

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
ADVANCED MICRO DEVICES, INC.
Petitioners,

v.

ADVANCED CLUSTER SYSTEMS, INC.,
Patent Owner.

IPR2025-00913
U.S. Patent No. 11,570,034

**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	4
I. Introduction.....	8
II. Grounds for standing	9
III. Note.....	9
IV. The Petition’s Reliance on Expert Testimony.....	9
A. Technological Background	9
B. Focused Expert Testimony.....	9
C. Detailed Expert Testimony.....	10
V. The ’034 patent.....	10
A. Overview	10
VI. Priority date of the ’034 patent	11
VII. Prosecution History of the ’034 patent.....	12
VIII. Level of ordinary skill in the art	12
IX. Claim construction.....	12
A. “ <i>peer-to-peer architecture</i> ”	13
X. Relief requested and the reasons for the requested relief.....	13
XI. Identification of how the claims are unpatentable.....	13
A. Challenged Claims	13
B. Statutory grounds for challenges.....	14

C.	Ground 1: Claims 1-25, 27-28, and 30 are obvious over Menon in view of Trefethen, RS6000, and POEref	16
1.	Menon	16
2.	Trefethen	20
3.	RS6000.....	21
4.	POEref.....	22
5.	Combining Trefethen with Menon	22
6.	Combining RS6000 with Menon	24
7.	Combining POEref with Menon	26
8.	Challenges.....	27
D.	Ground 2: Claims 26 and 29 are obvious over Menon in view of Trefethen, RS6000, and POEref, further in view of MPIref.....	87
9.	MPIref	87
10.	Combining MPIref with Menon	89
XII.	Sotera Stipulation.....	94
XIII.	Conclusion	95
XIV.	Mandatory notices	96
A.	Real party-in-interest.....	96
B.	Related matters	96
C.	Lead and back-up counsel and service information.....	97
	Certificate of Word Count	99
	Certificate of Service	100

PETITIONER’S EXHIBIT LIST

Ex.1001	U.S. Patent No. 11,570,034
Ex.1002	Prosecution History of U.S. Patent No. 11,570,034
Ex.1003	Declaration of Dr. Chandrajit L. Bajaj under 37 C.F.R. § 1.68
Ex.1004	<i>Curriculum Vitae</i> 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 against Intel in Western District of Texas
Ex.1020	Order Granting Joint Motion for Entry of Amended Scheduling Order for <i>Advanced Cluster Systems, Inc. v. Intel Corporation</i> , 7:24-CV-00245 (W.D.TX.)
Ex.1021	Federal Court Statistics
Ex.1022	Advanced Cluster Systems, Inc.’s Preliminary Infringement Contentions, January 22, 2025, <i>Advanced Cluster Systems, Inc. v. Intel Corporation</i> , 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 <i>Advanced Cluster Systems, Inc. v. NVIDIA Corporation</i> , 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 Trefethen
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, <i>Advanced Cluster Systems, Inc. v. Intel Corporation</i> , 7:24-cv-00245 (WDTX)
Ex.1043	September 26, 2024 Complaint against AMD in Western District of Texas
Ex.1044	Scheduling Order for <i>Advanced Cluster Systems, Inc. v. Advanced Micro Devices, Inc.</i> , 7:24-CV-00244 (W.D.TX.)
Ex.1045	Advanced Cluster Systems, Inc.'s Preliminary Infringement Contentions, February 24, 2025, <i>Advanced Cluster Systems, Inc. v. Advanced Micro Devices, Inc.</i> , 7:24-cv-00244 (WDTX)
Ex.1046	Sotera Stipulation, <i>Advanced Cluster Systems, Inc. v. Advanced Micro Devices, Inc.</i> , 7:24-cv-00244 (WDTX)
Ex.1047	Declaration of International Business Machines Corp.

Ex.1048	Affidavit of Tanya Zeif, Internet Archive
Ex.1049	MPIref at Argonne National Laboratory

I. INTRODUCTION

U.S. Patent No. 11,570,034 (the “’034 patent,” Ex.1001) contains long-winded claims that obfuscate the well-known concepts in those claims. The ’034 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 ’034 patent.

Anything purportedly novel about the ’034 patent was published in Menon and Trefethen. The ’034 patent recites cluster computing in a peer-to-peer architecture—the Cornell team used cluster computing in a peer-to-peer architecture. The ’034 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 ’034 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 teach all purportedly novel features of the '034 patent.

Accordingly, pursuant to 35 U.S.C. §§ 311, 314(a), Intel Corporation and Advanced Micro Devices, Inc. (“Petitioners”) respectfully request that the Board find claims 1-30 of the '034 patent unpatentable.

