UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
Petitioner,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2025-00795
U.S. Patent No. 10,333,768

PETITION FOR *INTER PARTES* REVIEW UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104

TABLE OF CONTENTS

Petitioner's Exhibit List 5

I. Introduction.....8

II. Grounds for standing 9

III. Note.....9

IV. The '768 patent 9

A. Technological Background 9

B. Overview 10

V. Priority date of the '768 patent 11

VI. History of the '768 patent 11

A. Prosecution 11

B. IPRs 12

VII. Level of ordinary skill in the art 12

VIII. Claim construction.....13

A. "*peer-to-peer architecture*" 13

B. "*a mechanism for …*" 14

IX. Relief requested and the reasons for the requested relief.....16

X. Identification of how the claims are unpatentable.....16

A. Challenged Claims 16

B. Statutory grounds for challenges..... 16

C. Ground 1: Claims 26, 29, 35-36, 39 are obvious over Menon in view of Trefethen, RS6000, and POEref 19

1. Menon 19

2. Trefethen 22

3. RS6000 24

4. POEref 24

5. Combining Trefethen with Menon 24

6. Combining RS6000 with Menon 26

7. Combining POEref with Menon 28

8. Challenges 30

D. Ground 2: Claims 27-28 and 37-38 are obvious over Menon in view of Trefethen, RS6000, and POEref, further in view of MPIref. 78

9. Summary of MPIref 78

10. Combining MPIref with Menon 80

XI. Discretionary denial is inappropriate 93

A. *Fintiv* 93

B. *Advanced Bionics* 95

C. *General Plastic* 96

XII. Conclusion 99

XIII. Mandatory notices 100

A. Real party-in-interest 100

B. Related matters 100

C. Lead and back-up counsel and service information 101

Certificate of Word Count 102

Certificate of Service 103

PETITIONER'S EXHIBIT LIST

Ex.1001	U.S. Patent No. 10,333,768
Ex.1002	Prosecution History of U.S. Patent No. 10,333,768
Ex.1003	Declaration of Dr. Chandrajit L. Bajaj under 37 C.F.R. § 1.68
Ex.1004	*Curriculum Vitae* of Dr. Chandrajit L. Bajaj
Ex.1005	"MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing," Menon et al., SC '97: Proceedings of the 1997 ACM/IEEE Conference on Supercomputing 1997
Ex.1006	"MultiMATLAB: MATLAB on Multiple Processors," Trefethen et al., 1996 Computer Science Technical Report, Cornell University 1996
Ex.1007	"RS/6000 SP: Planning Vol. 1, Hardware and Physical Environment," IBM 2001
Ex.1008	"Operation and Use, Volume 1, Using the Parallel Operating Environment," IBM 2001
Ex.1009	"Single program multiple data" in Dictionary of Algorithms and Data Structures, P. E. Black, (accessible at https://www.nist.gov/dads/HTML/singleprogrm.html) December 2004
Ex.1010	The RS/6000 SP Inside Out" to Barrios et al., IBM 1999
Ex.1011	"High Performance Cluster Computing: Programming and Applications," Rajkumar Buyya, Vol. 2, 1999
Ex.1012	"Analysis of 100Mb/s Ethernet for the Whitney Commodity Computing Testbed," Fineberg et al., NAS Technical Report NAS-97-025 1997
Ex.1013	"A Parallel Linear Algebra Server for Matlab-like Environments," G. Morrow et al., SC '98: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing 1998

Ex.1014	"Parallel MATLAB: Doing It Right," R. Choy et al., Proceedings of the IEEE, Vol. 93, No. 2, 2005
Ex.1015	"Mathematica Parallel Computing Toolkit - Unleash the Power of Parallel Computing," R. Maeder, 2005
Ex.1016	"Mastering MATLAB® 5 – A Comprehensive Tutorial and Reference," D. Hanselman et al., 1998
Ex.1017	"MPI: A Message-Passing Interface Standard," Message Passing Interface Forum, 1994
Ex.1018	"Modern Operating Systems," A.S. Tanenbaum, 2001
Ex.1019	September 26, 2024 Complaint in Western District of Texas
Ex.1020	Order Granting Joint Motion for Entry of Amended Scheduling Order for *Advanced Cluster Systems, Inc. v. Intel Corporation*, 7:24-CV-00245 (W.D.TX.)
Ex.1021	Federal Court Statistics
Ex.1022	Advanced Cluster Systems, Inc.'s Preliminary Infringement Contentions, January 22, 2005, *Advanced Cluster Systems, Inc. v. Intel Corporation*, 7:24-cv-00245 (WDTX)
Ex.1023	Cornell Websites
Ex.1024	"pyMPI–An introduction to parallel Python using MPI," P. Miller 2002
Ex.1025	Amended Joint Claim Construction Chart for *Advanced Cluster Systems, Inc. v. NVIDIA Corporation*, 1:19-cv-02032 (DDE)
Ex.1026	SC97: High Performance Networking and Computing Conference Website
Ex.1027	Declaration of IEEE
Ex.1028	ACM Digital Library - Citations to Menon
Ex.1029	Affidavit of Archive.org

Ex.1030	HPC Wire Article about the SC97 High Performance Computing Conference
Ex.1031	Flyer about the SC97 High Performance Computing Conference
Ex.1032	IEEE Xplore Website
Ex.1033	Article about Cornell Theory Center
Ex.1034	ACM Digital Library
Ex.1035	ACM Digital Library - Citations to Menon
Ex.1036	A Beginner's Guide to the IBM SP
Ex.1037	ACM Digital Library - Citations to MPIref
Ex.1038	IBM Website for RS6000
Ex.1039	IBM Website for POE
Ex.1040	ACS Preliminary Response, IPR2021-00019
Ex.1041	Decision Denying Institution, IPR2021-00019
Ex.1042	Sotera Stipulation, *Advanced Cluster Systems, Inc. v. Intel Corporation*, 7:24-cv-00245 (WDTX)

I. INTRODUCTION

U.S. Patent No. 10,333,768 (the "'768 patent," Ex.1001) contains long-winded claims that obfuscate well-known concepts contained within those claims. The '768 patent's purported invention takes a known mathematical tool (Mathematica) that runs on a single computer and enables it to run in parallel on multiple computers. A Cornell research team, however, made this same solution years earlier by enabling a known mathematical tool (MATLAB), which ran on a single computer, to run in parallel on multiple computers.

The Cornell team's implementation of MATLAB in parallel computers was named MultiMATLAB, and the principal researchers published a pair of papers titled "MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing" ("Menon") and "MultiMATLAB: MATLAB on Multiple Processors" ("Trefethen") more than five years before the earliest priority date of the '768 patent.

Anything purportedly novel about the '768 patent was published in Menon and Trefethen. The '768 patent recites cluster computing in a peer-to-peer architecture—the Cornell team used cluster computing in a peer-to-peer architecture. The '768 patent recites a module of code for communication among the nodes in the computer cluster—the Cornell team used a module of code for communication among the nodes. The '768 patent recites a sequence starting at

node 1, then node 2, then node 3—the Cornell team provided example software code that starts at node 1, then node 2, then node 3. Menon and Trefethen disclose the ’768 patent well before the earliest priority date.

Accordingly, pursuant to 35 U.S.C. §§ 311, 314(a), and 37 C.F.R. § 42.100, Intel Corporation (“Petitioner”) respectfully requests that the Board review and find unpatentable claims 26-29 and 35-39 (the “Challenged Claims”) of the ’768 patent.

II. GROUNDS FOR STANDING

Petitioner certifies the ’768 patent is eligible for IPR and that Petitioner is not barred or estopped from requesting an IPR. 37 C.F.R. § 42.104(a).

III. NOTE

Petitioner cites to exhibits’ original page numbers, unless noted otherwise. **Emphasis** in quoted material has been added. Claim terms are presented in *italics*.

IV. THE ’768 PATENT

A. Technological Background

Dr. Bajaj’s declaration describes the background technology predating the ’768 patent, including (i) cluster computing, (ii) parallelizing mathematical processing, (iii) peer-to-peer architecture, and (iv) sequential processing. *See* Ex.1003, ¶¶28-52.

B. Overview

The ’768 patent describes computer clusters as “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,” but that “[m]any computer application programs are not currently designed to benefit from advantages that computer clusters can offer.” Ex.1001, 1:19-24.

The purported novelty of the ’768 patent is to take a computer program, such as “Mathematica software,” as well as “Maple®, MATLAB®, … other applications employing an interpreter or a kernel,” and then “adding cluster computing functionality” to it. Ex.1001, 1:14-16, 4:14-25. Cluster computing functionality is shown in Figure 2, where “kernel modules 206 a-e” (e.g. Mathematica) are each “in communication with a single cluster node module 204 a-e, respectively.” Ex.1001, 5:19-20, 34-35.

The cluster node modules use Message-Passing Interfaces (“MPIs”) to “distribute tasks among the kernel modules 206 a-e.” Ex.1001, 6:16-21. Tasks include “performing calculations” from the “simple (for example, ‘1+1’), [to] entire subroutines and sequences of code (such as, for example, Mathematica code).” Ex.1001, 11:42-48, 25:41-42, 24:58-61.

There is nothing novel, however, about cluster node modules using MPI to

parallelize an existing mathematical software program on multiple computers. For example, the Cornell team's MultiMATLAB uses an "interface module" based on "the MPI communication standard" to provide "fast and convenient execution of easily parallelizable numerical computations on multiple processors." Ex.1005, Abstract; Ex.1006, Abstract.

V. PRIORITY DATE OF THE '768 PATENT

The '768 patent issued on June 25, 2019. The earliest possible priority date is June 13, 2006. It is unnecessary to determine whether the '768 patent is entitled to this date in this proceeding as all relied-upon prior art predates June 13, 2006. Petitioner does not waive any right or opportunity to dispute the '768 patent's priority date in this or another forum.

VI. HISTORY OF THE '768 PATENT

A. Prosecution

Applicant amended the claims to recite "a mechanism for the nodes to communicate … with each other using a peer-to-peer architecture." Ex.1002, 481. Applicant explained that "peer-to-peer architecture … makes intermediate results of computation available to other nodes." Ex.1002, 488. Applicant contrasted the peer-to-peer architecture to the "gridMathematica software package," which "connect Mathematica kernels in a master-slave relationship rather than a peer-to-peer relationship." Ex.1002, 488; Ex.1001, 12:26-32.

After this amendment, the Examiner allowed the claims. Ex.1002, 80.

B. IPRs

The ’768 patent was challenged in two related IPR proceedings: IPR2021-00019 and IPR2021-00020. In denying institution, the Board determined that claim 1 requires an order of operations involving the first, second and third nodes: “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.” Ex.1041, 26.

The prior art cited herein teaches this precise order of operations. Specifically, Trefethen describes a sequence, involving three nodes, in which a third node receives a result from a second node, performs a second calculation and communicates and passes the result to a first node for display to the user. Ex.1006, 6; Ex.1003, ¶88.

VII. LEVEL OF ORDINARY SKILL IN THE ART

A person of ordinary skill in the art (“POSITA”) in June 2006 would have been someone knowledgeable of and familiar with computer cluster systems and cluster computing techniques available at the time. Such a POSITA would have a bachelor’s degree in computer science, computer engineering, electrical engineering, or an equivalent training, and approximately two years of experience working in the field of cluster computing or parallel processing and would be

knowledgeable regarding high-level scientific computing languages. Additional work experience can substitute for specific educational background, and vice versa. Ex.1003, ¶¶18-21.

VIII. CLAIM CONSTRUCTION

In IPR proceedings, claim terms receive their "ordinary and customary meaning[s]" from the perspective of those of ordinary skill in the art. 37 C.F.R. § 42.100(b). For purposes of this proceeding, Petitioner submits that no claim term requires express construction other than the following terms. *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017).

A. "*peer-to-peer architecture*"

The limitation "peer-to-peer architecture" appears in independent claims 26, 29 and 35.

In the previous IPR proceedings, Patent Owner construed "peer-to-peer architecture" as requiring "an architecture in which 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." Ex.1040, 8. For purposes of this proceeding, Petitioner and its expert used this construction when applying the prior art.

B. "*a mechanism for …*"

This limitation appears in independent claim 26 ("*a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls*"), claim 29 ("*a mechanism for the nodes to communicate results of mathematical expression evaluation with each other*"), and claim 35 ("*a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture*").

The Federal Circuit has repeatedly held that "the term 'mechanism'—without more" does not connote an identifiable structure." *Media Rights Technologies, Inc. v. Capital One Financial Corp.*, 800 F.3d 1366, 1372-73 (Fed. Cir. 2015). Here, the claims recite that the "mechanism" performs a function, e.g. "to communicate results of mathematical expression evaluation," but do not recite specific structural terms. Further, the specification only mentions the term "mechanism" twice with reference to MPI calls in MPI module 302 but nowhere describes the mechanism's structure. Ex,1001, 14:36-43; 25:22-47; Ex.1003, ¶¶63-72.

The specification does not clarify the structure of "mechanism," which is generally considered a placeholder for "means," and thus should be interpreted as invoking pre-AIA 35 USC 112, sixth paragraph. *See generally*, *Welker Bearing*

Co. v. PHD, Inc., 550 F.3d 1090 (Fed. Cir. 2008) (holding that "mechanism for…" is a means-plus-function limitation).

The specification describes "MPI module 302" including "basic MPI calls such as, for example, relatively low-level routines that map MPI calls." Ex.1001, 13:9-12. "In some embodiments, basic MPI calls include calls that send data, equations, formulas, and other/or other expressions." Ex.1001, 13:9-16, Table B; Ex.1003, ¶¶63-72.

Accordingly, for purposes of this Petition, a POSITA would have understood the specification of the '768 patent to provide the following structure and respective functions for "mechanism":

Structure:

- An MPI module.

Function:

- Claim 26: communicate results of mathematical expression evaluation with each other using asynchronous calls.
- Claim 29: communicate results of mathematical expression evaluation with each other.
- Claim 35: communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture. Ex.1003, ¶¶72.

To the extent that 35 USC 112, sixth paragraph is not invoked, the analysis

below nevertheless shows that the prior art meets the plain and ordinary meaning of "mechanism."

IX. RELIEF REQUESTED AND THE REASONS FOR THE REQUESTED RELIEF

Petitioner asks that the Board institute an IPR trial and cancel the Challenged Claims in view of the analysis below.

X. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

A. Challenged Claims

Petitioner challenges claims 26-29 and 35-39. Claims 1-25 and 30-34 are challenged in a separate, contemporaneously-filed petition

B. Statutory grounds for challenges

Grounds	Claims	Basis (pre-AIA)
#1	26, 29, 35-36, 39	35 U.S.C. §103 over Menon in view of Trefethen, RS6000 and POEref
#2	27-28 and 37-38	35 U.S.C. §103 over Menon in view of Trefethen, RS6000 and POEref further in view of MPIref

"MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing" ("Menon") is a paper authored by Vijay Menon and Anne Trefethen of the Cornell research team. It qualifies as a printed publication because it was published and distributed at the Proceedings of the 1997 ACM/IEEE Conference on Supercomputing 1997. Ex.1005; Ex.1026; Ex.1027. Menon was also published

and distributed on Cornell's website at least as early as 2001. Ex.1023; Ex.1029. Menon was reasonably located by POSITAs because it had been cited 17 times in papers prior to the priority date of the '768 patent. Ex.1028, 4-9, Ex.1029, 1-3; *see also* Ex.1003, ¶84-86.

"MultiMATLAB: MATLAB on Multiple Processors" ("Trefethen") is a second paper authored by Vijay Menon, Anne Trefethen, and other members of the Cornell research team. It qualifies as a printed publication because it was published and distributed on the webpage of one of its authors at least as early as 2001. Ex.1023; Ex.1029. Trefethen was also published on a Cornell University website no later than May 10, 1996. Ex.1023; Ex.1029. Trefethen was reasonably located by POSITAs because it had been cited 10 times in papers prior to the priority date of the '768 patent. Ex.1035, 2-3; *see also* Ex.1003, ¶¶90-91.

"RS/6000 SP: Planning Vol. 1, Hardware and Physical Environment" ("RS6000") is a product manual by IBM in support of the IBM RS/6000 SP computing solution. Ex.1007. RS6000 qualifies as a printed publication because it was catalogued and indexed in IBM's online document library at least as early as 2002 on IBM's website. Ex.1007; Ex.1029; Ex.1038; *see also* Ex.1003, ¶¶93-94

"Operation and Use, Volume 1, Using the Parallel Operating Environment" ("POEref") is a product manual published by IBM in support for the Parallel Operating Environment running on an RS/6000 system. POEref qualifies as a

printed publication because it was catalogued and indexed in IBM's online document library at least as early as 2001. Ex.1039; Ex.1029. POEref was distributed with the IBM SP2 systems and published at least as early as 2001. Ex.1008, 157; Ex.1036, 14; *see also* Ex.1003, ¶¶97-99.

"MPI: A Message-Passing Interface Standard" ("MPIref") is the message passing standard written by the Message Passing Interface Forum. MPIref qualifies as a printed publication because it was distributed to POSITAs using FTP mail servers by the University of Tennessee and Oak Ridge National Library in 1994. Ex.1017, cover page; Ex.1029; Ex.1003, ¶103. It was published on MPI Forum's website no later than July 3, 1998. Ex.1029, 1-3. MPIref was reasonably located by POSITAs because it had been cited 74 times in papers with online publications dates before the earliest priority date of the '768 patent. Ex.1037, 2-16; *see also* Ex.1003, ¶¶103-104.

Further, each prior art reference is (i) more than 20 years old; (ii) is regular on its face with no signs of obvious alterations; and (iii) was found in a place where it could be expected to be found. Accordingly, each document is prima facie authenticated as an ancient document. *See* Fed. R. Evid. 901(8).

Petitioner cites additional documents to evidence a POSITA's knowledge and provide context regarding a POSITA's understanding of the art. *See, e.g., Yeda Research v. Mylan Pharm. Inc.*, 906 F.3d 1031, 1041-42 (Fed. Cir. 2018).

