

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NVIDIA CORP.,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2021-00019
Patent 10,333,768 B2

Before KARL D. EASTHOM, ARTHUR M. PESLAK, and
SEAN P. O'HANLON, *Administrative Patent Judges*.

PESLAK, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

Petitioner, NVIDIA Corporation, filed a Petition (Paper 1, “Pet.”) requesting an *inter partes* review of claims 1, 4–10, 18–22, 24, 25, 30, 31, 33, and 34 (the “challenged claims”) of U.S. Patent No. 10,333,768 B2 (Ex. 1001, “the ’768 patent”). Patent Owner, Advanced Cluster Systems, Inc., timely filed a Preliminary Response (Paper 5, “Prelim. Resp.”). With our permission, Petitioner filed a Reply to the Patent Owner Preliminary Response (Paper 7, “Pet. Reply”). Patent Owner filed a Sur-reply (Paper 8, “Sur-reply”).

We have authority, acting on the designation of the Director, to determine whether to institute an *inter partes* review under 35 U.S.C. § 314(a). *See also* 37 C.F.R. § 42.4(a) (2020) (“The Board institutes the trial on behalf of the Director.”). Under 35 U.S.C. § 314(a), an *inter partes* review may not be instituted unless the information presented in the Petition shows “there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” Taking into account the Petition, the arguments presented in the Preliminary Response, the additional briefing, as well as all supporting evidence, we deny institution of *inter partes* review because Petitioner fails to establish a reasonable likelihood that at least one of the challenged claims is unpatentable. In particular, we determine that Petitioner fails to show that the cited references disclose the “third node” as recited in independent claim 1 (“the third node limitation”).

A. Related Matters

The parties state that the ’768 patent is asserted in *Advanced Cluster Systems, Inc. v. NVIDIA Corporation*, Case No. 19-cv-2032-CFC (D. Del.). Pet. 3; Paper 3, 1. The ’768 patent is also the subject of IPR2021-00020

filed by Petitioner on October 10, 2020. Paper 3, 1. Patent Owner also identifies IPR2020-01608, IPR2021-00075, and IPR2021-00108 as involving patents related to the '768 patent. *Id.*

B. Real Parties in Interest

Petitioner identifies itself as the only real party-in-interest. Pet. 3. Patent Owner identifies itself as the only real party-in-interest. Paper 3, 1.

C. The '768 Patent

The '768 patent is titled Cluster Computing. Ex. 1001, code (54). The '768 patent issued on June 25, 2019 from an application filed on February 14, 2014. *Id.* at codes (45), (22).

The '768 patent “relates to the field of cluster computing generally and to systems and methods for adding cluster computing functionality to a computer program, in particular.” *Id.* at 1:14–17. Embodiments of the '768 patent include enabling a software package to benefit from a plurality of nodes in a cluster. *Id.* at 2:6–10. For example, “[o]ne embodiment allows a user to create applications, using a high-level language such as Mathematica, that are able to run on a computer cluster having supercomputer-like performance.” *Id.* at 2:10–13. “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.” *Id.* at 2:13–17.

Figure 1 of the '768 patent, reproduced below, “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.” *Id.* at 4:63–66.

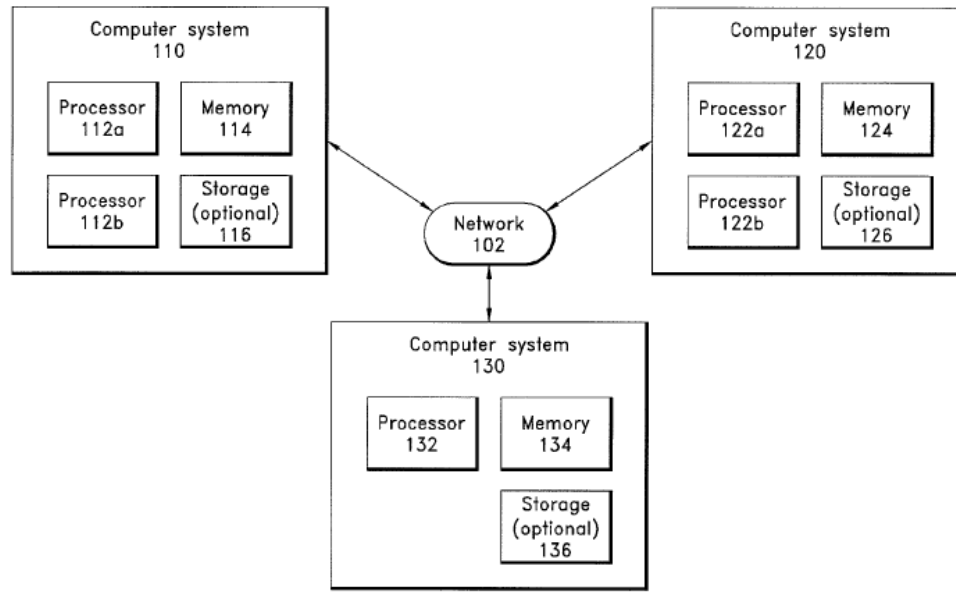


FIG. 1

Figure 1 of the '768 patent illustrates “a computer cluster 100 wherein computer systems 110, 120, 130 communicate with one another via a communications network 102.” *Id.* at 4:63–66.

In the embodiment shown in Figure 1, computer cluster 100 comprises first, second, and third computers systems 110, 120, and 130. *Id.* at 6:53–55. Each of the computer systems 110, 120, 130 includes one or more processors (112a, 112b, 122a, 122b, 132), a memory device (114, 124, 134), and an optional storage device (116, 126, 136). *Id.* at 6:56–63. Each of the computer systems 110, 120, 130 includes a network interface (not shown) for connecting to communications network 102, which can include one or more of a LAN, a WAN, an intranet, a wireless network, and/or the Internet. *Id.* at 6:63–67.

Figure 2 of the '768 patent, reproduced below, “is a block diagram showing relationships among software modules running on one embodiment of a computer cluster 100.” *Id.* at 5:16–18.

