of the send buffer after a nonblocking send operation is called, until the send completes.

A nonblocking receive call indicates that the system may start writing data into the receive buffer. The receiver should not access any part of the receive buffer after a nonblocking receive operation is called, until the receive completes.

Id. at 40.

```
int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status)

MPI_TEST(REQUEST, FLAG, STATUS, IERROR)
    LOGICAL FLAG
    INTEGER REQUEST, STATUS(MPI_STATUS_SIZE), IERROR
```

A call to MPI_TEST returns flag = true if the operation identified by request is complete. In such a case, the status object is set to contain information on the completed operation; if the communication object was created by a nonblocking send or receive, then it is deallocated and the request handle is set to MPI_REQUEST_NULL. The call returns flag = false, otherwise. In this case, the value of the status object is undefined. MPI_TEST is a local operation.

The return status object for a receive operation carries information that can be accessed as described in section 3.2.5. The status object for a send operation carries information that can be accessed by a call to MPI_TEST_CANCELLED (see Section 3.8).

One is allowed to call MPI_TEST with a null or inactive request argument. In such a case the operation returns with flag = true and empty status.

The functions MPI_WAIT and MPI_TEST can be used to complete both sends and receives.

Advice to users. The use of the nonblocking MPI_TEST call allows the user to schedule alternative activities within a single thread of execution. An event-driven thread scheduler can be emulated with periodic calls to MPI_TEST. (*End of advice to users.*)

Rationale. The function MPI_TEST returns with flag = true exactly in those situations where the function MPI_WAIT returns; both functions return in such case the same value in status. Thus, a blocking Wait can be easily replaced by a nonblocking Test. (*End of rationale.*)

Id. at 42.

The Asserted Patents themselves confirm that this functionality is lifted directly from the MPI specification, acknowledging that these three asynchronous calls "make it possible for the kernel to do work while communications are proceeding simultaneously." '768 patent, 13:41–47. It then describes mpiISend, mpiIRecv, and mpiTest using language that parallels the MPI standard: the mpiISend command "creates a packet containing" an expression "to be sent as payload and where the expression should be sent"; the mpiIRecv command "creates a packet specifying where it expects to receive an expression and from which processor this expression is expected;" and the mpiTest command "creates a packet specifying which message to test for completion, then waits for a reply expression to evaluate." *Id.* at 14:1–29. The '768 patent admits as much, stating that "[a] cluster node module 204 can include at least a portion of an application programming interface

('API') known as the Message Passing Interface ('MPI'), which is used in several supercomputer and cluster installations," and that these "fundamental calls mpiISend, mpiIRecv, and mpiTest" are standard functionality. *Id.* at 11:45–48, 14:29–35. Accordingly, it would have been obvious to a POSA that any system implementing MPI—or even a similar peer-to-peer protocol—would inherently or routinely include such asynchronous operations.

Moreover, the Claim Chart Exhibits describe prior art systems and references that already implemented these asynchronous techniques in parallel clusters. For example, MultiMATLAB supports asynchronous messaging across MATLAB processes. Specifically, MultiMATLAB provides for "send and receive commands, which can be executed on any of the MATLAB processes," and "[a]n optional argument can be added in Recv to specify the source" and these "processes run asynchronously." Trefethen at 6–7.

2.3. Send, Recv, Probe, Barrier

More general point-to-point communication is accomplished by send and receive commands, which can be executed on any of the MATLAB processes. For example, the sequence

```
x = [pi pi^2];
Send(3,x)
Eval(3, 'Recv' )
```

passes a message containing a 2-vector from the master process to process 3, leading to the output

```
3.1416 9.8696
```

An optional argument can be added in `Recv` to specify the source. Another optional argument may be added in both `Send` and `Recv` to specify a message tag so as to ensure that sends and receives are properly matched and to aid in error checking. The command

```
Probe,
```

run on any process, again with optional source process number and message tag, returns 1 (true) if a message has arrived from the indicated source with the indicated tag, otherwise 0 (false).

Id. at 6.

`cycle.m` consists of the following program:

```
if ID==0                  % first process: send
   a = 1
   Send(ID+1,a)
elseif ID == Nproc-1      % last process: receive and double
   a = 2*Recv
else                      % middle processes: receive, double, and send
   a = 2*Recv
   Send(ID+1,a)
end;
```

Process 0 creates the variable `a` with value 1 and sends it to process 1. Process 1 receives the message, doubles the value of `a`, and sends it along to process 2; and so on. If there are six processors the command `Eval( 'cycle' )` produces the output

```
a = 1
a = 2
a = 4
a = 8
a = 16
a = 32
```

The processes run asynchronously, but since each `Send` command is only executed after the corresponding `Recv` has completed, the proper sequence of computations and final value 32 are guaranteed so long as all of the nodes are functioning.

Id. at 7. It would be obvious to a POSA to implement such asynchronous messaging in parallel clusters similar to MultiMATLAB, including AppleSeed, Distributed Maple, Matlab*P, MatlabMPI, Merrimac, Pooch, SEM, and Whitney. One of ordinary skill would have understood that sequential communication defeats the very performance gains that cluster parallelization is intended to achieve. Thus, even in clusters that did not adopt the MPI standard per se, a POSA would have been motivated to implement asynchronous messaging using other conventional tools or APIs—precisely because this was a well-understood and widely adopted strategy for maximizing parallel efficiency. *See, e.g.*, Geist at 45 ("The PVM communication model provides asynchronous blocking send, asynchronous blocking receive, and nonblocking receive functions."); *id*. at 252 ("The pvm_send routine is asynchronous."). Plaintiff also asserts claims reciting a "data structure" in which messages are stored, including "received message queues" and "message receiving queues." *See* '621 patent, claims 1, 6, 7, 8; '582 patent, claims 6, 7, 8, 9; *see* '768 patent, 14:12–15; 14:20–24. This functionality was standard in systems implementing MPI, and was either expressly disclosed by the systems and references cited in the Claim Chart Exhibits

or would have been obvious for a POSA to implement in view of those references. *See* Section III.D.3.a (Peer-to-Peer Architecture), *supra*.

For example, Cownie explains that in MPI systems, "send and receive queues store information about all of the unfinished send and receive operations." Cownie at 2. These queues are essential to tracking the status of nonblocking or asynchronous communication operations across multiple processes. Cownie further describes how this queuing behavior is implemented in MPICH, a widely used MPI implementation. MPICH underlies or is compatible with many of the systems identified in the Claim Charts, including Appleseed, Matlab*P, MatlabMPI, MultiMATLAB, Pooch, pyMPI, SEM, and Whitney. *See id.*; pyMPI Introduction at 12–13; Menon at 5; Whitney Analysis at 7. Cownie explains that some or all of these queues can store "information that was passed to the function call that initiated the operation." Cownie at 2. This allows the runtime system to manage in-progress operations and match incoming messages with the appropriate receive call. A POSA would have found it obvious to use such data structures—including separate queues for received messages and expected messages—in any peer-to-peer messaging system, particularly one implementing MPI or a similar protocol. These data structures are not novel architectural choices; they are necessary to support asynchronous communication across distributed nodes and ensure orderly message handling when multiple messages may be in-flight simultaneously.

cator there are, conceptually at least, three distinct "message queues," which represent the state of the MPI subsystem. These are

Send Queue: represents all of the outstanding send operations.
Receive Queue: represents all of the oustanding receive operations.
Unexpected Message Queue: represents messages that have been sent to this process, but have not yet been received.

The send and receive queues store information about all of the unfinished send and receive operations that the process has started within the communicator. These might result either from blocking operations such as `MPI_Send` and `MPI_Recv` or nonblocking operations such as `MPI_Isend` or `MPI_Irecv`. Each entry on one of these queues contains the information that was passed to the function call that initiated the operation. Nonblocking operations will remain on these queues until they have completed and have been collected by a suitable `MPI_Wait`, `MPI_Test`, or one of the related multiple completion routines. The unexpected message queue represents a different class of information, since the elements on this queue have been created by MPI calls in other processes. Therefore, less information is available about these elements. In all cases the order of the queues represents the order that the MPI subsystem will perform matching (this is important where many entries could match, for instance when wild-card tag or source is used in a receive operation).

Cownie at 1–3, 6–8.

Accordingly, to the extent Plaintiff contends that any Asserted Claim is distinguished by the presence of a "message queue" or "data structure" for handling peer-to-peer communication, such functionality was conventional, well-documented, and would have been obvious to a POSA to include in any MPI- or PVM-based cluster system.

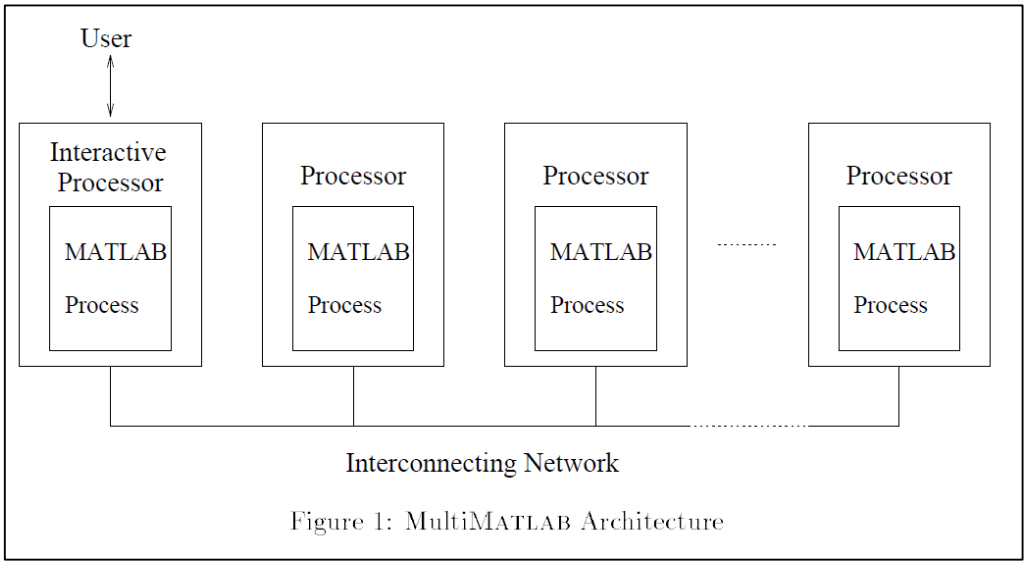
(c) Kernels

The claimed "kernel" in the Asserted Patents refers to "program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language." '768 patent, 23:9–12; *see also id.*, 1:29–42, 4:14–25. As the Asserted Patents acknowledge, such single-node kernels were well-known and conventional at the time of the alleged invention. The Asserted Patents concede, for example, that prior-art Mathematica software included "a kernel that is designed to communicate with a single

node," as did "[m]athematics software packages from other vendors and other types of software." *Id.* at 1:29–42.

It was also conventional in the art to deploy such single-node kernels across multiple nodes in a parallel cluster, and to coordinate them using peer-to-peer messaging or middleware. Accordingly, to the extent Plaintiff alleges that any limitation related to kernels is not disclosed in a particular system or reference in the Claim Chart Exhibits, it would have been obvious to a POSA to implement or combine that functionality using other well-known systems and references.

As just one example, Menon describes how MultiMATLAB implemented a MATLAB process—a type of single-node kernel—on each of a set of nodes connected by an interconnecting network.



Menon at 3, FIG. 1.

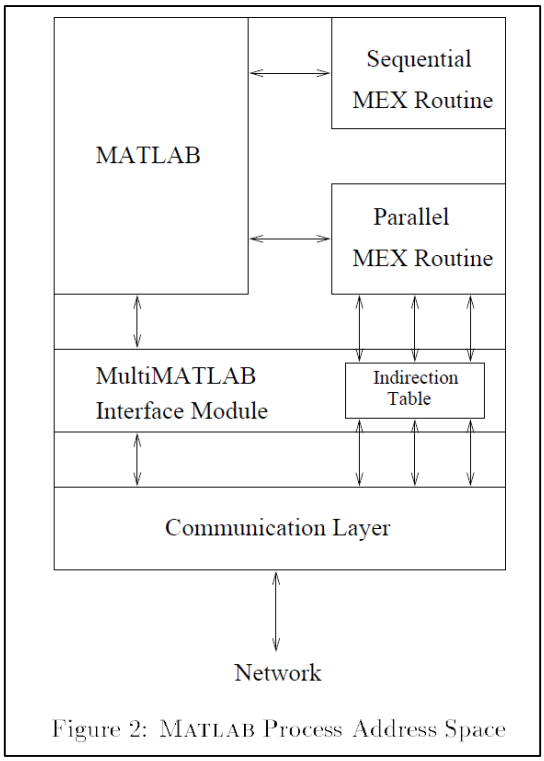
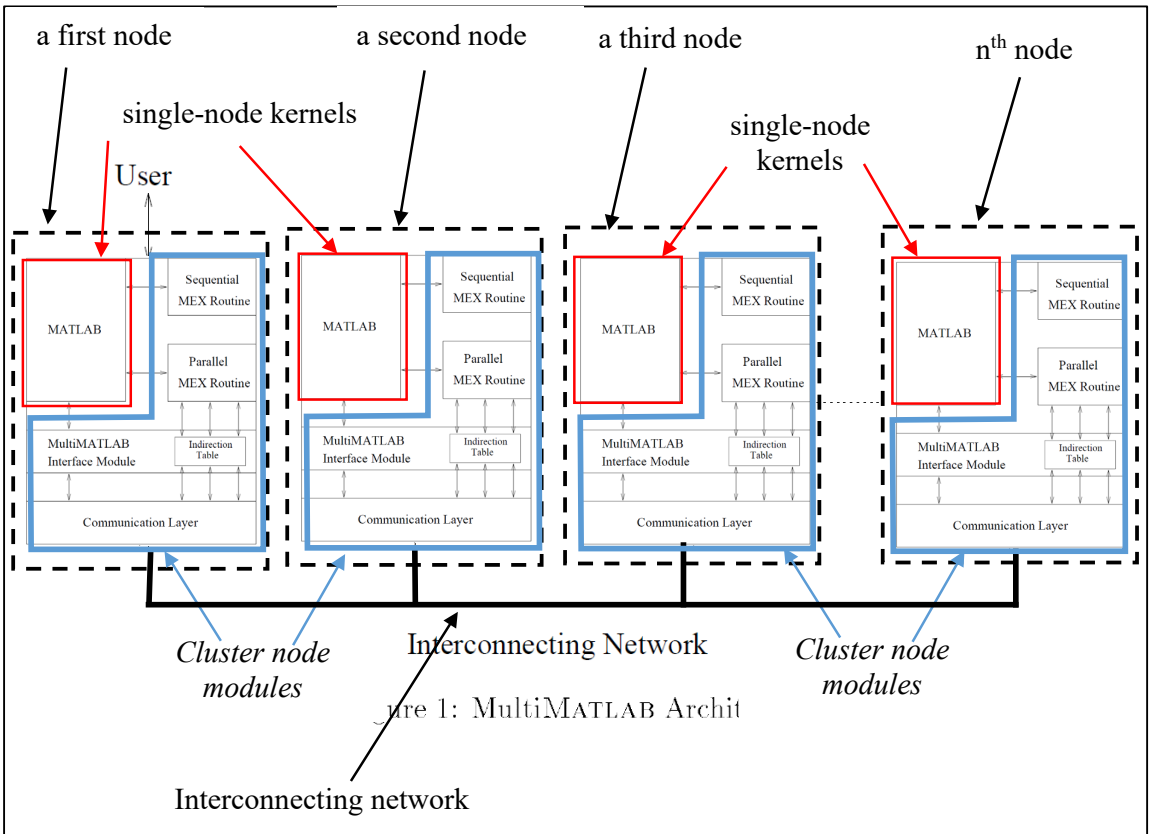


Figure 2: MATLAB Process Address Space

Id. at 4, FIG. 2. Each of the nodes in this cluster had a MATLAB process. "The user interacts directly with one Matlab process"—the leftmost node in FIG. 1—while "[t]hese other processes are used either by explicitly sending Matlab commands to them or by executing routines that, in turn, use them." *Id.* at 3–4.

MultiMATLAB also teaches that it was well-understood, routine, and conventional in the prior art before the earliest-claimed priority date of the Asserted Patents to have a first cluster node module receive commands from a user interface without the commands first passing through the described first kernel. Menon discloses a set of components to provide interconnection with other nodes, including the MEX Routines, the MultiMATLAB Interface Module, and the Indirection Table. Menon at 3–5, FIG. 1, FIG. 2. Users can execute commands using an "Eval" routine, which "is implemented as a MEX routine which accepts a vector of

MATLAB process IDs and a MATLAB command" and "sends the commands over the underlying communication layer to the processes corresponding to the specified IDs." *Id.* at 4. The cluster node module first maps MPI calls for MEX routines "to the appropriate table offset" using the indirection table. *Id.* at 6. This process is depicted in the annotated Figures 1 and 2 from Menon:



Menon at FIGs. 1 and 2 (combined and annotated)

As another example, Distributed Maple and PVMaple both operated in conjunction with Mathematica and its front end. *See, e.g.*, Schreiner7 at 2–3 ("The user interacts with Distributed *Mathematica* via the *Mathematica* frontend which communicates with the scheduling program running in the background in order to transfer tasks to other Mathematica kernels."); Petcu2 at 217

("PVMaple system proposes an extension of Maple capabilities to distributed computations for workstations grouped into a Parallel Virtual Machine. The user interacts with the system via the text oriented Maple front-end."). To the extent any of the computer cluster systems and references in these Invalidity Contentions do not expressly or inherently disclose that a user interface may transmit commands directly to a cluster node module without first passing through a kernel, a POSA would have found it obvious to implement such functionality based on examples like MultiMATLAB and Distributed Maple, and doing so would not require undue experimentation. These architectures were known, functionally interchangeable, and widely used in the relevant art.

Distributed Maple also implemented a kernel on each node of the cluster, consistent with the Asserted Patents' description of a distributed, single-node kernel model. The kernel "contains fundamental and primitive commands: the Maple language interpreter (which converts the commands you type into machine instructions your computer processor can understand), algorithms for numerical calculation, and routines to display results and perform other input and output operations." Maple Guide at 88. Dr. Schreiner likewise disclosed implementing such kernels on distinct nodes of a distributed cluster:

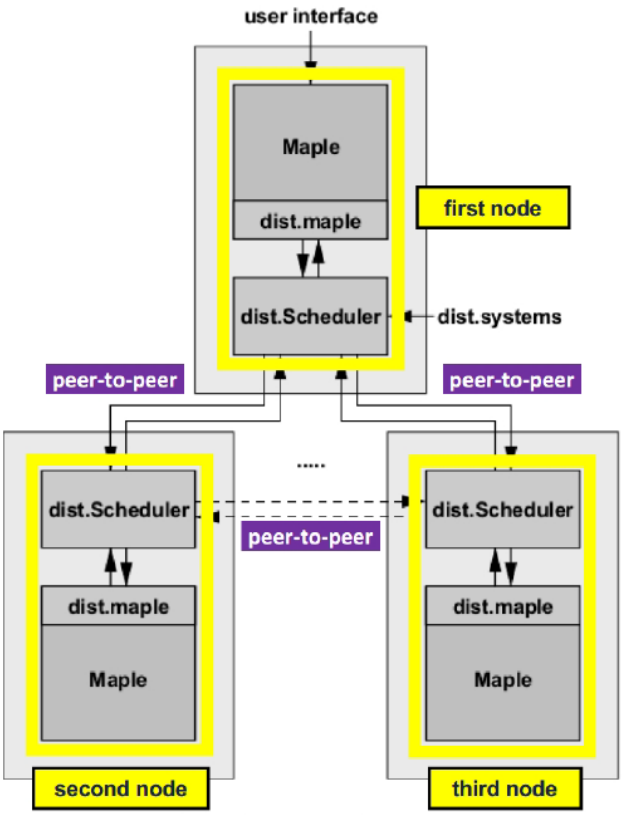


Fig. 1. Software architecture.

Schreiner1 at 311 (annotated).

In addition, each of the other systems and references in the Claim Chart Exhibits teaches or inherently discloses the use of distributed kernels, whether they be environments like Mathematica, Maple, MATLAB, or Merrimac or scientific computing libraries such as FFTW or NumPy. These systems consistently reflect the well-known practice of deploying single-threaded evaluation engines across multiple nodes for parallel computation.

(d) Claimed Three-Node Sequence of Operations

Well before the earliest alleged priority date of the Asserted Patents, it was widely known in the art that data communication between cluster nodes can proceed in a sequential or pipelined fashion. *See* Buyya at 21–22. In this approach, "the tasks of the algorithm, which are capable of

concurrent operation, are identified and each processor executes a small part of the total algorithm." *Id.* at 21. Data, such as the results of a process's computation of a task at a cluster node, is then sent to the process at the next cluster node in the pipeline, as explained by Buyya:

> Processes are organized in a pipeline - each process corresponds to a stage of the pipeline and is responsible for a particular task. The communication pattern can be very simple since the data flows between the adjacent stages of the pipeline. For this reason, this type of parallelism is also sometimes referred to as data flow parallelism. The communication may be completely asynchronous.

Id. at 21. The "pipeline is one of the simplest and most popular functional decomposition paradigms." *Id.* Figure 1.6 of Buyya below shows a schematic representation of the data pipeline structure.

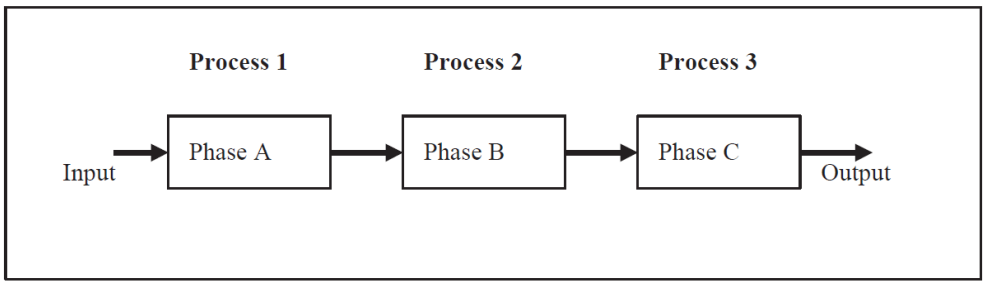


Figure 1.6 Data pipeline structure.