C. Ground 1: Claims 26, 29, 35-36, 39 are obvious over Menon in view of Trefethen, RS6000, and POEref

1. Menon

Menon discloses "MultiMATLAB," a platform that enables MATLAB to operate in parallel on multiple processors nodes." Ex.1005, Abstract, Title.

a) Menon describes cluster computing

Menon discloses MultiMATLAB as "a general extension" of MATLAB Ex.1005, Abstract. MultiMATLAB comprises several processors, each configured to run a MATLAB process. Ex.1005, 3, FIG. 1. Each MATLAB process communicates "with other processes through a communication layer." Ex.1005, 3. The communication layer uses "MPI [Message Passing Interface]" to communicate "over the parallel platform's interconnection network." Ex.1005, 3, 5. Figure 1 of Menon illustrates an example MultiMATLAB architecture. Ex.1005, FIG. 1.

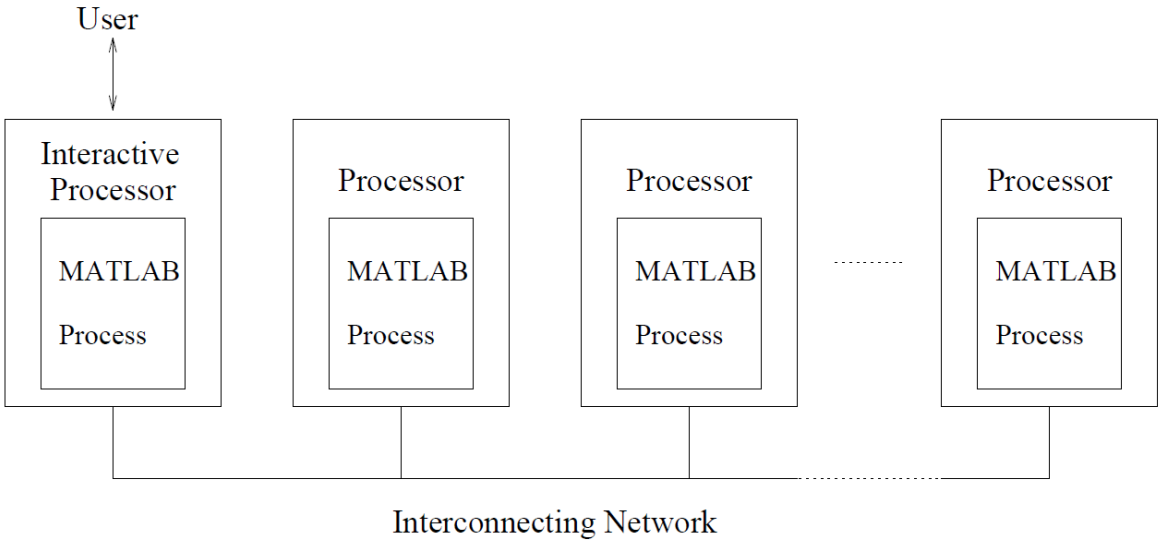


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1.

A user "interacts directly with … the interactive process," and the "other processes wait for commands from the interactive MATLAB process." Ex.1005, 3, 6.

b) Menon describes parallelizing mathematical calculations

Menon describes using MultiMATLAB on a "parallel platform," such as the "the IBM SP2" to calculate mathematical expressions, such as matrix equations, in parallel. Ex.1005, 2, 5, 11.

c) Menon describes a peer-to-peer architecture

Menon explains that MultiMATLAB works with different parallel programming paradigms. Ex.1005, 8. MultiMATLAB can operate as a

"master/slave" environment" Ex.1005, 8. Alternatively, MultiMATLAB can provide peer-to-peer MATLAB process communication using "SPMD/Message passing" to provide to "point to point and collective communication." Ex.1005, 9-10.

To provide "point to point" communication among MATLAB routines, Menon describes "message passing routines" that achieve "finer grain communication … between any processes." Ex.1005, 9, 8. Table 1 of Menon depicts the message passing routines.

MultiMATLAB commands	
Starting and stopping MultiMATLAB (MPICH only)	
Start(n)	Initializes n remote MATLAB processes.
Quit	Closes remote MATLAB processes.
Running commands on multiple processors	
Nproc	Returns number of MATLAB processes.
ID	Returns ID of calling MATLAB process.
Eval(pid,'command')	Evaluates a command on one or more MATLAB process.
Master/Slave Communication	
Put(pid,dataname)	Put data from the master process onto one or more remote processes.
Get(pid,dataname)	Get data from a remote process onto the master process.
SPMD Communication	
Send(pid,data)	Send data from one process to another.
Recv(pid)	Receive data sent from another process.
Barrier	Synchronize processes.
Bcast(pid,data)	Broadcasts data from processor pid to all processes.
Sum(data)	Adds data across all processors to form a global sum.
Collect(data)	Collects local data segments into a single global one.

Table 1: Selected commands in the MultiMATLAB system

Commands for point to point communication (including Send and Recv)

Ex.1005, Table 1 (annotated); Ex.1003, 82

Menon also discloses that MultiMATLAB uses a set of components to provide direct communication with other modules, namely, the "MEX Routines," the "MultiMATLAB interface module," an "indirection table" and "communication layer." Ex.1005, 3, 5.

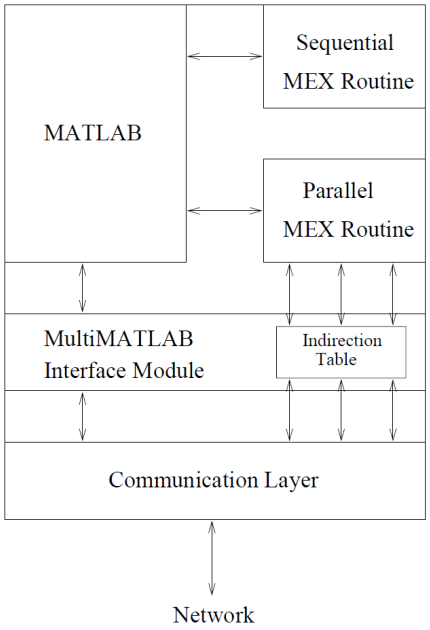


Figure 2: MATLAB Process Address Space

Ex.1005, FIG. 2.

2. Trefethen

Trefethen describes an implementation of MultiMATLAB. Trefethen discloses how a user can interact with a single node and run MATLAB on multiple IBM SP2 nodes. Ex.1006, 3-9.

a) Trefethen describes sequential processing from one node to the next

Trefethen describes "point-to-point communication is accomplished by send

and receive commands." Ex.1006, 5. "SPMD programs can be built upon Send and Recv commands." Ex.1006, 6. Trefethen describes an example script that demonstrates nodes directly communicating with each other using the Send and Recv commands. Ex.1006, 6:

user instructions

Instruction for MATLAB process ID=0 to create a=1 and send "a" to MATLAB process ID=1

```
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;
```

Instruction for the last MATLAB process to receive the variable a and double it

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

Instruction for a middle MATLAB process ID to receive the variable "a," double it, and send the doubled variable to the next MATLAB process ID=ID+1

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

Output of final result (a=32)

Ex.1006, 6 (annotated); Ex.1003, ¶88.

The sample code shows:

- node 0 sending data to node 1,
- node 1 performing a mathematical calculation and then sending that result to node 2,
- node 2 performing a mathematical calculation and then sending that result to the next node ("and so on"), and

- the result is output to the user. Ex.1006, 6; Ex.1003, ¶88.

3. RS6000

"RS/6000 SP: Planning Vol. 1, Hardware and Physical Environment" ("RS6000," Ex.1007) is part of an IBM installation guide for the IBM SP2 system used in Menon. Ex.1010, 6; Ex.1007, 1-2.

4. POEref

"Operation and Use, Volume 1, Using the Parallel Operating Environment" ("POEref," Ex.1008) is IBM product documentation for "the IBM Parallel Environment (PE)" and "its Parallel Operating Environment (POE)." POE discussed in Menon as initializing the MATLAB processes. Ex.1008, ix; Ex.1005, 6.

5. Combining Trefethen with Menon

a) Analogous art

Menon and Trefethen are analogous prior art because they pertain to the same field of endeavor as the '768 patent, namely, the "the field of cluster computing" and are reasonably pertinent to the problem of the '768 patent, namely, executing a single program on multiple nodes. Ex.1005, 2; Ex.1006, Abstract; Ex.1001, 1:14-15; Ex.1003, ¶108-112.

b) Motivation to combine

A POSITA would have combined Trefethen with Menon for several reasons.

First, Menon suggests the combination by expressly citing to Trefethen and explaining that Menon builds on the teaching of Trefethen. Ex.1005, 1-2.

Second, Trefethen describes an implementation of MultiMATLAB and "how the [MultiMATLAB] system is used" with examples of code executed on a MultiMATLAB system to perform mathematical calculations, Ex.1006, 2, while Menon describes "tak[ing] advantage of high performance distributed memory processors" to enable MATLAB to perform mathematical calculations in parallel. Ex.1005, Abstract, 16. Accordingly, Menon and Trefethen together represent a combination of prior art elements according to known methods (executing MATLAB code on a multiprocessor system) to yield the predictable result discussed in both papers: the MultiMATLAB system providing parallel execution of MATLAB on a high-performance distributed memory multiprocessor to perform mathematical calculations. Ex.1005, Abstract; Ex.1006, 2-7.

Third, a POSITA seeking to understand MultiMATLAB's capabilities would have consulted the Cornell website dedicated to MultiMATLAB, which contained links to both papers. Ex.1023. Co-author Vijay Menon likewise included links to both papers within his Cornell web page. Ex.1023. Further, the authors "Vijay Menon" and "Anne E. Trefethen" are common across both papers. *See* Ex.1005, 1 and Ex.1006, 1. Accordingly, any POSITA seeking to understand MultiMATLAB would have encountered and considered both Menon and

Trefethen.

c) Predictable and reasonable expectation of success

The combination of Menon and Trefethen to parallelize MATLAB would have been predictable because researchers had a strong desire to parallelize MATLAB. Ex.1006, 11 ("Many people must have thought about parallelizing MATLAB"). The goal of parallelizing MATLAB was "inevitable." Ex.1006, 12. The combination of Menon and Trefethen succeeded in achieving this goal: the MultiMATLAB system provides parallel execution of MATLAB on a high-performance distributed memory multiprocessor to perform mathematical calculations. Ex.1005, 2 ("In this paper, we … achieve … MATLAB code with high-performance parallel routines").

Accordingly, a POSITA would have been motivated and found it predictable to combine Trefethen with Menon. Ex.1003, ¶¶113-116.

6. Combining RS6000 with Menon

a) Analogous art

RS6000 is analogous prior art because it pertains to the same field of endeavor as the '768 patent, namely, the "the field of cluster computing" and is reasonably pertinent to the problem of the '768 patent, namely, executing a single program on multiple nodes. Ex.1007, 1; Ex.1001, 1:14-15; Ex.1003, ¶¶117-120.

b) Motivation to combine

A POSITA would have combined the teachings of RS6000 with Menon because Menon expressly suggests the combination. Specifically, Menon describes using the "IBM SP2, a modern high performance distributed memory multiprocessor." Ex.1005, 5. RS6000 is part of an installation guide for the IBM RS/6000 SP providing "an overview of the IBM RS/6000 SP."[1] Ex.1007, 1.

A POSITA seeking to understand MultiMATLAB would be motivated to seek the documentation corresponding to the systems for which "MultiMATLAB was design[ed] specifically." Ex.1005, 6. Additionally, the IBM SP2 was considered "a state-of-the-art parallel computing system" designed to "execute serial and parallel applications simultaneously." Ex.1007, 1.

A POSITA would also have combined RS6000 with Menon because Menon and RS6000 together represent a combination of prior art elements (Menon describes the MultiMATLAB architecture on an "IBM SP2" and RS6000 provides details of that system) according to known methods (as taught by Menon) to yield the predictable result of MultiMATLAB implemented on an IBM SP2 having the physical attributes disclosed by RS6000. Ex.1005, Abstract, 6; Ex.1007, 1, 11.

[1] "IBM SP2" is the same as "IBM RS/6000 SP." Ex.1010, 6 ("the SP2 was renamed to simply the SP").

Combining RS6000 with Menon (and Trefethen) would have been predictable and would have had a reasonable expectation of success at least because Menon expressly discloses implementing its MultiMATLAB architecture on an IBM SP2 and that MultiMATLAB "was design[ed] specifically for the IBM SP2." Ex.1005, 6, 14 ("These results demonstrate that the MultiMATLAB system can be an effective platform for parallel computing"). Ex. 1003, ¶¶121-124.

7. Combining POEref with Menon

a) Analogous art

POEref is analogous prior art pertaining to the same field of endeavor as the '768 patent, namely, "the field of cluster computing" and is reasonably pertinent to the problem of the '768 patent, namely, executing a single program on multiple nodes. Ex.1008, ix ("PE Software is designed to run on an…RS/6000 network cluster"); Ex.1001, 1:14-15; Ex.1003, ¶¶125-128.

b) Motivation to combine

A POSITA would have combined the teachings of POEref with Menon because Menon suggests the combination by teaching that MultiMATLAB uses "POE, IBM's Parallel Operating Environment." Ex.1005, 6. A POSITA seeking to understand MultiMATLAB would be motivated to review the documentation corresponding to the systems expressly discussed in Menon. Ex.1005, 6.

Further, Menon describes using POE to initialize the MATLAB processes.

Ex.1005, 6. POEref describes that the "Parallel Operating Environment (POE)" provides the command "poe" "to load and execute programs on remote nodes" and "initializ[e] the local environment." Ex.1008, ix, 28.

A POSITA would also have considered and combined POEref with Menon because Menon and POEref together represent a combination of prior art elements (Menon describing the MultiMATLAB architecture using IBM's POE and POEref "describ[ing] … [the] Parallel Operating Environment (POE)") according to known methods (as taught by Menon) to yield the predictable result of an implementation of MultiMATLAB utilizing the Parallel Operating Environment. Ex.1005, 6; Ex.1008, ix, 28.

The results of the combination would have been predictable and there would have been a reasonable expectation of success at least because Menon expressly discusses using POE, which is the subject of POEref, to implement MultiMATLAB. Ex.1005, 6, 14 ("These results demonstrate that the MultiMATLAB system can be an effective platform for parallel computing"). *See* Ex.1003, ¶¶129-132.

8. Challenges

Claim 26

[26.0] ***A computer cluster comprising:***

Menon discloses a "MultiMATLAB architecture" that provides "high-performance parallel routines." Ex.1005, 2. MultiMATLAB's cluster comprises multiple processor nodes, where "each processor in a parallel platform individually runs a MATLAB process" and each process is "provided with the ability to communicate with other processes through a communication layer. Ex.1005, 3.

Figure 1 illustrates the MultiMATLAB architecture:

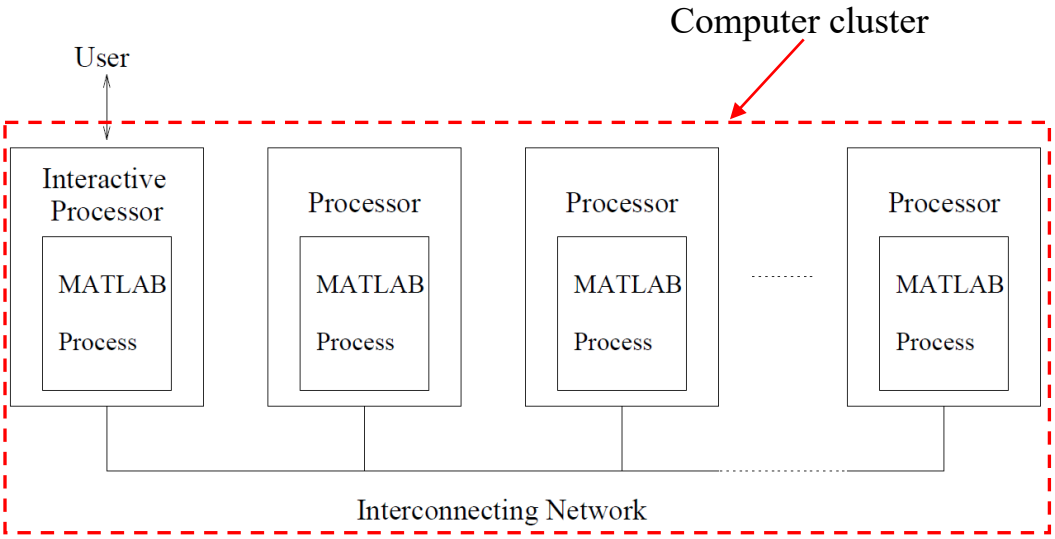


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶¶134

Menon discloses that MultiMATLAB was designed "specifically for the IBM SP2." Ex.1005, 5, 6. The IBM SP2 system "execute[s] parallel programs on … a **networked cluster** of RS/6000 processors." Ex.1008, 1.

Menon also describes using a plurality of processors to perform mathematical calculations. Ex.1005, 13, FIGs. 7, 8 (using "8 processors" and "32 processors," respectively).

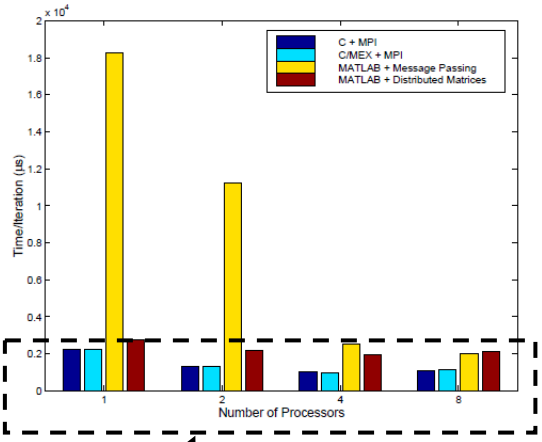


Figure 7: Parallel Conjugate Gradients on the IBM SP-2: 256 × 256 matrix

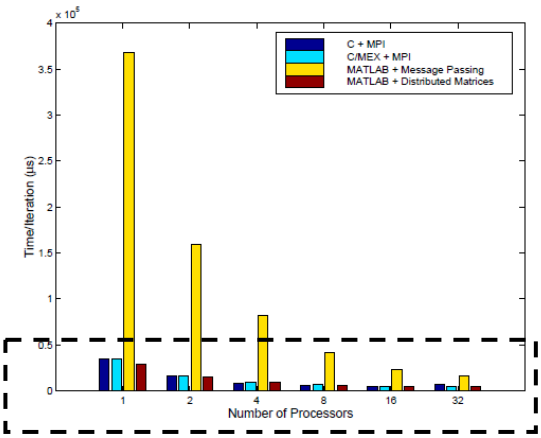


Figure 8: Parallel Conjugate Gradients on the IBM SP-2: 1024 × 1024 matrix

8 processors

32 processors

Ex.1005, FIGs. 7 and 8 (annotated)

Accordingly, Menon's MultiMATLAB cluster with multiple MATLAB processes implemented on IBM SP2 processor nodes to carry out mathematical computations in parallel, in view of RS6000's description that the IBM SP2 is a networked cluster computing system for executing software in parallel, renders obvious this limitation. Ex.1003, ¶¶133-139, 323.