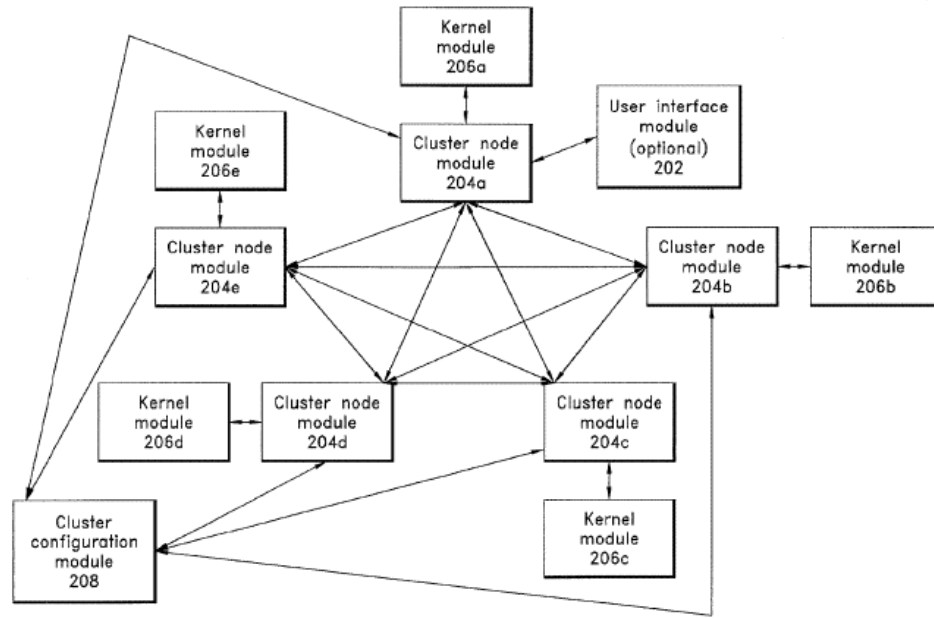


FIG. 2

Figure 2 of the '768 patent illustrates “relationships among software modules running on one embodiment of a computer cluster 100.” *Id.* at 5:16–18.

In the embodiment shown in Figure 2, a cluster system 100 includes a user interface module 202 that is able to access a plurality of kernel modules 206a – 206e by communicating with a first cluster node module 204a. *Id.* at 11:15–18. Kernel modules 206a – 206e can reside in the memory of one or more computer systems on which they run. *Id.* at 11:23–25. Kernel modules 206a – 206e include single-threaded program code and are each associated with one of the processors 112a, 112b, 122a, 122b, 132. *Id.* at 11:30–32. In the embodiment shown in Figure 2, each of the kernel modules 206a – 206e is in communication with a single cluster node module 204a – 204e, respectively. *Id.* at 5:33–35. Cluster node module 204a is in communication with each of the other cluster node modules 204b – 204e. *Id.* at 11:21–23. Cluster configuration module 208 is stored on one or more of the computer systems 110, 120, 130 or on a remote computer system, and

can establish communication with the cluster node modules 204a – 204e. *Id.* at 11:32–36. In one embodiment, communication between the cluster configuration module 208 and the cluster node modules 204a – 204e initializes the cluster node modules 204a – 204e to provide cluster computing support for the computer cluster 100. *Id.* at 11:36–40. In one embodiment, the cluster node modules 204a – 204e provide a way for many kernel modules 206a – 206e such as, for example, Mathematica kernels, running on a computer cluster 100 to communicate with one another. *Id.* at 11:42–45.

D. Challenged Claims

Claim 1 is the only independent claim challenged in the Petition.

Claim 1 is reproduced below:

1. A computer cluster comprising:
 - 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
 - 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; and

- 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 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.

Ex. 1001, 30:23–31:4.

E. Prior Art and Asserted Grounds

Petitioner asserts that claims 1, 4–10, 18–22, 24, 25, 30, 31, 33, and 34 would have been unpatentable on the following grounds:

Claim(s) Challenged	35 U.S.C. §	Reference(s)/Basis
1, 4–10, 18–22, 24, 25, 30, 31, 33, 34	§ 103(a) ¹	Schreiner1, ² Schreiner2, ³ Schreiner3, ⁴ Maple Guide, ⁵ Distributed Maple Code, ⁶ SPARC IV Article, ⁷ AMD Article ⁸
30, 31, 34	§ 103(a)	Schreiner1, Schreiner2, Schreiner3, Maple Guide, Distributed Maple Code, SPARC IV Article, AMD Article, MPI Standard ⁹

¹The Leahy-Smith America Invents Act, Pub. L. No. 112-29, 125 Stat. 284 (2011) (“AIA”), included revisions to 35 U.S.C. § 103 that became effective after the effective filing date of the challenged patent. Therefore, we apply the pre-AIA version of 35 U.S.C. § 103.

² Schreiner et al., “Distributed Maple: parallel computer algebra in networked environments,” Journal of Symbolic Computation, published in 2003 (Ex. 1008) (“Schreiner1”).

³ Schreiner, “Distributed Maple – User and Reference Manual (V 1.1.12),” July 6, 2001 (Ex. 1009) (“Schreiner2”).

⁴ Károly Bósa, Wolfgang Schreiner, “Task Logging, Rescheduling and Peer Checking in Distributed Maple,” March 18, 2002 (Ex. 1010) (“Schreiner3”).

⁵ Waterloo Maple Inc., “Maple 5 Learning Guide,” 1996, 1998 (Ex. 1011) (“Maple Guide”).

⁶ Exhibits 1012 – 1018.

⁷ “SUN Debuts UltraSPARC IV,” EE Times, October 15, 2003 (Ex. 1019) (“SPARC IV Article”).

⁸ “AMD to Unveil Dual-Core PC Chips,” Los Angeles Times, May 31, 2005 (Ex. 1020) (“AMD Article”).

⁹ MPI: A Message-Passing Interface Standard, 1995 (Ex. 1021) (“MPI Standard”).

II. ANALYSIS

A. Overview

A petition must show how the construed claims are unpatentable under the statutory grounds it identifies. 37 C.F.R. § 42.104(b)(4). Petitioner bears the burden of demonstrating a reasonable likelihood that it would prevail with respect to at least one challenged claim for a petition to be granted. 35 U.S.C. § 314(a).

A claim is unpatentable under § 103(a) if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations, including (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of skill in the art; and (4) when in evidence, objective indicia of non-obviousness (i.e., secondary considerations). *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

We analyze the asserted grounds with these principles in mind.

B. Level of Ordinary Skill in the Art

Petitioner contends that a skilled artisan “would have had a Bachelor’s degree in computer science, electrical engineering, or an equivalent field, and two years of academic or industry experience working with distributed computing.” Pet. 14 (citing Ex. 1005 ¶¶ 38–40).

Patent Owner does not explicitly address the level of ordinary skill in the art. *See generally* Prelim. Resp.