Buyya at FIG. 1.6

As is notable from Figure 1.6 of Buyya, POSAs knew and understand that a pipeline paradigm has communication flow from the first process to a second process to a third process and then provides the resulting output.

Buyya teaches combining paradigms (e.g., such as the SPMD cluster computing paradigm and the data pipeline structure for data flows) when discussing "the need, [in some applications,] to mix elements of different paradigms," which may involve "situations where it makes sense to mix data and task parallelism simultaneously." Buyya at 23. For example, the SPMD cluster

computing paradigm and the data pipeline structure for data flows, depicted above in Buyya's Figures 1.5 and 1.6, respectively, can be combined to implement an example situation where data is distributed, as shown in Figure 1.5, to one process (e.g., process 1 in Figure 1.6 above), and then data propagates from this process to all the other processes on the cluster nodes, as shown in Figure 1.6 above, while the processes perform calculations on their respective received data, as shown in Figure 1.5.

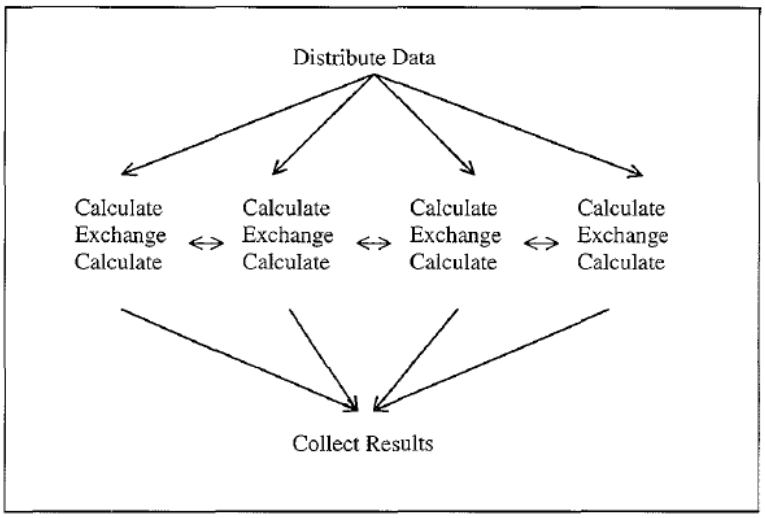


Figure 1.5 Basic structure of a SPMD program.

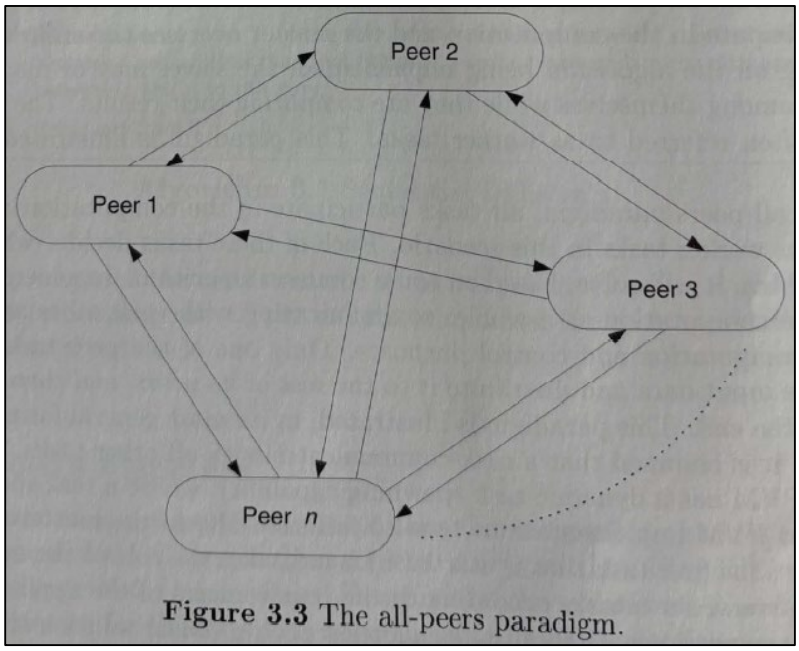
Buyya at FIG. 1.5.

Buyya explains that the SPMD cluster computing paradigm (and its combination with the data pipeline structure for data flows) can be implemented in computer clusters with the cluster nodes communicating with each other "in the all-peers paradigm," i.e., via a peer-to-peer network. Buyya at 53–55. In a peer-to-peer network setting, one process at a cluster node may function as a coordinating process that, for example, receives user instructions and distributes commands or calls to the other processes of the cluster computer:

> In the all-peers paradigm, all tasks participate in the computation and are referred to as worker tasks in this scenario. Each of these tasks decides what portion of the problem it

> will solve, based on some common algorithm. In general, all tasks control the computation as a whole, communicating with each other as necessary both for computation and control purposes. Only one of the peer tasks may still read in the input data and distribute it to the rest of its peers, and then collect the results in the end.

Id. at 53. Figure 3.3 of Buyya below shows a peer-to-peer network comprising *n* cluster nodes where "a task communicates with all other tasks." *Id.*



Buyya at FIG. 3.3

The aforementioned MultiMATLAB architecture employs a combination of a SPMD cluster computing paradigm with a data pipeline structure for data flows, as illustrated with the following example script (m-file), "cycle.m," executed on 6 processors that constitute a computer cluster implementing a MultiMATLAB architecture:

```
if ID==0                    % first p
    a = 1
    Send(ID+1,a)
elseif ID == Nproc-1        % last process: receive and double
    a = 2*Recv
else                        % middle
    a = 2*Recv
    Send(ID+1,a)
end;
```

MATLAB process ID=0 creates variable a=1 and sends it to MATLAB process ID=1

Last MATLAB process receives the variable "a" and doubles it

MATLAB processes ID receive the variable "a," double it, and send the doubled variable "a" to the next MATLAB process ID+1

Process 0 creates the variable a with value 1 and sends it to process 1. Process 1 receives the message, doubles the value of a, and sends it along to process 2; and so on. If there are six processors the command `Eval( 'cycle' )` produces the output

```
a = 1
a = 2
a = 4
a = 8
a = 16
a = 32
```

Trefethen at 7 (annotated). The m-file "cycle.m," when executed in a MultiMATLAB architecture comprising multiple processors communicating in a SPMD paradigm, results in the sequential evaluations of mathematical expressions by processors that make up the computer cluster, where the final result is communicated to the node that distributed the call for the execution of the mathematical expressions. *Id.*

Accordingly, it was well known, and a POSA would have understood, before the earliest-claimed priority date of the Asserted Patents for a computer cluster to have a plurality of nodes in a peer-to-peer network each having a processor and a memory, where each node has a single-node program that executes using the processor and communicates with any other node using MPI, and where one of the nodes functions as a coordinating node of the computer cluster during parallel computation in that it receives user instructions, distributes calls to all the other nodes based on the user instructions, and then collects results from the other nodes upon completion of each node's computation.

Plaintiff has asserted that the '768 patent claims recite a "precise order of operations involving three specific nodes: (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation; (2) the third node performs a different calculation based on the received result; and (3) the third node sends the new result to the first node." IPR2021-00019, Paper No. 5 (POPR filed Feb. 10, 2021) at 26–27. Notwithstanding this representation, Plaintiff now appears to allege in its Infringement Contentions that a system merely configured to enable communications between three nodes infringes the Asserted Claims—regardless of whether that system is actually configured to perform the claimed "precise order of operations involving three specific nodes."

To the extent Plaintiff alleges that the accused Intel products, in combination with software libraries, somehow satisfy these claim limitations, the same sequence was well-known and disclosed in multiple prior art systems—including several expressly discussed in these Invalidity Contentions.

For example, the prior-art FFTW library, as discussed further below in Section III.D.3.e (Fourier Transforms and Other Computational Libraries), discloses sequential operations consistent with the claimed 1–2–3–1 structure. In particular, Cramer and FFTW Manual describe a parallel implementation for calculating fast Fourier transforms that implements sequential operation. One processor computes a 2D FFT on its set of slices, sends intermediate results to the other processors, and those processors then compute a transposed 3D FFT. Cramer at 2; FFTW Manual at 43–49.

Choy likewise described this type of parallelized FFTW implementation, where processors pass intermediate results between stages of distributed computation, as well as additional parallelized FFT functionalities:

Functionalities provided

- PBLAS, ScaLAPACK
  - solve, eig, svd, chol, matmul, +, -, …
- FFTW
  - 1D and 2D FFT
- PARPACK
  - eigs, svds
- Sparse Matrix Routines
- 'MultiMATLAB' mode

Choy 3 at 37.

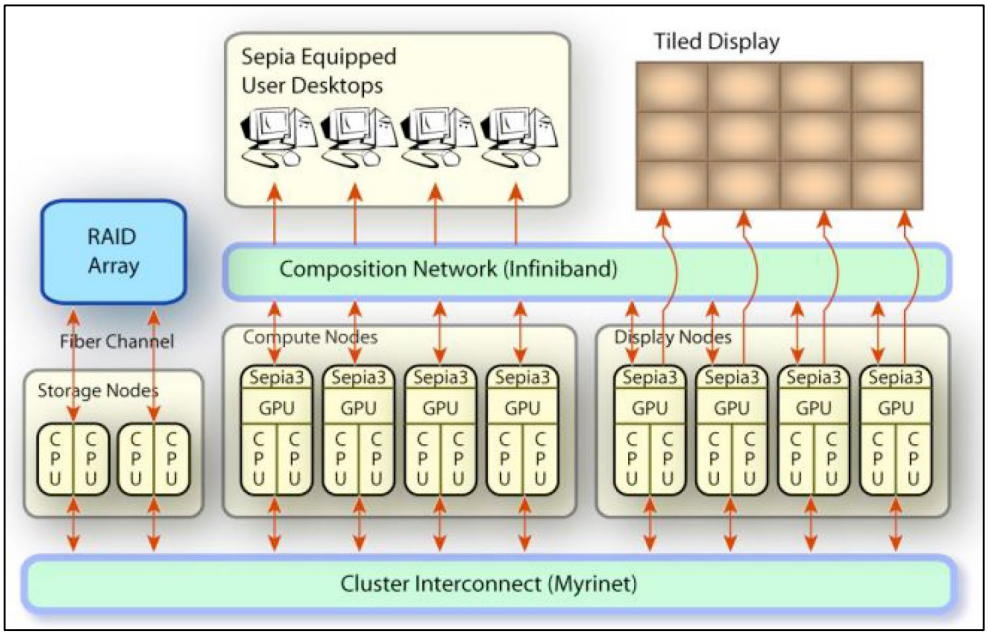
Example 3

% FFT2 in 4 lines

- A = randn(10000,10000*p);
- B = mm('fft',A);
- C = B.';
- D = mm('fft',C);
- F = D.';

Id. at 47.

Other systems followed similar patterns. Chen used the Brook programming language to implement parallelized FFT on a cluster of NVIDIA GPUs—again using multiple stages of computation distributed across separate nodes:



FFT with Brook GPU Presentation at 11.

As another example, PVMaple discloses the claimed sequence of three-node communications with a basic example illustrating the obviousness and triviality of this limitation:

Example 1: communication between processes

```
> restart; read 'pvm.m';                          load the PVMaple functions
> pvm[settime]();                                 start time chronometer
> pvm[spawn]([dana,1],[paula,1]);                 start two tasks
      Tasks: [['local', 1], [dana, 1], [paula, 1]]  result: pvm[tasks] values
> mess1:='pvm[send]([paula,1],"global S;          prepare a message to be sended
  S:=2"); S:=5;':                                 and finish with an assignation
> M:=pvm[send]([dana,1],mess1):                   send command to the first task
  pvm[receive](M,[dana,1]);                       receive last command's result
                          [5]                     1st task last command's result
> mess2:='S:=3; pvm[receive]([dana,1],'all');     prepare receive from task 1
  S+10;';                                         and finish with an addition
> N:=pvm[send]([paula,1],mess2):                  send command to the second task
  pvm[receive](N,[paula,1]);                      receive last command's result
                          [12]                    2nd task last command's result
> pvm[exit]();                                    kill the tasks and stop the time
                 PVMaple quitted                  PVMaple stopped
> pvm[time]();                                    show the time graph
```

Petcu2 at 220.

In the foregoing example, PVMaple instantiates three nodes on three processors: 'local,' 'dana,' and 'paula," comprising first, second, and third nodes. The second node, *e.g.*, 'dana,' receives calls from the first node, *e.g.*, pvm[spawn] or pvm[send] calls. The second node executes at least a first mathematical expression evaluation, e.g., assignment S:=2. The second node sends the result S to the third node, *e.g.*, 'paula,' with a pvm[send] call. The third node executes at least a second mathematical expression evaluation using the received result, *e.g.*, it executes the expression S+10. The third node then returns the result of the second mathematical expression evaluation to the first node, *e.g.*, 'local,' which receives it (the result is equal to 12) with a pvm[receive] call. Additional examples 2–4 in Petcu2 further illustrate this limitation. *See* Petcu2 at 219–22.

The claimed sequence of communications between the three nodes is further illustrated in the following annotated figure from Petcu2 at 219 (annotated):

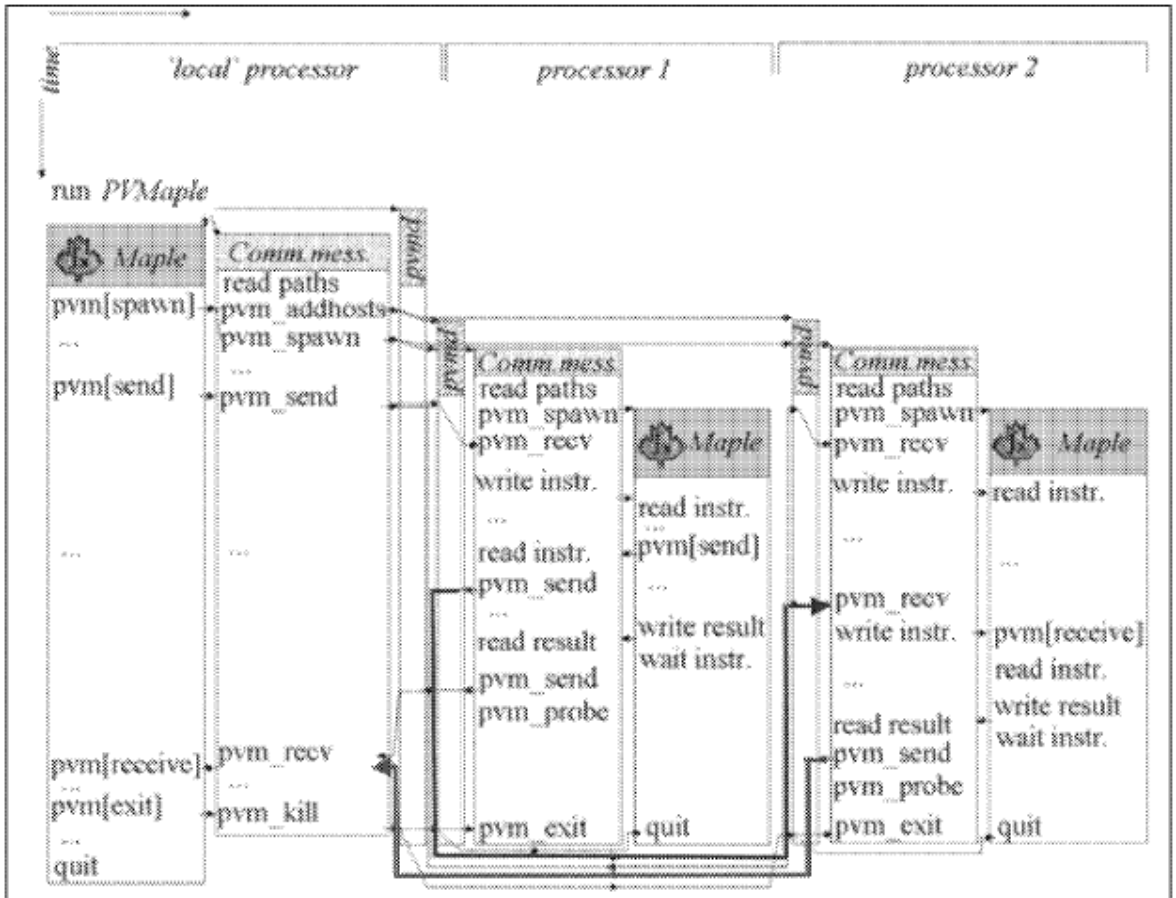


Fig. 2. Communications and components inter-dependencies

The upper arrow shows a second node, *e.g.*, processor 1, communicating with a third node, *e.g.*, processor 2 with a pvm[send] call which is typically used to send results of mathematical evaluations. The third node receives the message from the second node, performs an evaluation, and writes results to a first node, *e.g.*, local, with a 'write result' Maple call and corresponding pvm[send] PVMaple call, illustrated by the lower arrow.

This structure exactly mirrors the "precise order of operations" emphasized by Plaintiff, and confirms that the claimed sequence of operations was already a trivial instance of standard multi-node workflows in cluster computing. Indeed, this kind of node-to-node message chaining—

where intermediate results are passed from one peer node to another, then returned to the initiating node—was a commonplace pattern in data-parallel and task-pipelined systems. It was well within the capabilities of any POSA familiar with MPI, PVM, or standard peer-to-peer messaging libraries. Accordingly, the claimed three-node communication sequence does not reflect any inventive concept. It is a predictable and routine application of known messaging techniques, and would have been obvious to implement in any of the systems disclosed in these Invalidity Contentions.

Distributed Maple also disclosed mechanisms for implementing the claimed sequence of operations across three specific node. As discussed in these Invalidity Contentions, Distributed Maple provided functions—such as dist[start], dist[wait], dist[data], dist[put], and others—that allowed the first node (*e.g.*, the current root node) to create and manage tasks distributed across second and third nodes by a central scheduler.

For example, the following illustrative code creates on the first node a task t1 computing function f1(x) and a task t2 computing f2(t1):

```
t1 := dist[start](f1, x);
t2 := dist[start](f2, t1);
r := dist[wait](t2);
```

The first node waits for termination of task t2, and the root scheduler then sends tasks t1 and t2 to the second and third processors, respectively. On receipt of task t1, the second processor performs a first mathematical evaluation of the Maple function f1 passed to it by the first processor. The following exemplary code shows a simple and common function f1, which allocates on node 2 a shared data object d and fills it with the value g1(x); upon the termination of f1, a reference to object d is sent to node 1 and stored there in data structure t1:

```
f1 := proc(x)
 d := dist[data]();
```

```
 dist[put](d, gl(x));
return d;
end:
```

Task t2 is passed by the root scheduler to execute on processor node 3.

The following exemplary code shows a simple and common function f2:

```
f2 := proc(t1)
 d := dist[wait](t1);
 y := dist[get](d);
 return g2(y);
end:
```

This function waits for the result of task t1; for this purpose, it sends a message to processor node 1 (where the data structure t1 is stored) and receives by a reply message the values stored in t1, i.e., a reference to the data object d stored on processor 2. To get the value stored in d, processor node 3 sends a message to processor node 2, receives the value g1(x) and stores it in local variable y; it then computes g2(y) using the result of the first mathematical evaluation (the value g1(x), now locally stored in y) performed by the second processor node. Upon the termination of f2 on the third processor node, this value is sent to processor node 1 and stored there in the data structure t2. From there node 1 takes it and writes it into the local variable r.

Moreover, these patterns were not unique to Distributed Maple. Comparable messaging and task distribution systems existed in AppleSeed, Matlab*P, MatlabMPI, MultiMATLAB, Pooch, PVMaple, SEM, and other cluster environments cited in these Invalidity Contentions. A POSA would have understood that the messaging and shared object capabilities of these prior art systems, including those reflected in the exemplary code above, could be used to implement or adapt computational libraries, such as those taught by FFTW, Cramer, and others. As discussed above, adaptation of such libraries was within the ordinary skill, and resulted in the claimed sequence of three-node operations.

The exemplary code above is illustrative of a broader set of algorithms known in the art, and obvious to implement on any of the computer clusters in these contentions at least because such algorithms are efficiently solved through parallelization. For example, Distributed Maple also disclosed solutions to mathematical computation problems involving dataflow parallelism approach, where tasks running on different nodes depend on results from other tasks, *e.g.*, a task running on a third node is dependent on result of computation of a second node, as indicated by the arrows in the following illustration:

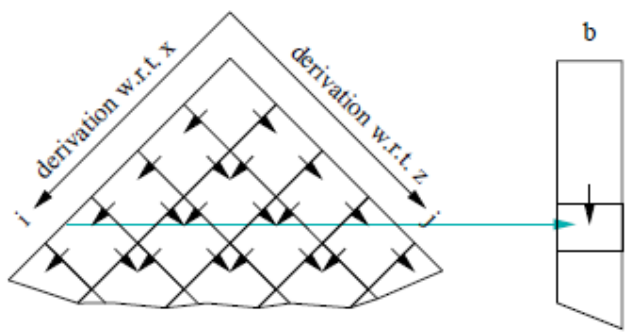


Figure 5: Data Dependencies

Schreiner4 at 8.

In the dataflow parallelism example, the main program running on the first (e.g., root) node distributes tasks to other nodes, including a second and third nodes, together with task dependencies. The dependencies signal to each node which other nodes may provide it with a result needed for the node's own computation: All of the nodes, including the second and third nodes, then send results back to the first node, which waits for them in order to compute the full mathematical result and display it to the user:

> The manager program starts the tasks in some order compatible with the task dependencies and passes to each task (apart from the initial one which receives the input polynomial p) the references to those tasks that the new task potentially depends on. Then the program waits for all tasks (i, j) in the order of increasing $i + j$, combines the computed gcds with the vector b and signals by

Schreiner4 at 9.

A task on the third node that depends on the result of a first mathematical evaluation on the second node will receive the result sent by the second node:

> An "inner" task selects non-deterministically the result of one of the two neighbor tasks (whichever happens to be available first) and, depending on the selection, proceeds with the derivation in the style of a task on the right boundary respectively on the left boundary.

Id. at 10.

The second node may communicate the results to the third node through a task result request and messages, or through a shared data object and corresponding messages:

> As an optimization, a task (i, j) actually needs not return the boundary derivatives d^j_{i+m} and d^{j+m}_i to the manager program (which is only interested in the gcd vector g); instead it may put the results into a shared space from which the other tasks may retrieve then (actually a task should do this as early as possible such that subsequent tasks become enabled even *before* the predecessor task has terminated). Consequently, each task needs not receive as arguments references to other tasks but only the locations of shared data where it may find the boundary derivatives when they have been computed. The main program

Id.