II. GROUNDS FOR STANDING

Petitioners certify the '034 patent is eligible for IPR and that Petitioners are not barred or estopped from requesting an IPR. 37 C.F.R. § 42.104(a).

III. NOTE

Petitioners cite exhibits' original page numbers, unless noted otherwise.

Emphasis in quoted material has been added. Claim terms are presented in *italics*.

IV. THE PETITION'S RELIANCE ON EXPERT TESTIMONY

A. Technological Background

Dr. Bajaj's declaration describes the background technology predating the '034 patent, including (i) cluster computing, (ii) parallelizing mathematical processing, (iii) peer-to-peer architecture, and (iv) sequential processing. *See* Ex.1003, ¶¶28-47.

B. Focused Expert Testimony

Dr. Bajaj's declaration provides focused expert testimony on the following

topics: (i) the level of skill in the relevant art (Ex.1003, ¶¶24-27); (ii) claim construction (Ex.1003, ¶¶54-56); (iii) reasons to combine the different prior art combinations (Ex.1003, ¶¶65-96), and (iv) key teachings of the prior art (Ex.1003, ¶¶57-64), including:

- Kernel (Ex.1003, ¶¶57-59);
- Cluster node module (Ex.1003, ¶¶60-62); and
- Peer-to-peer mathematical computation (Ex.1003, ¶¶63-64).

C. Detailed Expert Testimony

To the extent the Board is interested in more detailed analysis, Dr. Bajaj’s declaration also includes a detailed claim-by-claim analysis of the prior art. Ex.1003, ¶¶97-544.

V. THE ’034 PATENT

A. Overview

The ’034 patent describes computer clusters as “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.” Ex.1001, 1:22-27.

The purported novelty of the ’034 patent is to take a computer program, such as “Mathematica software,” as well as “Maple®” or “MATLAB®,” and then “adding cluster computing functionality” to it. Ex.1001, 1:16-17, 4:21-26. Figure

2 shows “kernel modules 206 a-e” (e.g. Mathematica) each “in communication with a single cluster node module 204 a-e.” Ex.1001, 5:21-40.

The cluster node modules use Message-Passing Interfaces (“MPIs”) to “distribute tasks among the kernel modules 206 a-e.” Ex.1001, 6:19-23. 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, 25:31-32, 24:48-51.

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 provides an “interface module” using “the MPI communication standard” to enable MATLAB to perform “easily parallelizable numerical computations on multiple processors.” Ex.1005, Abstract; Ex.1006, Abstract.

VI. PRIORITY DATE OF THE ’034 PATENT

The ’034 patent’s earliest possible priority date is June 13, 2006. It is unnecessary to determine whether it is entitled to this date in this proceeding as all relied-upon prior art predates June 13, 2006. Petitioners do not waive any right to dispute the ’034 patent’s priority date in this or another forum.

VII. PROSECUTION HISTORY OF THE '034 PATENT

The Examiner allowed the claims without rejection. In the Notice of Allowance, the Examiner provided analysis only with respect to three patent references, finding that the prior art failed to teach independent claim 2 (which would become claim 1). No specific reasons were identified, and the Examiner provided no analysis with respect to non-patent literature. *See generally*, Ex.1002.

VIII. 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, ¶¶18-23.

IX. 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, Petitioners submit 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 1 and 30.

In a previous IPR proceeding for the parent patent (IPR2021-00019), 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, Petitioners and its expert used this construction.

X. RELIEF REQUESTED AND THE REASONS FOR THE REQUESTED RELIEF

Petitioners ask that the Board institute an IPR trial and cancel the challenged claims in view of the analysis below.

XI. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

A. Challenged Claims

Petitioners challenge claims 1-30.

B. Statutory grounds for challenges

Grounds	Claims	Basis (pre-AIA)
#1	1-25, 27-28, 30	35 U.S.C. §103 over Menon in view of Trefethen, RS6000 and POEref
#2	26, 29	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 ’034 patent. Ex.1028, 4-9, Ex.1029, 1-3; *see Blue Calypso, LLC. v. Groupon, Inc.*, 815 F.3d 1331, 1350 (Fed. Cir. 2016) (public accessibility can be shown via an express citation); *see also* Ex.1003, ¶¶110-112.

“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 '034 patent. Ex.1035, 2-3; *see also* Ex.1003, ¶¶116-117.

“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, ¶¶119-120.

“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; Ex.1047; *see also* Ex.1003, ¶¶123-125.