For the purposes of this Decision, we agree with Petitioner’s proposed level of ordinary skill in the art because it comports with the level of skill reflected in the ’768 patent and the prior art of record.

C. Claim Construction

We apply the same claim construction standard used by Article III federal courts and the ITC, both of which follow *Phillips v. AWH Corp.*, 415 F.3d 1303 (Fed. Cir. 2005) (en banc), and its progeny. 37 C.F.R. § 42.100(b). Accordingly, we construe each challenged claim of the ’768 patent to generally be “the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent.” *Id.*

Petitioner contends that no claim terms require express construction and proffers no proposed constructions. Pet. 14.

Patent Owner provides contentions concerning various claim limitations. Prelim. Resp. 6–16. In connection with the claim 1 limitation “a mechanism for the nodes to communicate . . . with each other using a peer-to-peer architecture,” Patent Owner contends that “[a] POSITA would understand that, in the context of cluster computing, the adjective ‘peer-to-peer’ ordinarily means that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” *Id.* at 8. In connection with claim 4, Patent Owner contends that “cluster node module” should be construed as “a module that cooperates with other cluster node modules to establish intercommunication among nodes in a computer cluster and to exchange messages such that each node can communicate tasks and data with other nodes without the tasks and data being required to go through a central server or master node.” *Id.* at 13.

After reviewing the evidence and argument presented by the parties, we determine that it is not necessary to our decision to construe any claim terms. *See Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (noting that “we need only construe terms ‘that are in controversy, and only to the extent necessary to resolve the controversy’”) (citing *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999)).

D. Ground 1: Alleged Obviousness over Schreiner1, Schreiner2, Schreiner3, Distributed Maple Code, Maple Guide, SPARC IV Article, and AMD Article

Petitioner contends that claims 1, 4–10, 18–22, 24, 25, 30, 31, 33, and 34 are unpatentable over Schreiner1, Schreiner2, Schreiner3, Distributed Maple Code, Maple Guide, SPARC IV Article, and AMD Article. Pet. 14–75. In support thereof, Petitioner identifies the disclosures in Schreiner1, Schreiner2, Schreiner3, Distributed Maple Code, Maple Guide, SPARC IV Article, and AMD Article alleged to describe the subject matter in the challenged claims. *Id.* Additionally, Petitioner offers declaration testimony from Dr. Henry Tufo Ph.D. (“Tufo Declaration”)¹⁰ (Ex. 1005) and Dr. Wolfgang Schreiner Ph.D. (“Schreiner Declaration”)¹¹ (Ex. 1006) in support of the Petition.

Patent Owner contends, *inter alia*, that the Petition fails to show that the prior art references disclose the third node limitation. Prelim. Resp.

¹⁰ Dr. Tufo is “presently employed as Professor of Computer Science at the University of Colorado at Boulder (‘UCB’).” Ex. 1005 ¶ 5.

¹¹ Dr. Schreiner is “a Professor in the Research Institute for Symbolic Computation (‘RISC’) at Johannes Kepler University in Austria.” Ex. 1006 ¶ 1. Dr. Schreiner “designed and programmed Distributed Maple from 1998-2001.” *Id.* ¶ 15.

26–29. Patent Owner supports its contentions with the Declaration of Jaswinder Pal Singh, Ph.D. Ex. 2001.¹²

1. Schreiner1 – Ex. 1008¹³

Schreiner1 is entitled “Distributed Maple: parallel computer algebra in networked environments” and bears a copyright date of 2003. Ex. 1008.

Schreiner1 is a journal article authored by Dr. Schreiner, Christian Mittermaier, and Karoly Bosa. *Id.* at 305. Schreiner1 “gives a comprehensive overview on the design and the use of ‘Distributed Maple,’ an environment for parallel computer algebra on multiprocessors and heterogeneous computer clusters.” *Id.* Schreiner1 explains that Distributed Maple was developed on the basis of the computer algebra system Maple (*id.*) and that Distributed Maple is built on top of the Maple kernel and does not require any kernel extensions. *Id.* at 306. According to Schreiner1, Distributed Maple is “so portable that applications can be executed in many different environments” and “so general that it can be applied to schedule tasks of other computer algebra systems (e.g., Mathematica).” *Id.* Schreiner1 describes Distributed Maple as providing “a programming model which is based on functional/logic/dataflow parallelism” that “allows the creation of a large number of implicitly scheduled tasks with automatic resolution of data dependencies and of globally shared data structures with implicit synchronization.” *Id.* The authors describe using Distributed Maple “to develop the first parallel versions for a number of non-trivial applications from algebraic geometry (parallel curve and surface plotting and parallel

¹² Dr. Singh is “currently a Professor of Computer Science at Princeton University.” Ex. 2001 ¶ 4.

¹³ Ex. 1008 contains two sets of page numbers. We refer to the page numbers added by Petitioner.

neighborhood analysis).” *Id.* at 307. Schreiner1 discloses that “[t]he user interacts with Distributed Maple via a conventional Maple frontend (text or graphical).” *Id.* at 309. Schreiner1 explains that “[t]he core of Distributed Maple is a scheduler program which is completely independent and even *unaware* of Maple” and “can in fact embed and schedule tasks from any kind of computation kernels that implement a specific communication protocol.” *Id.* at 310. Schreiner1 discloses that a Distributed Maple session comprises two components: a scheduler and a Maple interface. *Id.* Figure 1 of Schreiner1, reproduced below, depicts a software architecture for a Distributed Maple session. *Id.* at 311.

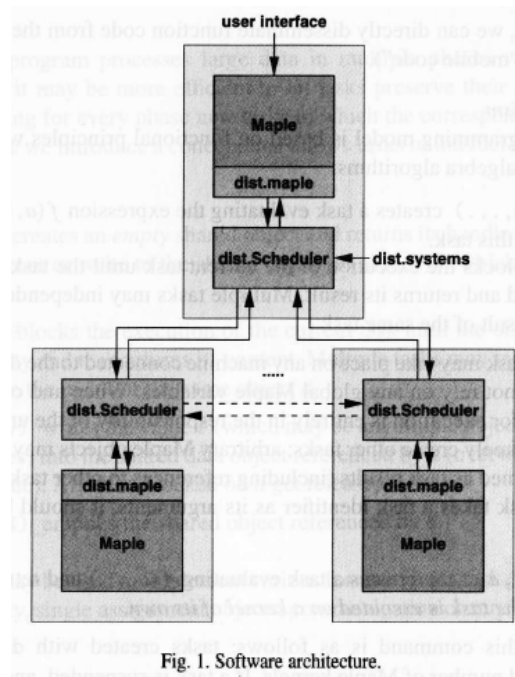


Figure 1 of Schreiner1 illustrates a software architecture for a Distributed Maple session. Ex. 1008, 311.