Thus, in the dataflow parallelism examples, a first node (*e.g.*, main or root node) distributes calls to second and third nodes; a second node computes its result and sends it to a third node, one

that depends on the result from the second node; the third node then computes a second result and sends it back to the first node for presentation to the user.

One example of this approach is found in CASA Code, including the "dist/casa/neighbGraph/GCDs" function, CASA Code at 8–10, and related functions called by it, which coordinated distributed computation across multiple dependent nodes. Other examples of source code may be found in, e.g., Schreiner1 and Schreiner4, as discussed above. For example, the following exemplary pseudo code is executed by the first node and distributes tasks to perform evaluations of mathematical expressions to a set of other nodes, including second and third nodes:

```
(b, n) := derivatives(p)
  n := min(n_x, n_z)
  for k from 0 to n by m do
    for (i, j) in tasks(k) do
      if i = 0 ∧ j = 0 then t_{i,j} := start task(i, j, m, p)
      else if i = 0 then t_{i,j} := start task_Right(i, j, m, t_{i,j-1})
      else if j = 0 then t_{i,j} := start task_Left(i, j, m, t_{i-1,j})
      else t_i := start task_Inner(i, j, m, t_{i,j-1}, t_{i-1,j})
    end
  end
  for k from 0 to n by m do
    tset := {t_{i,j} : (i, j) ∈ tasks(k)}
    while tset ≠ ∅ ∧ n = min(n_x, n_y) do
      t := select tset
      (_, _, g) := wait t
      update b_{i+j} ... b_{(i+m)+(j+m)} and n
      tset := tset-{t}
    end
  end
  for k from n + m to min(n_x, n_y) by m do
    for (i, j) ∈ tasks(k) do stop t_{i,j}; end
  end
  for t ∈ tset do stop t end
end
```

Schreiner4 at 9. The remote tasks then wait for mathematical evaluation results on which they depend from other nodes, *e.g.*, a third node waits to receive results from a second node, evaluate their mathematical expression task, and return results to the main program on the first node:

$$
\begin{array}{l}
(p_1, p_2, g) := \mathbf{task}_{\mathrm{Left}}(i, j, m, t) \\
\quad ((p, \_), \_) := \mathbf{wait}\ t \\
\quad (p_1, p_2, g) := \mathrm{task}(i, j, m, \frac{\partial p}{\partial x}) \\
\mathbf{end} \\
\\
(p_1, p_2, g) := \mathbf{task}_{\mathrm{Right}}(i,\ j,\ m,\ t) \\
\quad ((\_, p), \_) := \mathbf{wait}\ t \\
\quad (p_1, p_2, g) := \mathrm{task}(i, j, m, \frac{\partial p}{\partial z}) \\
\mathbf{end} \\
\\
(p_1, p_2, g) := \mathbf{task}_{\mathrm{Inner}}(i, j, m, tr, tl) \\
\quad t := \mathbf{select}\{tr, tl\} \\
\quad \mathbf{if}\ t = tr\ \mathbf{then} \\
\quad\quad (p_1, p_2, g) := \mathrm{task}_{\mathrm{Right}}(i, j, m, t) \\
\quad \mathbf{else} \\
\quad\quad (p_1, p_2, g) := \mathrm{task}_{\mathrm{Left}}(i, j, m, t) \\
\quad \mathbf{end} \\
\mathbf{end}
\end{array}
$$

Id. at 10. For example, the wait call executed by the third node will trigger a message from the second node to deliver the result of the first evaluation to the third node, which then proceeds with its own second evaluation of a mathematical expression using the received result:

`dist[wait]`: **Wait for a Task Result**

```
result := dist[wait](task)
```

This call suspends the current execution until *task* has completed and then returns its *result*. While the current execution is suspended, other tasks can be executed by the corresponding Maple kernel.

```
> t := dist[start](int, x^n, x):
> r := 1 + dist[wait](t);
```

Schreiner2 at 15.

WaitD: Wait for the Result of a Task

result=WaitD[*task*]

This call blocks the current execution until the result of *task* is available and returns the result. As long as the execution is blocked, the kernel can execute other tasks assigned by the scheduler.

```
In[3]:=t=StartD[Integrate,x^n,x];
In[4]:=r=1+WaitD[t]
           x^(1 + n)
Out[4]=1 + ---------
             1 + n
```

Schreiner7 at 14.

These systems illustrate that dataflow parallelism—implemented through peer-to-peer communication and shared data structures—was conventional in the art. A POSA would have readily applied the same techniques to any of the prior art clusters disclosed in these Invalidity Contentions. No undue experimentation would be required, as the messaging primitives (e.g., MPI, PVM) and parallelization strategies were well-documented and widely implemented across computer algebra systems and scientific computing environments. Accordingly, the claimed three-node messaging sequence was not an inventive departure, but rather a routine expression of standard cluster programming paradigms, and would have been obvious to a POSA using any of the referenced systems.

Distributed Maple also implemented recursive mathematical computation functions involving the communication of intermediate results among first, second and third nodes, as required by the Asserted Claims. For example, Schreiner2 discloses code for an exemplary Distributed Maple program that performs a distributed factorization of a list of integers. Schreiner2 at 7–9. The factorization program defines a procedure called "*mai,*" with parameters "*length*" and "*size,*" which invoke a recursive function called "*dfactors*" to factor integers using a parallel, cluster-based approach. *Id.* The *dfactors* procedure performs the following steps:

(1) Creates a list of *length* elements of random integers no bigger than *size*;

(2) Splits the list into two halves;

```
# divide problem
m := trunc(length/2);
numbers1 := [ seq(numbers[i], i=1..m) ];
numbers2 := [ seq(numbers[i], i=m+1..length) ];
```

(3) Processes one half locally (on the root node) using *dfactors*, which may recursively divide and distribute tasks to other nodes:

```
# conquer in parallel
task1 := dist[start](dfactors, numbers1);
result2 := dfactors(numbers2);
```

(4) Issues a *dist[start]* command, which causes one or more remote kernels to determine the factorization of the other half of the list (the elements contained in the numbers1 variable) by performing the *dfactors* function on that half of the list, which then causes the second half of the list to be further split by each remote node and distributed to yet other nodes in a recursive fashion:

```
# conquer in parallel
task1 := dist[start](dfactors, numbers1);
result2 := dfactors(numbers2);
```

(5) Combines results from the local and remote tasks, with intermediate values propagated back to the local (root) kernel, which combines the results of the initial two halves into a final list of results:

```
# combine results
result1 := dist[wait](task1);
RETURN ([op(result1), op(result2)]);
```

(6) Displays the final result in the user interface of the local kernel when *dfactors* returns them to the *main* procedure.

In this regard, a Distributed Maple cluster comprising three nodes and implementing recursive mathematical computations discloses or renders obvious the claimed limitation where (1) the third node receives, from the second node, a result of a calculation that was performed by the second node in a different claim limitation (*e.g.*, it receives results of *dfactors* from the second node); (2) the third node performs a different calculation based on the received result (*e.g.*, it performs a calculation based on the received *dfactors* result); and (3) the third node sends the new result to be combined with other results and displayed by the first node (*e.g.*, the root node).

As yet another example of the obviousness of the claimed sequence of three-node operations, in or around 2000, Mr. Brent Seidel created and publicly released a PDE Solver program, which ran on a cluster of four Macintosh computers configured as an AppleSeed cluster using MPI. *See* Seidel.

The PDE Solver software included a computational kernel (at least under Plaintiff's apparent interpretation of the claim term "kernel") for solving partial differential equations ("PDEs"). The PDE kernel was deployed as one kernel instance per node across four nodes, with peer-to-peer communication between all nodes using asynchronous MPI calls—e.g., MPI_ISend, MPI_IRecv, and MPI_IWait—the same mechanisms described in the Asserted Patents' preferred embodiment.

In the PDE Solver software, node "0" functioned as the "first" node, providing a user interface and assembling the final results for display to the user. The three remaining nodes, designated nodes "1," "2," and "3," executed PDE computations. Any selection of these nodes maps to the claimed second and third nodes. The program's *pde_main_c* routine ran on node "0' and initialized the cluster, distributed data, and coordinated execution with the other nodes using the following functions:

```
33 void    main(void)
34 {
35
36     printf("Generic grid solver (PDE)\n\n");
37     pde_initialize();
38     pde_load_data();
39     pde_compute();
40     pde_write_data();
41 }
```

PDE Solver Code at pde_main.c.

Once initialized, each node, including the first, second and third nodes (e.g., nodes "0," "l," and "2" or "0," 2," "3") executed computational kernel code comprising, for example, the pde_compute() and pde_write_data() functionalities.

Once the first node (*e.g.*, node "0") sent commands and PDE data to the other nodes, the second node computed a first PDE result and communicated it to other nodes, including the third node:

```
79  **  Send and receive data from neighbour processors
80  */
81              if (pde_processor > 0)
82              {
83                  MPI_Isend(pde_data[0], pde_grid_cols*sizeof(grid_type), MPI_BYTE,
84                      pde_processor - 1, TAG_XTALK, MPI_COMM_WORLD, &xtalk_upper_send);
85                  MPI_Irecv(pde_prev_stripe, pde_grid_cols*sizeof(grid_type), MPI_BYTE,
86                      pde_processor - 1, TAG_XTALK, MPI_COMM_WORLD, &xtalk_upper_receive);
87              }
88              if (pde_processor < (pde_cluster_size - 1))
89              {
90                  MPI_Isend(pde_data[pde_grid_rows - 1], pde_grid_cols*sizeof(grid_type),
91                      MPI_BYTE, pde_processor + 1, TAG_XTALK, MPI_COMM_WORLD, &xtalk_lower_send);
92                  MPI_Irecv(pde_next_stripe, pde_grid_cols*sizeof(grid_type), MPI_BYTE,
93                      pde_processor + 1, TAG_XTALK, MPI_COMM_WORLD, &xtalk_lower_receive);
94              }
95  /*
96  **  Wait for communication to complete
97  */
98              if (pde_processor > 0)
99              {
100                 MPI_Wait(&xtalk_upper_send, &status);
101                 MPI_Wait(&xtalk_upper_receive, &status);
102             }
103             if (pde_processor < (pde_cluster_size - 1))
104             {
105                 MPI_Wait(&xtalk_lower_send, &status);
106                 MPI_Wait(&xtalk_lower_receive, &status);
107             }
108         }
```

PDE Solver Code at pde_compute.c.

The second processor used MPI_ISend to send the PDE result data to the third processor. On receiving the data, the third processor computed its PDE solution using the received result,

```
109         update_pass();
110         move_values();
```

and sent the result to the first node (e.g., node "0") using another asynchronous or non-blocking MPIISend call, identical to the Asserted Patents' preferred embodiment, as shown here:

```
124         else
125         {
126             remote[0] = sum;
127             remote[1] = sum2;
128             MPI_Send(remote, 2*sizeof(REAL), MPI_BYTE, 0, TAG_STATS, MPI_COMM_WORLD);
129         }
```

PDE Solver Code at pde_compute.c. The first node then assembled the results, including from the third node, for display to a user, with the functionality in pde_write_data(). *See id.* at pde_write.c.

This sequence again mirrors the claimed communication sequence, with explicit MPI messaging and asynchronous control of multi-node evaluation. Mr. Seidel's PDE Solver illustrates the routine nature of this architecture. A POSA could readily adapt the PDE Solver to any MPI-based cluster discussed in these contentions. Doing so would not require undue experimentation: the software architecture, messaging primitives, and task orchestration were all well-understood in the field by the early 2000s. Moreover, a POSA would be motivated to use such a system in conjunction with any of the clusters described herein, because parallelized PDE solving is a standard workload in scientific computing, and MPI-based clusters provide an ideal environment for scaling such computations.

As another example, Abdennadher discloses the "XWCH-p2p" system, an extension of the "XW" system architecture. *See* Abdennadher at 1. The XWCH-p2p system includes one coordinator node and multiple worker nodes, with "direct worker-to-worker communications" (*id.* at 2–4)—i.e., a peer-to-peer architecture. The XWCH-p2p system executes "Parallel and Distributed Applications (PDA)," which are "set[s] of communicating modules that can be represented by a data flow graph where nodes are modules and edges are communications inter-modules (Figure 1)." *Id.* at 3. Abdennadher further discloses an XWCH-p2p platform with one coordinator and three workers executing a PDA where module A communicates its results to modules B and C. *See id.* at 4. A POSA would have understood from this structure that:

- The coordinator assigns module A to worker A, which performs an initial computation;
- Worker A sends the result directly to worker B, which accepts the result and executes further tasks using that input; and

- Worker B then returns a result to the coordinator.

See Abdennadher at 4. This A → B → Coordinator flow exactly mirrors the claimed second node → third node → first node sequence. A POSA would have recognized that Abdennadher discloses the claimed communication order through its use of explicit task dependency, peer messaging, and coordinator-based orchestration.

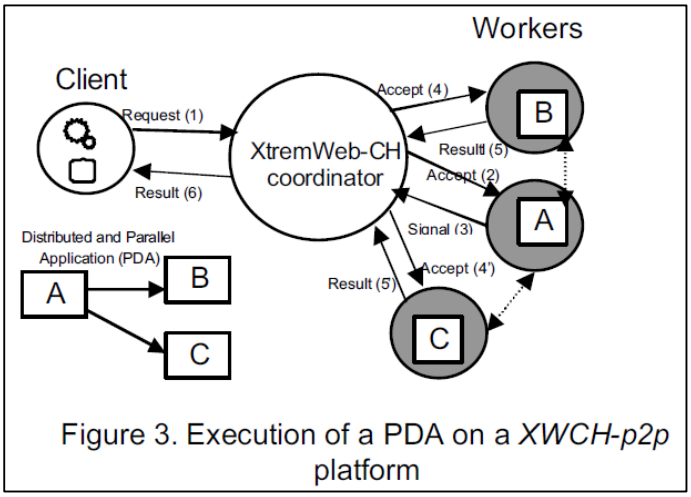


Figure 3. Execution of a PDA on a *XWCH-p2p* platform

Abdennadher at 4.

Similarly, Jia confirms that the claimed sequence of operations was conventional in the context of peer-to-peer (P2P) computing. *See* Jia at ii. Jia provides a specific example involving three peers—*Peer1*, *Peer2*, and *Peer3*—each with an input/output port and handler. *See id.* at 31–32. "[T]he P2p computation is started by the main thread of the active peer *Peer1*." *Id.* at 31. A message is sent from *Peer1's* output port to *Peer2's* input port via the channel *c1* between them. *See id.* at 31–32. *Peer2* receives the message and activates an input handler "to handle the received messages and execute the corresponding computations. During computation, the input handler sends data to its output port." *Id.* at 32. "The output port correspondingly sends the data as messages to *Peer3* through the channel *c2* between *Peer2* and *Peer3*. Again, the passive *Peer3* starts the computation once it receives the input messages." *Id.* "In the same way as *Peer2*, *Peer3*

sends the computation data to its connected peer, *Peer1* through channel *c3*. *Peer1* receives the final computation data which is collected from all of the three peers. Thus, the whole computation task is completed and the P2P application is terminated." *Id.*

Jia explains that this "simple example" illustrates how peer nodes perform dependent computations across a closed messaging loop, where each node passes results to the next in sequence. Jia at 31. Jia's disclosure is functionally identical to the claimed order of operations. And because Jia's example is expressly presented as foundational to the field, it demonstrates that the claimed ordering was well-understood, routine, and conventional. A POSA could have implemented the same logic on any cluster using standard message-passing protocols, such as MPI or PVM, without undue experimentation. Moreover, a POSA would have been motivated to do so. The sequence described by Jia enables the execution of dependent computation chains, where later-stage processing relies on intermediate results—a common structure for workloads that are difficult to parallelize using simple data partitioning.

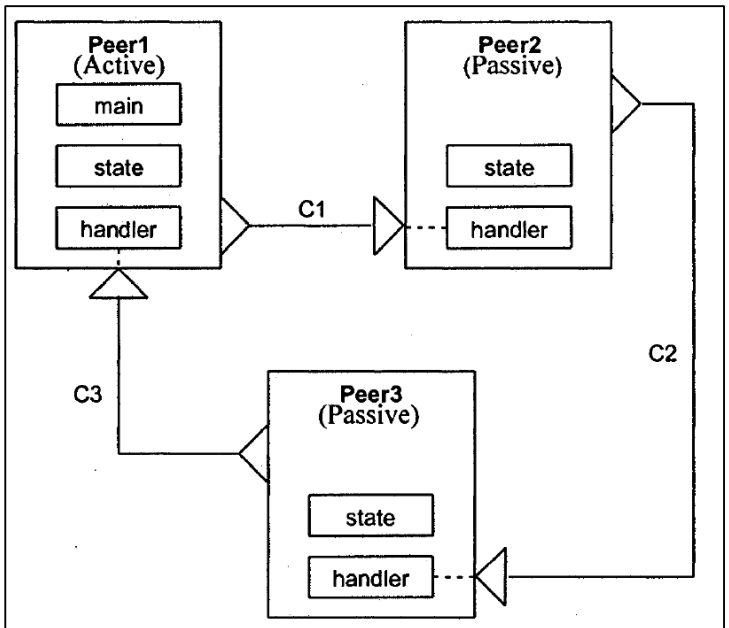


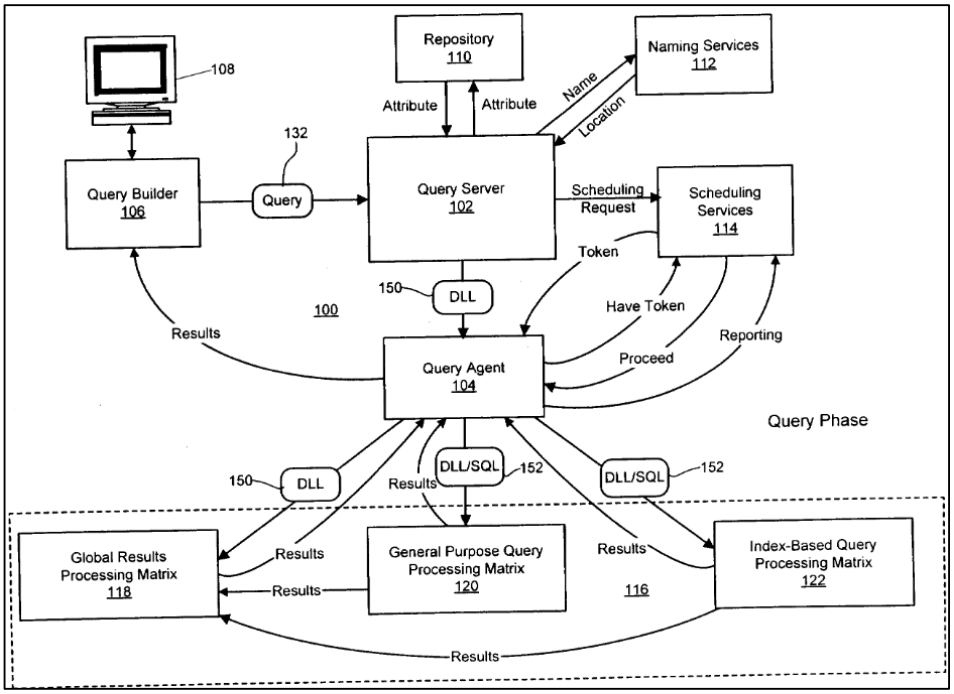
Figure 3.2 Abstract P2P Computing Example

Jia at 32.

Yet another example confirming the obviousness of the claimed three-node communication sequence appears in Bayliss 359, which is directed to "parallel processing of database queries in a parallel processing system." Bayliss 359 at [0001]. Bayliss 359 discloses a cluster-based architecture in which a query agent receives user instructions and generates an executable query, which it transmits to a general-purpose query processing matrix. *See id.* at [0013]-[0014]; *see also id.* at [0160]. Bayliss 359 then teaches a multi-stage dataflow model, where:

- A first processing matrix executes a first database operation based on the query;
- That result is passed to a second processing matrix, which performs additional computation; and
- The second matrix returns the result to the query agent.

Id. at [0087], [0089], FIG. 1.



Bayliss 359 at FIG. 1.

A POSA would have understood this model as another routine expression of staged computation involving intermediate result passing, implemented across distributed nodes. As with the other prior art systems discussed herein, a POSA could readily have adapted any of the disclosed computing clusters to follow the architecture described in Bayliss 359, using standard message-passing protocols such as MPI or PVM. Such adaptation would not have required undue experimentation. Moreover, a POSA would have been motivated to combine teachings from systems like Bayliss 359 with other clusters because the underlying model supports dependent computation, where later stages must operate on the results of earlier evaluations. This paradigm applies across computing contexts—from scientific workloads to distributed query processing—and the fact that Bayliss 359 teaches the same operational flow confirms the breadth and conventionality of the claimed sequence.

(e) Fourier Transforms and Other Computational Libraries

The specification of the Asserted Patents provides little disclosure on parallelizing Fourier transforms, describing ParallelFourier[list] as based on an existing Mathematica function: "[l]ike Fourier[], except that list is a two-or-three-dimensional list represented across the cluster, like for matrices." '768 patent at 20:20–25. However, a POSA at the time would have been deeply familiar with Fourier transforms. Given the exponential speed-up offered by Fast Fourier Transform ("FFT") and the increasing size of datasets, the motivation to parallelize FFTs on a cluster was self-evident. This motivation was widely discussed in the literature and implemented using established libraries like FFTW and communication protocols like MPI.

The Discrete Fourier Transform ("DFT") takes a function from its original domain, typically time or space, to the frequency domain. This is accomplished by sampling the original

function at a discrete set of evenly spaced points and then converting this information into a representation of the function as a series of sines and cosines of varying frequency.

The Fast Fourier Transform ("FFT") is an algorithm for computing the Discrete Fourier Transform that reduces the computational complexity of the DFT from order N^2 to order N Log N, where N is the number of points in the sample sequence. This is important as N becomes large (e.g., if N = 1,000 then N^2 = 1,000,000 and N Log_{10} N = 3,000), as the time to compute the DFT is proportional to these values. Though performing a DFT or FFT without the aid of a computer or calculating device is possible, the time to do so would be prohibitive and the process prone to errors.