[26.1.0] ***a plurality of nodes:***

Menon describes running "MultiMATLAB" on "the IBM SP2" with "32 processors" (*plurality of nodes*). Ex.1005, 5, 13.

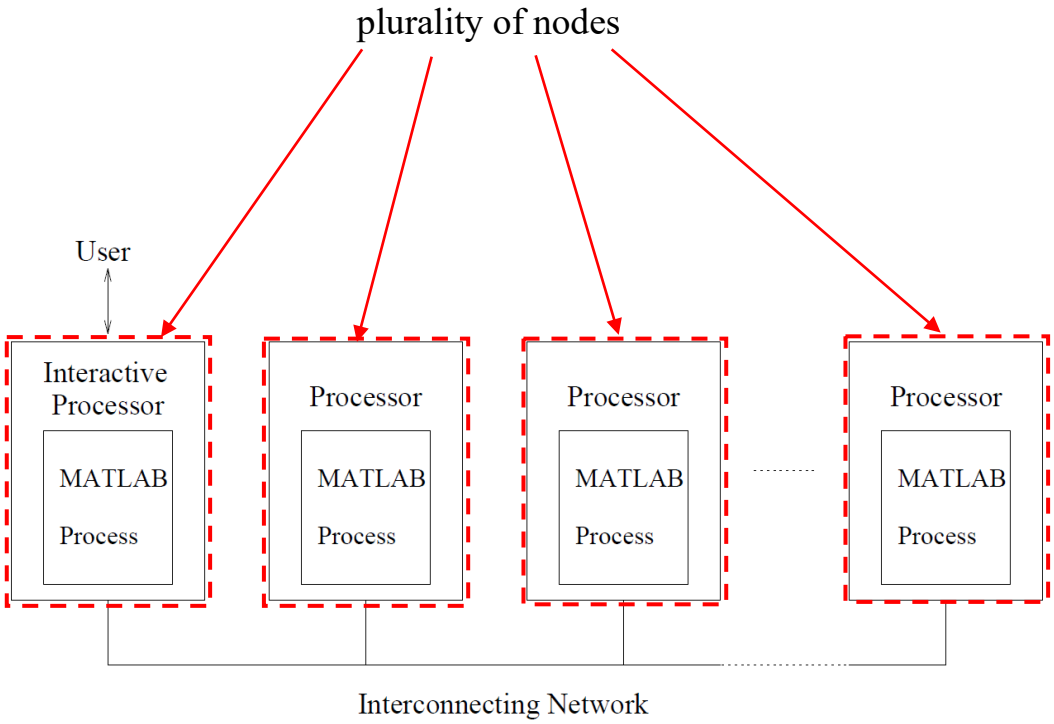


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶¶140

RS6000 describes the IBM SP2 providing a plurality of nodes, each node having a "Symmetric MultiProcessor (SMP)" ("*hardware processor*") for executing "parallel applications." Ex.1007, 1.

Thus, Menon discloses using 32 processor nodes of an IBM SP2 system for executing a series of parallel processes and RS6000 teaches that each node of the

IBM SP2 has a symmetric multiprocessor, rendering obvious this limitation. Ex.1003, ¶¶140-142, 324.

[26.1.1] ***wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster,***

First, Menon discloses that the user "interacts directly with" a MATLAB process called the "interactive process" (*node*). Ex.1005, 3. Menon discloses that, in MultiMATLAB implemented on the IBM SP2, "all MATLAB process are started via POE, IBM's Parallel Operating Environment." Ex.1005, 6. Then, "[o]n non-interactive processes, a separate top-level MEX Routine is immediately run upon initialization of the system. This MEX Routine runs in a loop in which it waits for MATLAB commands to arrive from the interactive process and executes them in its own MATLAB environment." Ex.1005, 4.

Second, POEref discloses an initialization process where, in IBM's Parallel Operating Environment, the user enters the "poe" command, explaining that "[w]hen you invoke **poe**, <u>the Partition Manager</u> allocates processor nodes for each task and <u>initializes the local environment</u>. It <u>then loads your program, and reproduces your local environment, on each processor node</u>." Ex.1008, 28.

Accordingly, Menon discloses that the interactive process receives a command from the user to start the MATLAB processes using IBM's POE, where, upon initialization, the non-interactive MATLAB processes wait for further

commands, and POEref further discloses the "poe" command in POE that initializes a local environment and reproduces that local environment on each remote processor node rendering obvious *wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster*. Ex.1003, ¶¶143-146, 325.

[26.1.2] *wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and*

As shown in [26.1.1], Menon discloses a user using IBM's POE to launch the MATLAB processes on the IBM SP2 processor nodes.

POEref provides additional details of the cluster initialization process described in Menon. POEref describes that POE "ensures that each API initializes properly." Ex.1008, 3. One such API is "MPI." Ex.1008, 3. Further, POEref describes that POE "connects standard I/O to each remote node so the parallel tasks can communicate with the home node." Ex.1008, 3.

Accordingly, Menon's teaching that the user employs POE to start MATLAB processes on each processor node of the MultiMATLAB architecture (*the cluster initialization process*) in view of POEref's disclosure that the initialization process ensures proper initialization of the communication API for MPI and communication among the processor nodes (*establishing communication*

among two or more of the nodes) renders obvious this limitation. Ex.1003, ¶¶267-270, 326.

[26.1.3] *[wherein one or more of the nodes are configured to receive:] an instruction from a user interface or a script; and*

First, Menon describes one of the MATLAB processes as an interactive MATLAB process with which a user interacts directly. Ex.1005, 3, 6. "For each of *p* MATLAB processes, POE assigns an identification number between 0 and *p* – 1. The process numbered 0 is considered the interactive MATLAB." Ex.1005, 6.

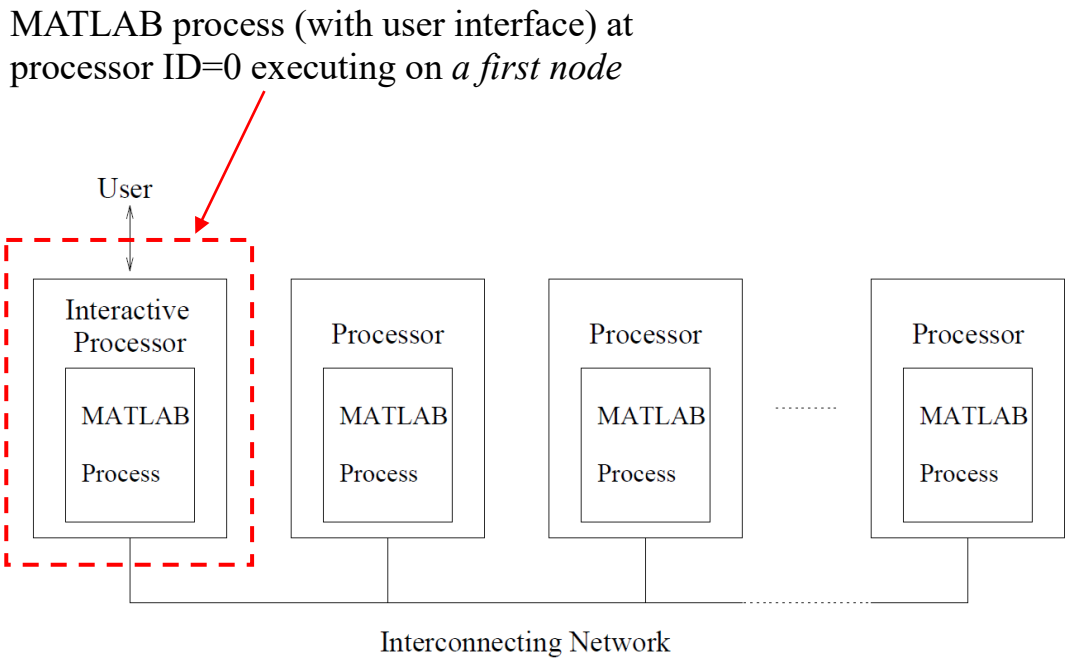


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶328

Menon discloses "MATLAB's Partial Differential Equation Toolkit," which "provides an intuitive graphical interface allowing users to specify geometries, differential equations, and boundary conditions." Ex.1005, 6. Ex.1003, ¶174.

Second, Trefethen describes that, rather than entering explicit MATLAB commands as an argument string, the user can use an "m-file" to provide instructions. Ex.1006, 5. That is, a user may "want to execute a program (an m-file) rather than a single line of text," and "[a] command such as Eval('filename') achieves this effect." Ex.1006, 5.

A POSITA would recognize and know that the "m-file" described in Trefethen is a "script." *See*, e.g., Ex.1016, 5 ("These files are called ***script files*** or simply ***M-files***") (emphasis original).

Accordingly, Menon's teaching that interactive processor node received user instructions, in view of Trefethen's teaching of a user inputting an m-file into MATLAB that is used by the main MATLAB process to assign different actions to different processes, renders obvious this limitation. Ex.1003, ¶¶327-333.

[26.2] ***a mechanism for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;***

First, Menon discloses that MultiMATLAB uses "MEX routines" and that "[a]ll parallel functionality in the MultiMATLAB system is provided through MEX routines." Ex.1005, 3. The MEX routines send the "commands over the

underlying communication layer to the processes corresponding to the specified IDs." Ex.1005, 4.

MultiMATLAB commands	
Starting and stopping MultiMATLAB (MPICH only)	
Start(n)	Initializes n remote MATLAB processes.
Quit	Closes remote MATLAB processes.
Running commands on multiple processors	
Nproc	Returns number of MATLAB processes.
ID	Returns ID of calling MATLAB process.
Eval(pid,'command')	Evaluates a command on one or more MATLAB process.
Master/Slave Communication	
Put(pid,dataname)	Put data from the master process onto one or more remote processes.
Get(pid,dataname)	Get data from a remote process onto the master process.
SPMD Communication	
Send(pid,data)	Send data from one process to another.
Recv(pid)	Receive data sent from another process.
Barrier	Synchronize processes.
Bcast(pid,data)	Broadcasts data from processor pid to all processes.
Sum(data)	Adds data across all processors to form a global sum.
Collect(data)	Collects local data segments into a single global one.

Table 1: Selected commands in the MultiMATLAB system

Commands implemented in MEX Routines

Ex.1005, Table 1 (annotated); Ex.1003, ¶336

Second, Trefethen discloses that the commands used in MultiMATLAB are "written as a C file" and "interfaced to MATLAB via the standard MATLAB Fortran/C/C++ interface known as MEX." Ex.1006, 10. The "[h]igher-level MultiMATLAB commands are usually built on higher-level MPI commands." Ex.1006, 10. Trefethen teaches that "MultiMATLAB is currently built upon the standard send and receive variants" of MPI. Ex.1006, 10. Trefethen also discloses sending tasks to other processors using and provides the example "Eval([4 5] ,

'cond(hilb(ID))')". Ex.1006, 4. The MATLAB processes on processor ID=4 and ID=5 then each perform the task "cond(hilb(ID))." Ex.1006, 4; Ex.1003, ¶339.

Figure 2 and Figure 1 of Menon show the MEX routines and the interconnection network linking the processors.

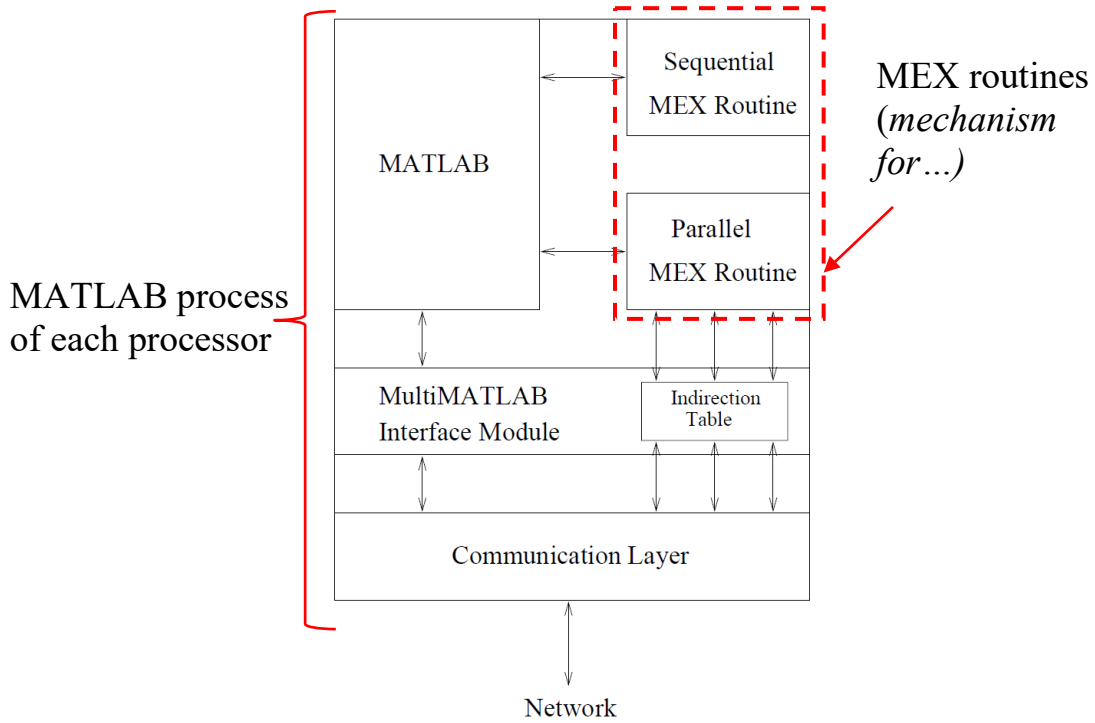


Figure 2: MATLAB Process Address Space

Ex.1005, FIG. 2 (annotated); Ex.1003, ¶336

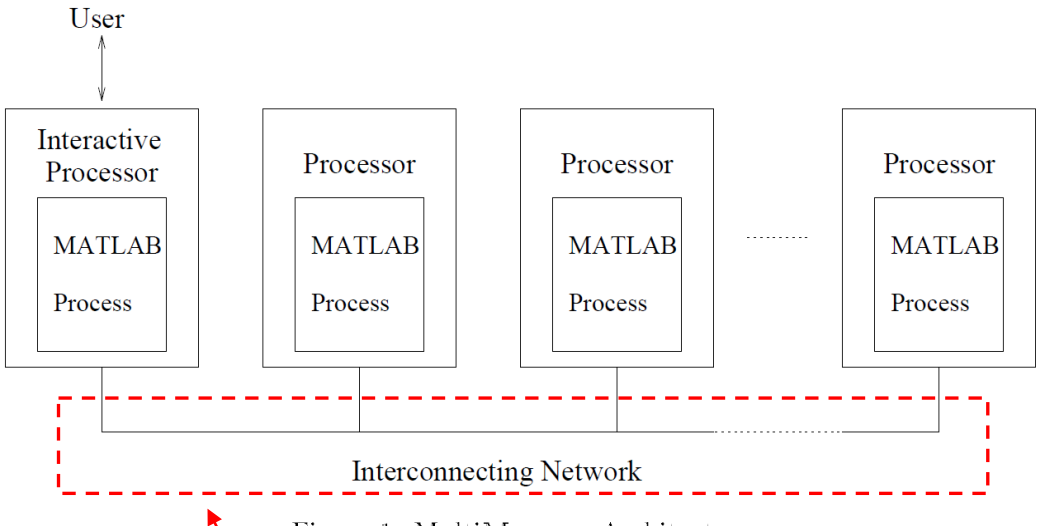


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶336

Interconnecting network connecting each MATLAB process on a processor to any other in the MultiMATLAB architecture

Thus, the MEX Routines that contain program code to implement the standard MPI send and receive calls to send commands (e.g., tasks) and data directly to other processes renders obvious an MPI module. Ex.1003, ¶340.

Trefethen also provides details of the SPMD communication commands. Ex.1006, 5-7. Specifically, Trefethen provides an example where mathematical results are communicated from one MATLAB process to another using "asynchronous" calls such as Send and Recv:

```
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
```

Asynchronous calls

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.

Ex.1006, 6 (annotated); Ex.1003, ¶341.

While Trefethen's example uses six processors, it would have been obvious to a POSITA that Trefethen contemplates any number of processors, including three processors. Ex.1003, ¶167.

Trefethen further discloses that MultiMATLAB commands also deliver a mathematical result to a designated process. For example, the instruction "Eval ('Sum(1, [1 ID Nproc])') returns the result to process 1. "If the first argument ["1,"] is omitted, the result is returned (broadcast) to all processes." Ex.1006, 7.

Accordingly, Menon's MEX routines, having code to implement MPI calls, that enable SPMD point-to-point communication so that each processor can communicate directly with the others over an interconnecting network, further in view of Trefethen's teaching that the SPMD communication enables mathematical

expression evaluation results from a first MATLAB process on a first processor to be communicated to a second MATLAB process on a second processor, and the mathematical expression results obtained there to be communicated to a third MATLAB process on a third processor, and so on, renders obvious this limitation. Ex.1003, ¶¶334-343.

[26.3] ***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;***

First, Menon discloses that each of the IBM SP2 processors "individually runs a MATLAB process." Ex.1005, 3. The '768 patent identifies MATLAB as an example of a single-node kernel. *See, e.g.,* Ex.1001, 4:14-21.

Figure 1 below shows a MATLAB process executing on each of multiple processors.