As shown in Figure 1, a Distributed Maple session “comprises a set of *nodes* each of which holds a pair of processes: a *kernel* and a *scheduler*.” *Id.* at 319. “Initially, a single task runs on the root kernel; this task may subsequently create new tasks which are distributed via the schedulers to

other kernels and may in turn create new tasks.” *Id.* With reference to Figure 1, Schreiner1 explains that “every scheduler instance accepts tasks from the attached computation kernel and schedules these tasks among all machines connected to the session.” *Id.* at 311. Schreiner1 further explains that “[t]he Maple kernel is a single-threaded process which communicates by a simple communication protocol with the scheduler on the same node” and “[a]ll capabilities for parallel and distributed program execution are embedded in this scheduler.” *Id.* at 314.

2. *Schreiner2 – Ex. 1009*

Schreiner2 is entitled “Distributed Maple – User and Reference Manual (V 1.1.12)” and bears a publication date of July 6, 2001. Ex. 1009. Like Schreiner1, Schreiner2 describes the Distributed Maple system. More particularly, Schreiner2 “describes the use of a system for writing distributed Maple applications and sketches its implementation.” *Id.* at 4.

3. *Schreiner3 – Ex. 1010*

Schreiner3 is entitled “Task Logging, Rescheduling and Peer Checking in Distributed Maple” and bears a publication date of March 18, 2002. Ex. 1010, 1. Schreiner3 describes extending the Distributed Maple system by adding “fault tolerance mechanisms such that the time spent in a long running computation is not any [sic] wasted by the eventual occurrence of session failure.” *Id.* Schreiner3 describes a first fault tolerance mechanism as “the logging of task return values and of shared object values such that after a failure the newly started session can “transparently to the application program reuse already computed result.” *Id.* A second fault tolerance mechanism is described as “the migration of tasks such that a session may tolerate the failure of individual nodes without overall failures.” *Id.* A third fault tolerance mechanism is described as “the redirection of the

messages such that a session may tolerate also the failure of the connections between nodes without overall failure.” *Id.* Figure 1 of Schreiner3 illustrates an Execution Model of the Distributed Maple system.

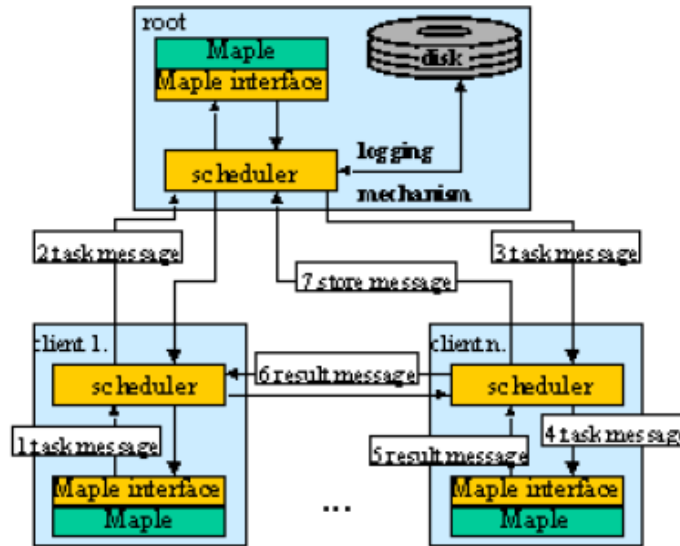


Figure 1: Execution Model

Figure 1 of Schreiner3 illustrates an Execution Model of the Distributed Maple system. Ex. 1010, 5.

Figure 1 depicts the passing of messages between nodes in a Distributed Maple system, and a logging mechanism in the root node. Schreiner3 explains that “[t]he logging mechanism in Distributed Maple is a fault tolerance mechanism for saving the results of intermediate tasks and the values of shared objects during the computation” and allows the system “to restore the results of computed tasks in a later session, if the current session crashes.” *Id.* at 3.

4. Maple Guide – Ex. 1011

Maple Guide is entitled “Maple V Learning Guide, Release 5” published by Waterloo Maple, Inc., and bears a copyright date of 1998. Ex.

1011, 5.¹⁴ Maple Guide explains that “Maple V is a *Symbolic Computation System* or *Computer Algebra System*” and that “[b]oth phrases refer to Maple V’s ability to manipulate information in a symbolic or algebraic manner.” *Id.* at 11.

5. *Distributed Maple Code – Exs. 1012–1018*

Distributed Maple Code is a collection of the following computer code files embodying a distribution of Distributed Maple for installation on a computer: dist.maple5 file (Ex. 1012), source code for parallel versions of Maple functions in the distsoft directory (Ex. 1013), source code for parallel versions of CASA functions in the distsoft directory (Ex. 1014), an “Install” file for Distributed Maple (Ex. 1015), a “ReadMe” file for Distributed Maple (Ex. 1016), an “Install” file for the source code in the distsoft directory (Ex. 1017), and a “ReadMe” file for the source code in the distsoft directory (Ex. 1018).

6. *Maple Guide – Ex. 1011*

Maple Guide is entitled “Maple V Learning Guide, Release 5” published by Waterloo Maple, Inc., and bears a copyright date of 1998. Ex. 1011, 5.¹⁵ Maple Guide is a manual authored by K. M. Heal, M. L. Hansen, and K. M. Rickard. *Id.* at 4. Maple Guide explains that “Maple V is a *Symbolic Computation System* or *Computer Algebra System*” and that “[b]oth phrases refer to Maple V’s ability to manipulate information in a symbolic or algebraic manner.” *Id.* at 11.

¹⁴ Ex. 1011 contains two sets of page numbers. We refer to the page numbers added by Petitioner.

¹⁵ Ex. 1011 contains two sets of page numbers. We refer to the page numbers added by Petitioner.