The FFT is widely used in a variety of disciplines (e.g., engineering, mathematics, physics, music, etc.) and has numerous applications to audio, video, noise reduction/cancelation, radar, etc. A classic use case is taking an analog recording of a piece of music and turning it into a digital recording, for example, converting a song on a record to an MP3 digital audio file.

The typical undergraduate in the physical sciences, engineering, etc., at the earliest-claimed priority date would most certainly have known what an FFT/DFT is, how to employ it in practice, and would likely have had experience implementing it or would have been able to do so without difficulty. *See, e.g.*, Trefethen at 2 (describing MATLAB as "containing dozens of high-level commands such as . . . fft (fast Fourier transform)"). Parallelizing an FFT on a computer cluster with MPI or similar peer-to-peer communications was well-known in the art and would have been obvious to apply to the systems and references disclosed in these Invalidity Contentions.

Cramer, for example, teaches a method for parallelizing FFT on a "cluster of workstations" interconnected with "commodity networking hardware" such as Ethernet and using MPI or PVM protocols for message passing. Cramer at 1–2. This type of cluster is substantially similar to the

expected configuration of a cluster in the Claim Chart Exhibits. Cramer describes his approach as based on the FFTW library, discussed below. Cramer's example is a 3D FFT, and it follows the data "slice" approach used by most of the leading prior art, as well as the apparent implementation used by the preferred embodiment of the Asserted Patents. "The FFT starts with the assumption that each processor contained one slab of the 3D data space (see Figure 1)[:]"

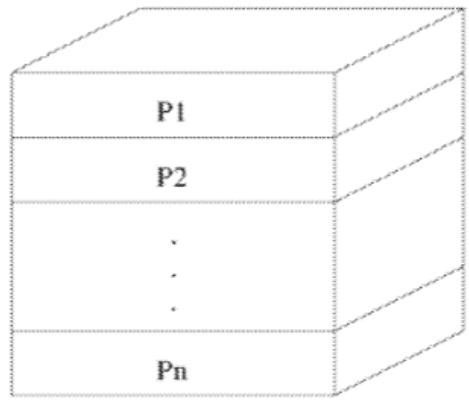


Figure 1: Processor representation of the 3D data space.

Cramer at 2.

Each processor then computes a 2D FFT on its set of slices, and sends the results to the other processors. *Id.* The processors then compute a transposed 3D FFT. *Id.* Thus, this algorithm makes straightforward use of MPI or PVM inter-node communication and computes a Fourier transform across the cluster nodes in parallel. Cramer's algorithm is an example of knowledge in the art that was obvious to use with, for example, AppleSeed, Distributed Maple, Intel Paragon Maple, Matlab*P, MatlabMPI, MultiMATLAB, Pooch, SEM, PVMaple, Whitney, and the Stony Brook cluster.

Cramer based his approach on the well-established FFTW (Fastest Fourier Transform in the West) software package that implemented FFT in an MPI cluster. It was well-documented, and

its source code was available for anyone of skill to download, review and use, motivating a POSA to apply its teachings to clusters implementing MPI or any similar protocol, such as PVM or the message passing scheduler protocol in Distributed Maple. The obviousness of such a combination is confirmed by the fact that some of the prior art MPI clusters in these Invalidity Contentions, such as Matlab*P, implemented the actual FFTW library:

> The main idea in MATLAB∗P is that data exist on the parallel server as distributed matrices. Any operations on a distributed matrix (which exists in the MATLAB front end only as a handle) will be relayed to the server transparently. The server calls the appropriate routine from a parallel numerical library (e.g. ScaLAPACK, FFTW, . . .) and the results stay on the server until explicitly requested.

Choy 2 at 336.

FFTW teaches a data slicing approach similar to Cramer, and would have been obvious to implement for the same reasons:

4.2.3 MPI Data Layout

The transform data used by the MPI FFTW routines is *distributed*: a distinct portion of it resides with each process involved in the transform. This allows the transform to be parallelized, for example, over a cluster of workstations, each with its own separate memory, so that you can take advantage of the total memory of all the processors you are parallelizing over.

In particular, the array is divided according to the rows (first dimension) of the data: each process gets a subset of the rows of the data. (This is sometimes called a "slab decomposition.") One consequence of this is that you can't take advantage of more processors than you have rows (e.g. `64x64x64` matrix can at most use 64 processors). This isn't usually much of a limitation, however, as each processor needs a fair amount of data in order for the parallel-computation benefits to outweight the communications costs.

Below, the first dimension of the data will be referred to as '`x`' and the second dimension as '`y`'.

FFTW supplies a routine to tell you exactly how much data resides on the current process:

```
void fftwnd_mpi_local_sizes(fftwnd_mpi_plan p,
                            int *local_nx,
                            int *local_x_start,
                            int *local_ny_after_transpose,
                            int *local_y_start_after_transpose,
                            int *total_local_size);
```

Given a plan `p`, the other parameters of this routine are set to values describing the required data layout, described below.

`total_local_size` is the number of `fftw_complex` elements that you must allocate for your local data (and workspace, if you choose). (This value should, of course, be multiplied by `n_fields` if that parameter to `fftwnd_mpi` is not `1`.)

The data on the current process has `local_nx` rows, starting at row `local_x_start`. If `fftwnd_mpi` is called with `FFTW_TRANSPOSED_ORDER` output, then `y` will be the first dimension of the output, and the local `y` extent will be given by `local_ny_after_transpose` and `local_y_start_after_transpose`. Otherwise, the output has the same dimensions and layout as the input.

FFTW Manual at 43.

The teachings of Cramer and/or FFTW could be implemented by a POSA without undue experimentation by a set of send calls and FFT computations (available in each of the computer clusters disclosed in these Invalidity Contentions), causing a subset of data to be processed by each of the cluster nodes in parallel, such that the Fourier transform is evaluated across the cluster in parallel.

Other well-known systems discussed similar parallel computation of FFTs. MultiMATLAB teaches "parallel MEX routines [that] would perform matrix multiplications, solve for eigenvalues, compute Fourier transforms, and so on." Menon at 10. Whitney similarly teaches the use of NAS benchmarks—widely used to measure parallel performance—which teach

a parallel implementation "of the 3-D FFT PDE benchmark [that] follows a fairly standard scheme."

> The 3-D array of data is distributed according to-planes of the array– one or more planes are stored in each processor. The forward 3-D FFT is then performed as multiple 1-D FFTs in each dimension.

NAS Benchmark at 8.

Another exemplar reference is Howard 938. Howard 938 teaches a toolkit of algorithms for parallelizing the Fourier transform and describes motivations for doing so.

For example, Howard 938, like Cramer and FFTW, also describes performing a 2D Fast Fourier Transform in two stages, first by performing a one-dimensional Fast Fourier Transform ("FFT") on the rows of the input array by splitting the rows and their associated calculations among multiple nodes; and, second, performing a one-dimensional FFT on the columns of the input array by splitting the columns and their associated calculations among multiple nodes and then combining the results. Howard 938 at [0142]–[0150], Figs. 67–69. In this approach, each computer node operates on a portion of the FFT dataset in parallel. Together, the nodes comprise "a call that calculates a Fourier transform across the computer cluster in parallel" as '582 claim 10 and '621 claim 9 require. The distributed data "slice" approach also discloses '768 patent claim 36, which requires "the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node."

Howard 938's approach is also consistent with the preferred embodiment of the Asserted Patents, in which each node "returns the portion of the Fourier-transformed array meant for that processor." '768, 20:20–25. Unlike the '768 specification, Howard 938's teaching is much more detailed.

Thus, Howard 938 teaches the "slicing" approach also taught by Cramer and FFTW. A POSA would have understood this approach to be especially applicable to computer clusters such as those disclosed in these Invalidity Contentions, including the Claim Chart Exhibits. This applicability was known in the art and described in publications such as Cramer and FFTW Manual. Accordingly, a POSA would have found it obvious to implement Howard's "slicing" approach with the computer clusters in these Invalidity Contentions. Howard 938's teachings could be implemented by a POSA without undue experimentation by a set of send calls and FFT computations (available in each of the computer clusters disclosed in these Invalidity Contentions), causing a subset of data to be processed by each of the cluster nodes in parallel, such that the Fourier transform is evaluated across the cluster in parallel.

A POSA would have been motivated to combine the computer clusters in these Invalidity Contentions with Howard 938 to parallelize Fourier transforms. These clusters and Howard 938 all address the same solution to the same problem—using parallelization to speed up mathematical operations—with Howard 938 motivating the use of parallelization to speed up Fourier transforms, specifically. Howard 938 at [0142]-[0147], [0437]-[0447]. In fact, during the prosecution of the '768 patent, the Examiner rejected claims reciting the parallelized performance of Fourier transforms based in part on Howard 938, finding:

> In analogous art Howard teaches the computer cluster node of claim 37, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.

ACS_0000566 ('768 patent file history).

Howard 938 contains express motivation for performing parallel Fourier transforms, which would have motivated a POSA to add Fourier transforms to the functions already available for the

computer clusters disclosed in these Invalidity Contentions, including the Claim Chart Exhibits. For example, Howard 938 teaches that parallelizing Fourier operations using "slicing" provides flexibility to handle arbitrarily large images: "The partitioning can handle an arbitrary number of parallel processing nodes and arbitrarily large images." Howard 938 at [0438].

Yet other references provide additional motivation. Chu provides further motivation, as it likewise explains the wide applicability of Fourier transforms and the benefits of parallelizing them, stating that "many researchers have recognized the practical importance of minimizing computing time by parallelizing sequential FFT algorithms in various ways for today's high-performance multiprocessor computers," thus proposing the same solution to the same problem as in the clusters disclosed in these Invalidity Contentions, using parallelization to speed up mathematical operations. Chu at 95–96.

More generally, the benefits of parallelizing FFTs on computer clusters were well known, and would have motivated a POSA. Cramer, for example, teaches that such clusters "are capable of achieving the performance of traditional supercomputers (on certain problems) at a significantly reduced cost" and states that "the use of this parallel FFT in our code allowed for a great deal of improvement in its performance." Cramer at 1. In addition, FFTW provides further motivation, as, like Cramer, it teaches parallelizing FFT in a cluster with MPI messaging, which is similar to the messaging scheme used by the clusters disclosed in the Claim Chart Exhibits.

For example, FFTW documentation from 2003 teaches a parallelized FFT implementation using MPI:

4.2.3 MPI Data Layout

The transform data used by the MPI FFTW routines is *distributed*: a distinct portion of it resides with each process involved in the transform. This allows the transform to be parallelized, for example, over a cluster of workstations, each with its own separate memory, so that you can take advantage of the total memory of all the processors you are parallelizing over.

In particular, the array is divided according to the rows (first dimension) of the data: each process gets a subset of the rows of the data. (This is sometimes called a "slab decomposition.") One consequence of this is that you can't take advantage of more processors than you have rows (e.g. `64x64x64` matrix can at most use 64 processors). This isn't usually much of a limitation, however, as each processor needs a fair amount of data in order for the parallel-computation benefits to outweight the communications costs.

Below, the first dimension of the data will be referred to as '`x`' and the second dimension as '`y`'.

FFTW supplies a routine to tell you exactly how much data resides on the current process:

```
void fftwnd_mpi_local_sizes(fftwnd_mpi_plan p,
                            int *local_nx,
                            int *local_x_start,
                            int *local_ny_after_transpose,
                            int *local_y_start_after_transpose,
                            int *total_local_size);
```

Given a plan `p`, the other parameters of this routine are set to values describing the required data layout, described below.

`total_local_size` is the number of `fftw_complex` elements that you must allocate for your local data (and workspace, if you choose). (This value should, of course, be multiplied by `n_fields` if that parameter to `fftwnd_mpi` is not `1`.)

The data on the current process has `local_nx` rows, starting at row `local_x_start`. If `fftwnd_mpi` is called with `FFTW_TRANSPOSED_ORDER` output, then `y` will be the first dimension of the output, and the local `y` extent will be given by `local_ny_after_transpose` and `local_y_start_after_transpose`. Otherwise, the output has the same dimensions and layout as the input.

FFTW Manual at 43.

Moreover, if the claims are construed to encompass stream computing, BLAS and FFTW libraries were implemented with Brook stream computing language for parallel execution on stream multiprocessors:

To test the effectiveness of our pass-splitting technique, we applied it to two kernels: Mat4Mult, which multiplies two streams of 4x4 matrices, producing a single 4x4 matrix (4 `float4`s) output stream; and Cloth, which simulates particle-based cloth with spring constraints, producing updated particle positions and velocities. We tested two versions of each kernel. Mat4Mult4 and Cloth4 were compiled with hardware support for 4 `float4` outputs, requiring only a single pass to complete. The Mat4Mult1 and Cloth1 were compiled for hardware with only a single output, forcing the runtime to generate separate shaders for each output.

Brook for GPU 2004 at 781.

BLAS **SAXPY** and **SGEMV** routines: The BLAS (Basic Linear Algebra Subprograms) library is a collection of low-level linear algebra subroutines [Lawson et al. 1979]. SAXPY performs the vector scale and sum operation, $y = ax + y$, where x and y are vectors and a is a scalar. SGEMV is a single-precision dense matrix-vector product followed by a scaled vector add, $y = \alpha Ax + \beta y$, where x, y are vectors, A is a matrix and α, β are scalars. Matrix-vector operations are critical in many numerical applications, and the double-precision variant of SAXPY is a core computation kernel employed by the LINPACK Top500 benchmark [2004] used to rank the top supercomputers in the world. We compare our performance against that of the optimized commercial Intel Math Kernel Library[Intel 2004] for SAXPY and the ATLAS BLAS library[Whaley et al. 2001] for SGEMV, which were the fastest public CPU implementations we were able to locate. For a reference GPU comparison, we implemented a hand-optimized DirectX version of SAXPY and an optimized OpenGL SGEMV implementation. For these tests, we use vectors or matrices of size 1024^2.

Id. at 782.

FFT: Our Fourier transform application performs a 2D Cooley-Tukey fast Fourier transform (FFT) [1965] on a 4 channel 1024 by 1024 complex signal. The fast Fourier transform algorithm is important in many graphical applications, such as post-processing of images in the framebuffer, as well as scientific applications such as the SETI@home project [Sullivan et al. 1997]. Our implementation uses three kernels: a horizontal and vertical 1D FFT, each called 10 times, and a bit reversal kernel called once. The horizontal and vertical FFT kernels each perform 5 floating-point operations per output value. The total floating point operations performed, based on the benchFFT [Frigo and Johnson 2003] project, is equal to $5 \cdot w \cdot h \cdot channels \cdot \log_2(w \cdot h)$. To benchmark Brook against a competitive GPU algorithm, we compare our results with the custom OpenGL implementation available from ATI at [ATI 2004a]. To compare against the CPU, we benchmark the heavily optimized FFTW-3 software library compiled with the Intel C++ compiler [INTEL 2003].

Id. at 782.

More generally, a POSA would have been motivated to implement an "advanced functions module" like that disclosed in the Asserted Patents for evaluating mathematical expressions across a computer cluster in parallel. Numerous distributed computing systems and computational libraries available before the priority date—including systems based on MPI, PVM, and stream computing—were explicitly designed to enable the distributed execution of mathematical computations. These systems provided foundational infrastructure for message passing, task coordination, and load balancing across multiple nodes. At the same time, well-established numerical libraries such as BLAS, LAPACK, FFTW, NumPy, and others offered a rich set of optimized routines for common linear algebra, statistical, and transform-based operations.

A POSA would have recognized that combining distributed computing frameworks with such numerical libraries would enable the efficient evaluation of complex expressions across multiple nodes, thereby improving scalability and performance. Indeed, these combinations were widely taught, documented, and implemented in numerous academic and commercial systems—including AppleSeed, Distributed Maple, Matlab*P, MatlabMPI, Merrimac, MultiMATLAB, Pooch, SEM, and others. The ability to wrap or script such functions as callable routines in higher-level environments (such as Maple, Matlab, Mathematica, or Python) also was well-known, and tools like MPI bindings, parallel toolkits, and scripting language extensions enabled practitioners to express distributed function calls in a modular and reusable way.

Accordingly, implementing an "advanced functions module" comprising calls that evaluate mathematical expressions in parallel across a computer cluster would have been a natural and obvious extension of the capabilities available in the art. A POSA would have had a reasonable expectation of success in developing such a module using conventional programming techniques, given the mature state of MPI-based parallel processing, the availability of optimized

computational libraries, and the well-understood mechanisms for distributing workloads and aggregating results in cluster environments. No specialized hardware or undisclosed techniques would have been required to do so, and a POSA would have been able to implement such a system without undue experimentation.

(f) Special Purpose and Multicore Processors

Plaintiff also asserts claims that recite processors with a plurality of processing cores and/or multiple processors on one die. *See, e.g.*, '768 patent, claim 1; '621 patent, claim 20. As discussed in Section II.C (Admitted Prior Art), *supra*, the Asserted Patents admit that such processors were conventional and not an inventive feature. *See id.* at 7:10–32 ("For example, if the processors **112***a-b* were each Core 2 Duo(R) processors having two processing cores, computer system **110** would have four nodes in all."). In addition, Plaintiff asserts claims reciting a "special purpose microprocessor," which the common specification teaches can be a digital signal processor. *See, e.g.*, *id.* at 7:21–23, claim 39. Digital signal processors were also conventional and not an inventive feature. *See* Section II.C (Admitted Prior Art), *supra*.

Processors with multiple cores were conventional and common well before the alleged inventions by Plaintiff. Such processors were widely used in computing in general, and cluster computing specifically. As just one example, the Matlab*P parallel cluster system and publications disclose the use of dual core processors:

> *A. Test Platform*
>
> - Beowulf cluster with 9 nodes (2 nodes are used in the tests).
> - Dual-processor nodes, each with two 1.533GHz Athlon MP processors.
> - 1GB DDR RAM on each node. No swapping occurred during benchmarks.
> - Fast ethernet (100Mbps/sec) interconnect. Intel Etherfast 410T switch.
> - Linux 2.4.18-4smp

Choy 1 at 4.

Likewise, DSPs were known to be advantageous in cluster computing for certain types of problems. As one example, Nayak teaches the advantages of cluster computing on DSPs:

> The DSP architecture has some custom hardware for multiply and accumulate type of operations, which make them ideal for signal and image processing applications in comparison to the general purpose processors. Hence, it is wise to implement computation intensive matrix operations on the DSP as they would take a longer time on general purpose processors.

Nayak at 4.

Nayak teaches a cluster with a combination of DSP and/or FPGA nodes, in addition to other processor nodes. It uses a Matlab kernel and MPI for peer-to-peer message passing, making it obvious to combine with Matlab*P and other references using MPI and similar peer-to-peer protocols, such as PVM or Distributed Maple's scheduler messaging mechanism:

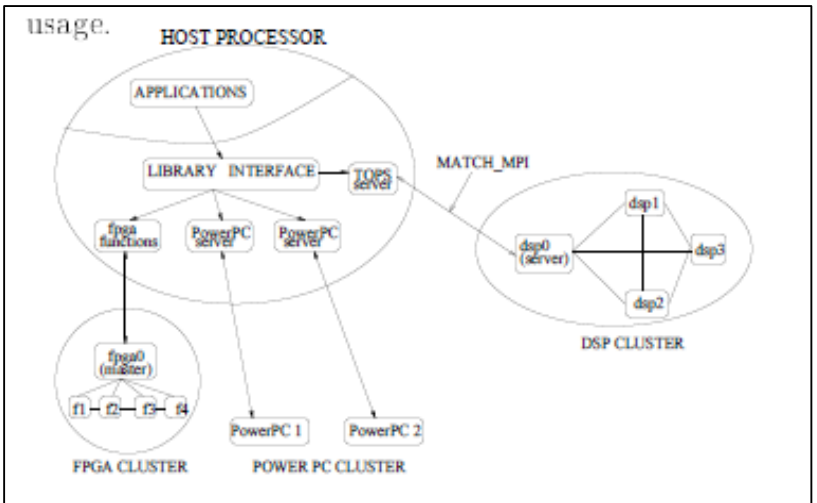


Figure 4: Software Architecture

Nayak, Fig. 4.

As a further example of knowledge in the art regarding multiple core and special purpose processors, Kim 592 teaches: "When a particular application/project/job requires more processing power than a single processor is capable of providing, it becomes necessary to provide a coprocessor, such as a digital signal processor (DSP) or a floating point unit (FPU)." Kim 592 at 1:10–15. Accordingly, the use of special purpose processors was conventional, and obvious whenever extra computational power was desired.

Moreover, the Stony Brook cluster was implemented and publicly known and used at Stony Brook University as early as 2003. It comprised over 30 nodes, each node comprising a CPU with at least a plurality of processing cores and an NVIDIA GPU: "We have built a cluster with 32 computation nodes connected by a 1 Gigabit Ethernet switch. Each node consists of a dual-CPU HP PC with an nVIDIA GeForce FX 5800 Ultra." Fan 1 at Introduction. Each node was interconnected with each other node with a peer-to-peer communication mechanism comprising a high-speed network for exchanging MPI protocol messages between the nodes:

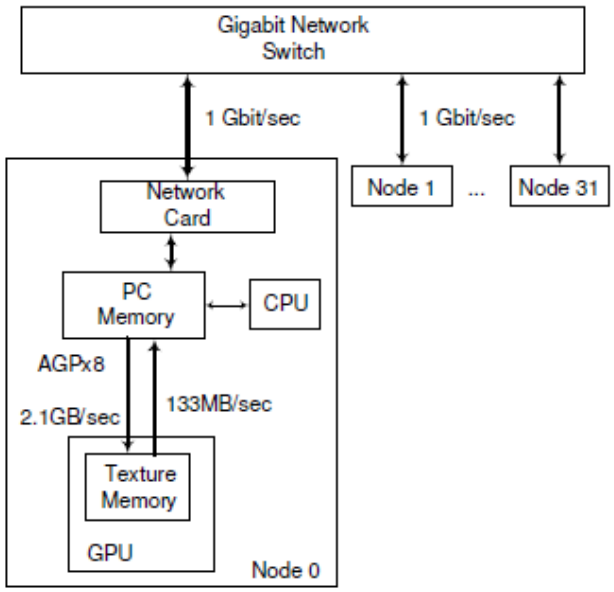


Figure 3: The architecture of our GPU cluster. (Although all 32 nodes have the same configuration, we show only node 0 in detail.)

Fan 1 at Fig. 3; *id.* at Our GPU Cluster.