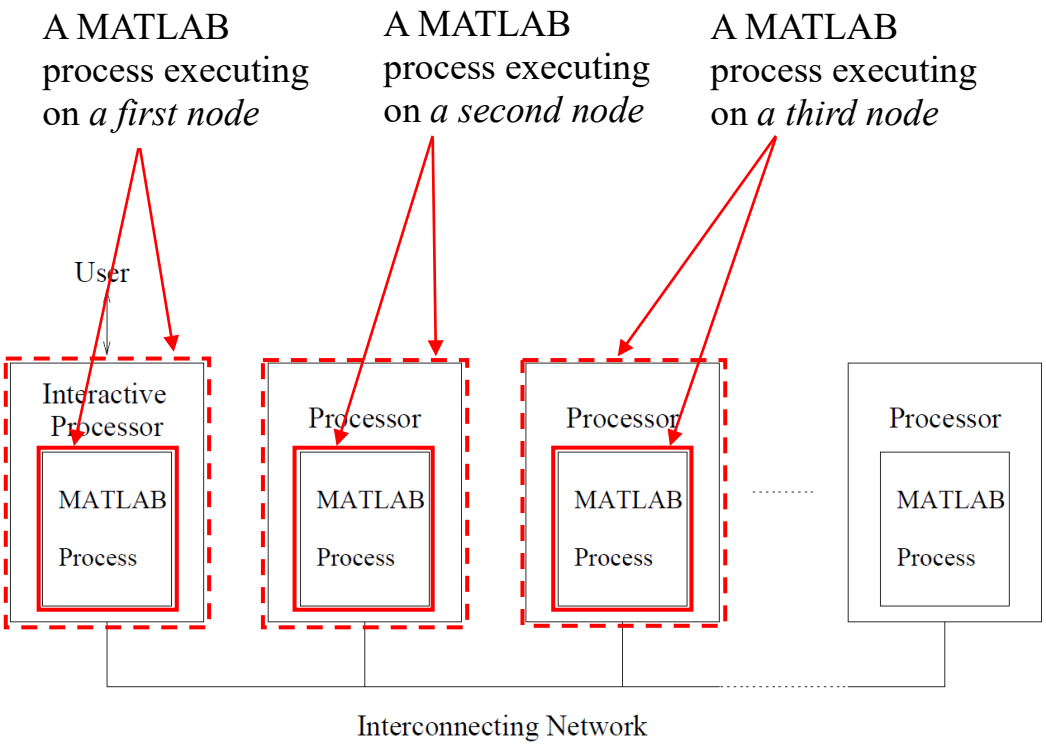


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶148

Menon explains, with reference to Figure 2, that each MATLAB process contains MATLAB and that MEX Routines "may be run directly from [the] MATLAB" environment. Ex.1005, Abstract, 3, FIG. 2. The MEX commands are "interfaced to MATLAB via the standard MATLAB [] interface system known as MEX." Ex.1006, 10.

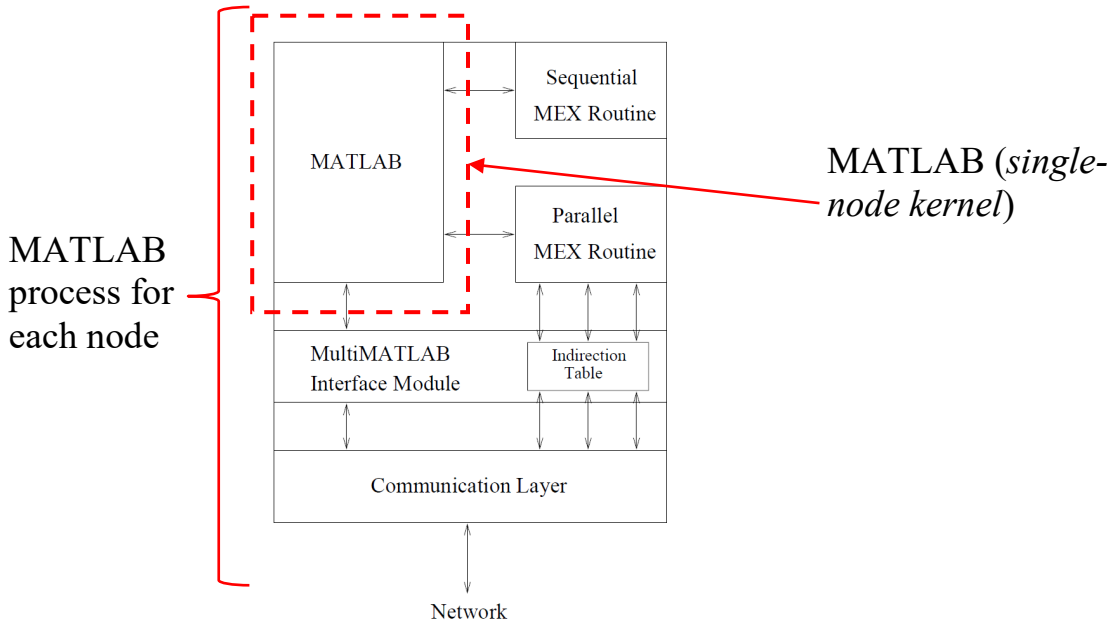


Figure 2: MATLAB Process Address Space

Ex.1005, FIG. 2 (annotated); Ex.1003, ¶149

Because MATLAB is "MATLAB software," a POSITA would recognize MATLAB is *program code* executed by *a hardware processor*. Ex.1006, 2; Ex.1005, 3, Ex.1007, 3; Ex.1003, ¶150. Accordingly, each node executing MATLAB discloses each node executing *a single-node kernel*.

Second, Menon describes MATLAB as "the numerical computing environment of choice on uniprocessors for hundreds of thousands of engineers and scientists" and provides an "extensive library of high quality numerical routines." Ex.1005, 1. As one example, Menon describes implementations of "the conjugate gradients algorithm." Ex.1005, 11-13. Trefethen provides additional

examples, such as each node doubling a variable, "fast Fourier transform," and "polynomial zerofinding." Ex.1006, 2-6.

Each MATLAB process "only does the computation required for its local data" but exchanges data with other MATLAB processes. Ex.1005, 13, 11-13. Thus, MATLAB (*a single-node kernel*) causes the node of the IBM SP2 to perform its mathematical computation (*program code … capable of causing a hardware processor to evaluate mathematical expressions)*.

Third, Menon teaches that a processor accesses MATLAB from a *computer-readable medium*, because Menon describes the MATLAB process, which includes MATLAB, as being in an "address space." Ex.1005, FIG. 2, 5.

As discussed previously, a POSITA would understand Menon's "address space" to be a "memory locations [of a process] … contain[ing] the executable program." Ex.1018, 67-68. Because Menon's address space refers to a computer's memory, a POSITA would recognize that the MATLAB process and MATLAB are in, and accessed from, a *computer-readable medium*. The *computer-readable medium* of the processors of the IBM SP2 can be in the form of hard disk drives, or memory cards, as taught by RS6000. Ex.1007, 6, 10 (IBM SP2 nodes have disk drives), 10 (IBM SP2 nodes have "memory cards" that support a "maximum of 64 GB").

Accordingly, Menon's disclosure of each processor accessing the MATLAB process address space (*computer-readable medium*) with MATLAB (*computer program code for a single-node kernel*) for performing mathematical computations (*evaluate mathematical expressions*) in view of RS6000's teaching that each node of the IBM SP2 has disk drives and memory cards for storing program code accessible by that node, renders obvious this limitation. Ex.1003, ¶¶147-158, 344

[26.4.0] ***wherein the plurality of nodes comprises:***

Limitation [26.5.0] is rendered obvious for the same reasons presented above for [26.1.0]. Ex.1003, ¶¶342.

[26.4.1] ***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,***

First, as shown in limitation [26.1.0], the combination of Menon and RS6000 renders obvious *a node comprising a hardware processor*. Further, as shown in limitation [26.3], the combination of Menon and RS6000 renders obvious *a node configured to access* Menon's address space, which refers to a computer's memory, comprising *program code for a single-node kernel*, where the memory can be in the form of memory cards having up to 64 GB memory for storing program code accessible by that processor. Thus, Menon and RS6000 render obvious *a first node comprising a first hardware processor configured to access a first memory comprising … program code for a single-node kernel*.

Second, Menon describes one of the MATLAB processes as an interactive MATLAB process with which a user interacts directly. Ex.1005, 3, 6. "For each of *p* MATLAB processes, POE assigns an identification number between 0 and *p* – 1. The process numbered 0 is considered the interactive MATLAB." Ex.1005, 6.

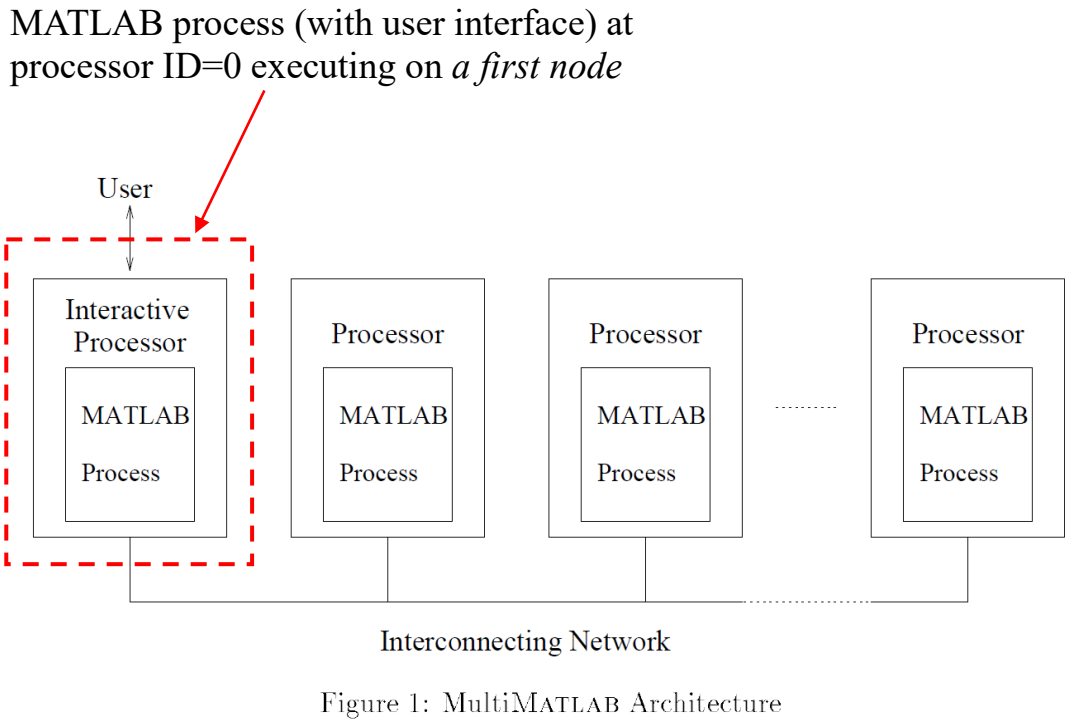


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated)

Menon discloses "MATLAB's Partial Differential Equation Toolkit," which "provides an intuitive graphical interface allowing users to specify geometries, differential equations, and boundary conditions." Ex.1005, 6.

Accordingly, Menon and RS6000 describe a node having (i) a 64-bit *processor* that accesses and executes *program code* stored in *memory*, and (ii) the

MATLAB code stored in the address space of, and accessed by, the node includes a toolkit with a graphical interface to allow users to provide equations and directly interact with the interactive processor node (*user interface*), rendering obvious this limitation. Ex.1003, ¶¶171-176, 346.

[26.4.2] *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*

As shown in limitation [26.3], MATLAB teaches *a first single-node kernel* and is configured so that the MEX routines "may be run directly from [the] MATLAB" environment. Ex.1005, Abstract, 3. Menon teaches one example of a user providing user instructions by using "an Eval routine for parallel evaluation" on the interactive MATLAB process. Ex.1005, 4. "[T]he Eval routine is implemented as a MEX Routine which accepts a vector of MATLAB process IDs and a MATLAB command. This routine sends the commands over the underlying communication layer to the processes corresponding to the specified IDs." Ex.1005, 4.

Trefethen also describes example *user instructions* that are interpreted in order to send different actions to different MATLAB processes. Ex.1006, 3-7. For example, Trefethen discloses a command with data, "Eval([4 5] , 'cond(hilb(ID))')" (*user instructions*). In accordance with that instruction, the calls

"cond(hilb(ID))" are then distributed to the MATLAB processes on processor ID=4 and ID=5 for each to perform that call. Ex.1006, 4; Ex.1003, ¶179.

As another example, Trefethen discloses the m-file "cycle.m." Ex.1006, 6. In this example, the user instructions instruct: (i) the interactive MATLAB process (ID=0) to create a variable *a* with a value of 1 and send the variable *a* to the next MATLAB process (ID+1), (ii) each middle MATLAB process to receive the variable *a* from the immediately preceding MATLAB process, double the value of the received variable *a*, and send the result to the next MATLAB process, and (iii) the last MATLAB process to receive the variable *a* and double its value. Ex.1006, 6. The *user instructions* cycle.m and the final output from a MultiMATLAB architecture that has six MATLAB processes are shown below:

user instructions

Instruction for MATLAB process ID=0 to create a=1 and send "a" to MATLAB process ID=1

```
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;
```

Instruction for the last MATLAB process to receive the variable a and double it

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

Instruction for a middle MATLAB process ID to receive the variable "a," double it, and send the doubled variable to the next MATLAB process ID=ID+1

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

Output of final result (a=32)

Ex.1006, 6 (annotated); Ex.1003, ¶181.

Accordingly, Menon's teaching that MATLAB receives user commands, interprets such commands into actions that MATLAB and MEX Routines perform and then causes calls to be sent to the MATLAB processes on other nodes in accordance with the IDs specified by the user instructions, in view of Trefethen's teaching that MATLAB interprets the commands in order to assign different actions to different processes, renders obvious this limitation. Ex.1003, ¶¶177-182, 347.

[26.5.1] ***a second node comprising a second hardware processor with a plurality of processing cores,***

As shown in limitation [26.1.0], Menon discloses the MultiMATLAB architecture executing a series of MATLAB processes on 32 processors on an IBM SP2, and RS6000 discloses that the IBM SP2 is a cluster processing system having up to 128 or more processor nodes, where each processor node has up to sixteen 64-bit processors. Further, as shown in [26.1.0], each processor node of the IBM SP2 has hard drives and up to 64 GB memory for storing program code accessible by the processor nodes. A non-interactive processor node of the MultiMATLAB architecture, i.e., a processor node other than the interactive processor (*first node*) of Figure 1 discussed in limitation [1.3.1] (e.g., processor ID=1), teaches *a second node comprising a second hardware processor*.

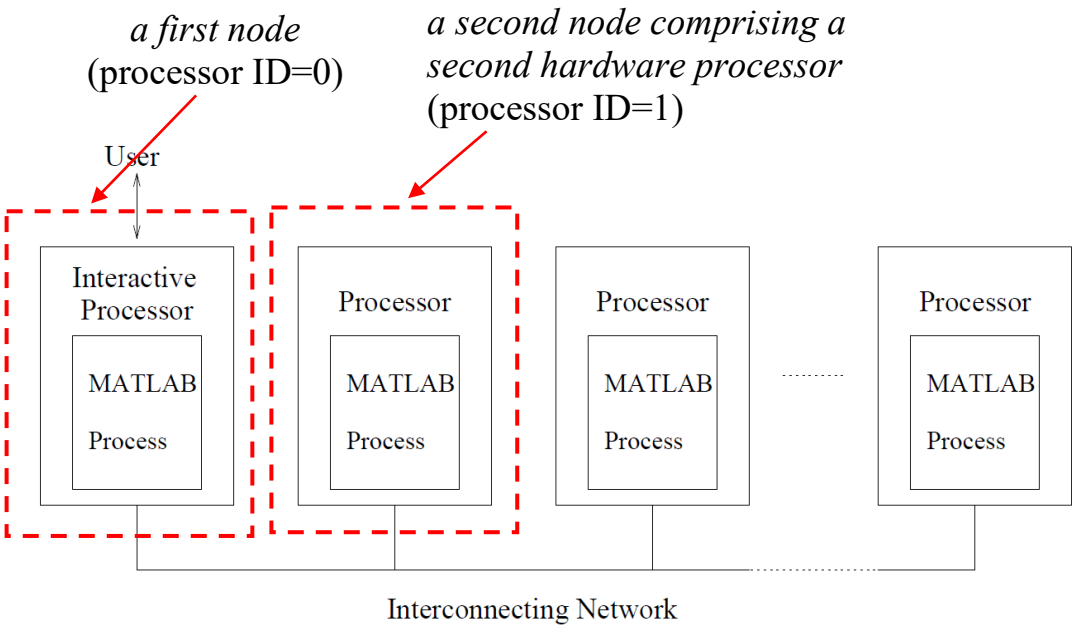


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶184

RS6000 also teaches that each of the nodes has a "Symmetric MultiProcessor (SMP)" ("*hardware processor*") and that the SMP includes "four, eight, twelve, or sixteen 375 MHz 630FP 64-bit processors." Ex.1007, 2, 9. Thus, each node has *hardware processor with a plurality of processing cores*.

Accordingly, Menon discloses executing a series of parallel processes on the 32 processors of the IBM SP2 system and RS6000 teaches that each node of the IBM SP2 has a symmetric multiprocessor having up to sixteen processors (each having a core), rendering obvious this limitation. Ex.1003, ¶¶183-186, 348.

[26.5.2] ***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;***

As shown in [26.2], Menon teaches MEX routines having code that implement standard variants of MPI calls to enable SPMD point-to-point communication so that the MATLAB processes can communicate tasks and data with each other. Ex.1005, 9. Each MATLAB process is assigned an "identification number" and "wait[s] for commands from the interactive MATLAB process." Ex.1005, 6.