7. *SPARC IV Article – Ex. 1019*

The SPARC IV Article is an article published in the EE Times news journal and bears a publication date of October 15, 2003. Ex. 1019. The SPARC IV Article describes Sun Microsystems’ “new UltraSPARC IV processor.” *Id.* It explains that the UltraSPARC IV processor utilizes “Chip Multithreading (CMT), a design concept that allows the processor to execute tens of threads simultaneously, thus enabling increases in application throughput.” *Id.*

8. *AMD Article – Ex. 1020*

The AMD Article is entitled “AMD to Unveil Dual-Core PC Chips.” Ex. 1020. The AMD Article was published in the Los Angeles Times newspaper and bears a publication date of May 31, 2005 and a copyright date of 2005. *Id.* The AMD Article discloses that “Advanced Micro Devices Inc. is set to launch its first PC microprocessors with two computing engines on a single chip.” *Id.* The article explains that AMD and Intel Corp. had “discovered that performance could be boosted by creating two computing cores and running them at a slower speed on a single chip.” *Id.*

9. *Analysis of Claim 1*

Petitioner provides a number of reasons why a skilled artisan would have been motivated to combine the various references relied on in this challenge. Pet. 15–17. Petitioner also provides a limitation-by-limitation analysis of claim 1. *Id.* at 23–52. Petitioner supports its contentions with the testimony of Dr. Tufo. Ex. 1005 ¶¶ 43–111.

Patent Owner provides three specific contentions regarding claim 1. First, Patent Owner argues the Petition fails to show that the prior art discloses “peer-to-peer architecture.” Prelim. Resp. 20–24. Second, Patent Owner argues the Petition fails to show that the prior art discloses

communication of “user instructions” using “peer-to-peer architecture.” *Id.* at 24–26. Third, Patent Owner argues the Petition fails to show that the prior art discloses the limitation in claim 1 “wherein the third node comprises a third hardware processor . . . 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.” *Id.* at 26–29. Patent Owner supports these contentions with the testimony of Dr. Singh. Ex. 2001 ¶¶ 55–70.

Patent Owner’s first two contentions are based on its proposed construction of “peer-to-peer architecture.” We do not reach the question of the construction of “peer-to-peer architecture,” but rather focus our analysis on Patent Owner’s third contention, which is dispositive of our determination not to institute *inter partes* review.

In order to provide the basis for our determination, we set forth Petitioner’s contentions concerning how the prior art allegedly satisfies the limitations of claim 1 preceding the third node limitation. Other than argument based on its proposed construction of “peer-to-peer architecture,” which we do not adopt for the purposes of this decision, Patent Owner does not dispute these contentions in the Preliminary Response. *See generally* Prelim. Resp.

a) “A computer cluster comprising:”

Petitioner contends that, to the extent the preamble is limiting, Schreiner¹ discloses “an environment for executing **parallel computer algebra problems on multiprocessors and heterogeneous clusters.**” Pet. 23 (citing Ex. 1006 ¶ 61; Ex. 1008, Abstract, Fig. 1, 324–344; Ex. 1009, Abstract; Ex. 1010, 2).

b) “a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor,”

Petitioner provides the following annotated version of Figure 1 of Schreiner1:

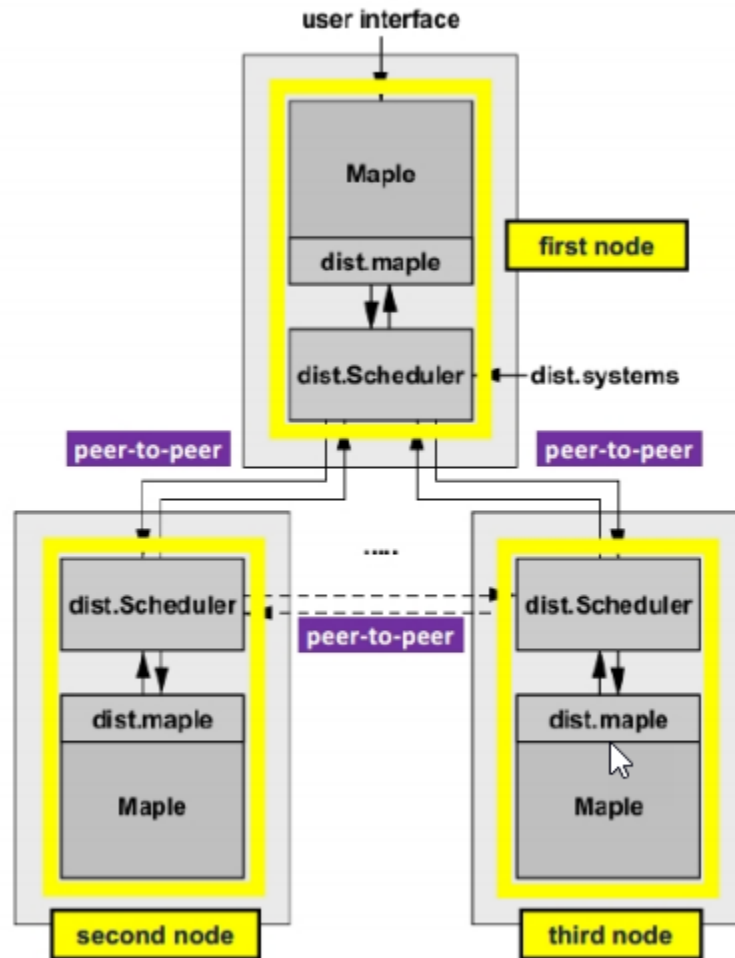


Fig. 1. Software architecture.

The preceding image is Figure 1 of Schreiner illustrating Schreiner’s Software Architecture with color annotations added by Petitioner. Pet. 24.

Petitioner contends that Figure 1 “shows a cluster with three nodes depicted and five dots indicating other nodes can be present. *Id.* at 23. Petitioner further contends that “[e]ach Maple kernel and the dist.maple and dist.Scheduler code was executed by a hardware processor.” *Id.* at 24–25 (citing Ex. 1005 ¶ 62; Ex. 1006 ¶ 47). Petitioner further contends that

“Schreiner1 discusses several implementations of its design, including a **128 processor** SGI Origin 3800 distributed shared memory multiprocessor cluster, a **24-processor heterogeneous** computer cluster, an **18-processor** Sun HPC 6500 system, and a Linux-based Beowulf cluster with **16 compute** nodes. *Id.* (citing Ex. 1005 ¶ 63; Ex. 1008, 324–325, 327; Ex. 1010, 2).

c) *“wherein one or more of the nodes are configured to receive a command to start a cluster initialization process for the computer cluster, and”*

Petitioner contends that “Schreiner1 teaches the dist.maple and scheduler processes on the root node initializing the cluster nodes upon a command by the user.” Pet. 25 (citing Ex. 1005 ¶ 64; Ex. 1008, 314; Ex. 1009, 14). Petitioner further contends that “Schreiner1 provides a specific example of the initialization command being received by a root node from a user, which causes a cluster of computers nicknamed aquila and andromeda to be configured for Distributed Maple operation.” *Id.* (citing Ex. 1005 ¶ 65; Ex. 1008, 310; Ex. 1009, 5, 13–14).

d) *“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”*

Petitioner contends that “the nodes in Schreiner1 comprise[] a hardware processor with access to storage, such as a hard disk drive (a non-transitory computer-readable medium), containing . . . program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions.” Pet. 26 (citing Ex. 1005 ¶ 66). Petitioner further contends that “Maple software is program code for a single-node kernel that, when executed, is capable of causing the hardware processor to evaluate mathematical expressions” and in that regard is similar to the Mathematica kernel referenced in the Specification of the ’768 patent.