Figure 2: The Stony Brook Visual Computing Cluster.

Fan 1 at Fig. 2; *id.* at Scaling LBM onto the GPU Cluster ("We performed experiments on the GPU cluster using MPI and replicated these experiments using communication code that we developed using TCP/IP sockets.").

As an example, the use of GPUs for parallel processing in computer clusters with peer-to-peer communications was known in the art well before the alleged inventions of the Asserted Patents. To the extent Plaintiff contends that the Asserted Patents enable and describe the application of their claims to GPU-based parallel processing, it would have been obvious to combine the teachings of any of the computer cluster systems and references in these contentions with at least the Stony Brook cluster as described by Fan et al. For example, the Stony Brook cluster implemented the MPI messaging protocol, making it obvious to combine with other MPI-based systems and references such as AppleSeed, Matlab*P, MatlabMPI, MultiMATLAB, Pooch, or SEM, or with references with similar protocols, such as PVMaple or Distributed Maple.

As another example, Plaintiff has admitted that Athlon processors, which are a family of processors by AMD, constitute prior art to all Asserted Patents. *See* '768 patent at 7:19. One member of this family, the "Athlon 64 X2," was a dual-core processor released before the earliest-claimed priority date. *See* AMD Article. Similarly, Intel's Pentium D dual-core processors and AMD's 8xx series dual-core Opteron chips were commercially available before the priority date. *See id.*; Erez. The benefits of using multicore processors such as the ones listed above were well-known and would have motivated a POSA. For example, AMD Article notes that dual core processors have improved performance over single core processors, as "engineers realized that simply ratcheting up the clock speed of single-core chips was creating too much heat and not producing the same improvements seen in previous models. They discovered that performance could be boosted by creating two computing cores and running them at a slower speed on a single

chip." *See* AMD Article at 2. A POSA would also have understood that multicore processors typically placed multiple processing cores on a single die, allowing for independent or parallel execution of software threads. *See* Erez at 24 ("Modern processors often contain multiple cores on the same die . . . ."). Accordingly, it would have been obvious to a POSA to use these multicore processors in any of the distributed computing systems disclosed in these Invalidity Contentions—including those implementing FFT, peer-to-peer messaging, or kernel parallelization—given their widespread availability, performance advantages, and compatibility with standard clustering frameworks.

(g) Cluster Initialization and Establishing Communication Among Nodes

Plaintiff has asserted claims that recite a cluster initialization process, which may comprise establishing communication among two or more nodes. *See, e.g.*, '768 patent, claims 1, 26. As discussed in Section II.C (Admitted Prior Art), *supra*, the Asserted Patents themselves admit that such initialization processes were conventional and not inventive. *See, e.g.*, *id.* at 22:26–47.

For example, the Asserted Patents acknowledge that Dauger 924 (also known as the "'924 patent")—which "discloses a method and system for parallel operation and control of computer clusters"—constitutes prior art. *See* Dauger 924 at 22:29–33. The '924 patent discloses a method where "'node 0,' or the computer controlling the cluster," discovers another node and receives information from the other node about its capabilities, after which "[n]ode 0 breaks the active communications link with Node n 614." *See id.* at Fig. 6, 4:55–56, 5:34–6:27. A POSA would have understood that at least this method of the '924 patent comprises a cluster initialization process. A POSA would readily have understood that this process constitutes a cluster initialization routine, and that for node 0 to later "break" a communications link, it must have first established

that link. Thus, the ’924 patent confirms that initialization and inter-node communication setup were well-understood, routine, and conventional in the prior art.

Similarly, Distributed Maple discloses a built-in command—*dist[initialize]*—for launching a cluster session and establishing communication between nodes. *See* Schreiner2 at 12–14. The *dist[initialize]* command “initializes a distributed Maple session by starting remote processes and establishing the corresponding network connections.” *Id.* In a documented example, Schreiner1 describes a session in which machine “aquila” uses *dist[initialize]* to start remote Maple kernels on “virgo” and “andromeda”, forming a connected distributed session. *See* Schreiner1 at 309–10. The session proceeds to evaluate mathematical expressions across the nodes and returns results before shutting down—precisely the kind of initialization and distributed operation described in the Asserted Claims. *See id.*

```
aquila!33> maple
    |\^/|     Maple 6 (IBM INTEL LINUX22)
._|\|   |/|_. Copyright (c) 2000 by Waterloo Maple Inc.
 \  MAPLE  /  All rights reserved. Maple is a registered trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> read `dist.maple`;
Distributed Maple V1.1.7 (c) 1998-2001 Wolfgang Schreiner (RISC-Linz)
See http://www.risc.uni-linz.ac.at/software/distmaple
> dist[initialize]([[virgo,linux], [andromeda,octane]]);
```

Schreiner1 at 309.

```
connecting virgo...
connecting andromeda...
                                        okay

> t1 := dist[start](int, x^n, x):
> t2 := dist[start](int, x^n, n);
> dist[wait](t1) + dist[wait](t2);
                                        (n + 1)      n
                                   x           x
                                   -------- + -----
                                    n + 1      ln(x)

> dist[terminate]();
                                        okay

> quit;
```

Id. at 310.

PVMaple further reinforces that cluster initialization and inter-node communication were well known. Petcu2, which describes the PVMaple system, explains that its design principles were "very similar to that of Distributed Maple." Petcu2 at 217. Petcu2 further discloses that "[i]nitialization of a Parallel Virtual Maple session, creation of Maple processes and inter-processes communications are provided by pvm.m package." *Id.* PVMaple used the *pvm.m* package to support cluster session initialization, creation of Maple processes, and setup of inter-process communications. *See id.* Petcu2 describes the use of *pvm[spawn]* to launch Maple processes on remote machines and includes example sessions where a local node initializes and communicates with two remote nodes via PVM messaging. *See id.* at 218–19.

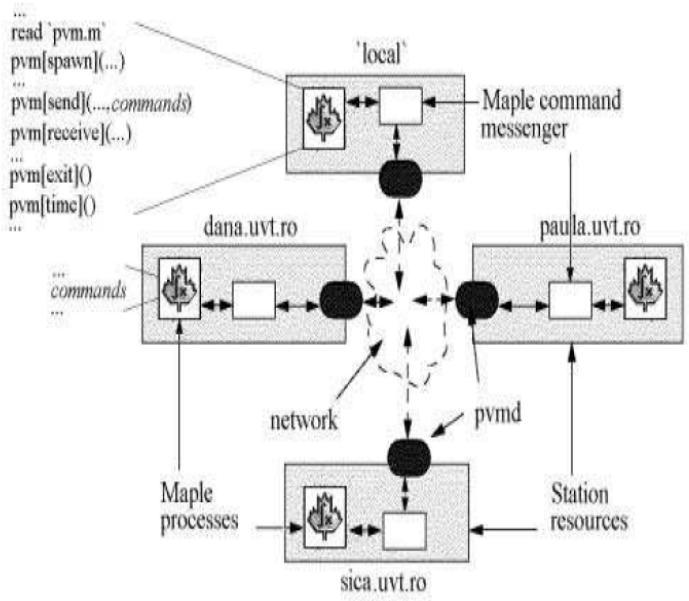


Fig. 1. Processes and inter-communication routes

Table 2. Functions and constants

Function Synopsis		Meanings/Parameters
spawn	pvm[spawn](*sequence*)	create Maple processes; *sequence* is a sequence of elements like [*station_name*,*processes_no*]

Petcu2 at 218.

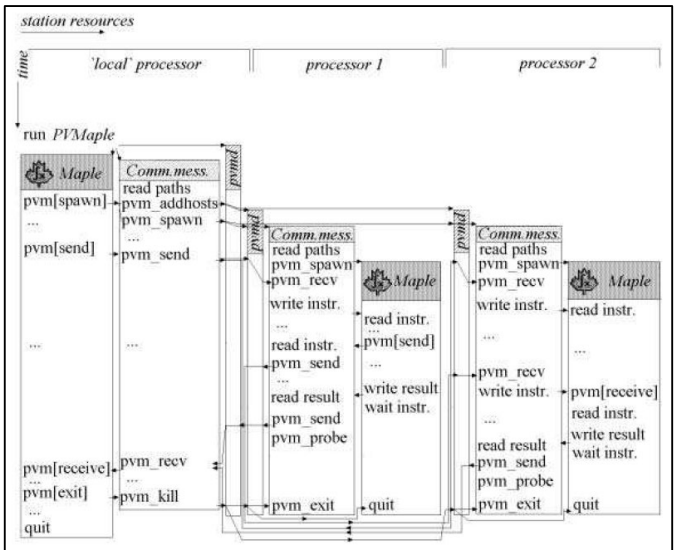


Fig. 2. Communications and components inter-dependencies

Id. at 219.

These examples show that cluster initialization processes involving two or more nodes were routine and expected in distributed computing systems prior to the priority date of the Asserted Patents. To the extent any of the systems disclosed in these Invalidity Contentions do not expressly or inherently disclose a cluster initialization process that includes communication among nodes, a POSA would have found it obvious to implement one based on teachings from AppleSeed, Dauger, Distributed Maple, Matlab*P, MatlabMPI, MultiMATLAB, Pooch, PVMaple, pyMPI, SEM, Whitney, or any number of other cluster systems in the art. A POSA would have been motivated to implement such functionality, as initializing a cluster and establishing communication among nodes is a necessary precondition for distributed operation. Doing so would have required no inventive insight or undue experimentation.

E. Lack of Secondary Indicia of Nonobviousness

Intel is not aware of any evidence that would tend to establish any secondary considerations of non-obviousness. This lack of evidence further renders the Asserted Claims obvious. Proving

any such secondary considerations is Plaintiff's burden. *See, e.g.*, *ZUP, LLC v. Nach Mfg., Inc.*, 896 F.3d 1365, 1373 (Fed. Cir. 2018) ("[A] patentee bears the burden of production with respect to evidence of secondary considerations of nonobviousness."). Intel expressly incorporates by reference arguments NVIDIA's argument in its Final Invalidity Contentions that Plaintiff has not, and cannot, establish secondary considerations of nonobviousness. Accordingly, Intel reserves all rights regarding its full contention in this respect until after Plaintiff completes its final and binding disclosure of any such evidence and contentions. In the meantime, Intel notes the complete lack of any such evidence in the record.

With regard to the "commercial success" factor, Plaintiff has not shown any commercial success of the alleged inventions of the Asserted Patents, including of the SET or SEM products that it alleges embody the '768 patent claims. For example, despite submitting voluminous argument and multiple declarations in the NVIDIA IPRs, including IPR2021-00019 and IPR2021-00020, Plaintiff failed to address the commercial success prong. Plaintiff has not presented any evidence of commercial sales. Furthermore, Plaintiff has failed to establish any nexus between the claimed invention and any alleged commercial success. *Ormco Corp. v. Align Technology, Inc.*, 463 F.3d 1299, 1311–12 (Fed. Cir. 2006). Commercial success cannot be based on unclaimed features or those known in the art. *Id*.

While Plaintiff has not shown any actual commercial success, it also has not articulated any nexus between any alleged commercial success and any claimed feature that is unknown in the art. As set forth in these Invalidity Contentions, the Asserted Claims do not include any novel elements, but are, at best, an obvious combination of an existing Mathematica kernel and conventional MPI communication protocol. Likewise, the SEM and SET products alleged by Plaintiff to practice the '768 patent claims are merely a combination of a Mathematica or another

prior art kernel and the MPI communication protocol, precluding any showing of commercial success based on claimed novel features.

Alleged Commercial Success Via Licensing. Plaintiff has presented no evidence of commercial success via a licensing program.

Long-felt But Unresolved Needs. Plaintiff has presented no evidence of any long felt and unresolved need. In the POPRs filed in NVIDIA's IPRs, named-inventor Dauger claimed that SEM had advantages over the prior art because it "interposes a communication layer between the front end user interface (view) and the back end Mathematical kernels (model) on each node." EX-2006, ¶ 24. However, Dauger provided no evidence or explanation for why interposing the communications software between the user interface and the kernel, as compared to connecting the communications software in some other way, would make any difference at all to the user. Moreover, Dauger's statement rested on an incorrect claim construction requiring "interposition" and thus failed to establish a nexus to the challenged claims. *Kennametal*, 780 F.3d at 1385. In addition, Plaintiff did not claim that any such "interposition" is a requirement of the '768 patent claims, despite alleging that it is an essential feature of its alleged invention. Accordingly, there is no nexus between this alleged improvement and any claimed feature. In addition, the argument fails to provide any relevant secondary consideration because the element was already in the prior art. For example, Distributed Maple and PVMaple were used with Mathematica and its front end, as established in NVIDIA's invalidity contentions. Schreiner 7, 3 ("The user interacts with Distributed Mathematica via the Mathematica frontend which communicates with the scheduling program running in the background in order to transfer tasks to other Mathematica kernels.").

No Industry Praise. There is also no evidence of industry praise for the alleged invention of the Asserted Patents or any functionality that allegedly practices the Asserted

Patents. To the extent any praise is related to any functionality that allegedly practices the Asserted Patents, that praise is not due to the allegedly novel features of the Asserted Patents, but instead only to features present in the prior art, which is not a sufficient nexus to be relevant to the question of industry praise for purposes of obviousness. *See Muniauction, Inc. v. Thomson Corp.*, 532 F.3d 1318, 1328 (Fed. Cir. 2008). Praise of Intel's processors or software is not praise of the Asserted Patents.

In the POPRs filed in NVIDIA's IPRs, Plaintiff cited the declaration of Dr. Vineer Bhansali as evidence of "praise." Bhansali focuses on "load balancing," asserting that with SEM "I did not have to figure out how to load balance my code." EX-2007, ¶10. However, for secondary-considerations evidence to be relevant to obviousness, it must have a "nexus" to the claimed invention. *Ormco, Inc. v. Align Tech., Inc.*, 463 F.3d 1299, 1311–12 (Fed. Cir. 2006). Load balancing has no nexus because the Asserted Patents do not assert that load balancing was novel or non-obvious. *Kennametal, Inc. v. Ingersoll Cutting Tool Co.*, 780 F.3d 1376, 1385 (Fed. Cir. 2015) (rejecting evidence lacking a nexus to "what is both claimed and novel in the claim"). Indeed, the prior art cited in the NVIDIA petitions and in these Invalidity Contentions also teaches load balancing. For example, load balancing in Distributed Maple operates in substantially the same manner as the Asserted Patents' preferred embodiment. EX-1008, 316. *Geo. M. Martin Co. v. Alliance Machine Sys. Int'l*, 618 F.3d 1294, 1304–05 (Fed. Cir. 2010) (rejecting secondary-consideration evidence where the supposed need had already "been met by prior art machines").

In the POPRs filed in NVIDIA's IPRs, Plaintiff also cited the declaration of John Bancroft as further evidence of "praise." Yet, he admitted that his laboratory during the relevant time period "did not itself utilize" Plaintiff's product. EX-2008, ¶28. Instead, he claimed that, by "keeping the kernel intact and adding a separate inter- node parallelization and communication layer," SET

"did not require the programmer to have any specialized skill in writing parallel tasks." *Id.*, ¶¶25–26. This evidence is irrelevant because it ignores the closest prior art cited in these Invalidity Contentions. The publications cited in the petitions and NVIDIA's invalidity contentions teach the same solution of keeping Maple or Mathematica kernels intact and providing inter-node parallelization with the scheduler layer.

For example, Distributed Maple was built on the same principle of parallelizing instances of Maple or Mathematica kernels without requiring any modifications to the kernels:

> **Time-safety and portability**. Distributed Maple is built on top of a state of the art commercial computer algebra system. In contrast to other developments, the system only relies on basic interfaces to the Maple kernel and does not require any kernel extensions. Consequently, Distributed Maple has since 1998 survived four release changes (Maple 4–7) without major changes. The environment is so portable that applications can be executed in many different environments (from wide-area heterogeneous clusters to shared-memory multiprocessors). It is so general that it can be applied to schedule tasks of other computer algebra systems (e.g. Mathematica).

Schreiner 1 at 306.

Likewise, PVMaple used the same principle of adding parallelization software to unmodified Maple, Mathematica, or MATLAB kernels:

> We make the following assumptions about the software module that is to be interfaced with the system. (1) It has a user interactive interface, (2) it can be installed on several machines of a cluster or is accessible by each cluster machine via an NFS, (3) it has TCP/IP communication facilities, or its source code is available in a language having TCP/IP communication library, or it has I/O facilities, and (4) user knows how to split the problem into subproblems. Specifically, user wants to use the interface of an existing software module to launch several copies on nodes of a cluster and, in the next step, to send to them separate subproblems to be solved. Separately, we assume that cooperation between instances is possible. Finally, other users of the same cluster may want to use (within the same scenario) the same software module(s) within their applications. For example, each node of an 8-processor cluster has a copy of Maple running and various users may utilize groups of 1 through 8 kernels.

Petcu 6 at 453.

Unexpected Results. No evidence of any such unexpected results is known. As discussed above, the concepts contained in the Asserted Claims were already combined in the same manner as the asserted. These prior art systems, as described in the above-referenced exhibits, disclosed the same combination of elements, and the same result of that combination, that is recited in the claim. Thus, there were no unexpected results that arose from combining the well-known elements in the Asserted Claims.

In its POPRs in the NVIDIA IPRs, Plaintiff cited the "surprise" of one of the named inventors: "[u]pon testing SEM, I was *surprised* to learn that a cluster executing Mathematica and SEM performed much better than gridMathematica." EX-2006, ¶39 (emphasis added). The named inventor's testimony is the type of "self-serving" statement the Board has correctly rejected in the past. *In re Cree, Inc*., 818 F.3d 694, 702 (Fed. Cir. 2016). The testimony is also irrelevant because it compares the claimed invention against gridMathematica instead of the "closest prior art" cited in these Invalidity Contentions. *In re Harris*, 409 F.3d 1339, 1344 (Fed. Cir. 2005); *Trs. of Columbia Univ. v. lllumina, Inc.*, 2015 WL 4385761, *14 (Fed. Cir. 2015).

Failure of Others. No evidence of any such failure is known. In its POPRs in the NVIDIA IPRs, Plaintiff pointed to an alleged "failure of others," criticizing unnamed "automatic parallelizers or universal compilers" as supposedly unable "to produce accurate parallel code that was sufficiently optimized or advantageous enough to catch on." EX-2006, ¶37. Nevertheless, this argument is irrelevant because the claims neither recite "automatic" or "universal" parallelization nor require a specific level of optimization or advantageousness. *ABT Sys., LLC v. Emerson Elec. Co.*, 797 F.3d 1350, 1362 (Fed. Cir. 2015). Moreover, Plaintiff's argument again ignores the closest prior art cited in these Invalidity Contentions.

Skepticism by Experts. No experts or person of skill expressed skepticism about implementing the alleged inventions.

In the POPRs in NVIDIA's IPRs, Plaintiff cited criticisms leveled against Plaintiff and the alleged inventor Dauger by the Department of Energy ("DOE"). Far from favoring validity, those criticisms cut strongly against Plaintiff's attempts to establish any secondary considerations. The Federal Circuit has rejected alleged "skepticism" where it did not address issues of "technical infeasibility or manufacturing uncertainty." *In re Magna Elecs.*, 2015 WL 2110525, *4 (Fed. Cir. 2015). Here, the DOE did not express any skepticism that "auto-parallelization tools" like those proposed by the alleged inventors would work. To the contrary, the DOE recognized that "[n]umerous other auto-parallelization tools exist within the HPC community and have had various degrees of success." EX-2021, 1. Indeed, DOE reviewers expressly commented that the alleged inventors' proposal included nothing inventive over those other tools: "The development of tools such as SET are not new" (*id.*); "it is not innovative research or development" (*id.*); and "this proposal does not move beyond the state of the art" EX-2019, 2.

Instead, the DOE reviewers expressed skepticism over the bold performance claims made by the alleged inventors. One reviewer described the alleged inventors as presenting "an unverified solution to making parallel computing 'easy,'" stating that the proposal "provides no quantitative or qualitative evidence" supporting its claims. EX-2019, 2. Another reviewer stated that, "despite claims of 'quantum leap' in this proposal," "[t]his idea is not unique, and there is nothing in the proposal that moves beyond the current state-of-the-art." *Id.*, 3–4. This objective third party analysis shows that Plaintiff's performance claims are not credible and should be rejected.

The DOE reviewers also explained how the lack of commercial success cut against its claims of "quantum leap" technology. As one reviewer concluded, "if it was as easy as the applicants claim, HPC [High-Performance Computing] developers would be using SET instead of other traditional approaches." EX-2019, 3. But instead, Plaintiff had an "absence of identified customers" and "no sound plan to showing that this SET-based approach can be commercially viable." *Id.*, 2, 4. To this day, Plaintiff has not claimed any commercial success, severely undercutting any alleged secondary considerations.

Teaching Away by Others. No evidence of any such teaching is known.

Recognition of a Problem. As discussed above, the industry recognized the problem and had already discussed multiple approaches that implemented the Asserted Claims to solve that problem.

Copying of the Alleged Invention by Competitors. No credible evidence of any such copying is known. *See Amazon.com, Inc. v. Barnesandnoble.com, Inc.*, 239 F.3d 1343, 1366 (Fed. Cir. 2001) (allegedly copied feature must be an embodiment of the patented claims).

In sum, Intel is not aware of any credible evidence of secondary considerations that would favor a finding of validity.

IV. DOUBLE PATENTING

The Asserted Claims are invalid due to obviousness-type double patenting.

As the Federal Circuit discussed in *In re Hubbell*, obviousness-type double patenting has two separate and independent rationales:

> There are two justifications for obviousness-type double patenting. The first is "to prevent unjustified timewise extension of the right to exclude granted by a patent no matter how the extension is brought about." *Van Ornum*, 686 F.2d at 943-44 (quotation and citation omitted). The second rationale is to prevent multiple infringement suits by different assignees asserting essentially the same patented

> invention. *Fallaux*, 564 F.3d at 1319 (recognizing that "harassment by multiple assignees" provides "a second justification for obviousness-type double patenting"); *see also* Chisum on Patents § 9.04[2][b][ii] ("The possibility of multiple suits against an infringer by assignees of related patents has long been recognized as one of the concerns behind the doctrine of double patenting.").

See In re Hubbell, 709 F.3d 1140, 1145 (Fed. Cir. 2013).