As shown in [26.4.2], Trefethen discloses that MultiMATLAB can be configured with user instructions that instruct: (i) the interactive MATLAB process to create variable *a* with a value and send the variable to the next MATLAB process, and (ii) the middle MATLAB processes to receive the variable *a* from the previous MATLAB process, double the value of the received variable *a*, and send the result to the next MATLAB process. Ex.1006, 6. That is, Trefethen teaches that the MATLAB process at processor ID=1 (*the second node*), receives variable a (*is configured to receive calls*) from the processor ID=0 (*from the first node*), uses its processor to double the value of received variable *a* (*execute at least a first mathematical expression evaluation*), and send the result of the mathematical operation (*and communicate a result of the first mathematical expression*

evaluation) to the next MATLAB process at the next processor, processor ID=2 (*to a third node*), as shown below:

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

processor ID=0 (*first node*) sends a=1 (*calls*) to processor ID=1 (*second node*)

processor ID=Nproc-1 (last *node*) receives a (*receives calls*) and doubles it (*execute at least a first mathematical expression evaluation*)

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
```

processor ID=ID+1 (*second node*) sends *(calls)* a=2 (*a result of the first mathematical expression evaluation)* to processor ID=2 (*third node*)

Output of third node (a=4)

Ex.1006, 3-4 (annotated); Ex.1003, ¶190.

Accordingly, Menon's teaching of a point-to-point system for passing messages among processors with identification numbers, in view of Trefethen's disclosure of example instructions that use point-to-point message passing to send calls from a first processor that are received by a second processor, where the second processor then sends a mathematical result to a third processor, renders obvious this limitation. Ex.1003, ¶¶187-191, 349.

[26.6.1] ***wherein the third node comprises a third hardware processor with a plurality of processing cores,***

Menon describes a third processor in its MultiMATLAB architecture running on the IBM SP2, as shown in Figure 1:

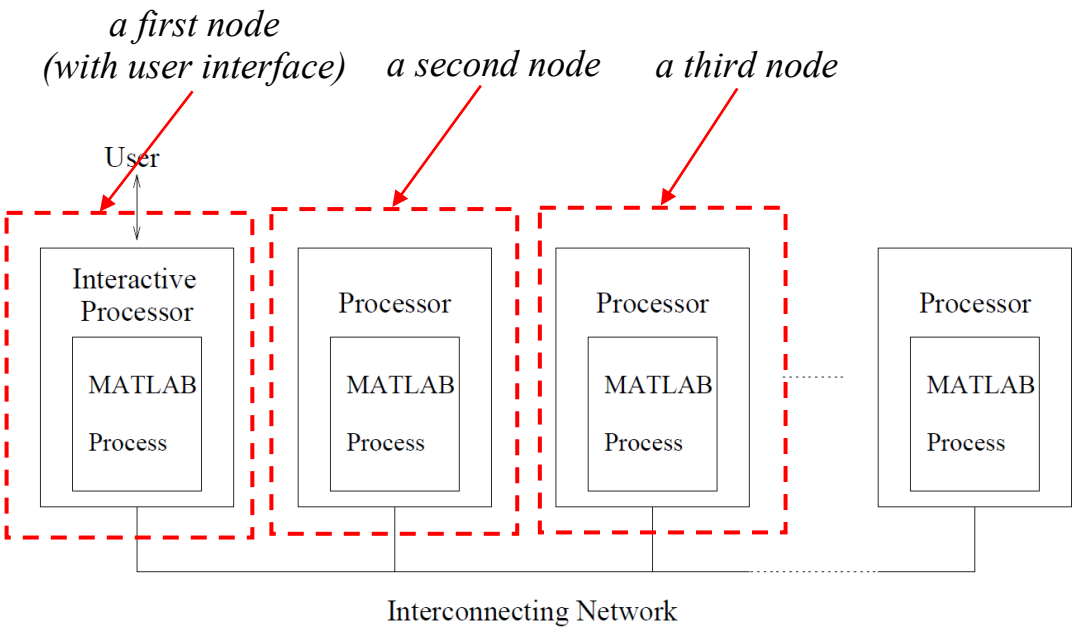


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶193

As shown in limitation [26.5.1], each processor node of the IBM SP2 has hard drives, memory for storing program code accessible by the processor nodes, and a hardware processor having up to sixteen processor cores. Ex.1007, 9. Ex.1003, ¶¶192-193, 350.

[26.6.2] ***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;

As shown in limitation [26.5.2], Trefethen teaches the MATLAB process at processor ID=2 (*a third node*) receives (*is configured to receive*) a variable that was doubled by the previous MATLAB process at processor ID=1 (*the result of the first mathematical expression evaluation*) and sent by processor ID=1 (*from the second node*).

Trefethen also teaches that the MATLAB process at processor ID=2 uses its processor to double the received variable, as shown below:

```
if ID==0                    % fir
   a = 1
   Send(ID+1,a)
elseif ID == Nproc-1        % las
   a = 2*Recv
else                        % mid
   a = 2*Recv
   Send(ID+1,a)
end;
```

processor ID=2 (*third node*) receives the variable a doubled by processor ID=1 (*a result of the first mathematical expression evaluation*) from processor ID=1 (*second node*) and doubles it (*execute at least a second mathematical expression evaluation using the received result*)

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
```

Output of third node (a=4)

Output of final result (a=32)

Ex.1006, 6 (annotated); Ex.1003, ¶196.

Trefethen further discloses that each MATLAB process transmits its output "to the master process as soon as it is ready."[2] Ex.1006, 4. A POSITA would recognize that Trefethen's "master process" as Menon's interactive MATLAB process that provides the output to the user. Ex.1003, ¶197.

Further, Menon describes an iterative algorithm using matrix-vector multiplication. Ex.1005, 11, Figure 4. Menon teaches that, during the execution of this example algorithm, "the matrix *A* and the different vectors are each distributed by row over all processes as in Figure 6," shown below:

[2] While Trefethen uses the term "master" process, the cycle.m example uses Send() and Recv(), which are the peer-to-peer paradigm and not the master/slave paradigm. Here, the "master" process is the process with which the user is directly interacting. Ex.1003, ¶197.

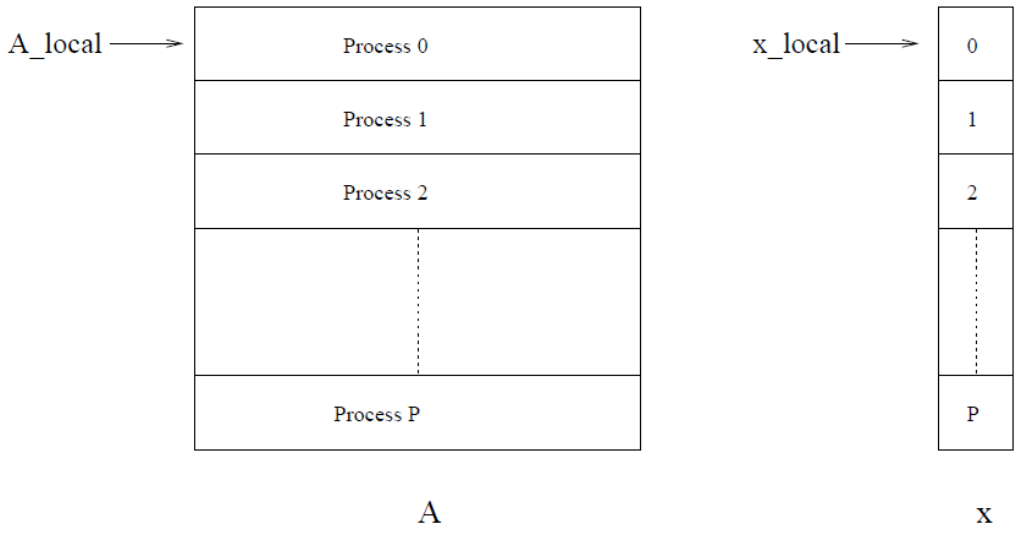


Figure 6: Matrix A and vector x distributed by row over P processors

Ex.1005, FIG. 6

Accordingly, Menon's teaching of a point-to-point computing system for passing messages among processors with identification numbers in view of Trefethen's disclosure of (i) a second processor passing mathematical results to a third processor; and (ii) the third processor performing a second calculation and sending the output, when ready, to the interactive MATLAB process (*first node*), renders obvious this limitation. Ex.1003, ¶¶194-201, 351.

[26.7] ***wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;***

Menon discloses that "[t]he user interacts directly with one MATLAB process, called the interactive process, and operates within that process's MATLAB environment." Ex.1005, 3. Trefethen discloses an example scenario

where plots are produced "in a distributed fashion that are then sent to the user's screen," and notes that "[t]his can be particularly useful when one wishes to monitor the progress of computations on several processors graphically." Ex.1006, 7. In particular, Trefethen teaches "set[ting] up a MATLAB figure window in each process and arrang[ing] them in a grid on the screen. This is easily done using standard MATLAB handle graphics commands." Ex.1006, 7. In other words, "the progress of computations on several processors"—such as the results of computations performed by the processors of the MultiMATLAB architecture and returned to the interactive processor—are presented to the user at the user's screen. Ex.1006, 7.

As shown in limitation [26.6.2], as soon as the MATLAB process at processor ID=2 completes the operation of doubling the variable that it has received from processor ID=1, it sends the result to the master process (interactive processor) to provide the result to the user.

Therefore, Menon and Trefethen render it obvious that the interactive processor (*the first node*) displays (*is configured to return*) the received variable (*the result of the second mathematical expression evaluation*) on the user's screen (*to the user interface*). Ex.1003, ¶¶202-204, 352.

[26.8.0] ***wherein one or more of the nodes are configured to:***

Limitation [26.8.0] is rendered obvious for the same reasons presented above for [26.1.0]. Ex.1003, ¶353.

[26.8.1] ***accept user instructions;***

As shown in [26.4.1], Menon discloses the interactive processor of the MultiMATLAB architecture has a MATLAB process with MATLAB that provides a toolkit with a graphical interface to allow users to provide equations. Figure 1 of Menon below shows the MATLAB process at the interactive processor of the MultiMATLAB architecture.

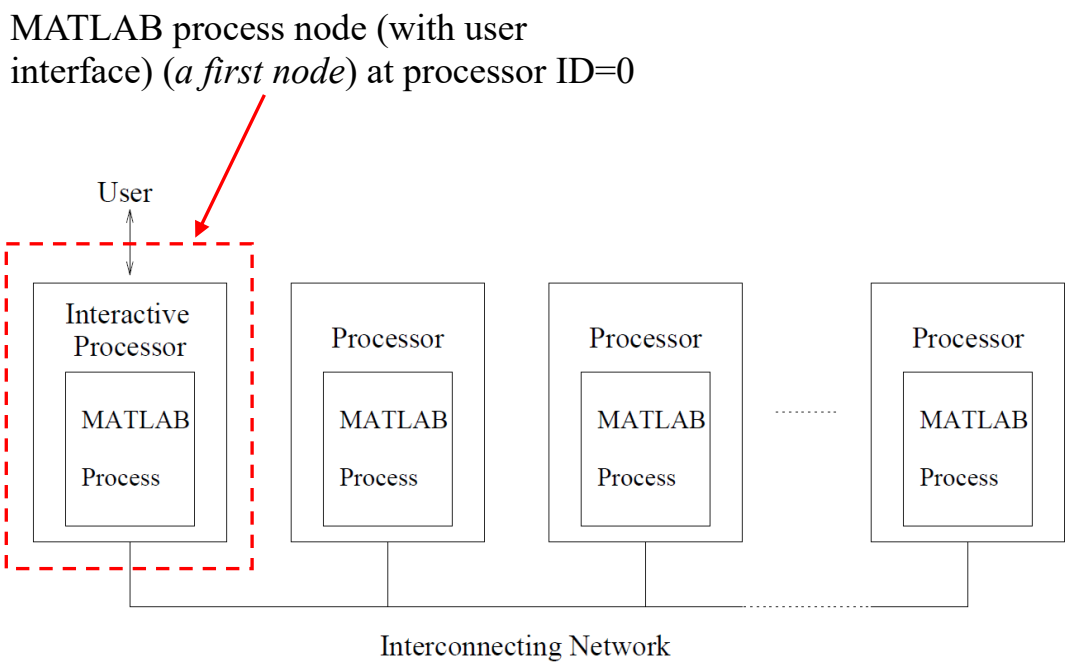


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated)

Menon explains that users can provide the graphical interface "geometries, differential equations, and boundary conditions." Ex.1005, 6. Further, as discussed in [26.4.2], a user can also provide *user instructions* via an "m-file." Trefethen teaches that m-files can contain "dozens of high-level commands such as svd (singular value decomposition), fft (fast Fourier transform), and roots (polynomial zerofinding)." Ex.1006, 2. Trefethen explains that "for most applications, … a user will want to execute a program (an m-file) rather than a single line of text. A command such as Eval('filename') achieves this effect." Ex.1006, 5.

Accordingly, Menon's disclosure of an interactive processor (*one or more of the nodes*) that includes a user interface for receiving *user instructions*, in view of Trefethen's teaching of the interactive MATLAB process receiving instructions input via an m-file into MATLAB, renders obvious this limitation. Ex.1003, ¶¶206-210, 354.

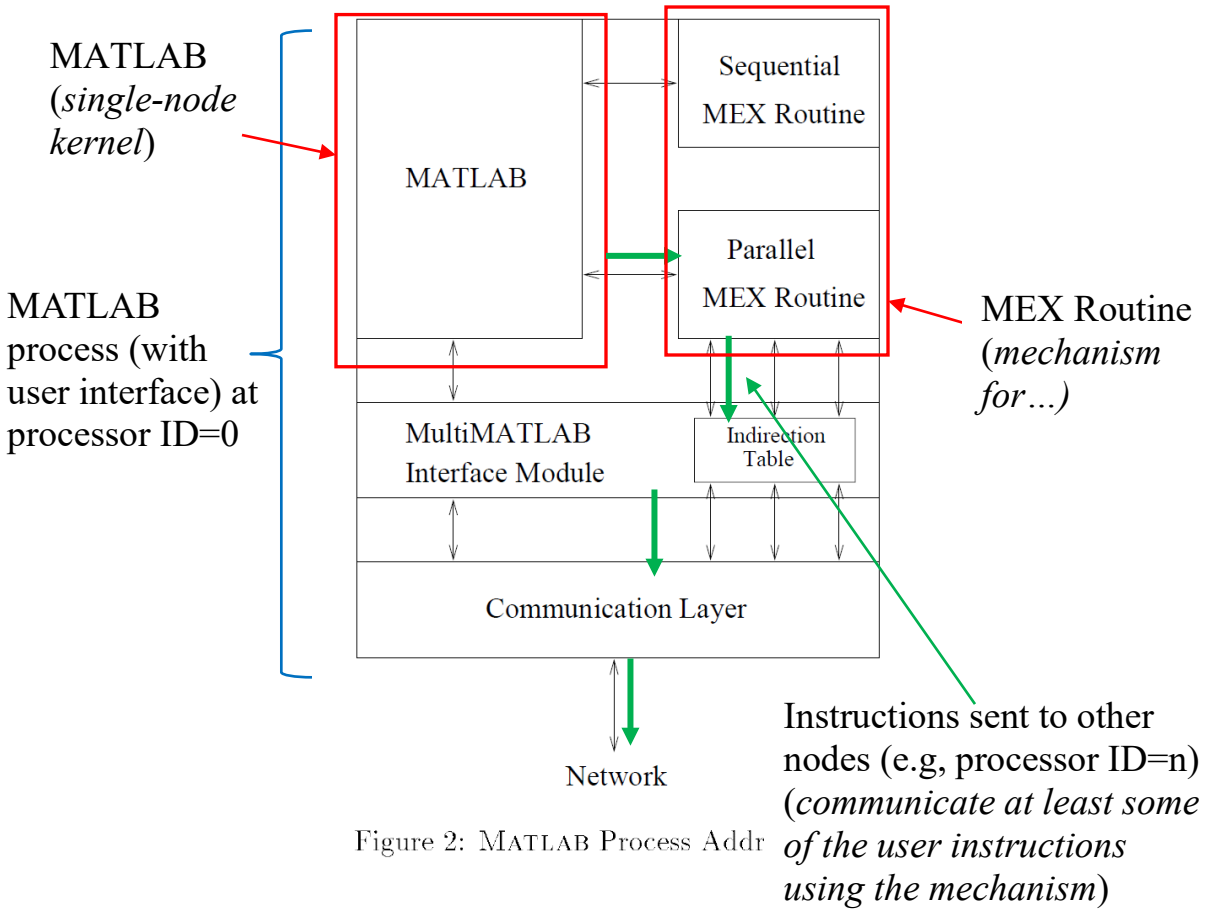
[26.8.2] *after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and*

First, as discussed in [26.4.2] and [26.8.1], *user instructions* can be provided via m-files that specify different actions for different MATLAB processes.

Second, as shown in [26.2], Menon renders obvious a *mechanism for…* because it teaches MEX routines having code that implement standard variants of

MPI calls to enable SPMD point-to-point communication so that the MATLAB processes can communicate tasks and data with each other.

Third, Menon describes how the MATLAB process at the interactive processor (e.g., ID=0) sends commands to the other MATLAB processes, stating that the Eval routine "is implemented as a MEX routine which accepts a vector of MATLAB process IDs and a MATLAB command. This routine sends the commands over the underlying communication layer to the processes corresponding to the specified IDs." Ex.1005, 4.