Id. at 26–27 (citing Ex. 1001, 1:42–43, 2:21–24, 4:14–23; Ex. 1005 ¶ 68; Ex. 1011, 88). Petitioner also contends that “Schreiner1 teaches that Maple kernels were installed on each node of the Distributed Maple cluster.” *Id.* at 27 (citing Ex. 1008, 310–311, Fig. 1). Lastly, Petitioner contends that “[t]he processor would execute the Maple Code to evaluate mathematical expressions.” *Id.* at 27–28 (citing Ex. 1008, Abstract; Ex. 1005 ¶ 70; Ex. 1010, 1).

e) *“a mechanism¹⁶ for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;”*

Petitioner contends that “[t]he claimed mechanism is disclosed in the Distributed Maple Publications as a combination of two software components: dist.Scheduler, which ‘coordinates node interaction,’ and dist.maple, which ‘implements the interface between kernel and scheduler.’” Pet. 28 (citing Ex. 1006 ¶ 14; Ex. 1008, 310). Petitioner further contends that its annotated Figure 1 of Schreiner1, reproduced above, shows arrows that “indicate . . . all nodes are connected to each other, allowing peer-to-peer communications, as expressly taught by Schreiner1.” *Id.* at 28–29 (citing Ex. 1005 ¶ 72; Ex. 1008, 315, 317–318). Petitioner emphasizes that Schreiner1 discloses that “[w]hen a node needs to send a message to one of its peers, it can thus establish a direct connection for message transfers.” *Id.* at 29 (citing Ex. 1008, 315). Petitioner further contends “Schreiner1 teaches that the Distributed Maple dist[start] command is used to create tasks for

¹⁶ Neither party addresses whether this limitation should be construed as a means plus function limitation under 35 U.S.C. § 112(f) despite the fact that “mechanism” is a well-known nonce word. *See Williamson v. Citrix Online, LLC*, 792 F.3d 1339, 1350 (Fed. Cir. 2015). Given our disposition of this case, it is not necessary to reach this question.

evaluating mathematical expressions and distribute them to cluster nodes” and that “a result message is communicated whenever a node finishes evaluating a task.” *Id.* at 31 (citing Ex. 1005 ¶ 76; Ex. 1008, 312).

- f) “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”*

Petitioner parses this limitation into subsections as discussed below.
Pet. 32.

- (1) “a first node comprising a first hardware processor”*

Petitioner contends that “[t]he first node is the Distributed Maple root node.” Pet. 33 (citing Ex. 1005 ¶ 80).

- (2) “configured to access a first memory comprising program code for a user interface and program code for a first single-node kernel”*

Petitioner contends that each node including the root node in Distributed Maple “comprises a hardware processor configured to access memory.” Pet. 33 (citing Ex. 1005 ¶ 81; Ex. 1008, 306, 324–325, 327, 332). Petitioner further contends that “like other installed software programs, Maple is stored in a non-volatile storage medium, such as a disk drive” and “[w]hen a user starts Maple, the code is loaded into memory and accessed by a processor for execution.” *Id.* (citing Ex. 1005 ¶ 82; Ex. 1006 ¶ 40). Petitioner further contends that “the first kernel receives commands from a ‘user interface’ and sends them to the first cluster node module.” *Id.* at 34 (citing Ex. 1005 ¶ 83; Ex. 1008, 309–310).

- (3) *“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”*

Petitioner contends “[t]he Maple kernel is configured to interpret user instructions received in the user interface and to distribute calls to at least one of the other nodes for execution.” Pet. 34 (citing Ex. 1005 ¶ 85); *see also id.* (citing Ex. 1005 ¶ 86; Ex. 1011, 88). Petitioner further contends that “Schreiner¹ describes a user entering the Distributed Maple ‘dist[all]’ instruction into the root Maple kernel user interface: ‘dist[all](*command*) lets the Maple statement *command* be executed on every Maple kernel connected to the distributed session.’” *Id.* at 35 (citing Ex. 1008, 311; Ex. 1009, 30). Petitioner further contends that a skilled artisan would have understood “that, in order for this Distributed Maple instruction and its embedded Maple statement (‘*command*’) to be executed on every kernel, the instruction is distributed to every other node” and “the root kernel, through its user interface, distributed calls – in this case, the Distributed Maple dist[all] command and the Maple *command* to be evaluated – to the other cluster nodes.” *Id.* (citing Ex. 1005 ¶ 87).

- g) *“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;”*

Petitioner parses this limitation into subsections as discussed below.
Pet. 38.

(1) a second node

Petitioner contends that “[o]ne of the plurality of nodes comprising a hardware processor discussed above . . . is the ‘second node.’” Pet. 38 (citing Ex. 1005 ¶ 91).

(2) “plurality of processing cores”

Petitioner contends that “Schreiner1 discloses the use of multi-processor systems such as the ‘128 processor SGI Origin 3800 distributed shared memory multiprocessor’ and ‘an 18-processor Sun HPC 6500 system.’” Pet. 38 (citing Ex. 1008, 324–325, 327). Petitioner further contends that “[i]t would have been obvious to implement the system of Schreiner1 using a hardware processor with a plurality of processing cores for one or more of the nodes, including the second node, as taught in the SPARC IV Article and the AMD article.” *Id.*

(3) “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”

Petitioner contends that Schreiner1’s Figure 5 discloses “the second node receives a call from the first node, indicated by task <t,d>, it then executes the mathematical expression indicated by the task call; and the second node then returns a result to the first node in result <t,r>.”

Pet. 41–42 (citing Ex. 1005 ¶ 95; Ex. 1008, 319–320, Fig. 5).

h) “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;”

Patent Owner contends that the Petition fails to establish this limitation because it fails “to show where the prior art teaches the precise order of operations, involving three specific nodes, specified by the claim.” Prelim. Resp. 27. Patent Owner further contends “the Petition refers to the prior art’s general disclosure that multiple nodes may be involved in evaluating mathematical expressions and then speculates that the prior art could work in a manner that would meet the claim limitation.” *Id.* (citing Pet. 44–47; Ex. 2001 ¶ 67).