“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 and was also made available on an Argonne National Laboratory website in 1997. Ex.1017, cover page; Ex.1029; Ex.1003, ¶129. 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 ’034 patent. Ex.1037, 2-16; *see also* Ex.1003, ¶¶129-130.

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).

Petitioners cite additional documents to evidence a POSITA’s knowledge and 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 1-25, 27-28, and 30 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 processor 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 that communicates “with other processes through a communication layer.” Ex.1005, 3, FIG. 1. The communication layer uses “MPI” to communicate “over the parallel platform’s interconnection network.” Ex.1005, 3, 5.

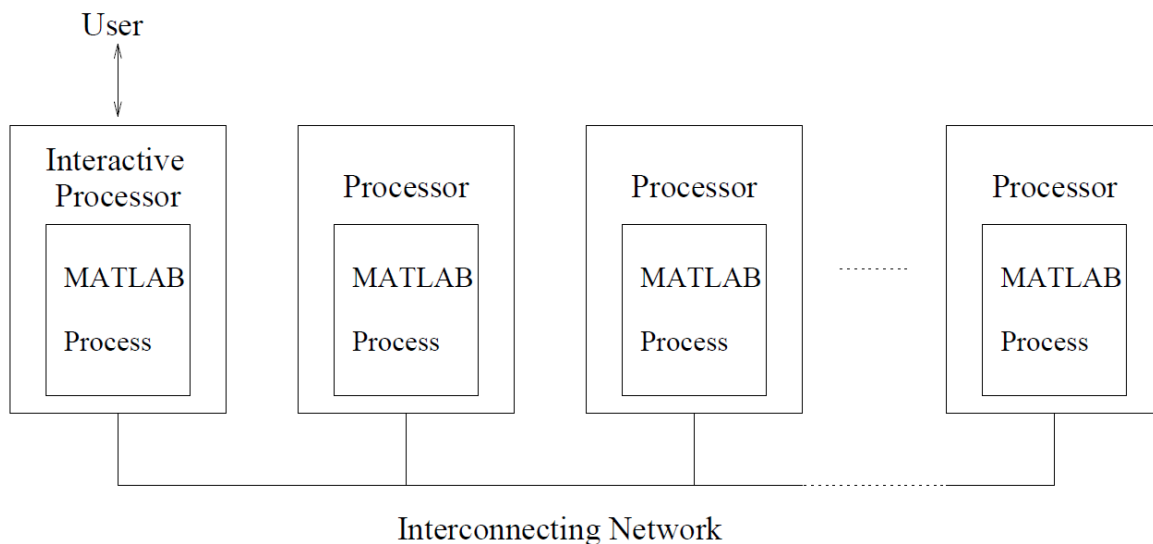


Figure 1: MultiMATLAB Architecture

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 in parallel. Ex.1005, 2, 5, 11.

c) Menon describes peer-to-peer architecture

Menon explains that MultiMATLAB works with different parallel programming paradigms. Ex.1005, 8. Notably, MultiMATLAB enables peer-to-peer communication through “SPMD/Message passing,” which provides “point to point” communication. Ex.1005, 9-10.

To provide “point to point” communication, Menon describes “message passing routines” that achieve “finer grain communication ... between any processes.” Ex.1005, 9, 8; Table 1¹.

¹ Menon’s Table 1 describes commands for two distinct paradigms: a master/slave paradigm and SPMD/Message Passing. The SPMD/Message Passing paradigm uses Eval(), Send() and Recv() for peer-to-peer communication. Ex.1003, ¶¶107-108.

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, ¶108

Menon discloses that MultiMATLAB uses a set of components to provide direct communication with other modules: “MEX Routines,” “interface module,” “indirection table” and “communication layer.” Ex.1005, 3, 5. Ex.1003, ¶¶104-106.