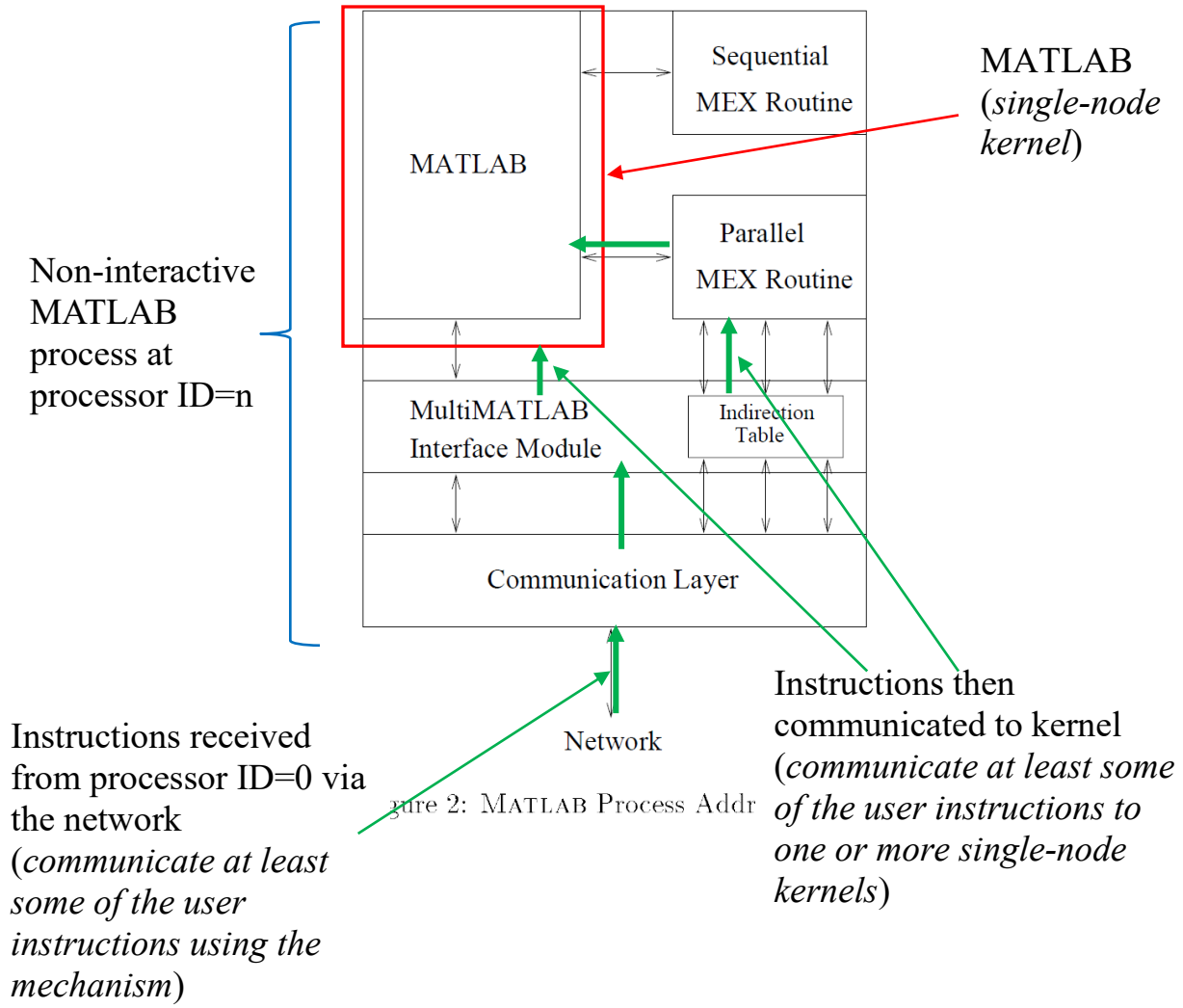
Ex.1005, FIG. 2 (annotated); Ex.1003, ¶214

Accordingly, Menon in view of Trefethen teaches that (i) the MATLAB process at the interactive processor receives the user instructions including IDs identifying MATLAB processes in the MultiMATLAB architecture and (ii) those instructions are communicated over the communication layer to the other MATLAB processes corresponding to the IDs specified in the user instructions, thereby rendering obvious this limitation. Ex.1003, ¶¶211-215, 355.

[26.8.3] ***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.***

First, as shown in [26.2], Menon renders obvious a *mechanism for*… because it teaches MEX routines having code that implement standard variants of MPI calls to enable SPMD point-to-point communication so that the MATLAB processes can communicate tasks and data with each other.

Second, Trefethen provides example illustrations showing that commands sent to the other or non-interactive MATLAB processes corresponding to the IDs specified in the *user instructions* are processed by MATLAB at the respective MATLAB processes identified by the IDs. Ex.1006, 3-7. For example, as discussed in [1.3.2], when the user provides the *user instructions*, such as Eval([4 5] , 'cond(hilb(ID))'), to the interactive MATLAB process, the MATLAB processes on processor ID=4 and ID=5 receive their respective commands and use their respective MATLAB instances to execute the commands. Ex.1006, 4. Figure 2 of Menon also shows MATLAB and MEX Routines (*one or more single-node kernels*) of a non-interactive MATLAB process which execute received commands.



Ex.1005, FIG. 2 (annotated); Ex.1003, ¶218

Accordingly, Menon and Trefethen teach that commands are communicated using the MEX routines from the interactive processor (ID=0), and once received by each non-interactive node (ID=n), are communicated to MATLAB (*single-node kernel*) for processing, thereby rendering obvious this limitation. Ex.1003, ¶¶216-219, 356.

Claim 29

[29.0] ***A computer cluster comprising:***

Limitation [29.0] is rendered obvious for the same reasons presented above for [26.0]. Ex.1003, ¶357.

[29.1.1] ***a plurality of nodes,***

Limitation [29.1.1] is rendered obvious for the same reasons presented above for [26.1.0]. Ex.1003, ¶358.

[29.1.2] ***wherein one or more of the nodes are configured to receive: a command to start a cluster initialization process for the computer cluster,***

Limitation [29.1.2] is rendered obvious for the same reasons presented above for [26.1.1]. Ex.1003, ¶¶359.

[29.1.3] ***wherein the cluster initialization process comprises establishing communication among two or more of the nodes; and***

Limitation [29.1.3] is rendered obvious for the same reasons presented above for [26.1.2]. Ex.1003, ¶¶360.

[29.1.4] ***[wherein one or more of the nodes are configured to receive:] an instruction from a user interface or a script; and***

Limitation [29.1.4] is rendered obvious for the same reasons presented above for [26.1.3]. Ex.1003, ¶¶361.

[29.2] ***a mechanism for the nodes to communicate results of mathematical expression evaluation with each other;***

Limitation [29.2] is rendered obvious for the same reasons presented above for [26.2]. Ex.1003, ¶¶362.

[29.3] ***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;***

Limitation [29.3] is rendered obvious for the same reasons presented above for [26.3]. Ex.1003, ¶¶363.

[29.4.0] ***wherein the plurality of nodes comprises:***

Limitation [29.4.0] is rendered obvious for the same reasons presented above for [26.4.0]. Ex.1003, ¶¶364.

[29.4.1] ***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,***

Limitation [29.4.1] is rendered obvious for the same reasons presented above for [26.4.1]. Ex.1003, ¶¶365.

[29.4.2] ***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***

Limitation [29.4.2] is rendered obvious for the same reasons presented above for [26.4.2]. Ex.1003, ¶¶366.

[29.5.1] ***a second node comprising a second hardware processor with a plurality of processing cores,***

Limitation [29.5.1] is rendered obvious for the same reasons presented above for [26.5.1]. Ex.1003, ¶¶367.

[29.5.2] ***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;***

Limitation [29.5.2] is rendered obvious for the same reasons presented above for [26.5.2]. Ex.1003, ¶¶368.

[29.6.1] ***wherein the third node comprises a third hardware processor with a plurality of processing cores,***

Limitation [29.6.1] is rendered obvious for the same reasons presented above for [26.6.1]. Ex.1003, ¶¶369.

[29.6.2] ***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***

Limitation [29.6.2] is rendered obvious for the same reasons presented above for [26.6.2]. Ex.1003, ¶¶370.

[29.7] ***wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface or the script;***

Limitation [29.7] is rendered obvious for the same reasons presented above for [26.7]. Ex.1003, ¶¶371.

[29.8.0] ***wherein one or more of the nodes are configured to:***

Limitation [29.8.0] is rendered obvious for the same reasons presented above for [29.1.0]. Ex.1003, ¶¶372.

[29.8.1] ***accept user instructions;***

Limitation [29.8.1] is rendered obvious for the same reasons presented above for [26.8.1]. Ex.1003, ¶¶373.

[29.6.2] ***after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and***

Limitation [29.8.2] is rendered obvious for the same reasons presented above for [26.8.2]. Ex.1003, ¶¶374.

[29.6.3] ***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.***

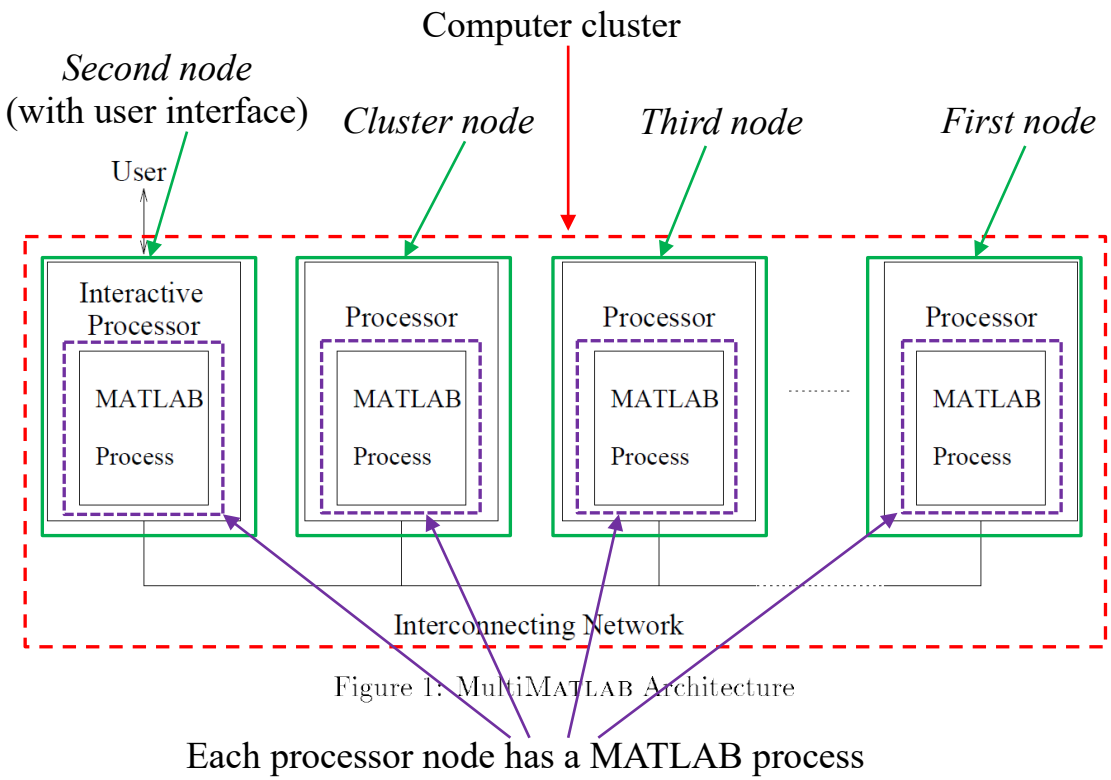
Limitation [29.8.3] is rendered obvious for the same reasons presented above for [26.8.3]. Ex.1003, ¶¶375.

Claim 35

[35.0] ***A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:***

As discussed in [26.0], Menon discloses a computer cluster having a plurality of nodes within the computer cluster and that "each processor in a parallel platform individually runs a MATLAB process." Ex.1005, 3.[3]

[3] The functionality recited for the "first," "second," and "third" nodes of claim 35 is different than the functionality recited for the similarly named "first," "second," and "third" nodes of claim 1.



Ex.1005, FIG. 2 (annotated); Ex.1003, ¶384.

Menon provides an example where the nodes compute the conjugate gradients algorithm in parallel. Ex.1005, 11-13; FIG. 4.

Accordingly, Menon's individual processor nodes of the MultiMATLAB architecture that evaluate mathematic expressions in parallel as implemented on an IBM SP2 multiprocessor system, in view of RS6000 describing 128 nodes for parallel processing, renders obvious this limitation. Ex.1003, ¶¶382-386.

[35.1.1] ***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,

First, as discussed in [26.1.3] and [26.3], Menon teaches that each MATLAB process runs on *a hardware processor* that (i) accesses memory having the program for the MATLAB process and (ii) executes that the program to evaluate mathematical expressions.

Second, Trefethen discloses that the MultiMATLAB architecture is capable of "produc[ing] plots in a distributed fashion that are then sent to the user's screen. This can be particularly useful when one wishes to monitor the progress of computations on several processors graphically." Ex.1006, 7.

Accordingly, IBM SP2's multiprocessor implementation having a node with a processor that accesses hard drives and RAM in view of Menon teaching that each node of the IBM SP2 executes a MATLAB process for evaluating mathematical expressions input by a user via the interface processor, further in view of Trefethen's teaching that MultiMATLAB displays the results of the mathematical computations, renders obvious this limitation. Ex.1003, ¶¶387-402.

[35.1.2] ***wherein the hardware processor comprises multiple processor cores;***

Limitation [35.1.2] is rendered obvious for the same reasons presented above for [26.5.1]. Ex.1003, ¶¶403.

[35.2] ***a user connection interface configured to receive a command to start a cluster initialization process for a computer cluster;***

First, as discussed in [26.1.1], Menon discloses initializing the cluster using IBM's POE.

Second, Menon further discloses that each processor of "the MultiMATLAB architecture … individually runs a MATLAB process" having a set of components to provide interconnection with other modules, including the "MEX Routines," "MultiMATLAB interface module," an "indirection table" and "communication layer" (collectively, *a user connection interface*). Ex.1005, 3, 5. Menon describes that the MultiMATLAB interface module on each processor is responsible for the underlying communication layer with other processors. Ex.1005, 3.

Accordingly, Menon teaches an interface module that communicates the initialization command from the user to the other processors of the MultiMATLAB architecture, rendering obvious this limitation. Ex.1003, ¶¶404-408.

[35.3] ***a mechanism to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and***

First, as shown in [26.2] and [26.6.2], Menon renders obvious *a mechanism to communicate results of evaluation with other computer cluster nodes* because it teaches MEX routines having code that implement standard variants of MPI calls to enable SPMD point-to-point communication so that the MATLAB processes can communicate tasks and data with each other.

Additionally, Menon's MEX routines are implemented using a "point to point … communication" paradigm. Ex.1005, 9. Unlike a "master/slave paradigm," which "is often adequate in situations where little or no communication is required," Menon's SPMD/Message Passing paradigm "accommodate[s] applications requiring a finer degree of communication" and provides "point to point … communication." Ex.1005, 9. Table 1 of Menon lists example point-to-point commands for the SPMD communication paradigm, including "Send(pid,data)," which "send[s] data from one process to another," and "Recv(pid)," which "receive[s] data sent from another process." Ex.1005, Table 1.

MultiMATLAB commands	
Starting and stopping MultiMATLAB (MPICH only)	
Start(n)	Initializes n remote MATLAB processes.
Quit	Closes remote MATLAB processes.
Running commands on multiple processors	
Nproc	Returns number of MATLAB processes.
ID	Returns ID of calling MATLAB process.
Eval(pid,'command')	Evaluates a command on one or more MATLAB process.
Master/Slave Communication	
Put(pid,dataname)	Put data from the master process onto one or more remote processes.
Get(pid,dataname)	Get data from a remote process onto the master process.
SPMD Communication	
Send(pid,data)	Send data from one process to another.
Recv(pid)	Receive data sent from another process.
Barrier	Synchronize processes.
Bcast(pid,data)	Broadcasts data from processor pid to all processes.
Sum(data)	Adds data across all processors to form a global sum.
Collect(data)	Collects local data segments into a single global one.

Table 1: Selected commands in the MultiMATLAB system

Commands for point-to-point communication

Ex.1005, Table 1 (annotated); Ex.1003, ¶411

Menon's SPMD point-to-point communication paradigm where each node can communicate tasks and data directly with each other node renders obvious *a peer-to-peer architecture*. Ex.1003, ¶¶409-419.

[35.4.1] ***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;***[4]

First, as discussed in [26.1.3] and [26.3], Menon teaches that each MATLAB process runs on a hardware processor that (i) accesses memory having the program for the MATLAB process and (ii) executes that program to evaluate mathematical expressions.

[4] In addition to the arguments presented here, this limitation describing the "second node" lacks patentable weight because it does not confer any structure on the claimed "computer cluster node," which is a separate node. See *In re Miller*, No. 2021-1599, 2022 WL 594417, at *3 (Fed. Cir. Feb. 28, 2022); *Edgewell Pers. Care Brands, LLC v. Munchkin, Inc.*, 998 F.3d 917, 920–21 (Fed. Cir. 2021) ("An apparatus claim is generally to be construed according to what the apparatus is, not what the apparatus does.").

Second, as discussed in [26.4.2], Menon describes an interactive processor node that interprets user instructions and distributes calls to the other nodes for execution.

Accordingly, Menon's (i) cluster node has program code executed by a processor, (ii) the cluster node receives calls from an interactive processor (*second node*) of the MultiMATLAB architecture, (iii) the interactive processor (second node) has a user interface, (iv) the MATLAB process on the interactive processor (*second node*) has MATLAB (*second single-node kernel*), and (v) the interactive processor (*second node*) receives user instructions, interprets them, and send commands to the other processors of the MultiMATLAB architecture for execution, rendering obvious this limitation. Ex.1003, ¶¶420-427.

[35.4.2] ***[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***

Limitation [35.4.2] is rendered obvious for the same reasons presented above for [26.5.1]-[26.5.2]. Ex.1003, ¶¶428-430.

[35.4.3.1] ***[program code that, when executed, is capable of causing the hardware processor to:] communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores,***

Limitation [35.4.3.1] is rendered obvious for the same reasons presented above for [26.5.1]-[26.5.2]. Ex.1003, ¶¶431-438.

[35.4.3.2] ***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;***

Limitation [35.4.3.2] is rendered obvious for the same reasons presented above for [26.6.1]-[26.6.2]. To the extent Patent Owner argues that the result of the second mathematical expression should be communicated to the *second node*, Trefethen discloses sending the result to the node that has the user interface (i.e., the second node as recited in claim 35). Ex.1006, 4. Trefethen also discloses that the node to which the result is to be sent can be specified in the instructions. Ex.1006, 7. Ex.1003, ¶¶439-443.

[35.5] ***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***

As discussed in [35.2], Menon discloses the MultiMATLAB interface module as the user connect interface. Menon teaches that the MultiMATLAB interface module on each processor is responsible for the underlying communication layer with other processors. Ex.1005, 3.

Further, as discussed in [26.6.2], Trefethen discloses that each MATLAB process transmits its output to the interactive processor where the user interface resides as soon as that MATLAB process's computation is completed. Ex.1006, 4.

Accordingly, Menon's "MultiMATLAB interface module," which is responsible for the underlying communication layer with other nodes (*user connection interface*), in view of Trefethen's teaching that each MATLAB process can transmit its mathematical results back to the processor with the user interface, renders obvious this limitation. Ex.1003, ¶¶444-448.

[35.6.0] ***wherein the computer cluster node is configured to:***

Limitation [35.6.0] is rendered obvious for the same reasons presented above for [35.0]. Ex.1003, ¶449.

[35.6.1] ***accept user instructions;***

Limitation [35.6.1] is rendered obvious for the same reasons presented above for [26.8.1]. Ex.1003, ¶450, 206-209.