Analyzing obviousness-type double patenting takes a two-step process. First, the differences between the claims of the patent being used as a reference and the claims at issue are identified. *Georgia-Pacific v. US Gypsum*, 195 F.3d 1322, 1326 (Fed. Cir. 1999) ("[A]nalysis of the claims at issue is the first step in determining if the second invention is merely an obvious variation of the first . . . . Because these two claims are so similar, we must look to see if there is anything to distinguish claim 1 of the '989 patent from claim 1 of the '569 patent. There are two differences between these claims."), *amended on rehearing*, 204 F.3d 1359 (Fed. Cir. 2000).

Second, whether those differences raise a patentable distinction must be determined. *Georgia-Pacific*, 195 F.3d at 1328 (finding that "[t]hese differences are not sufficient to render the claims patentably distinct"). "A later patent claim 'is not patentably distinct from an earlier claim if the later claim is obvious over, or anticipated by, the earlier claim.'" *Hubbell*, 709 F.3d at 1145 (*quoting Eli Lilly & Co. v. Barr Labs., Inc.*, 251 F.3d 955, 968 (Fed. Cir. 2001)). Where the recited claims at issue are not explicitly disclosed in the claims of the patent being used as a reference, Intel may rely on the knowledge of a POSA, admitted prior art, or disclosure in the prior art references relied upon by Intel.

Here, the Asserted Claims of the '768 and '621 patents are invalid for obviousness-type double patenting over the claims of the '034, '582, and/or '679 patents. The '034, '582, and/or '679 patents expire on May 4, 2027, before the '768 patent is set to expire on October 25, 2027, and also before the '621 patent is set to expire on May 29, 2027.

As shown in the following chart, the claims of the ’034 patent render the claims of the ’768 patent obvious. The Asserted Claims of the ’768 patent therefore are invalid for obviousness-type double patenting. *See Gilead Scis., Inc. v. Natco Pharma Ltd.*, 753 F.3d 1208, 1217 (Fed. Cir. 2014) (holding that a later-issued, but earlier expiring patent can invalidate a first-issued, but later expiring patent).

’768 Claim	**’034 Claim**
[1PRE] A computer cluster comprising:	[1PRE] A computer cluster comprising:
[1A] a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor, wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions; and	[1A] a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor, wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions; and
[1B] a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;	[1B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;
[1C1] wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and	[1C1] wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and
[1C2] a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is	[1C2] a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is

’768 Claim	’034 Claim
configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;	configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node;
[1C3] wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;	[1C3] wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;
[1D] wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;	[1D] wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;
[1E] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.	[1E] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels.
[4] The computer cluster of claim 1, wherein each of the nodes comprises one or more cluster node modules.	[3] The computer cluster of claim 1, wherein each of the nodes comprises one or more cluster node modules.
[20] The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.	[1E] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to

’768 Claim	’034 Claim
	communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels.
[21] The computer cluster of claim 20, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other.	[1E] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels.
[26PRE] A computer cluster comprising:	[25PRE] A computer cluster comprising:
[26A] a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script; and	[25A] a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script; and
[26B] a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;	[25B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;
[26C] wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions; wherein the plurality of nodes comprises: a first node comprising a first hardware processor	[25C] wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions wherein the plurality of nodes comprises:

’768 Claim	’034 Claim
configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and	a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and
[26D] a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node; wherein	[25D] a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node; wherein
[26E] the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;	[25E] the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;
[26F] wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;	[25F] wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;
[26G] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user	[25G] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instruction using the communications network interface, communicate at least some of the

’768 Claim	’034 Claim
instructions to one or more single-node kernels.	user instructions to one or more since-node kernels.
[27] The computer cluster of claim 26, wherein the asynchronous calls comprise a first command to create a first packet containing: an expression to be sent as payload; and a target node where the expression should be sent; wherein the first command is configured to be called from within a single-node kernel; wherein the single-node kernel is configured to send the first packet to a local cluster node module; and wherein the local cluster node module is configured to forward the expression to the target node.	[26] The computer cluster of claim 25, wherein the asynchronous calls comprise a first command to create a first packet containing: an expression to be sent as payload; and a target node where the expression should be sent; wherein the first command is configured to be called from within a single-node kernel; wherein the single-node kernel is configured to send the first packet to a local cluster node module; and wherein the local cluster node module is configured to forward the expression to the target node.
[29PRE] A computer cluster comprising:	[28PRE] A computer cluster comprising:
[29A] a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script; and	[28A] a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script; and
[29B] a mechanism for the nodes to communicate results of mathematical expression evaluation with each other;	[28B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other, wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface;
[29C] wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions; wherein	[28C] wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions

’768 Claim	’034 Claim
the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and	[28D1] wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and
[29D] a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;	[28D2] a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node; wherein
[29E] wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node; and	[28D3] the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node; and
[29F] wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;	[28E] wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;
[29G] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism,	[28F] wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user

’768 Claim	’034 Claim
communicate at least some of the user instructions to one or more single-node kernels.	instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels.
[30] The computer cluster of claim 4, wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another.	[25B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;
[31] The computer cluster of claim 4, wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules, and wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation.	[1B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;
[33] The computer cluster of claim 1, wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation.	[1B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;
[34] The computer cluster of claim 1, wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier.	[25B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;
[35PRE] A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:	[30PRE] A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:
[35A] a hardware processor configured to access one or more non-transitory memory	[30A] a hardware processor configured to access one or more non-transitory memory

’768 Claim	’034 Claim
devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores; a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;	devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores; a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;
[35B] a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and	[30B] a communications network interface to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture, wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface; and
[35C1] program code that, when executed, is capable of causing the hardware processor to: receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;	[30C1] program code that, when executed, is capable of causing the hardware processor to: receive calls from a second node comprising a second hardware processor configures to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;
[35C2] execute, using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node comprising	[30C2] execute, using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node comprising
[35C3] a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a	[30C3] a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a

’768 Claim	’034 Claim
second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;	second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node;
[35D] wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and	[30D] wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script; and
[35E] wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel.	[30E] wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to the single-node kernel.
[36] The computer cluster node of claim 35, wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node.	[30A] a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores; a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;
[37] The computer cluster node of claim 35, wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation.	[1B] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;

’768 Claim	’034 Claim
[39] The computer cluster node of claim 35, wherein the hardware processor comprises a special purpose microprocessor.	[30A] a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores; a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;

Similarly, as shown in the following chart, the claims of the ’582 patent render the claims of the ’621 patent obvious. Indeed, Plaintiff implicitly acknowledged that the claims of the ’582 and ’621 patents are not patentably distinct when it filed a terminal disclaimer during prosecution of the ’582 patent, disclaiming any patent term for the ’582 patent that exceeded the term of the earlier-issued ’621 patent. *See* ACS_0065861 (terminal disclaimer of patent term for ’582 patent over the ’621 patent). The Asserted Claims of the ’621 patent therefore are invalid for obviousness-type double patenting. *See Gilead*, 753 F.3d at 1217.

’621 Claim	’582 Claim
[1PRE] A computer cluster comprising:	[1PRE] A computer cluster comprising:
[1A] a first processor; a second processor; a third processor;	[1A] a first processor; a second processor; a third processor;
[1B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first	[1B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first

’621 Claim	’582 Claim
kernel configured to translate commands into code for execution on the first processor;	kernel configured to translate user commands into calls for execution of matrix operations on an array of data elements;
[1C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to: receive first commands from a user interface without the first commands first passing through the first kernel, and after receiving the first commands from the user interface, send second commands to the first kernel	[1C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to communicate with the first kernel and receive commands from a user interface;
[1D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor;	[1D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to execute the matrix operations on the array of data elements and return a first result using the second processor;
[1E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module;	[1E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicate with the first cluster node module;
[1F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor; and	[1F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to receive the first result and execute additional matrix operations on the first result and return a second result to one or more of the first, second, or third cluster node modules using the third processor; and
[1G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;	[1G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;

’621 Claim	’582 Claim
[1H] wherein each of the first, second, and third cluster node modules can transmit tasks to any other of the first, second, and third cluster node modules without transmitting the tasks through a master node, and	[1H] wherein the first cluster node module is configured to return, based on the second result, a final result.
[1I] wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.	[6] The computer cluster of claim 1, further comprising a data structure in which messages originating from the second and third cluster node modules are stored.
[3] The computer cluster of claim 1, wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor, wherein the first kernel and the first cluster node module reside in the first computer-readable medium, and wherein the second kernel and the second cluster node module reside in the second computer-readable medium.	[3] The computer cluster of claim 1, wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor, wherein the first kernel and the first cluster node module reside in the first computer-readable medium, and wherein the second kernel and the second cluster node module reside in the second computer-readable medium.
[4] The computer cluster of claim 1, further comprising a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.	[4] The computer cluster of claim 1, further comprising a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.
[6] The computer cluster of claim 1, wherein the data structure of the first cluster node module comprises a received message queue.	[7] The computer cluster of claim 6, wherein the data structure of the first cluster node module comprises a received message queue.
[7] The computer cluster of claim 6, wherein the first cluster node module further comprises a message receiving queue, in which data specifying a location to which the first cluster node module can expect to receive an expression and an identifier of a processor from which the expression is expected to be sent are stored.	[8] The computer cluster of claim 7, wherein the first cluster node module further comprises a message receiving queue, in which data specifying a location to which the first cluster node module can expect to receive an expression and an identifier of a processor from which the expression is expected to be sent are stored.
[8] The computer cluster of claim 1, wherein the second cluster node module further	[9] The computer cluster of claim 1, wherein the second cluster node module further

’621 Claim	’582 Claim
comprises a received message queue in which messages originating from the first and third cluster node modules are stored.	comprises a received message queue in which messages originating from the first and third cluster node modules are stored.
[9] The computer cluster of claim 1, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel.	[10] The computer cluster of claim 1, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel.
[10] The computer cluster of claim 9, wherein the advanced functions module comprises a call that calculates a Fourier transform across the computer cluster in parallel.	[11] The computer cluster of claim 10, wherein the advanced functions module comprises a call that calculates a Fourier transform across the computer cluster in parallel.
[11] The computer cluster of claim 9, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.	[12] The computer cluster of claim 10, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.
[13PRE] A computer cluster comprising:	[1PRE] A computer cluster comprising:
[13A] a first processor; a second processor; a third processor;	[1A] a first processor; a second processor; a third processor;
[13B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor;	[1B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate user commands into calls for execution of matrix operations on an array of data elements;
[13C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to: first receive first commands from a user interface, and after receiving the first	[1C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to communicate with the first kernel and receive commands from a user interface;

’621 Claim	’582 Claim
commands from the user interface, send second commands to the first kernel	
[13D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor;	[1D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to execute the matrix operations on the array of data elements and return a first result using the second processor;
[13E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module;	[1E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicate with the first cluster node module;
[13F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor; and	[1F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to receive the first result and execute additional matrix operations on the first result and return a second result to one or more of the first, second, or third cluster node modules using the third processor; and
[13G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;	[1G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;
[13H] wherein each of the first, second, and third cluster node modules can transmit tasks to any other of the first, second, and third cluster node modules without transmitting the tasks through a master node, and	[15] The computer cluster of claim 1, further comprising a communications network interface for the cluster node modules to communicate, with each other using a peer-to-peer architecture, results of execution of the matrix operations.
[13I] wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.	[6] The computer cluster of claim 1, further comprising a data structure in which messages originating from the second and third cluster node modules are stored.

’621 Claim	’582 Claim
[15] The computer cluster of claim 13, further comprising a communications network for connecting at least two of the first processor, the second processor, or the third processor to one another.	[15] The computer cluster of claim 1, further comprising a communications network interface for the cluster node modules to communicate, with each other using a peer-to-peer architecture, results of execution of the matrix operations.
[16] The computer cluster of claim 13, wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor, wherein the first kernel and the first cluster node module reside in the first computer-readable medium, and wherein the second kernel and the second cluster node module reside in the second computer-readable medium.	[3] The computer cluster of claim 1, wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor, wherein the first kernel and the first cluster node module reside in the first computer-readable medium, and wherein the second kernel and the second cluster node module reside in the second computer-readable medium.
[17] The computer cluster of claim 13, further comprising a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.	[4] The computer cluster of claim 1, further comprising a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.
[20] The computer cluster of claim 13, wherein the first processor and the second processor reside on the same die.	[17] The computer cluster of claim 15, wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface.
[21] The computer cluster of claim 13, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel.	[10] The computer cluster of claim 1, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel.
[22] The computer cluster of claim 21, wherein the advanced functions module comprises a call that calculates a Fourier	[11] The computer cluster of claim 10, wherein the advanced functions module comprises a call that calculates a Fourier

’621 Claim	’582 Claim
transform across the computer cluster in parallel.	transform across the computer cluster in parallel.
[23] The computer cluster of claim 21, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.	[12] The computer cluster of claim 10, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.
[24PRE] A computer cluster comprising:	[1PRE] A computer cluster comprising:
[24A] a first processor; a second processor; a third processor;	[1A] a first processor; a second processor; a third processor;
[24B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor;	[1B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate user commands into calls for execution of matrix operations on an array of data elements;
[24C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to send the commands to the first kernel, wherein the first cluster node module is logically positioned between a user interface and the first kernel;	[1C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to communicate with the first kernel and receive commands from a user interface;
[24D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor;	[1D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to execute the matrix operations on the array of data elements and return a first result using the second processor;
[24E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and	[1E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and

’621 Claim	’582 Claim
communicates with the first cluster node module;	communicate with the first cluster node module;
[24F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor; and	[1F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to receive the first result and execute additional matrix operations on the first result and return a second result to one or more of the first, second, or third cluster node modules using the third processor; and
[24G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;	[1G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;
[24H] wherein each of the first, second, and third cluster node modules can transmit tasks to any other of the first, second, and third cluster node modules without transmitting the tasks through a master node, and	[1H] wherein the first cluster node module is configured to return, based on the second result, a final result.
[24I] wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.	[6] The computer cluster of claim 1, further comprising a data structure in which messages originating from the second and third cluster node modules are stored.
[25] The computer cluster of claim 24, wherein the first processor and the second processor reside on the same die.	[17] The computer cluster of claim 15, wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface.
[26] The computer cluster of claim 24, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel, wherein the advanced functions module	[10] The computer cluster of claim 1, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel.

’621 Claim	’582 Claim
comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.	[12] The computer cluster of claim 10, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.
[27PRE] A computer cluster comprising:	[1PRE] A computer cluster comprising:
[27A] a first processor; a second processor; a third processor;	[1A] a first processor; a second processor; a third processor;
[27B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor;	[1B] at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate user commands into calls for execution of matrix operations on an array of data elements;
[27C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to send the commands to the first kernel, wherein the first cluster node module is logically positioned between a user interface and the first kernel;	[1C] a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to communicate with the first kernel and receive commands from a user interface;
[27D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor;	[1D] a second kernel residing in the at least one computer-readable medium, said second kernel configured to execute the matrix operations on the array of data elements and return a first result using the second processor;
[27E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module;	[1E] a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicate with the first cluster node module;

’621 Claim	’582 Claim
[27F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor; and	[1F] a third kernel residing in the at least one computer-readable medium, said third kernel configured to receive the first result and execute additional matrix operations on the first result and return a second result to one or more of the first, second, or third cluster node modules using the third processor; and
[27G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;	[1G] a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module;
[27H] wherein each of the first, second, and third cluster node modules can directly transmit tasks to any other of the first, second, and third cluster node modules, and	[1H] wherein the first cluster node module is configured to return, based on the second result, a final result.
[27I] wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored.	[6] The computer cluster of claim 1, further comprising a data structure in which messages originating from the second and third cluster node modules are stored.
[28] The computer cluster of claim 27, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.	[10] The computer cluster of claim 1, wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel. [12] The computer cluster of claim 10, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel.
[29] The computer cluster of claim 27, further comprising a communications network for connecting at least two of the first processor, the second processor, or the third processor to one another.	[15] The computer cluster of claim 1, further comprising a communications network interface for the cluster node modules to communicate, with each other using a peer-to-

’621 Claim	’582 Claim
	peer architecture, results of execution of the matrix operations.
[30] The computer cluster of claim 27, wherein at least one of the first, second, or third cluster node modules transmits tasks directly to another of the first, second, and third cluster node modules via one or more network components.	[15] The computer cluster of claim 1, further comprising a communications network interface for the cluster node modules to communicate, with each other using a peer-to-peer architecture, results of execution of the matrix operations.

V. PRIOR ART UNDER 35 U.S.C. §§ 102(F) AND (G)

A. Prior Art Under 35 U.S.C. § 102(f)

Under 35 U.S.C. § 102(f), a person shall be entitled to a patent unless “he did not himself invent the subject matter sought to be patented.” *In re VerHoef;* 888 F.3d 1362, 1365 (Fed. Cir. 2018). Those who “invent” patentable subject matter are entitled to a patent, 35 U.S.C. § 101, and when “an invention is made by two or more persons jointly, they shall apply for patent jointly,” 35 U.S.C. § 116. The “[i]nventors may apply for a patent jointly even though (1) they did not physically work together or at the same time, (2) each did not make the same type or amount of contribution, or (3) each did not make a contribution to the subject matter of every claim of the patent.” *Id*.

Failing to name the correct inventor(s) on a patent renders the patent invalid or unenforceable. *See Bd. of Educ. ex rel. Bd. of Trustees of Fla. State U. v. Am. Bioscience, Inc.*, 333 F.3d 1330, 1344 (Fed. Cir. 2003) (finding patent unenforceable “for failure to correctly name inventors in cases where the named inventors acted in bad faith or with deceptive intent”). A patent also is invalid for naming a non-inventor. *See Pannu v. Iolab Corp.*, 155 F.3d 1344, 1349 (Fed. Cir. 1998) (“Accordingly, if nonjoinder of an actual inventor is proved by clear and convincing evidence . . . a patent is rendered invalid.”). To prove that a person is not an inventor,

one need not prove who is the true inventor, just that the named inventor contributed nothing to the claimed invention. *Belcher Pharms., LLC v. Hospira, Inc.*, No. l 7-CV0775-LPS, 2019 WL 2503159, at *l n.l (D. Del. June 5, 2019) ("A patent may be invalid simply because it names the wrong inventors.").

In addition, Intel expressly incorporates by reference all lack-of-inventorship arguments identified in the Other Plaintiff Cases, including NVIDIA's argument in its Final Invalidity Contentions that neither named inventor Dean Dauger nor named inventor Zvi Tannenbaum invented or conceived of anything disclosed or claimed in the Asserted Patents, and that the true inventor(s) of the Asserted Patents, if any, are at least Victor Decyk, Rob Raguet-Schofield, Theo Gray, Bedros Afeyan, Kirk Won, and/or Vlad Savanchenko. Most if not all the evidence relied upon by NVIDIA was obtained during discovery in its case, and ACS and other relevant third parties have not yet produced it to Intel. Intel further incorporates by reference NVIDIA's argument that Plaintiff's failure to name the correct inventors of the Asserted Patents renders the patents invalid and that, even if the inventorship of the Asserted Patents is corrected based on the addition of any of the above-named inventors, the Asserted Patents still would be unenforceable because they would no longer be co-owned.

Intel expressly reserves the right to assert that the Asserted Claims are invalid under 35 U.S.C. § 102(f) in the event Intel obtains additional evidence that the inventors named in any of the Asserted Patents did not invent the subject matter claimed therein. Should Intel obtain such evidence, it will provide the name of the person(s) from whom and the circumstances under which the alleged invention or any part of it was derived. Because fact discovery has not yet opened and Intel has not yet taken discovery in this case—including seeking discovery from Plaintiff, taking depositions of the named inventors of the Asserted Patents, and seeking discovery of prior

inventions by third parties—Intel reserves the right to supplement these Contentions with facts, documents, or other information learned at a later point through discovery or further investigation.

B. Prior Art Under 35 U.S.C. § 102(g)

Each prior art patent, publication, or product identified above was either effectively filed or issued, published, or known, used, offered for sale or sold before the earliest-claimed priority date of the Asserted Patents to which it is applied for invalidity, and none was abandoned, suppressed, or concealed, so each such reference also constitutes evidence of prior invention pursuant to 35 U.S.C. § 102(g), if it is in the U.S. The persons or entities involved with each such invention include the named inventors on the above-identified patents, the authors listed on the above-identified publications, and the entities and individuals identified in connection with the above-identified products. In addition, each of the prior art systems and products identified above constitutes a prior invention pursuant to 35 U.S.C. § 102(g).

VI. INVALIDITY UNDER 35 U.S.C. § 112

As detailed below, certain of the Asserted Claims lack written description or enablement, are indefinite, and/or fail to claim what the inventors regard as the invention under provisions of 35 U.S.C. § 112. The § 112 Invalidity Contentions set forth below are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e) and the Orders of record in this matter to the extent appropriate in light of, without limitation, information provided by Plaintiff concerning its infringement allegations, further investigation and discovery, the Court's constructions of the claims at issue, positions taken by Plaintiff in this action or any other proceeding, review and analysis of expert witnesses, and/or as other circumstances warrant.

A. Lack of Enablement/Written Description

35 U.S.C. § 112(a) requires that the specification contain a written description of the invention. "[T]he hallmark of written description is disclosure." *Bos. Sci. Corp. v. Johnson & Johnson*, 647 F.3d 1353, 1361–62 (Fed. Cir. 2011) (citation omitted) (alteration in original). The test for whether a specification adequately describes an invention is "whether the disclosure of the application relied upon reasonably conveys to those skilled in the art that the inventor had possession of the claimed subject matter as of the filing date . . . . [T]he test requires an objective inquiry into the four corners of the specification from the perspective of a POSA . . . [It] is a question of fact." *Ariad Pharms., Inc. v. Eli Lilly & Co.*, 598 F.3d 1336, 1351 (Fed. Cir. 2010) (en banc); *Bos. Sci.*, 647 F.3d at 1362. "Adequate written description means that the applicant, in the specification, must 'convey with reasonable clarity to those skilled in the art that, as of the filing date sought, it or she was in possession of the [claimed] invention.'" *Agilent Techs., Inc. v. Affymetrix, Inc.*, 567 F.3d 1366, 1379 (Fed. Cir. 2009). It is well settled that "one cannot disclose a forest in the original application, and then later pick a tree out of the forest and say here is my invention." *Purdue Pharma L.P. v. Faulding Inc.*, 230 F.3d 1320, 1326 (Fed. Cir. 2000). Rather, the specification must provide some guides or "blaze marks" that disclose the claimed invention "specifically, as something appellants actually invented." *Ariad,* 598 F.3d at 1348.