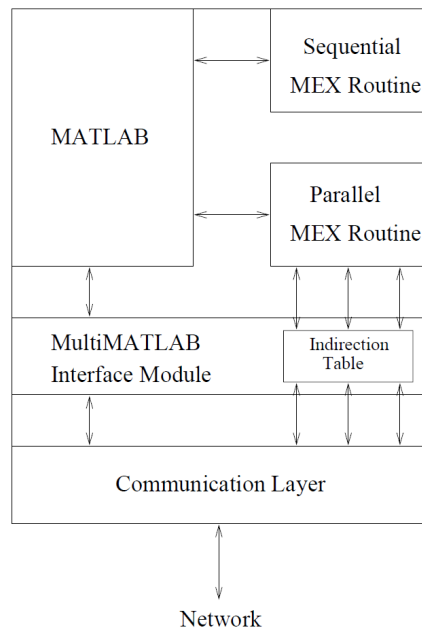


Figure 2: MATLAB Process Address Space

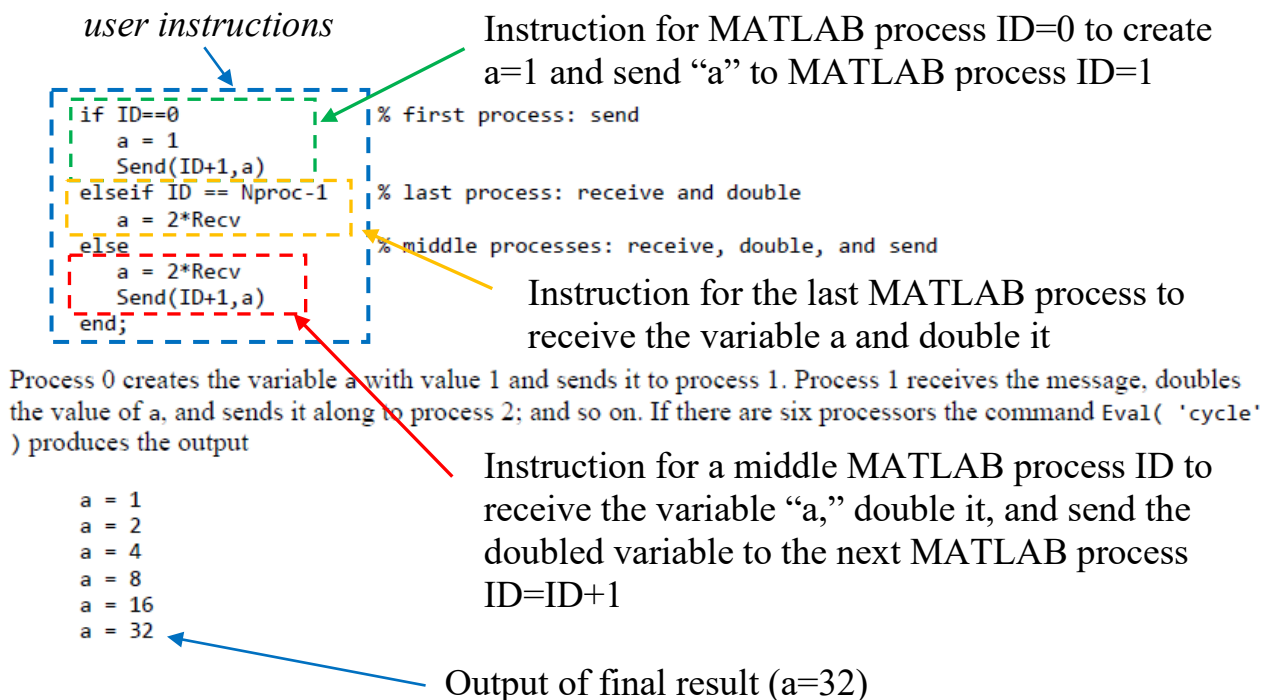
Ex.1005, FIG. 2.

2. Trefethen

Trefethen describes how MultiMATLAB runs 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, 6. Trefethen demonstrates that the Send and Recv commands allow direct communication among nodes. Ex.1006, 6:



Ex.1006, 6 (annotated); Ex.1003, ¶¶113-115.

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, ¶115.

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; Ex.1003, ¶118.

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).” Menon discusses using POE to initialize the MATLAB processes. Ex.1008, ix; Ex.1005, 6; Ex.1003, ¶¶121-122.

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 '034 patent, namely, the “the field of cluster computing” and are reasonably pertinent to the problem of the '034 patent, namely, executing a single program on multiple nodes. Ex.1005, 2; Ex.1006, Abstract; Ex.1001, 1:14-15; Ex.1003, ¶¶65-69.

b) Motivation to combine

Menon suggests the combination by expressly citing Trefethen and explaining that Menon builds on the teachings of Trefethen. Ex.1005, 1-2. Ex.1003, ¶71.

Additionally, Trefethen describes “how the [MultiMATLAB] system is used” using code examples that perform parallel mathematical calculations using

MATLAB, Ex.1006, 2, while Menon describes “tak[ing] advantage of high performance distributed memory processors” to enable MATLAB to perform parallel mathematical calculations. 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 executing MATLAB in parallel to perform mathematical calculations. Ex.1005, Abstract; Ex.1006, 2-7; Ex.1003, ¶70.

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. Ex.1003, ¶72.