[35.6.2] ***after accepting user instructions, communicate at least some of the user instructions using the mechanism for the nodes to communicate with each other; and***

Limitation [35.6.2] is rendered obvious for the same reasons presented above for [26.8.2]. Ex.1003, ¶¶451, 211-215.

[35.6.3] ***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.***

Limitation [35.6.3] is rendered obvious for the same reasons presented above for limitation [26.8.3]. Ex.1003, ¶¶452, 216-219.

Claim 36

[36.0] ***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.***

First, Menon teaches MATLAB in conjunction with the MEX Routines is capable of causing a processor of the MultiMATLAB architecture to evaluate mathematical expressions having the data "distributed by row over all processes." Ex.1005, 11-13.

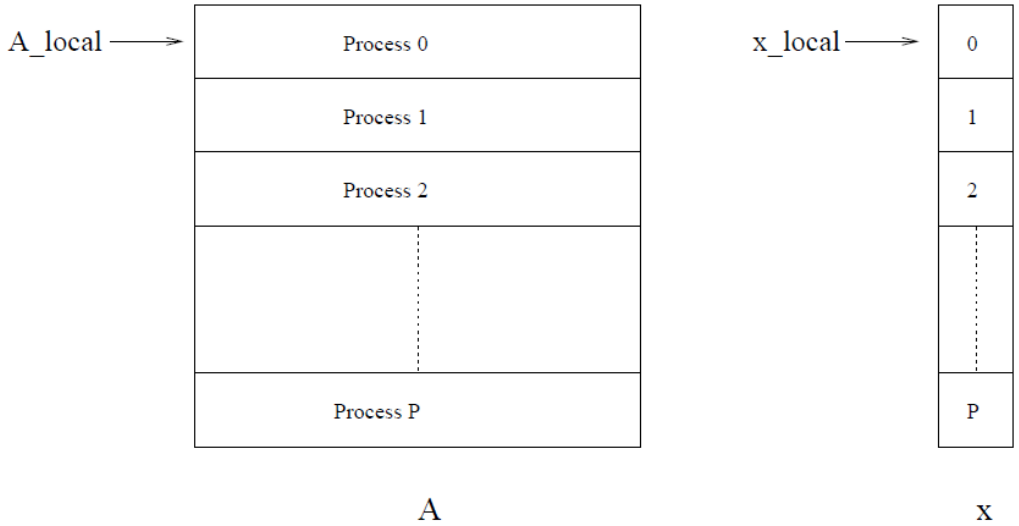


Figure 6: Matrix A and vector x distributed by row over P processors

Ex.1005, FIG. 6

Menon teaches that each MATLAB process on a processor performs "computation required for its local data." Ex.1005, 11-13. A POSITA would recognize that each MATLAB process, when computing "local data," would only

use the data on its local node and would not compute on data on another node. Ex.1003, ¶455.

Second, Trefethen discloses that MATLAB (in the MATLAB processes of the MultiMATLAB architecture) can perform "fast Fourier transform." Ex.1006, 2. Menon also teaches that the "parallel MEX routines" of MATLAB processes of the MultiMATLAB architecture "would perform matrix multiplications, solve for eigenvalues, compute Fourier transforms, and so on." Ex.1005, 10.

Accordingly, Menon's teaching that (i) data is stored in a matrix distributed across the MultiMATLAB architecture's different processors for parallel processing using MATLAB, (ii) that the MATLAB processes compute Fourier transforms, and (iii) each MATLAB process performs computation for its local data, in view of Trefethen teaching that MATLAB of the MATLAB processes can perform fast Fourier transforms, renders obvious this limitation. Ex.1003, ¶¶453-457.

Claim 39

[39.0] ***The computer cluster node of claim 35, wherein the hardware processor comprises a special purpose microprocessor.***

RS6000 discloses that each symmetric multiprocessor ("*hardware processor*") has "four, eight, twelve, or sixteen 375 MHz 630FP 64-bit processors per node." Ex.1007, 2, 9. A POSITA would recognize then that the 375 MHz

630FP 64-bit processors are an integrated circuit designed for high-speed data manipulation and render obvious a *special purpose microprocessor*. Ex. 1003, ¶459.

Accordingly, RS6000's 375 MHz 630FP 64-bit processors render obvious this limitation. Ex.1003, ¶¶458-460.

D. Ground 2: Claims 27-28 and 37-38 are obvious over Menon in view of Trefethen, RS6000, and POEref, further in view of MPIref

9. Summary of MPIref

"MPI: A Message-Passing Interface Standard" ("MPIref," Ex.1017) is a published report from the Message Passing Interface Forum "contain[ing] all the technical features proposed for the [message passing] interface." Ex.1017, ii. MPI is used as a communication layer allowing all the MATLAB processes on the MultiMATLAB architecture to communicate with each other. Ex.1005, 3.

MPIref discloses that MPI includes "point-to-point communication," and that the "basic point-to-point communication operations are **send** and **receive**." Ex.1017, iii, 15. The **send** operation MPI_SEND "specifies a **send buffer** in the sender memory from which the message data is taken." Ex.1017, 15. The send buffer "consists of **count** successive entries of the type indicated by **datatype**, starting with the entry at address **buf**." Ex.1017, 16. MPIref also discloses that that message includes a "destination" field to specify the "message destination."

Ex.1017, 18. (emphases original).

The syntax of the **send** operation MPI_SEND is shown below:

Send buffer information Message envelope

The syntax of the blocking send operation is given below.

MPI_SEND(buf, count, datatype, dest, tag, comm)

IN	buf	initial address of send buffer (choice)
IN	count	number of elements in send buffer (nonnegative 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)

Ex.1017, 16 (annotated); Ex.1003, ¶101.

MPIref discloses that processes receive a message "with the **receive** operation MPI_RECV," where "[t]he message to be received is selected according to the value of its [message envelope], and the message data is stored into the **receive buffer**." Ex.1017, 16. "The receive buffer consists of the storage containing **count** consecutive elements of the type specified by **datatype**, starting at address **buf**." Ex.1017, 16. The syntax of the **receive** operation MPI_RECV is shown below:

Recv buffer information Message envelope

The syntax of the blocking receive operation is given below.

MPI_RECV (buf, count, datatype, source, tag, comm, status)

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	status	status object (Status)

Ex.1017, 19 (annotated); (emphases original). Ex.1003, ¶102.

MPIref was made available and distributed by the University of Tennessee and Oak Ridge National Laboratory at the time of its publication in May 1994. Ex.1017, vi.

10. Combining MPIref with Menon

a) Analogous art

MPIref is analogous prior art because it pertains to the same field of endeavor as both the '768 patent and Menon, namely, the "the field of cluster computing" and is reasonably pertinent to the problem of the '768 patent, namely, executing a single program on multiple nodes Ex.1001, 1:14-15; Ex.1003, ¶462-464.

b) Motivation to combine

A POSITA would have considered and combined the teachings of MPIref for several reasons.

Menon suggests the combination by expressly citing to "the MPI communication standard." Ex.1005, 5. MPIref *is* the MPI communication standard referenced by Menon. Ex.1017, 1 ("MPI: A Message-Passing Interface Standard").

Menon discloses "two implementations of the MultiMATLAB architecture," both of which "use MPI as their underlying communication substrate." Ex.1005, 3, 5. Further, Menon describes that a POSITA would look to use the MPI standard with MultiMATLAB because of MPI's "broad popularity, its suitability to high performance parallel computing, and the virtually universal availability of vendor-optimized versions on modern parallel computers." Ex.1005, 5.

The combination MPIref with Menon would have been obvious to a POSITA because Menon and MPIref together represent a combination of known prior art elements (Menon describing an implementation of the MultiMATLAB architecture using MPI as its communication layer and MPIref describing "all the technical features" of MPI) to yield the predictable result of an implementation of the MultiMATLAB architecture utilizing various technical features of MPI disclosed by MPIref. Ex.1005, 3-10; Ex.1017, ii.

The results of the combination would have been predictable and there would have been a reasonable expectation of success at least because Menon expressly discusses using MPI, which is the subject of MPIref, to establish communication between all the MALAB processes of the MultiMATLAB architecture, and MPIref provided a "practical, portable, efficient and flexible standard for message passing." Ex.1005, 3; Ex.1017, 2. Ex.1003, ¶¶465-468.

Claim 27

[27.0] ***The computer cluster of claim 26, wherein the asynchronous calls comprise a first command to create a first packet containing:***

[27.1] ***an expression to be sent as payload; and***

[27.2] ***a target node where the expression should be sent;***

[27.3] ***wherein the first command is configured to be called from within a single-node kernel;***

[27.4] ***wherein the single-node kernel is configured to send the first packet to a local cluster node module; and***

[27.5] ***wherein the local cluster node module is configured to forward the expression to the target node.***

First, Menon discloses that each node includes a set of components to provide interconnection with other modules, namely, the "MEX Routines," the "MultiMATLAB interface module," an "indirection table" and "communication layer" (collectively, the *cluster node module*). Ex.1005, 3, 5. The "MEX routines" provide "[a]ll parallel functionality in the MultiMATLAB system." Ex.1005, 3. "The interface module, shown in Figure 2, is responsible for initializing an

underlying communication layer, such as MPI …, or any other package available on the platform, and exposing it to the rest of the system." Ex.1005, 3.

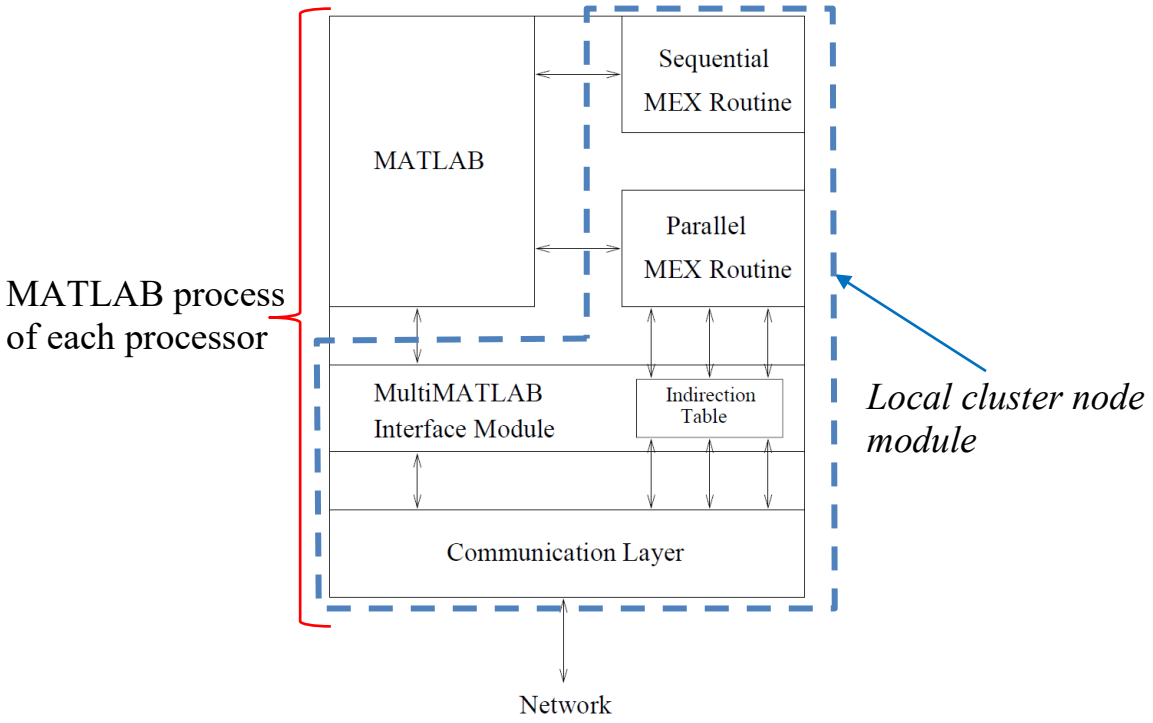
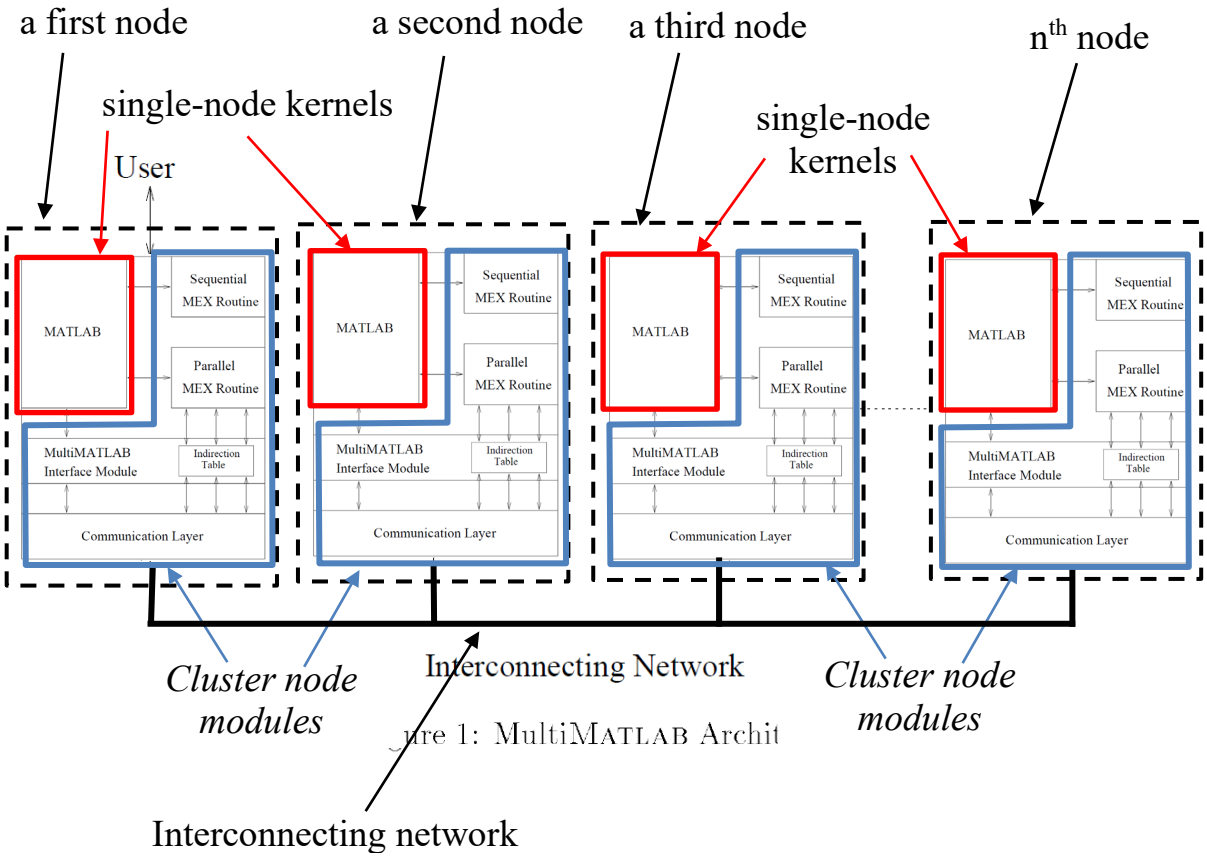


Figure 2: MATLAB Process Address Space

Ex.1005, FIG. 2 (annotated); Ex.1003, ¶470

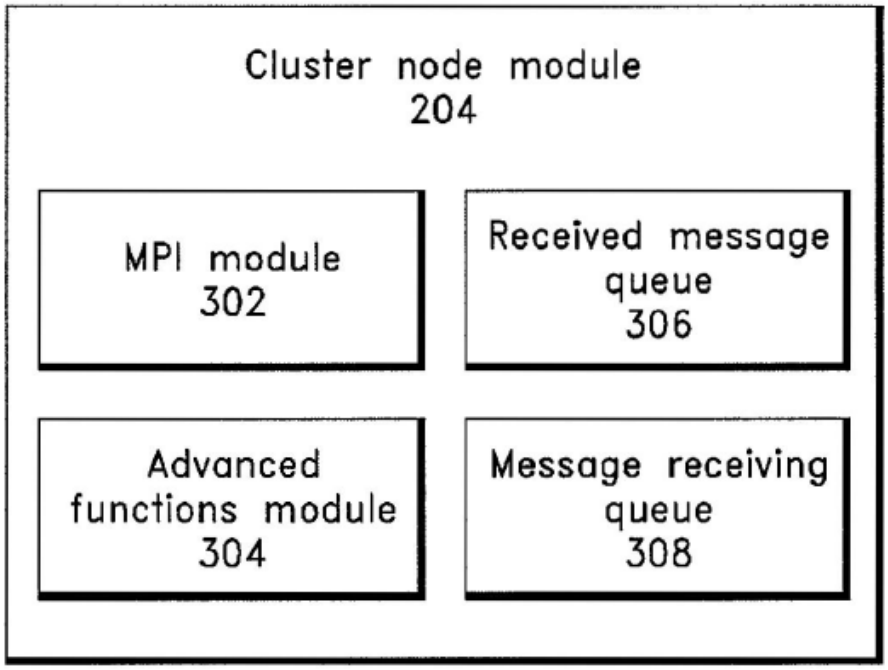
As discussed in [26.2], Menon describes the point-to-point commands in the MEX routines, such as "Send(pid,data)" that "send[s] data from one process to another," and "Recv(pid)" that "receive[s] data sent from another process." Ex.1005, Table 1. Trefethen teaches that the Send and Recv commands are "asynchronous." Ex.1006, 6.

Menon's Figure 1 and Figure 2, combined into a single figure as shown below, illustrate the relationship among the communication components (MEX routines, interface module, indirection table, and the communication layer) and the single-node kernel for each node of the MultiMATLAB architecture.



Ex.1005, FIGs. 1 and 2 (combined and annotated)

Correspondingly, the '768 patent depicts its cluster node module as a set of components to provide communication among the nodes:



Ex.1001, FIG. 3

Accordingly, Menon describes each processor of the MultiMATLAB architecture having a MATLAB process, which in turn has a set of components to provide interconnection with other modules (namely, the MEX routines, MultiMATLAB interface module, indirection table, and communication layer), rendering obvious a *local cluster node module*.