Petitioner provides three examples to establish that the cited references disclose this limitation. In the first example, Petitioner contends that “Schreiner¹ gives an example where ‘[a]fter the distributed session has been successfully established, two calls of dist[start] create two tasks evaluating the Maple expressions $\text{int}(x^n, x)$ and $\text{int}(x^n, n)$, respectively.’” Pet. 44 (citing Ex. 1008, 310). Petitioner further contends that the “tasks can be sent to one or more of the second, third, and other nodes for evaluation because ‘[t]he execution of a task may take place on any machine connected to the distributed session.’” *Id.* at 44–45 (citing Ex. 1008, 312). Petitioner further contends that the root node *receives results* from the nodes evaluating the mathematical expressions and “*puts them together into a final result* for display to the user” and “the root (first node) waits for the task results to be sent back by executing the wait instructions, and then displays the overall result of the mathematical expression.” *Id.* at 45 (citing Ex. 1005 ¶ 100; Ex. 1008, 310) (emphases added). Petitioner also contends that “Schreiner¹ teaches that task evaluation may depend on other task results” which “means that a third node performing a mathematical expression

evaluation may need to wait for a result from a second node performing another evaluation.” *Id.* (citing Ex. 1005 ¶ 101; Ex. 1008, 312). We are not persuaded that this contention satisfies the third node limitation for the following reasons.

We agree with Patent Owner that claim 1 requires a precise order of operations involving the recited first, second, and third nodes. The claim recites that the third node is configured to “receive the result of the first mathematical expression evaluation from the second node” with the third node required to “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.” Ex. 1001, 30:53–59. Petitioner specifically contends that the root node “receives results” which it “puts together for display to the user.” Pet. 45. Petitioner’s sequence of events does not satisfy this limitation because Petitioner does not direct us to evidence that the third node will receive a result from the second node, perform a second mathematical evaluation and communicate the result of the second mathematical evaluation to the first node. Petitioner’s attempt to cure this deficiency by claiming that the third node “*may* need to wait for a result from a second node performing another evaluation” does not persuade us because it is based on speculation of what may happen, but more pertinently is contrary to its contention that the results of the second and third node are both sent to the root node. *See* Pet. 45. Consequently, we determine that Petitioner has not sufficiently established that this example discloses the third node limitation.

Petitioner’s second example is based on Figure 5 of Schreiner¹ which is reproduced below:

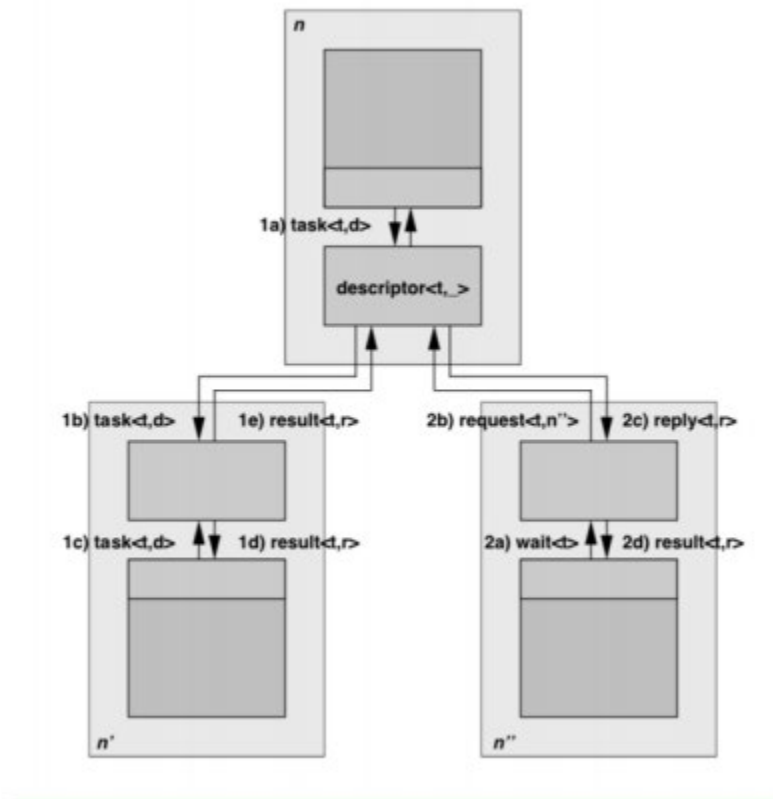


Figure 5 of Schreiner1 is an “execution model” of Distributed Maple.
Ex. 1008, 320.

Petitioner contends that Figure 5 teaches that “the third node (in this case, n'') requests a result from the first node (n), which sends it to the third node after receipt from the second (n').” Pet. 45. Petitioner further contends that the

third node initiates the request through the wait call, indicating that it was evaluating a second expression that was dependent on the result of a first evaluation performed on the second node. A [skilled artisan] would understand that the third node then *could* execute a second mathematical expression using that result, and in fact, that is *likely* why the third node requested the result in the first place. . . . [A] [skilled artisan] would further understand that the third node *could* send the result of the second mathematical expression to the first (root) node if the first node is the node that originally requested that the second mathematical expression be performed, a common case.

Id. at 46 (citing Ex. 1005 ¶ 102) (emphases added).

We note that this block quotation is repeated in the Petition verbatim from Dr. Tufo’s Declaration. Further, Dr. Tufo does not provide a citation to any portion of Schreiner¹ to support his testimony concerning the execution model shown in Figure 1. Because this testimony is conclusory and not supported by objective evidence, it is entitled to little or no weight. *See Velandier v. Garner*, 348 F.3d 1359, 1371 (Fed. Cir. 2003) (“[W]hat the [PTAB] consistently did was accord little weight to broad conclusory statements that it determined were unsupported by corroborating references. It is within the discretion of the trier of fact to give each item of evidence such weight as it feels appropriate.” (citation omitted)); *see also In re Am. Acad. of Sci. Tech Ctr.*, 367 F.3d 1359, 1368 (Fed. Cir. 2004) (“[T]he [PTAB] is entitled to weigh the declarations and conclude that the lack of factual corroboration warrants discounting the opinions expressed in the declarations” (citations omitted)). Because Petitioner has not provided sufficient objective evidence in support of this example, we determine that Petitioner has not shown that this example discloses the third node limitation.