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 achieved this goal: the MultiMATLAB system executes MATLAB in parallel to perform mathematical calculations.

Ex.1005, 2 (“In this paper, we ... achieve ... MATLAB code with high-performance parallel routines”). Ex.1003, ¶¶73.

Accordingly, a POSITA would have been motivated and found it predictable to combine Trefethen with Menon. *See also*, Ex.1003, ¶¶70-73.

6. Combining RS6000 with Menon

a) Analogous art

RS6000 is analogous prior art because it pertains to the same field of endeavor as the '034 patent, namely, the “the field of cluster computing” and is reasonably pertinent to the problem of the '034 patent, namely, executing a single program on multiple nodes. Ex.1007, 1; Ex.1001, 1:14-15; Ex.1003, ¶¶74-77.

b) Motivation to combine

A POSITA would have combined RS6000 with Menon because Menon suggests the combination by teaching that MultiMATLAB uses the “IBM SP2, a modern high performance distributed memory multiprocessor.” Ex.1005, 5.

RS6000 provides “an overview of the IBM RS/6000 SP.”² Ex.1007, 1; Ex.1003, ¶78.

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 “parallel applications.” Ex.1007, 1; Ex.1003, ¶79.

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; Ex.1003, ¶80.

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 MultiMATLAB on an IBM SP2 and that MultiMATLAB “was design[ed] specifically for the IBM SP2.” Ex.1005,

² “IBM SP2” is the same as “IBM RS/6000 SP.” Ex.1010, 6 (“the SP2 was renamed to simply the SP”).

6, 14 (“These results demonstrate that the MultiMATLAB system can be an effective platform for parallel computing”). *See*, Ex.1003, ¶81.

7. Combining POEref with Menon

a) Analogous art

POEref is analogous prior art pertaining to the same field of endeavor as the ’034 patent, namely, “the field of cluster computing” and is reasonably pertinent to the problem of the ’034 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, ¶¶82-85.

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; Ex.1003, ¶¶86-87.

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; Ex.1003, ¶86.

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; Ex.1003, ¶88.

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, ¶89.

8. Challenges

Claim 1

[1.0] *A computer cluster comprising:*

Menon discloses a “MultiMATLAB architecture” providing “high-performance parallel routines.” Ex.1005, 2. MultiMATLAB’s cluster comprises multiple nodes, where each node “runs a MATLAB process” and “communicate[s] with other processes through a communication layer.” Ex.1005, 3.

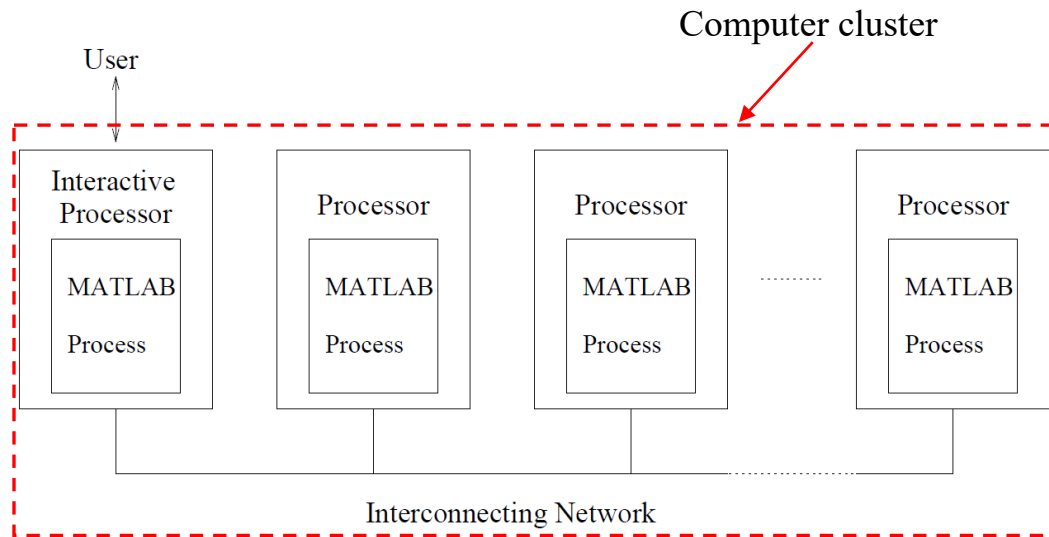


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶118

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 processor nodes to perform mathematical calculations. Ex.1005, 13, FIGs. 7, 8 (using “8 processors” and “32 processors,” respectively).