Second, Menon describes how the Send() command is called by the MATLAB process and communicates with the local cluster node module. The Send() command includes the "pid" value as the target node and the "data" value as the expression to be sent. Ex.1005, Table 1. MATLAB provides that packet of information (the pid value and data value) to the MEX routine (of the local cluster

node module). The MEX Routine decodes that packet by mapping the information to the corresponding MPI call (e.g., MPI_SEND). Ex.1005, 6. MPIref describes how the MPI_SEND call uses the data (i.e., the data value in the send call) and the identification of the destination node (i.e., the pid value in the send call) and forwards the data to the destination node. Ex.1003, ¶475.

Accordingly, Menon's description of Send command to send data from one process to another, where MATLAB (*single-node kernel*) generates a packet of information (including the expression and target node) that is sent to the MEX routines of the local cluster node module, which uses MPI to transmit the expression to the target, further in view of MPIref's description that the MPI_SEND call transmits the data to the destination node, renders obvious claim 27. Ex.1003, ¶¶469-498.

Claim 28

[28.0] ***The computer cluster of claim 27, wherein the asynchronous calls comprise a second command to create a second packet containing:***

[28.1] ***a location where the expression is expected to be received; and***

[28.2] ***a sending node from which the expression is expected to be received;***

[28.3] ***wherein the second command is configured to be called from within a single-node kernel;***

[28.4] ***wherein the single-node kernel is configured to send the second packet to a local cluster node module; and***

[28.5] ***wherein the local cluster node module is configured to store the second packet contents in a message receiving queue.***

First, as discussed in [26.2], Menon describes point-to-point commands that the MATLAB processes use to communicate with each other, such as "Send(pid,data)" that "send[s] data from one process to another," and "Recv(pid)" that "receive[s] data sent from another process." Trefethen further teaches communicating using the asynchronous Send and Recv commands. Ex.1006, 6.

Second, Menon describes how MultiMATLAB interprets the Recv() command. The Recv() command includes the "pid" value as the source of the expected data. Ex.1005, Table 1. MATLAB provides that packet of information (the pid value) to the MEX routine. The MEX routine decodes that packet, mapping the packet's information to the corresponding MPI call (e.g., MPI_RECV). MPIref discloses the details of the "MPI_RECV" call, which expects (i) the data transmitted and (ii) the identification of the source node.

Ex.1017, 15-21. MPIref teaches that a process receives a message that is transmitted by MPI_SEND call "with the receive operation MPI_RECV." Ex.1017, 16. "The message to be received is selected according to the value of its envelope, and the message data is stored into the receive buffer." Ex.1017, 16. "The receive buffer consists of the storage containing count consecutive elements of the type specified by datatype, starting at address **buf**." Ex.1017, 16. Ex.1003, ¶505.

Thus, a POSITA would recognize that executing the Recv command "Recv" (a second command) by MATLAB in a MATLAB process at processor ID+1 generates a collection or package (*second packet*) of information that the corresponding MPI call (MPI_RECV) needs for receiving the message transmitted by the MPI_SEND call from processor ID, including (i) the starting address of the receive buffer for storing the variable "a" (*expression*) that is transmitted by the processor ID-1 using the MPI call MPI_SEND, and (ii) an identification of the source processor ID-1 (*a sending node*) of the variable "a." Ex.1003, ¶509.

MPIref also describes that the MPI_RECV call "en_queues" (i.e., stores in a queue) the information received by the MPI_RECV call. Ex.1017, 164.

Accordingly, Menon's description of Send and Recv commands and Trefethen's description of those calls as asynchronous, in view of Menon describing that the execution of the command "Recv" (*a second command*) by

MATLAB in a MATLAB process (*single-node kernel*) of the MultiMATLAB architecture's processor ID would generate a collection or package (*second packet*) of information that is needed by the corresponding MPI call (MPI_RECV), as described by MPIref, including (i) the receive buffer (*a location where … is expected to be received*) for storing the variable "a" (*the expression*) and (ii) a source processor ID-1 (*a sending node from which … is expected to be received*) of the variable "a" (*the expression*), and (iii) storing the information received by the MPI_RECV call in a queue, renders obvious claim 28. Ex.1003, ¶¶499-531.

Claim 37

[37.0] ***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.***

First, as shown in [26.2], Menon teaches MEX routines having code that implement standard variants of MPI calls to enable SPMD point-to-point communication so that the MATLAB processes can communicate tasks and data with each other.

Second, Menon provides an example "parallel MATLAB implementation" where each MATLAB process performs its own computations while communicating values that may be needed for the computations with other MATLAB processes that are simultaneously processing its own calculations. Ex.1005, 11-13 (describing the operation of the conjugate gradients algorithm).

Third, Trefethen teaches that MultiMATLAB's commands are "evaluated simultaneously on all processes." Ex.1006, 6. Trefethen provides an example where the cluster is configured to exchange information between nodes during the course of parallel computation:

Instruction for MATLAB process ID=0 to create a=1 and send "a" to MATLAB process ID=1

Instruction for the last MATLAB process to receive the variable a and double it

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

Instruction for a middle MATLAB process ID to receive the variable "a," double it, and send the doubled variable to the next MATLAB process ID=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
```

SPMD point to point communication among processors of the MultiMATLAB architecture

user instructions

Ex.1006, 6 (annotated); Ex.1003, ¶537.

Accordingly, Menon teaching that the processors of MultiMATLAB have a set of components for communication with the other nodes (the cluster node module) that includes (i) MEX routines that enable intercommunication among the processors and (ii) a communication layer that uses MPI as the communication

substrate, further in in view of Trefethen teaching that MultiMATLAB's commands are executed simultaneously (*parallel*) and the nodes are configured to exchange data during the parallel computation, renders obvious this limitation. Ex.1003, ¶594, 532-539.

Claim 38

[38.0] ***The computer cluster node of claim 35, wherein the computer cluster node comprises instructions executable by the hardware processor and configured to implement asynchronous behavior, wherein the instructions comprise:***

[38.1] ***a first instruction to asynchronously send a payload to another node;***

[38.2] ***a second instruction to asynchronously receive a payload from another node; and***

[38.3] ***a third instruction to search for a payload matching a message specifier.***

First, as discussed in [26.3], the MATLAB process, MATLAB, and the MEX Routines are computer instructions executable by a hardware processor.

Second, as discussed in [26.2], the MATLAB processes described in Menon and Trefethen use asynchronous SPMD messaging commands, such as Send() and Recv() to communicate with each other.

Third, Menon describes how a cluster node module interprets message passing interface (MPI) calls for MEX routines using the "indirection table," which maps "MPI calls directly to the appropriate table offset." Ex.1005, 6.

A POSITA would recognize that the software on the nodes includes instructions for interpreting the Send command "Send (ID+1,a)" to generate a

collection of information that is used by the MEX routine to map to a corresponding MPI call (MPI_SEND). In this example, the MEX Routine for the asynchronous Send command causes the variable "a" (*payload*) to be transmitted to process ID+1 (*asynchronously send to another node*). Ex.1003, ¶546.

A POSITA would recognize that the software on the nodes includes instructions for interpreting the Recv command in a MATLAB process at processor ID+1 to receive a message. In this example, the MEX Routine for the asynchronous Recv command receives the variable "a" (*payload*) transmitted from another processor. Ex.1003, ¶505.

Trefethen teaches using a message specifier to search for matching expressions that have been received by using an "argument" to "specify a message tag so as to ensure that sends and receives are properly matched and to aid in error checking." Ex.1006, 6. Trefethen also describes a "Probe" command that returns "1 (true) if a message has arrived from the indicated source, otherwise (0) false." Ex.1006, 6.

Accordingly, Menon's teaching that the nodes include software instructions to perform message passing of mathematical results to the other processors, in view of Trefethen's teaching that a MATLAB process (i) uses an asynchronous send call to send a payload that (ii) is received using an asynchronous receive call, and (iii) uses an argument to specify a message to ensure that a payload is properly

sent and received, renders obvious claim 38. Ex.1003, ¶¶595-598, 582-593, 469-498, 499-531, 540-580.

XI. DISCRETIONARY DENIAL IS INAPPROPRIATE

Discretionary denial is not appropriate because each of *Fintiv*, *General Plastic* and *Advanced Bionics* favor institution and, as will be addressed in the discretionary briefing, the relevant considerations strongly favor institution, including the compelling economic and national security interests associated with Intel's production of semiconductors used in artificial intelligence. *See, e.g.,* Ex.1022.

A. *Fintiv*

The six factors for § 314 strongly favor institution given Petitioner's prompt filing, along with the precise teachings of the prior art. *See Apple Inc. v. Fintiv, Inc*., IPR2020-00019, Paper 11 (PTAB Mar. 20, 2020).

Further, to address any *Fintiv* concerns, Petitioner has **already filed** the following stipulation at the district court:

> *Sotera Stipulation:* Petitioner hereby stipulates that, if this proceeding is instituted, it will not pursue in the parallel district court case, any grounds for invalidating the '768 patent that was raised or could have reasonably been raised in this IPR. Ex.1042.

1. Motion to stay

No motion to stay has been filed, so the Board should not infer the outcome of such a motion. *Sand Revolution II, LLC v. Continental Intermodal Group – Trucking LLC*, IPR2019-01393, Paper 24 at 7 (PTAB 2020). Thus, this factor is neutral.

2. Parallel proceeding trial date

Trial is currently scheduled for November 16, 2026 (Ex.1020, 6), which would occur *after* a Final Written Decision. Further, scheduled trial dates often change and the current average time-to-trial for the district court hearing the co-pending litigation is 33.9 months, suggesting an expected trial date in July, 2027, well-after the Final Written Decision. Ex.1021, 37.

Thus, this factor weighs against discretionary denial.

3. Investment in the parallel proceeding

The investment in the co-pending litigation is minimal. No claim construction briefing has occurred, and expert reports are exchanged May 6, 2026. Ex.1020, 4.

Thus, this factor weighs against discretionary denial.

4. Overlapping issues with the parallel proceeding

To mitigate any overlap, Petitioner filed a *Sotera* stipulation at the District Court. Consequently, this factor favors institution.

5. Identity of parties

Petitioner is a defendant in the litigation. That is true of most Petitioners in IPR proceedings, making this factor neutral. *See HP Inc. v. Slingshot Printing LLC*, IPR2020-01084, Paper 13 at 9 (Jan. 14, 2021).

6. Other circumstances

The petition presents a particularly strong case on the merits. Indeed, the purported inventive concepts were published by a Cornell research team and the prior art references expressly suggest their combination.

B. *Advanced Bionics*

During prosecution, the Examiner did not consider the prior art of this petition. Further, the prior art in this petition is materially different than the prior art in the previous IPRs, namely Trefethen discloses the precise order of operations required by the claims. Thus, discretionary denial under §325(d) is inappropriate because the first part of the *Advanced Bionics* framework is not met. *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 (Feb. 13, 2020) (precedential).

Under the second prong of *Advanced Bionics*, the Office erred in allowing the ’768 patent because the Office overlooked and did not consider the specific teachings of these prior art references.

C. *General Plastic*

1. Whether the same petitioner previously filed a petition directed to the same claims of the same patent.

This factor weighs against discretionary denial because this petition is filed by Intel Corporation, a different petitioner than the prior petitioner, NVIDIA Corporation. *See Toshiba America Information Systems, Inc. et al. v. Walletex Microelectronics Ltd.*, IPR2018-01538, Paper 11 at 20-21 (PTAB Mar. 5, 2019).

Further, Petitioner has not coordinated with NVIDIA for the purposes of preparing this Petition. *See Twitter, Inc., v. Palo Alto Research Center Inc.* IPR2021-01458, Paper 11 at 33 (PTAB April 6, 2022)(declining to discretionarily deny the petition under *General Plastic* and finding that there was no evidence that Petitioner had coordinated with previous filers).

Lastly, multiple petitions are a result of Patent Owner's litigation activity, electing to file separate complaints against Intel and NVIDIA. *See Samsung Elecs. Co. v. Iron Oak Techs., LLC*, IPR2018-01554, Paper 9 at 31 (PTAB Feb. 13, 2019) (The Board, when evaluating the *General Plastic* factors, "decline[d] to wield [Patent Owner's] litigation activities as a shield.").

2. Whether at the time of filing of the first petition the petitioner knew of the prior art asserted in the second

petition or should have known of it.

At the time the previous IPRs were filed, Intel had not been sued and had not yet identified the combination of references proposed in this petition.

And, while Intel's petition relies on one reference in common with the NVIDIA IPR (i.e., MPIref, Ex.1017), Intel's grounds for unpatentability are materially different by showing (i) cluster computing, (ii) parallelizing mathematical processing, (iii) peer-to-peer architecture, and (iv) sequential processing. *See Streck, Inc. v. Ravgen, Inc.*, IPR2021-01577, Paper 20 at 20 (PTAB Apr. 22, 2022) (even in the case of overlapping prior art, different theories based on that art are "new before" the Board).

This factor weighs against discretionary denial.

3. Whether at the time of filing of the second petition the petitioner already received the Board's decision on whether to institute review in the first petition.

The decisions denying institution of the prior IPRs were issued before the filing of this petition. Petitioner, however, has used the claim language the claims to identify the relevant prior art and has not used the PTAB's prior decisions or Patent Owner's arguments as a "roadmap." *See General Plastic Indus. Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 16-19 (PTAB 2017); *see also Apple, Inc. v. Theta IP, LLC*, IPR2024-00818, Paper 11 at 18 (PTAB 2024)

(reviewing and considering a prior petition does not warrant a *General Plastic* denial).

This factor weighs against discretionary denial.

4. The length of time that elapsed between learning of the prior art asserted in the second petition and the filing of the second petition.

Petitioner is filing this IPR before the service of its invalidity contentions at the district court and well before the 1 year time bar. See Ex.1019, Ex.1022. Petitioner has acted with reasonable diligence to prepare this petition.

This factor weighs against discretionary denial.

5. Whether the petitioner provides adequate explanation for the time elapsed between the filings of multiple petitions directed to the same claims of the same patent.

As noted in factor 1, this is Petitioner's first IPR against the '768 patent. To the extent this factor requires explaining the time elapsed since NVIDIA's petitions, as explained in factor 4, Petitioner has been diligent in its analysis of prior art since being separately sued for patent infringement. Moreover, Petitioner did not copy NVIDIA's arguments but instead relies on distinct grounds.

6. The finite resources of the Board and the requirement under 35 U.S.C. § 316(a)(11) to issue a final determination not later than 1 year after the date on which the Director

notices institution of review.

Petitioner expects that instituting review would require only modest resources from the Board, which already is familiar with the ’768 patent, and would proceed in compliance with 32 U.S.C. § 316(a)(11). Accordingly, this factor weighs against discretionary denial.

For these reasons, *General Plastic* discretionary denial institution does not apply.

XII. CONCLUSION

Petitioner has established a reasonable likelihood that the Challenged Claims are unpatentable.

Respectfully submitted,

/Andrew S. Ehmke/
Andrew S. Ehmke
Lead Counsel for Petitioner
Registration No. 50,271

Dated: March 28, 2025
HAYNES AND BOONE, LLP
2801 N. Harwood Street, Suite 2300
Dallas, Texas 75201
Customer No. 27683

XIII. MANDATORY NOTICES

A. Real party-in-interest

Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner certifies that the real party-in-interest is Intel Corporation.

B. Related matters

Pursuant to 37 C.F.R. § 42.8(b)(2), to the best knowledge of the Petitioner, the ’768 patent is or was involved in the following cases:

Case Caption	Number	Court	Filed	Status
Advanced Cluster Systems, Inc. v. Intel Corporation	7:24-cv-00245	W.D. Tex.	9/26/24	Pending
Advanced Cluster Systems, Inc. v. Advanced Micro Devices, Inc.	7:24-cv-00244	W.D. Tex.	9/26/24	Pending
Advanced Cluster Systems, Inc. v. NVIDIA Corporation	1:19-cv-02032	D. DE	10/28/19	Dismissed
NVIDIA Corporation v. Advanced Cluster Systems, Inc.	IPR2021-00019	PTAB	10/6/20	Institution Denied
NVIDIA Corporation v. Advanced Cluster Systems, Inc.	IPR2021-00020	PTAB	10/10/20	Institution Denied

C. Lead and back-up counsel and service information

Lead Counsel

Andrew S. Ehmke	Phone: 214-651-5116
HAYNES AND BOONE, LLP	Fax: 214-200-0853
2801 N. Harwood Street, Suite 2300	andy.ehmke.ipr@haynesboone.com
Dallas, TX 75201	USPTO Reg. No. 50,271

Back-up Counsel

David L. McCombs	Phone: 972-651-5533
HAYNES AND BOONE, LLP	Fax: 214-200-0853
2801 N. Harwood Street, Suite 2300	david.mccombs.ipr@haynesboone.com
Dallas, TX 75201	USPTO Reg. No. 32,271
David O'Dell	Phone: 972-739-7635
HAYNES AND BOONE, LLP	Fax: 214-200-0853
2801 N. Harwood Street, Suite 2300	david.odell.ipr@haynesboone.com
Dallas, TX 75201	USPTO Reg. No. 42,044
Gregory Huh	Phone: 972-739-6939
HAYNES AND BOONE, LLP	Fax: 214-200-0853
2801 N. Harwood Street, Suite 2300	gregory.huh.ipr@haynesboone.com
Dallas, TX 75201	USPTO Reg. No. 70,480

Please address all correspondence in this proceeding to lead and back-up counsel. Petitioner consents to service in this proceeding by email at the addresses above.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner hereby certifies, in accordance with and reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 13,971. Pursuant to 37 C.F.R. § 42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: March 28, 2025

/Andrew S. Ehmke/
Andrew S. Ehmke
Lead Counsel for Petitioner
Registration No. 50,271

CERTIFICATE OF SERVICE

The undersigned certifies that, in accordance with 37 C.F.R. § 42.6(e) and 37 C.F.R. § 42.105, service was made on Patent Owner as detailed below.

Date of service	March 28, 2025
Manner of service	FEDERAL EXPRESS
Documents served	Petition for *Inter Partes* Review Under 35 U.S.C. § 312 and 37 C.F.R. § 42.104 of U.S. 10,333,768; Petitioner's Power of Attorney; Petitioner's Exhibit List; Exhibits 1001-1042.
Persons served	Knobbe, Martens, Olson & Bear, LLP 2040 Main Street, Fourteenth Floor Irvine, CA 92614

/Andrew S. Ehmke/
Andrew S. Ehmke
Lead Counsel for Petitioner
Registration No. 50,271