Petitioner’s third example is based on Figure 1 of Schreiner³ which is reproduced below:

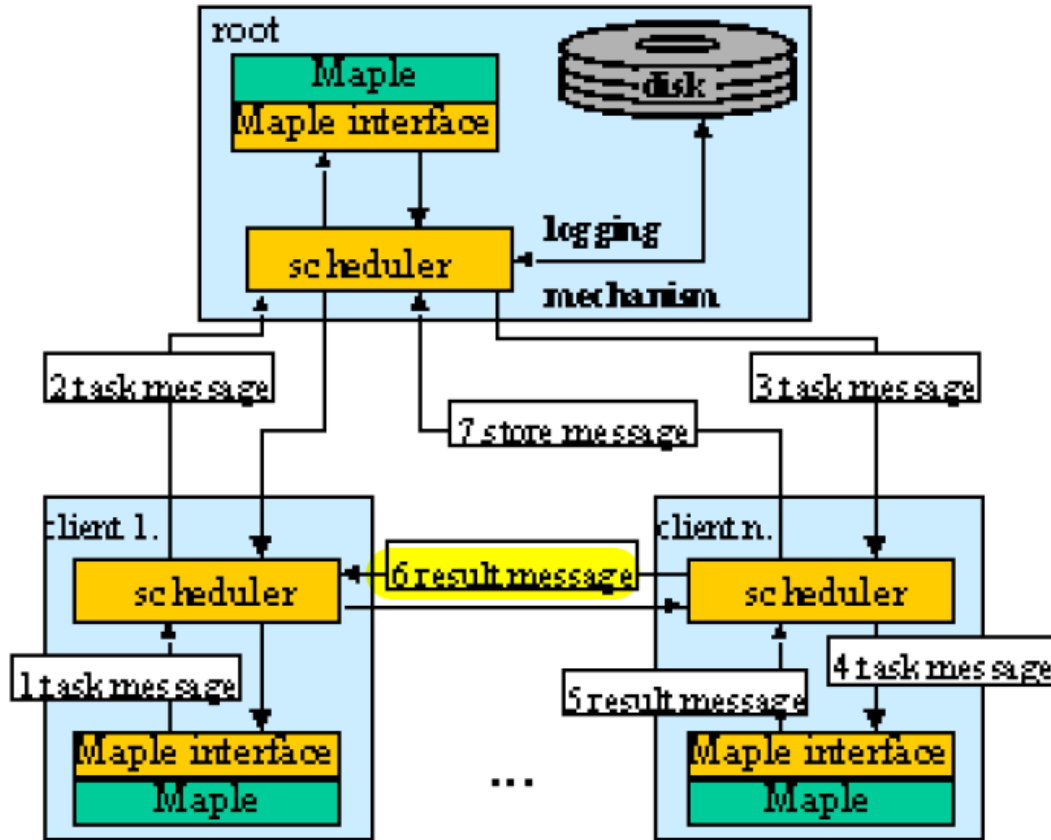


Figure 1: Execution Model

Figure 1 of Schreiner3 is an Execution Model of Distributed Maple.
Ex. 1010, 5.

Petitioner contends that in Figure 1 of Schreiner3, “node ‘client 1’ receives a result for node ‘client n.’” Pet. 47. Petitioner notes that in the case of Figure 1, “the initial tasks originating with the root Maple kernel are not shown.” *Id.* Petitioner further contends that because “the root initiates the mathematical evaluation by distributing tasks to other nodes, node ‘client 1’ is generating subsidiary tasks and evaluating expressions in order to eventually provide results to the root.” *Id.* Petitioner further contends that “once [‘client 1’] receives the result from node ‘client n,’ it may use it to perform at least a second mathematical expression evaluation in order to

process and return results for tasks sent to it by the root.” *Id.* (citing Ex. 1005 ¶ 103).

We note that this portion of the Petition is taken verbatim from Dr. Tufo’s Declaration. Further, Dr. Tufo does not provide a citation to any portion of any of Schreiner³ to support his testimony concerning the execution model shown in Figure 1 of Schreiner³. The testimony is, thus, entitled to little or no weight because it is conclusory and not based on objective evidence. In addition, neither the Petition nor Dr. Tufo attempt to reconcile the argument that client 1 returns a result to the root node when the box labelled “result message” indicates that result messages are sent between client nodes 1–n, not between the root node and client nodes. Because Petitioner has not provided sufficient evidence in support of this example, we determine that Petitioner has not shown that this example discloses the third node limitation.

Based on the foregoing, we determine that Petitioner fails to establish a reasonable likelihood that claim 1 is unpatentable.

10. Claims 4–10, 18–22, 24, 25, 30, 31, 33, 34

Claims 4–10, 18–22, 24, 25, 30, 31, 33, and 34 depend, directly or indirectly, from independent claim 1. Petitioner provides various contentions and cites additional evidence in connection with these dependent claims. Pet. 52–75. The additional contentions and evidence cited by Petitioner do not cure the deficiencies discussed above in connection with claim 1. We, thus, determine that Petitioner fails to establish a reasonable likelihood that claims 4–10, 18–22, 24, 25, 30, 31, 33, and 34 are unpatentable.

E. Ground 2: Alleged Obviousness over Schreiner1, Schreiner2, Schreiner3, Distributed Maple Code, Maple Guide, SPARC IV Article, AMD Article, and MPI Standard

Petitioner provides alternate challenges to claims 30, 31, and 34 in this ground. Pet. 10. We have reviewed Petitioner's contentions and additional evidence cited in support of the challenges to claims 30, 31, and 34, and determine that they do not cure the deficiencies discussed above in connection with independent claim 1. *See* Pet. 75–78. Therefore, we determine that Petitioner fails to establish a reasonable likelihood that claims 30, 31, and 34 are unpatentable.

III. CONCLUSION

Because Petitioner has not established a reasonable likelihood that any of the challenged claims are unpatentable, we do not institute *inter partes* review.

IV. ORDER

In consideration of the foregoing, it is hereby
ORDERED that institution of *inter partes* is denied.

IPR2021-00019
Patent 10,333,768 B2

FOR PETITIONER:

Brent Yamashita
Brennt.yamashita@dlapiper.com

Jonathan Hicks
Jonathan.hicks@dlapiper.com

FOR PATENT OWNER:

Jon Gurka
2jwg@knobbe.com

Ted M. Cannon
2tmc@knobbe.com

Cheryl Burgess
2ctb@knobbe.com