In addition, the enablement requirement of § 112 requires that the patent specification teach a skilled artisan how to "make and use" the claimed invention, which the Federal Circuit has held requires teaching "those skilled in the art how to make and use the full scope of the claimed invention without 'undue experimentation.'" *Genentech, Inc. v. Novo Nordisk A/S*, 108 F.3d 1361, 1365 (Fed. Cir. 1997) (quoting *In re Wright*, 999 F.2d 1557, 1561 (Fed. Cir. 1993)). "The scope

of the claims must be less than or equal to the scope of the enablement." *Nat'l Recovery Tech., Inc. v. Magnetic Separation Sys., Inc.*, 166 F.3d 1190, 1196 (Fed. Cir. 1999).

Intel also incorporates by reference and reserves the right to rely on the § 112 contentions served on Plaintiff by the defendant(s) in other cases, including in the "Other Plaintiff Cases."

1. Enablement/Written Description Contentions for the '768 Patent

Intel reserves the right to argue that the claims of the '768 patent, when read in light of the specification, prosecution history, and/or claim construction, infringement, and validity positions that ACS has taken in this case or other cases, fail to reasonably convey to one of ordinary skill that the inventor(s) had possession of the subject matter of the invention and/or fail to enable a person of ordinary skill to make and use the invention as claimed for at least the following reasons:

- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "hardware processor" in the context of the claimed invention.
- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture" (claim 1); "mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls" (claim 26); "mechanism for the nodes to communicate results of mathematical expression evaluation with each other" (claim 29); and "mechanism to

communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture" (claim 35).

- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of an apparatus that is both "configured to receive[:] a command to start a cluster initialization process" and comprises the other claimed structures, including, *inter alia,* the claimed "mechanism" and claimed configurations of nodes to receive and communicate the claimed results.
- Claims 1 and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "peer-to-peer architecture."
- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of a "single-node kernel."
- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of:

"a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node,

execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node" (claims 1, 26, 29);

"receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;

execute, using the hardware processor, at least a first mathematical expression evaluation; and

communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node" (claim 35).

- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of a single-

node kernel "configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution"

- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "communicate a result."
- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "one or more of the nodes" (claims 1, 26, and 29) or "the computer cluster node" (claim 35) configured to "accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions [to one or more single-node kernels] [claims 1, 26, and 29]/[to the single-node kernel] [claim 35]."
- Claims 1, 26, 29, and 35 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., to the extent they omit elements that render the claims overly broad relative to what is described in the specification, including a cluster node module interposed between the user interface and the kernel, and cluster node modules designed to intercommunicate with each other in a peer-to-peer architecture.

- Claim 4 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "each of the nodes comprises one or more cluster node modules."
- Claim 20 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "one or more of the nodes are configured to accept user instructions via one or more of the nodes."
- Claim 34 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier."
- Claim 35 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of:

  "program code that, when executed, is capable of causing the hardware processor to:

  receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to

interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution…

communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node." Additionally, these limitations lack patentable weight because they do not impose structure on the claimed "computer cluster node."

- Claim 36 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "performing a parallel fast Fourier transform, wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node."
- Claim 37 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation."
- Claim 39 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or

enabling disclosure of "the hardware processor comprises a special purpose microprocessor."

2. Enablement/Written Description Contentions for the '034 Patent

Intel reserves the right to argue that the claims of the '034 patent, when read in light of the specification, prosecution history, and/or claim construction, infringement, and validity positions that ACS has taken in this case or other cases, fail to reasonably convey to one of ordinary skill that the inventor(s) had possession of the subject matter of the invention and/or fail to enable a person of ordinary skill to make and use the invention as claimed for at least the following reasons:

- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "hardware processor" in the context of the claimed invention.
- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture" (claim 1); "communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls" (claim 25); "communications network interface for the nodes to communicate results of mathematical expression evaluation with each other" (claim 28); and "communications network interface to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture" (claim 30). Additionally, these limitations lack

patentable weight to the extent they fail to impose any structure upon the "communications network interface"

- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of an apparatus that is both "configured to receive[:] a command to start a cluster initialization process" and comprises the other claimed structures, including, *inter alia,* the claimed configurations of nodes to receive and communicate the claimed results.
- Claims 1 and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "peer-to-peer architecture."
- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of a "single-node kernel."
- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of:

  "a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node;

wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node" (claims 1, 25, 28);

"receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution;

execute, using the hardware processor, at least a first mathematical expression evaluation; and

communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node" (claim 30).

- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of a single-node kernel "configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution"

- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "communicate a result."
- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "one or more of the nodes" (claims 1, 25, 28) or "the computer cluster node (claim 30) configured to "after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to [one or more single-node kernels] [claims 1 and 28]/[one or more since-node kernels] [claim 25]/[the single-node kernel]/[the single-node kernel] [claim 30]."
- Claims 1, 25, 28, and 30 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because of the omission of elements that render the claims overly broad relative to what is described in the specification, including a cluster node module interposed between the user interface and the kernel, and cluster node modules designed to intercommunicate with each other in a peer-to-peer architecture.
- Claim 3 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "each of the nodes comprises one or more cluster node modules."

- Claim 24 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels."
- Claim 27 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface." Alternately, claim 27 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, fourth para., because it broadens the scope of the claimed "interface."
- Claim 30 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of:

  "program code that, when executed, is capable of causing the hardware processor to:

  receive calls from a second node comprising a second hardware processor configures to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution…

communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node." Additionally, these limitations lack patentable weight because they do not impose structure on the claimed "computer cluster node."

- Claim 30 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface."
- Claim 30 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script."

3. Enablement/Written Description Contentions for the '679 Patent

Intel reserves the right to argue that the claims of the '679 patent, when read in light of the specification, prosecution history, and/or claim construction, infringement, and validity positions that ACS has taken in this case or other cases, fail to reasonably convey to one of ordinary skill that the inventor(s) had possession of the subject matter of the invention and/or fail to enable a person of ordinary skill to make and use the invention as claimed for at least the following reasons:

- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "hardware processor" in the context of the claimed invention.
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture." Additionally, this limitation lacks patentable weight to the extent it fails to impose any structure upon the "communications network interface."
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "peer-to-peer architecture."
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of a "program code that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions comprising matrix operations."
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of:

"a first node comprising . . . second program code configured to interpret user instructions, identify an array of data elements, and distribute calls to at least one of a plurality of other nodes for execution of matrix operations on the array of data elements; and

a second node comprising a second hardware processor, wherein the second node is configured to receive the calls from the first node, execute the matrix operations on the array of data elements, and communicate a result of the matrix operations to a third node;

wherein the third node comprises a third hardware processor, wherein the third node is configured to receive the result of the matrix operations from the second node, execute at least a mathematical expression evaluation using the received result, and communicate the result of the mathematical expression evaluation to the first node."

- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "program code configured to interpret user instructions, identify an array of data elements, and distribute calls to at least one of a plurality of other nodes for execution"
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "communicate [a/the] result[s]."
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "one or more nodes

are configured to accept, via the user interface or via a script, user instructions, and after accepting the user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other."

- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because of the omission of elements that render the claims overly broad relative to what is described in the specification, including a single-node kernel, a cluster node module interposed between the user interface and the kernel, and cluster node modules designed to intercommunicate with each other in a peer-to-peer architecture.
- Claim 2 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface." Alternately, claim 2 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, fourth para., because it broadens the scope of the claimed "interface."
- Claim 3 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "each of the nodes comprises one or more cluster node modules."
- Claim 17 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "one or more of the nodes are configured to communicate at least some of the user instructions."

- Claim 18 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels."

4. Enablement/Written Description Contentions for the '621 Patent

Intel reserves the right to argue that the claims of the '621 patent, when read in light of the specification, prosecution history, and/or claim construction, infringement, and validity positions that ACS has taken in this case or other cases, fail to reasonably convey to one of ordinary skill that the inventor(s) had possession of the subject matter of the invention and/or fail to enable a person of ordinary skill to make and use the invention as claimed for at least the following reasons:

- Claims 1, 13, 24, and 27 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "processor" in the context of the claimed invention.
- Claims 1, 13, 24, and 27 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor."
- Claims 1, 13, 24, and 27 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "kernel configured to translate commands into code for execution."

- Claims 1 and 13 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of a "cluster node module configured to: receive first commands from a user interface without the first commands first passing through the first kernel, and after receiving the first commands from the user interface, send second commands to the first kernel" (claim 1) or "cluster node module configured to: first receive first commands from a user interface, and after receiving the first commands from the user interface, send second commands to the first kernel" (claim 13).
- Claims 1, 13, 24, and 27 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "each of the first, second, and third cluster node modules can transmit tasks to any other of the first, second, and third cluster node modules without transmitting the tasks through a master node" (claims 1, 13, 24) or "each of the first, second, and third cluster node modules can directly transmit tasks to any other of the first, second, and third cluster node modules" (claim 27).
- Claim 1, 13, 24, and 27 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., to the extent they omit elements that render the claims overly broad relative to what is described in the specification, including a single-node kernel and a cluster node module each on a separate node. Claim 13 and any claims dependent therefrom are additionally invalid to the extent they omit a cluster node module interposed between the user interface and the kernel.

- Claim 1, 13, 24, and 27 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of a "the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored."
- Claim 7 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a message receiving queue, in which data specifying a location to which the first cluster node module can expect to receive an expression and an identifier of a processor from which the expression is expected to be sent are stored."
- Claim 8 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "the second cluster node module further comprises a received message queue in which messages originating from the first and third cluster node modules are stored."
- Claims 9, 21, 26, and 28 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel."
- Claims 10 and 22 are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written

description and/or enabling disclosure of "a call that calculates a Fourier transform across the computer cluster in parallel."

- Claim 11, 23, 26, and 28 are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "calls that evaluate matrix operations across the computer cluster in parallel."

5. Enablement/Written Description Contentions for the '582 Patent

Intel reserves the right to argue that the claims of the '582 patent, when read in light of the specification, prosecution history, and/or claim construction, infringement, and validity positions that ACS has taken in this case or other cases, fail to reasonably convey to one of ordinary skill that the inventor(s) had possession of the subject matter of the invention and/or fail to enable a person of ordinary skill to make and use the invention as claimed for at least the following reasons:

- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "processor" in the context of the claimed invention.
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of the full scope of "at least one computer-readable medium in communication at least one of the first processor, the second processor, or the third processor," "second cluster node module configured to . . . communicate with the first cluster node module," and "third cluster node module

configured to communicate with the first cluster node module and the second cluster node module."

- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of a "kernel configured to translate user commands into calls for execution."
- Claim 1 any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., to the extent they omit elements that render the claims overly broad relative to what is described in the specification, including a single-node kernel, a cluster node module interposed between the user interface and the kernel, a kernel and a cluster node module each on a separate node, the first and second kernels executing on separate processors, and cluster node modules designed to intercommunicate with each other in a peer-to-peer architecture.
- Claim 1 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of:

  "a second kernel residing in the at least one computer-readable medium, said second kernel configured to execute the matrix operations on the array of data elements and return a first result using the second processor;

  …

  a third kernel residing in the at least one computer-readable medium, said third kernel configured to receive the first result and execute additional matrix operations

on the first result and return a second result to one or more of the first, second, or third cluster node modules using the third processor; and

…

the first cluster node module is configured to return, based on the second result, a final result."

- Claim 6 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a data structure in which messages originating from the second and third cluster node modules are stored."
- Claim 8 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a message receiving queue, in which data specifying a location to which the first cluster node module can expect to receive an expression and an identifier of a processor from which the expression is expected to be sent are stored."
- Claim 10 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel."
- Claims 11 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a call that calculates a Fourier transform across the computer cluster in parallel."

- Claim 12 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "calls that evaluate matrix operations across the computer cluster in parallel."
- Claims 13 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "the final result comprises at least a portion of the second result."
- Claims 15 and any claims dependent therefrom are invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "a communications network interface for the cluster node modules to communicate, with each other using a peer-to-peer architecture, results of execution of the matrix operations." Additionally, the limitation lacks patentable weight to the extent it fails to impose any structure upon the "communications network interface."
- Claim 16 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the cluster node modules to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more of the kernels."

- Claim 17 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, first para., because the specification as filed does not contain an adequate written description and/or enabling disclosure of "wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface." Alternately, claim 17 is invalid for failure to satisfy the requirements of 35 U.S.C. § 112, fourth para., because it broadens the scope of the claimed "interface."

* * *

Therefore, all of the Asserted Claims are invalid for lacking sufficient written description and/or an enabling disclosure.

B. Indefiniteness

Claims are indefinite under 35 U.S.C. § 112(b) when they "fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention." *Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 898–99 (2014). To the extent an asserted apparatus claim includes both apparatus and method limitations, that claim is invalid for indefiniteness under § 112(b) because it fails to identify or notify the public of what constitutes direct infringement. *See IPXL Holdings, L.L.C. v. Amazon.com, Inc.*, 430 F.3d 1377 (Fed. Cir. 2005). Section 112(b) also requires that the claimed subject matter be what "the inventor or a joint inventor regards as the invention." "Where it would be apparent to one of skill in the art, based on the specification, that the invention set forth in a claim is not what the patentee regarded as its invention, we must hold that claim invalid under § 112, paragraph 2." *Allen Eng'g Corp. v. Bartell Indus., Inc.*, 299 F.3d 1336, 1349 (Fed. Cir. 2002).

Intel expressly reserves the right to amend its Invalidity Contentions to pursue invalidity based on additional bases for indefiniteness as the case proceeds and after claim construction occurs, or in view of positions taken by Plaintiff regarding invalidity or infringement. Intel also reserves the right to argue that the limitations of the Asserted Claims listed below, when read in light of the specification, prosecution history, and/or claim construction, infringement, and validity positions that Plaintiff has taken in this case or other cases, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention. For instance, the written description does not use phrases that appear in these limitations, the claim language lacks antecedent basis, and/or the claim language, as phrased and read in context of other claim limitations, does not inform those skilled in the art about the scope of the invention with reasonable certainty. As one of ordinary skill would not understand the metes and bounds of the claims in which these limitations appear, each of the identified claims is invalid as indefinite pursuant to 35 U.S.C. § 112.

- "communicate a result of the second mathematical expression evaluation to the first node" (claim 35 of the '768 patent and any claims dependent therefrom).

Intel notes that detailed arguments as to indefiniteness, if any, will be presented in the claim construction meet-and-confer and briefing processes. Intel reserves the right to withdraw, modify, or refine any indefiniteness positions as the case progresses, including in light of the Court's claim construction, subsequent factual or expert discovery, or evolving positions taken by Plaintiff. Nothing herein should be construed as a waiver of Intel's ability to assert or retract indefiniteness arguments to ensure consistency with its positions across various stages of this proceeding. For claim terms that are indefinite at their boundaries, Intel contends that these claims are nevertheless invalid under §§ 102 and/or 103 because the prior art discloses the relevant subject matter that is undisputedly within the scope of these claims.

VII. SUBJECT-MATTER ELIGIBILITY

All the Asserted Claims of the Asserted Patents are further invalid for failure to meet the eligibility requirements of 35 U.S.C. § 101. Specifically, claims 1, 4, 20, 21, 26, 27, 29, 30, 31, 33–37, and 39 of the ’768 patent, claims 1, 3, 4, 6–11, 13, 15–17, and 20–30 of the ’621 patent, claims 1, 2, 3, 8, 10, 24, 25, 27, 28, and 30 of the ’034 patent, claims 1, 3, 4, and 6–20 of the ’582 patent, and claims 1, 2, 3, 8, 10, 17, and 18 of the ’679 patent are invalid because they fail to meet the patent-eligibility requirements.

These subject-matter eligibility contentions are subject to revision and amendment pursuant to Federal Rule of Civil Procedure 26(e) and the Orders in this matter to the extent appropriate in light of, without limitation, further investigation and discovery, the Court’s construction of the claims at issue, positions taken by Plaintiff in this action or any other proceeding, the review and analysis of expert witnesses, and/or as other circumstances warrant.

In support of its subject-matter eligibility contentions, Intel relies on the prior art identified in Section II (Identification of Relevant Prior Art), *supra.* Intel reserves the right to rely on percipient and/or expert testimony to supplement, summarize, and/or explain the state of the industry, and what was considered well understood, routine, and conventional, at the time the earliest applications to which the Asserted Patents claim priority were filed. Any such witnesses will be disclosed pursuant to the Federal Rules of Civil Procedure and any Order in this case for initial, additional, and expert disclosures.

A. The Asserted Claims Are Directed to an Abstract Idea

The Supreme Court’s *Alice* test sets forth a two-part standard for assessing subject-matter eligibility. In the first step, the court must determine whether the claims are “directed to” patent-ineligible subject matter, such as an abstract idea. *Alice Corp. Pty. Ltd. v. CLS Bank Int’l,* 134 S.

Ct. 2347, 2355 (2014). In deciding whether a claim is "directed to" an abstract idea, the court must look at the "focus of the claimed advance over the prior art to determine if the claim's character as a whole is directed to excluded subject matter." *Affinity Labs of Tex., LLC v. DIRECTV, LLC*, 838 F.3d 1253, 1257 (Fed. Cir. 2016) (internal quotation marks omitted). Limiting the claims directed to an abstract idea to a particular "technological environment" or "field of use" does "not render the claims any less abstract." *Id.* at 2359; *Alice*, 134 S. Ct. at 2358 ("the prohibition against patenting abstract ideas cannot be circumvented by attempting to limit the use of [the idea] to a particular technological environment") (internal quotation marks and citation omitted); *Smart Sys. Innovations, LLC v. Chi. Transit Auth*., 873 F.3d 1364, 1373 (Fed. Cir. 2017) (citing *Affinity Labs,* 838 F.3d at 1259) (holding that claims applying abstract idea to particular field of use, namely mass transit, were not patent eligible); *ChargePoint, Inc. v. SemaConnect, Inc.*, 920 F.3d 759, 768 (Fed. Cir. 2019), *cert. denied*, 140 S. Ct. 983 (2020) (limiting abstract idea of "communication over a network for interacting with a device" to electric vehicle charging stations did not make claims patent eligible).

The '768 patent claims are directed to the abstract idea of solving math problems sequentially by performing mathematical evaluations and then using those results for subsequent evaluations before returning a result. Beyond that abstract idea, everything else is well-understood, routine, and conventional. The claims merely recite performing these evaluations on a conventional computer cluster consisting of nodes that use conventional processors that can communicate with each other via peer-to-peer or all-to-all connections using standard protocols like MPI. Subsequently-issued patents take this idea to even greater levels of abstraction. The '034 patent replaces the routine and conventional "mechanism" for nodes to intercommunicate with an equally routine and conventional "communications network interface," which can be, for example,

a conventional bus. In the ’679 patent, the routine and conventional “single node kernel” that evaluates the mathematical expressions is replaced with even more generic “program code,” while the math problems are limited to “matrix operations,” a broad category of well-understood calculations performed on matrices that are routinely performed with pencil and paper. In the ’582 patent, the computer cluster likewise is configured to perform such “matrix operations.”

The independent claims of the ’621 patent drop any reference to mathematical evaluations or matrix operations and refer only to generic “commands.” Thus, the patent is directed to the abstract idea of performing user commands, such as math problems, by dividing them into tasks for separate evaluation and collecting the results. Beyond that abstract idea, everything else is well-understood, routine, and conventional. The claims merely recite performing these tasks on a conventional computer cluster consisting of nodes that use conventional processors running conventional kernels that can communicate with each other via peer-to-peer or all-to-all connections using conventional software “modules” that can communicate with each other and with the user.

Moreover, none of the claims specify any technological improvement. The specification cites routine and conventional cluster computing frameworks (*e.g.*, MPI, message queues) and describes typical cluster computing steps such as initializing the cluster, sending work, and collecting results. No specialized memory arrangement, unconventional OS, or improved network interface is required. *See, e.g.*, *Intellectual Ventures I LLC v. Symantec Corp.*, 838 F.3d 1316 (Fed. Cir. 2016) (finding claims directed to the abstract idea of “receiving, screening, and distributing e-mail” were not patent-eligible where the claims relied on generic computer components to perform the well-known task of distributing computational tasks among multiple processors); *Two-Way Media Ltd. v. Comcast Cable Commc'ns, LLC*, 874 F.3d 1329, 1334 (Fed. Cir. 2017) (holding that

claims directed to the abstract idea of "transmitting message packets over a communications network" were not patent-eligible where the claims failed to specify any particular technology for achieving the claimed result but instead relied on conventional computer and network components); see also Berkheimer v. HP Inc., 881 F.3d 1360, 1367 (Fed. Cir. 2018) ("Limiting the invention to a technological environment does not make an abstract concept any less abstract under step one.") (internal quotation marks and citation omitted). The dependent claims differ in minor ways that fail to transform the underlying abstract idea into patent-eligible subject matter.

The following charts identify with specificity the abstract idea in each Asserted Claim:

Abstract Ideas Claimed in the '768 Patent

'768 Patent Claim	Abstract Idea
All Asserted Claims	Solving math problems sequentially by performing mathematical evaluations and using the intermediate results for subsequent evaluations before returning a final result.

Abstract Ideas Claimed in the '621 Patent

'621 Patent Claim	Abstract Idea
All Asserted Claims	Performing algorithms, such as solving math problems, by dividing them into tasks for separate evaluation and collecting the results.

Abstract Ideas Claimed in the '034 Patent

'034 Patent Claim	Abstract Idea
All Asserted Claims	Solving math problems sequentially by performing mathematical evaluations and using the intermediate results for subsequent evaluations before returning a final result.

Abstract Ideas Claimed in the '582 Patent

'582 Patent Claim	Abstract Idea
All Asserted Claims	Performing matrix operations sequentially by performing evaluations and using the intermediate results for subsequent evaluations before returning a final result.

Abstract Ideas Claimed in the '679 Patent

'679 Patent Claim	Abstract Idea
All Asserted Claims	Performing matrix operations sequentially by performing evaluations and using the intermediate results for subsequent evaluations before returning a final result.