Accordingly, Menon’s cluster with multiple MATLAB processes implemented on IBM SP2 processor nodes to execute parallel mathematical computations, in view of RS6000’s description that the IBM SP2 is a networked cluster computing system for executing parallel programs, renders obvious this limitation. Ex.1003, ¶¶157-163.

[1.1.1] a plurality of nodes, wherein each of the plurality of nodes comprises a hardware processor

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

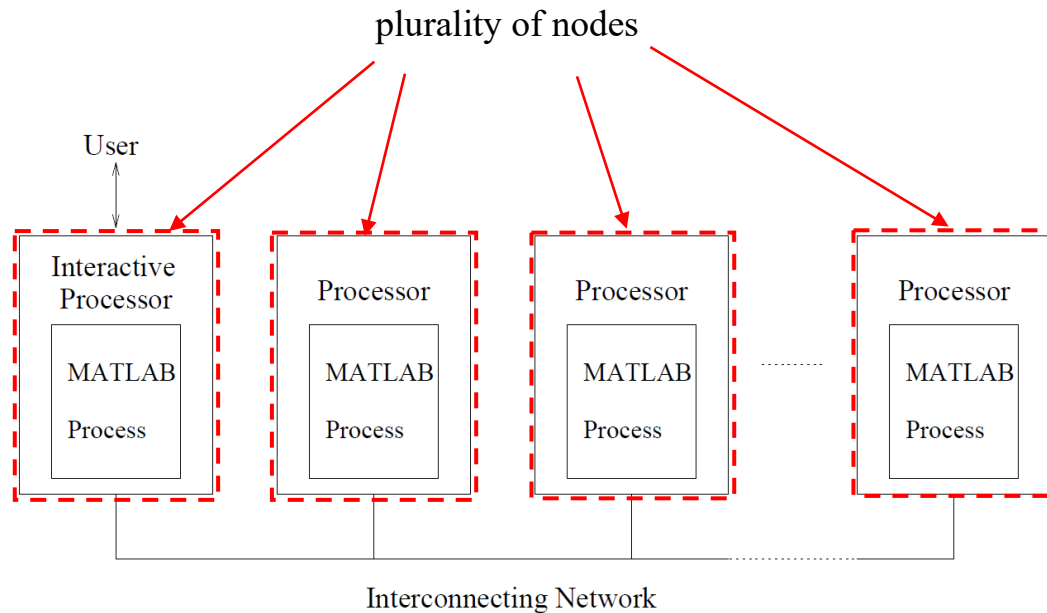


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶124

RS6000 describes that each IBM SP2 “processor node[]” has a “Symmetric MultiProcessor (SMP)” (“*hardware processor*”) for executing “parallel applications.” Ex.1007, 1-2.

Thus, Menon discloses using 32 processor nodes of an IBM SP2 and RS6000 teaches that each IBM SP2 node has a symmetric multiprocessor, rendering obvious this limitation. Ex.1003, ¶¶164-166.

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

Menon discloses that the user “interacts directly with” a MATLAB process called the “interactive process” node (*node*). Ex.1005, 3. Menon discloses that “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.” Ex.1005, 4.

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

Accordingly, Menon discloses that the interactive process node 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 commands, and POEref further discloses the “poe” command initializes a local environment and reproduces that local environment on each remote processor node, rendering obvious this limitation. Ex.1003, ¶¶167-170.

[1.1.3] *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 the hardware processor to evaluate mathematical expressions; and*

Menon discloses that each IBM SP2 node “individually runs a MATLAB process.” Ex.1005, 3. The ’034 patent identifies MATLAB as an example of a single-node kernel. *See, e.g.,* Ex.1001, 4:14-21.

Figure 1 shows a MATLAB process executing on processor nodes.