B. The Asserted Claims Lack an Inventive Step

Under *Alice* step two, the court must determine whether the patent claims include an "inventive concept" that is "sufficient to ensure that the patent in practice amounts to significantly more than a patent upon the [abstract idea] itself." *Alice*, 134 S. Ct. at 2355 (internal quotation marks and citations omitted). The inventive concept must be truly inventive; it is not sufficient to include "well-understood, routine, conventional" features. *Mayo Collaborative Servs. v. Prometheus Lab'ys., Inc.*, 566 U.S. 66, 79–80 (2012). For example, merely reciting conventional computer elements "cannot transform a patent-ineligible abstract idea into a patent-eligible invention." *Alice*, 134 S. Ct. at 2358. Moreover, "generic functional language" to achieve purported solutions to technical problems, without "any requirements for *how* the desired result is achieved" will not suffice for an inventive concept. *Two-Way Media*, 874 F.3d at 1339 (internal quotation marks and citation omitted).

The Asserted Claims fail to recite any elements that, either individually or as an ordered combination, transform the abstract idea of solving math problems sequentially—performing mathematical evaluations, then using those results for subsequent evaluations—into a patent-eligible application of that idea. *See Alice*, 573 S. Ct. at 2354–55. Beyond this abstract idea, the claims merely call upon generic computer hardware and routine high-performance computing techniques:

- **User Interface and Kernel:** The Asserted Patents describe using a user interface and a kernel (*e.g.*, a Mathematica kernel) to interpret program code instructions. Such components are standard in any computing environment and do not represent a novel improvement to the functioning of a computer system.
- **Communication Between Nodes:** The Asserted Patents acknowledge that computer clusters existed in the prior art. *See, e.g.*, '768 patent, 1:19–28 (admitting that prior-art "computer application programs" were capable of "running on a group of nodes that could act as a cluster"). The Asserted Patents rely on known methods of node-to-node communication, including protocols like MPI. This capability—allowing nodes to pass messages to each other—was undeniably conventional in distributed and high-performance computing at the earliest claimed priority date.
- **Parallelization and Distributed Computing:** The Asserted Patents generally describe parallelization and distributed computing, long-established techniques that were widely employed in high-performance computing systems. They do not recite any specific improvements to those techniques, such as specialized memory arrangements, unique OS layers, or improved network interfaces.

Indeed, the Federal Circuit found claims that issued before the earliest-claimed priority date here that were directed to "a plurality of processors collectively arranged to perform a parallel processing computation" invalid under Section 101. *SAP Am., Inc. v. InvestPic, LLC*, 898 F.3d 1161, 1170 (Fed. Cir. 2018) (finding claimed use of parallel processing computing architecture generic where "neither the claims nor the specification call[ed] for any parallel processing architectures different from those available in existing systems").

Moreover, the recited order of operations among the first, second, and third nodes also does not confer any inventive step. In high-performance computing, nodes exchanging messages directly without a master or root node was recognized as a conventional alternative to master-slave topologies. Before the earliest-claimed priority date, nodes would have been physically capable of using any one of numerous available peer-to-peer message-passing frameworks (*e.g.*, PVM, earlier MPI implementations, distributed Maple/Matlab). For example, it was widely known that MPI calls could pass data between any node pair, collectively or individually. The Asserted Claims "require no improved computer resources . . . , just already available computers, with their already available basic functions, to use as tools in executing the claimed process." *SAP Am.*, 898 F.3d at 1169–70; *buySAFE, Inc. v. Google, Inc.*, 765 F.3d 1350, 1355 (Fed. Cir. 2014) ("That a computer receives and sends the information over a network—with no further specification—is not even arguably inventive.").

Because the Asserted Claims rely on conventional hardware, standard software protocols, and known distributed computing practices to implement the abstract idea of sequentially solving math problems, they fail to supply the inventive concept required under *Alice* step two. *See Trinity Info Media, LLC v. Covalent, Inc.*, 72 F.4th 1355, 1367 (Fed. Cir. 2023) (explaining that this court has "found no inventive concept where claims merely recited 'generic features' or 'routine functions' to implement the underlying abstract idea" (citations omitted)).

More particularly, the following claim elements, both individually and as an ordered combination, were well-understood, routine, and conventional, as illustrated by the exemplary prior art listed in the tables below and as supported by the prior art references and disclosures detailed in Tables 1–3 of these Invalidity Contentions and accompanying Claim Chart Exhibits: Though each of the below elements is well-understood, routine, and conventional, there is no

requirement that each element is found routine and conventional to the extent it embodies the abstract idea itself (i.e., to the extent it is directed to an ordered sequence of evaluations and the generation of intermediate and final results). *BSG Tech LLC v. Buyseasons, Inc.*, 899 F.3d 1281, 1290 (Fed. Cir. 2018) ("[A] claimed invention's use of the ineligible concept to which it is directed cannot supply the inventive concept that renders the invention 'significantly more' than that ineligible concept.").

Well-Understood, Routine, and Conventional Elements in the '768 Patent

'768 Patent Claim	Element
1	"[a] computer cluster" "a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor," "wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster' "wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions" "a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture" "wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel," "the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution" "a second node comprising a second hardware processor with a plurality of processing cores," "wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node"

’768 Patent Claim	Element
	“wherein the third node comprises a third hardware processor with a plurality of processing cores,” “wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface” “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels”
4	“wherein each of the nodes comprises one or more cluster node modules”
20	“wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes”
21	“wherein one or more of the nodes are configured to communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other”
26	“[a] computer cluster” “a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster,” “wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script” “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls” “wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node

’768 Patent Claim	Element
	kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;” “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” “a second node comprising a second hardware processor with a plurality of processing cores,” “wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node” “the third node comprises a third hardware processor with a plurality of processing cores,” “wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script” “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels”
27	“wherein the asynchronous calls comprise a first command to create a first packet containing: an expression to be sent as payload; and a target node where the expression should be sent;” “wherein the first command is configured to be called from within a single-node kernel;”

’768 Patent Claim	Element
	“wherein the single-node kernel is configured to send the first packet to a local cluster node module; and wherein the local cluster node module is configured to forward the expression to the target node”
29	“[a] computer cluster” “a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster,” “wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script” “a mechanism for the nodes to communicate results of mathematical expression evaluation with each other” “wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions;” “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” “a second node comprising a second hardware processor with a plurality of processing cores,” “wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node” “wherein the third node comprises a third hardware processor with a plurality of processing cores,” “wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node”

’768 Patent Claim	Element
	“wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script” “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to one or more single-node kernels.”
30	“wherein each of the plurality of nodes implements asynchronous calls that enable the single-node kernel to perform computation tasks while the cluster node modules are simultaneously communicating with one another”
31	“wherein intercommunication among the plurality of single-node kernels during thread execution is enabled by the plurality of cluster node modules” “wherein the computer cluster is configured to permit exchange of information between nodes during the course of a parallel computation”
33	“wherein the plurality of nodes are configured to permit exchange of information between nodes during the course of parallel computation”
34	“wherein each of the plurality of nodes comprises instructions executable by the hardware processor and configured to implement asynchronous behavior,” “wherein the instructions comprise: a first instruction to asynchronously send a payload to another node; a second instruction to asynchronously receive a payload from another node; and a third instruction to search for a payload matching a message specifier”
35	“[a] computer cluster node for evaluating expressions in parallel with other computer cluster nodes” “a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation,”

’768 Patent Claim	Element
	“wherein the hardware processor comprises multiple processor cores; a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster” “a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture” “program code that, when executed, is capable of causing the hardware processor to: receive calls from a second node comprising a second hardware processor configured to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” “program code that, when executed, is capable of causing the hardware processor to: . . . execute, using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node” “a third hardware processor with a plurality of processing cores,” “wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node” “wherein the user connection interface is configured to return at least one result . . . to a user interface or a script” “wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and after communicating at least some of the user instructions using the mechanism, communicate at least some of the user instructions to the single-node kernel”
36	“wherein the one or more non-transitory memory devices comprise program code for performing a parallel fast Fourier transform” “wherein the program code causes the hardware processor to evaluate a command to perform a Fourier transform on an array comprising a first

’768 Patent Claim	Element
	data portion that is stored on the computer cluster node and a second data portion that is not stored on the computer cluster node”
37	“wherein the computer cluster node is configured to permit exchange of information with other computer cluster nodes during the course of parallel computation”
39	“wherein the hardware processor comprises a special purpose microprocessor”

Well-Understood, Routine, and Conventional Elements in the ’621 Patent

’621 Patent Claim	Element
1	“[a] computer cluster” “a first processor; a second processor; a third processor” “at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor” “a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to: receive first commands from a user interface without the first commands first passing through the first kernel, and after receiving the first commands from the user interface, send second commands to the first kernel” “a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor” “a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module” “a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor” “a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module”

’621 Patent Claim	Element
	“wherein each of the first, second, and third cluster node modules can transmit tasks to any other of the first, second, and third cluster node modules without transmitting the tasks through a master node” “wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored”
3	“wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor,” “wherein the first kernel and the first cluster node module reside in the first computer-readable medium,” “wherein the second kernel and the second cluster node module reside in the second computer-readable medium”
4	“a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module”
6	“wherein the data structure of the first cluster node module comprises a received message queue”
7	“wherein the first cluster node module further comprises a message receiving queue, in which data specifying a location to which the first cluster node module can expect to receive an expression and an identifier of a processor from which the expression is expected to be sent are stored”
8	“wherein the second cluster node module further comprises a received message queue in which messages originating from the first and third cluster node modules are stored”
9	“wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel”
10	“wherein the advanced functions module comprises a call that calculates a Fourier transform across the computer cluster in parallel”
11	“wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel”

’621 Patent Claim	Element
13	“[a] computer cluster” “a first processor; a second processor; a third processor” “at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor” “a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to: first receive first commands from a user interface, and after receiving the first commands from the user interface, send second commands to the first kernel” “a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor” “a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module” “a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor” “a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module” “wherein each of the first, second, and third cluster node modules can transmit tasks to any other of the first, second, and third cluster node modules without transmitting the tasks through a master node” “wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored”
15	“a communications network for connecting at least two of the first processor, the second processor, or the third processor to one another”
16	“wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor”

’621 Patent Claim	Element
	“wherein the first kernel and the first cluster node module reside in the first computer-readable medium, and wherein the second kernel and the second cluster node module reside in the second computer-readable medium”
17	“a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module”
20	“wherein the first processor and the second processor reside on the same die”
21	“wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel”
22	“wherein the advanced functions module comprises a call that calculates a Fourier transform across the computer cluster in parallel”
23	“wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel”
24	“[a] computer cluster” “a first processor; a second processor; a third processor” “at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor” “a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to send the commands to the first kernel, wherein the first cluster node module is logically positioned between a user interface and the first kernel” “a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor” “a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicates with the first cluster node module”

’621 Patent Claim	Element
	“a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor” “a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module” “wherein each of the first, second, and third cluster node modules can transmit tasks to any other of the first, second, and third cluster node modules without transmitting the tasks through a master node” “wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored”
25	“wherein the first processor and the second processor reside on the same die”
26	“wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel”
27	“[a] computer cluster” “a first processor; a second processor; a third processor” “at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor” “a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to send the commands to the first kernel,” “wherein the first cluster node module is logically positioned between a user interface and the first kernel” “a second kernel residing in the at least one computer-readable medium, said second kernel configured to translate commands into code for execution on the second processor” “a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send

’621 Patent Claim	Element
	commands to the second kernel and communicates with the first cluster node module” “a third kernel residing in the at least one computer-readable medium, said third kernel configured to translate commands into code for execution on the third processor” “a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module” “wherein each of the first, second, and third cluster node modules can directly transmit tasks to any other of the first, second, and third cluster node modules” “wherein the first cluster node module comprises a data structure in which messages originating from the second and third cluster node modules are stored”
28	“wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel, wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel”
29	“a communications network for connecting at least two of the first processor, the second processor, or the third processor to one another”
30	“wherein at least one of the first, second, or third cluster node modules transmits tasks directly to another of the first, second, and third cluster node modules via one or more network components”

Well-Understood, Routine, and Conventional Elements in the ’034 Patent

’034 Patent Claim	Element
1	“[a] computer cluster” “a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor,” “wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster,” “wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node

’034 Patent Claim	Element
	kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions” “a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture” “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” “a second node comprising a second hardware processor with a plurality of processing cores,” “wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of the first mathematical expression evaluation to a third node” “wherein the third node comprises a third hardware processor with a plurality of processing cores” “wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface” “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels”
2	“wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface”
3	“wherein each of the nodes comprises one or more cluster node modules”

’034 Patent Claim	Element
8	“wherein the computer cluster includes the user interface”
10	“wherein the cluster initialization process comprises establishing communication among two or more of the nodes”
24	“wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels”
25	“[a] computer cluster” “a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script” “a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls” “wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions” “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” “a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression evaluation, and communicate a result of mathematical expression evaluation to a third node” “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node”

’034 Patent Claim	Element
	“wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script” “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instruction using the communications network interface, communicate at least some of the user instructions to one or more since-node kernels”
27	“wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface”
28	“[a] computer cluster” “a plurality of nodes, wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster, wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and an instruction from a user interface or a script” “a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other, wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface” “wherein each of the nodes is configured to access a non-transitory computer-readable medium comprising program code for a single-node kernel that, when executed, is capable of causing a hardware processor to evaluate mathematical expressions” “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution” “a second node comprising a second hardware processor with a plurality of processing cores, wherein the second node is configured to receive calls from the first node, execute at least a first mathematical expression

’034 Patent Claim	Element
	evaluation, and communicate a result of mathematical expression evaluation to a third node” “wherein the third node comprises a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node” “wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script” “wherein one or more of the nodes are configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels”
30	“[a] computer cluster node for evaluating expressions in parallel with other computer cluster nodes” “a hardware processor configured to access one or more non-transitory memory devices comprising program code for a single-node kernel that, when executed, causes the hardware processor to interpret user instructions, to evaluate mathematical expressions, and to produce results of mathematical expression evaluation, wherein the hardware processor comprises multiple processor cores; a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster” “a communications network interface to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture,” “wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface” “program code that, when executed, is capable of causing the hardware processor to: receive calls from a second node comprising a second hardware processor configures to access a second memory comprising program code for a user interface and program code for a second single-node kernel, the second single-node kernel configured to interpret user

’034 Patent Claim	Element
	instructions and distribute calls to at least one of a plurality of other nodes for execution” “execut[ing], using the hardware processor, at least a first mathematical expression evaluation; and communicate a result of the first mathematical expression evaluation to a third node” “a third hardware processor with a plurality of processing cores, wherein the third node is configured to receive the result of mathematical expression evaluation from the computer cluster node, execute at least a second mathematical expression evaluation using the result of the first mathematical expression evaluation, and communicate a result of the second mathematical expression evaluation to the first node” “wherein the user connection interface is configured to return at least one result of mathematical expression evaluation to a user interface or a script” “wherein the computer cluster node is configured to: accept user instructions; after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to the single-node kernel”

Well-Understood, Routine, and Conventional Elements in the ’582 Patent

’582 Patent Claim	Element
1	“[a] computer cluster” “a first processor; a second processor; a third processor” “at least one computer-readable medium in communication with at least one of the first processor, the second processor, or the third processor; a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate user commands into calls for execution of matrix operations on an array of data elements” “a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to communicate with the first kernel and receive commands from a user interface” “a second kernel residing in the at least one computer-readable medium, said second kernel configured to execute the matrix operations on the

’582 Patent Claim	Element
	array of data elements and return a first result using the second processor” “a second cluster node module residing in the at least one computer-readable medium, said second cluster node module configured to send commands to the second kernel and communicate with the first cluster node module” “a third kernel residing in the at least one computer-readable medium, said third kernel configured to receive the first result and execute additional matrix operations on the first result and return a second result to one or more of the first, second, or third cluster node modules using the third processor” “a third cluster node module residing in the at least one computer-readable medium, said third cluster node module configured to send commands to the third kernel and configured to communicate with the first cluster node module and the second cluster node module” “wherein the first cluster node module is configured to return, based on the second result, a final result”
3	“wherein the at least one computer-readable medium comprises a first computer-readable medium coupled to at least the first processor and a second computer-readable medium coupled to at least the second processor, wherein the first kernel and the first cluster node module reside in the first computer-readable medium, and wherein the second kernel and the second cluster node module reside in the second computer-readable medium”
4	“a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module”
6	“a data structure in which messages originating from the second and third cluster node modules are stored”
7	“wherein the data structure of the first cluster node module comprises a received message queue”
8	“wherein the first cluster node module further comprises a message receiving queue, in which data specifying a location to which the first cluster node module can expect to receive an expression and an identifier of a processor from which the expression is expected to be sent are stored”

’582 Patent Claim	Element
9	“wherein the second cluster node module further comprises a received message queue in which messages originating from the first and third cluster node modules are stored”
10	“wherein the first cluster node module comprises an advanced functions module, and wherein the advanced functions module comprises a call that evaluates an expression across the computer cluster in parallel”
11	“wherein the advanced functions module comprises a call that calculates a Fourier transform across the computer cluster in parallel”
12	“wherein the advanced functions module comprises one or more calls that evaluate matrix operations across the computer cluster in parallel”
13	“wherein the final result comprises at least a portion of the second result”
14	“wherein the first cluster node module is configured to return the final result to the user interface”
15	“a communications network interface for the cluster node modules to communicate, with each other using a peer-to-peer architecture, results of execution of the matrix operations”
16	“wherein one or more of the cluster node modules are configured to: accept user instructions; after accepting the user instructions, communicate at least some of the user instructions using the communications network interface for the cluster node modules to communicate with each other; and after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more of the kernels”
17	“wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface”
18	“wherein at least one of the first, second, or third kernels comprises a single-node kernel”
19	“wherein the cluster node modules communicate with one another to act as a cluster”

’582 Patent Claim	Element
20	“wherein one or more of the cluster node modules are configured to receive a command to start a cluster initialization process for the computer cluster”

Well-Understood, Routine, and Conventional Elements in the ’679 Patent

’679 Patent Claim	Element
1	“[a] computer cluster” “a plurality of nodes each comprising a hardware processor and configured to access a non-transitory computer-readable medium comprising program code that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions comprising matrix operations” “a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture” “wherein the plurality of nodes comprises: a first node comprising a first hardware processor configured to access a first memory comprising first program code for a user interface and second program code configured to interpret user instructions, identify an array of data elements, and distribute calls to at least one of a plurality of other nodes for execution of matrix operations on the array of data elements” “a second node comprising a second hardware processor, wherein the second node is configured to receive the calls from the first node, execute the matrix operations on the array of data elements, and communicate a result of the matrix operations to a third node” “ wherein the third node comprises a third hardware processor, wherein the third node is configured to receive the result of the matrix operations from the second node, execute at least a mathematical expression evaluation using the received result, and communicate the result of the mathematical expression evaluation to the first node” “wherein the first node is configured to return the result of the mathematical expression evaluation to the user interface” “wherein one or more of the nodes are configured to: accept, via the user interface or via a script, user instructions; and after accepting the user instructions, communicate at least some of the user instructions

’679 Patent Claim	Element
	using the communications network interface for the nodes to communicate with each other”
2	“wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface”
3	“wherein each of the nodes comprises one or more cluster node modules”
8	“wherein the computer cluster includes the user interface”
10	“wherein the program code, when executed, is further capable of causing the hardware processor to execute a cluster initialization process configured to establish communication among two or more of the nodes”
17	“wherein one or more of the nodes are configured to communicate at least some of the user instructions”
18	“wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels”

VIII. DOCUMENT PRODUCTION

Pursuant to the Scheduling Order in this action, on April 9, 2025, Intel produced with its Preliminary Invalidity and Subject-Matter Eligibility Contentions prior art bearing the following Bates numbers: 98800DOC00034644 – 98800DOC00111775. At the time, Intel stated that it “also incorporates by reference, as if fully set forth herein, all prior art references identified in the Other Plaintiff Cases,” including *Advanced Cluster Systems, Inc., v. Advanced Micro Devices, Inc.,* No. 7:24-cv-244-ADA (W.D. Tex.). *See* 4/9/2025 Preliminary Invalidity and Subject-Matter Eligibility Contentions at 7, 303.

At this time, Intel produces with these Invalidity Contentions additional prior art bearing Bates numbers DEF_00053459 – DEF_00058895, which originally was produced by AMD with its invalidity contentions served on April 25, 2025.

Intel reserves the right to supplement its production of documents as the case progresses, including, but not limited to, producing or making additional documents or source code available for inspection. Intel also reserves the right to seek third-party discovery relevant to the accused instrumentalities and prior art, including the production of documents and source code from Intel's non-party suppliers, once fact discovery opens in this case.

Respectfully submitted,

DATED: April 25, 2025

/s/ Ashley Aull

Brian C. Nash
Texas Bar No. 24051103
MORRISON & FOERSTER LLP
300 Colorado Street, Suite 1800
Austin, TX 78701
Telephone: (737) 309-0700
bnash@mofo.com

Evan Mann (*pro hac vice*)
Susannah M. L. Gagnon (*pro hac vice*)
Kevin Yang
Benjamin Welton (*pro hac vice*)
MUNGER, TOLLES & OLSON LLP
560 Mission Street, 27th Floor
San Francisco, CA 94105
Telephone: (415) 512-4000
Fax: (415) 512-4077
evan.mann@mto.com
susannah.gagnon@mto.com
kevin.yang@mto.com
ben.welton@mto.com

Gregory P. Stone (*pro hac vice*)
Ashley Aull (*pro hac vice*)
Heather E. Takahashi
Faye P. Teller
Peter E. Gratzinger (*pro hac vice*)
Ted Kang (*pro hac vice*)
Adam W. Kwon (*pro hac vice*)
MUNGER, TOLLES & OLSON LLP
350 South Grand Avenue, Fiftieth Floor
Los Angeles, California 90071-3426
Telephone: (213) 683-9100
Fax: (213) 687-3702
gregory.stone@mto.com
ashley.aull@mto.com
heather.takahashi@mto.com
faye.teller@mto.com
peter.gratzinger@mto.com
ted.kang@mto.com
adam.kwon@mto.com

Attorneys for Defendant Intel Corporation

CERTIFICATE OF SERVICE

The undersigned hereby certifies that a true and correct copy of the foregoing document (and accompanying claim chart exhibits) was served by electronic mail to all counsel of record who are deemed to have consented to electronic service in accordance with Local Rule CV-5. I declare under penalty of perjury that the foregoing is true and correct.

Dated: April 25, 2025

/s/ Heather E. Takahashi
Heather E. Takahashi