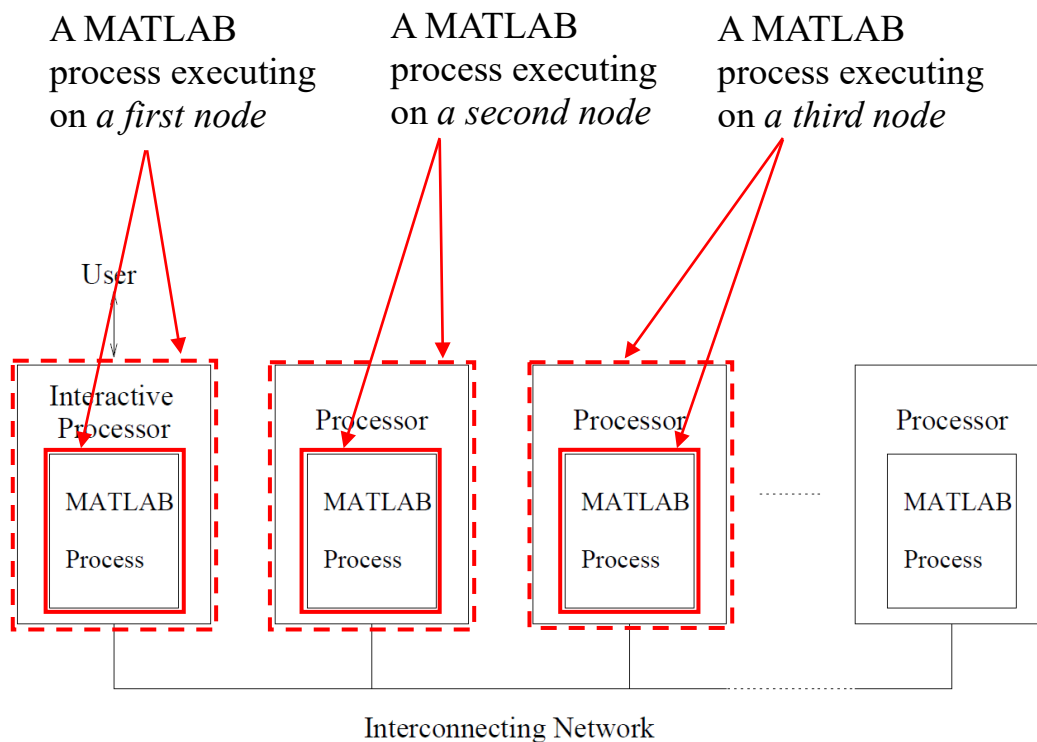


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶132

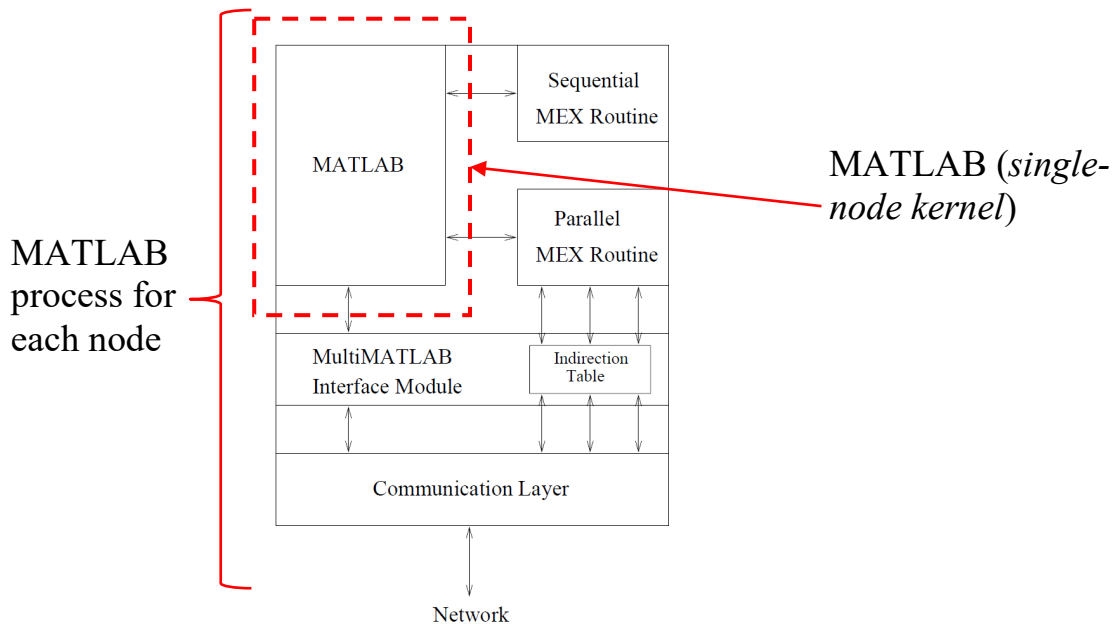


Figure 2: MATLAB Process Address Space

Ex.1005, FIG. 2 (annotated); Ex.1003, ¶133

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, ¶174. Accordingly, each node executing MATLAB discloses each node executing a *single-node kernel*.

Menon describes MATLAB as performing mathematical calculations for “engineers and scientists.” Ex.1005, 1. As one example, Menon describes calculating “the conjugate gradients algorithm.” Ex.1005, 11-13. Trefethen provides additional examples, such as each node doubling a variable, “fast Fourier transform,” and “polynomial zero-finding.” 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, 11-13. Thus, MATLAB (*a single-node kernel*) causes the hardware processor of the IBM SP2 node to perform mathematical calculations (*program code ... capable of causing a hardware processor to evaluate mathematical expressions*).

Menon describes MATLAB as being in an “address space.” Ex.1005, FIG. 2, 5. A POSITA would understand Menon’s “address space” to be “memory locations ... contain[ing] the executable program.” Ex.1018, 67-68. POSITA would then recognize that MATLAB is in, and accessed from, a *computer-readable medium*. The *computer-readable medium* of the IBM SP2 processor nodes can be disk drives or memory cards, as taught by RS6000. Ex.1007, 6, 10.

Accordingly, Menon’s disclosure of each hardware processor accessing the MATLAB process address space (*computer-readable medium*) containing MATLAB (*computer program code for a single-node kernel*) to perform mathematical calculations (*evaluate mathematical expressions*) in view of RS6000’s teaching that each IBM SP2 node has disk drives and memory cards for storing program code accessible by the node’s hardware processor, renders obvious this limitation. Ex.1003, ¶¶171-182.

[1.2] *a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using a peer-to-peer architecture;*³

Menon discloses that “[a]ll parallel functionality in the MultiMATLAB system is provided through MEX routines.” Ex.1005, 3. The MEX routines send “commands over the underlying communication layer to the processes corresponding to the specified IDs.” Ex.1005, 4. The underlying communication layer of MultiMATLAB “runs over the parallel platform’s interconnection network.” Ex.1005, 3. Menon describes MultiMatlab running on an “IBM SP2” and Figure 1 shows the interconnection network linking the processor nodes. Ex.1005, 5.

³ In addition to the arguments presented here, “*communications network interface*” lacks patentable weight because it does not confer any structure on the claimed “*communications network interface*.” 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.”).

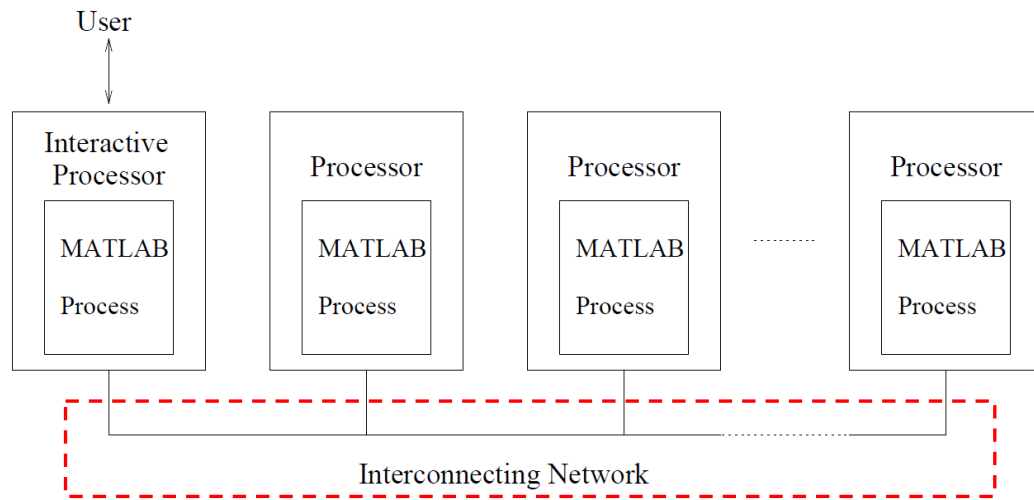


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶145

Interconnecting network connecting each MATLAB process on a processor node to the others

Trefethen discloses that MultiMATLAB’s commands 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 “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 and data to other processor nodes and provides the example “Eval([4 5] , ‘cond(hilb(ID))’)”. Ex.1006, 6. MATLAB on each of node ID=4 and ID=5 then perform the task “cond(hilb(ID))” using the data ID. Ex.1006, 4; Ex.1003, ¶188.

Third, Menon’s MEX routines use a “point to point ... communication” paradigm. Ex.1005, 9. Unlike a “master/slave paradigm,” Menon’s SPMD/Message Passing paradigm uses “point to point ... communication” to provide a “finer degree of communication.” Ex.1005, 9. Table 1 of Menon lists point-to-point commands, 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 1⁴.

⁴ Menon’s Table 1 describes commands for two distinct paradigms: a master/slave paradigm and SPMD/Message Passing. The SPMD/Message Passing paradigm uses Eval(), Send() and Recv() for peer-to-peer communication. Ex.1003, ¶65.

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.

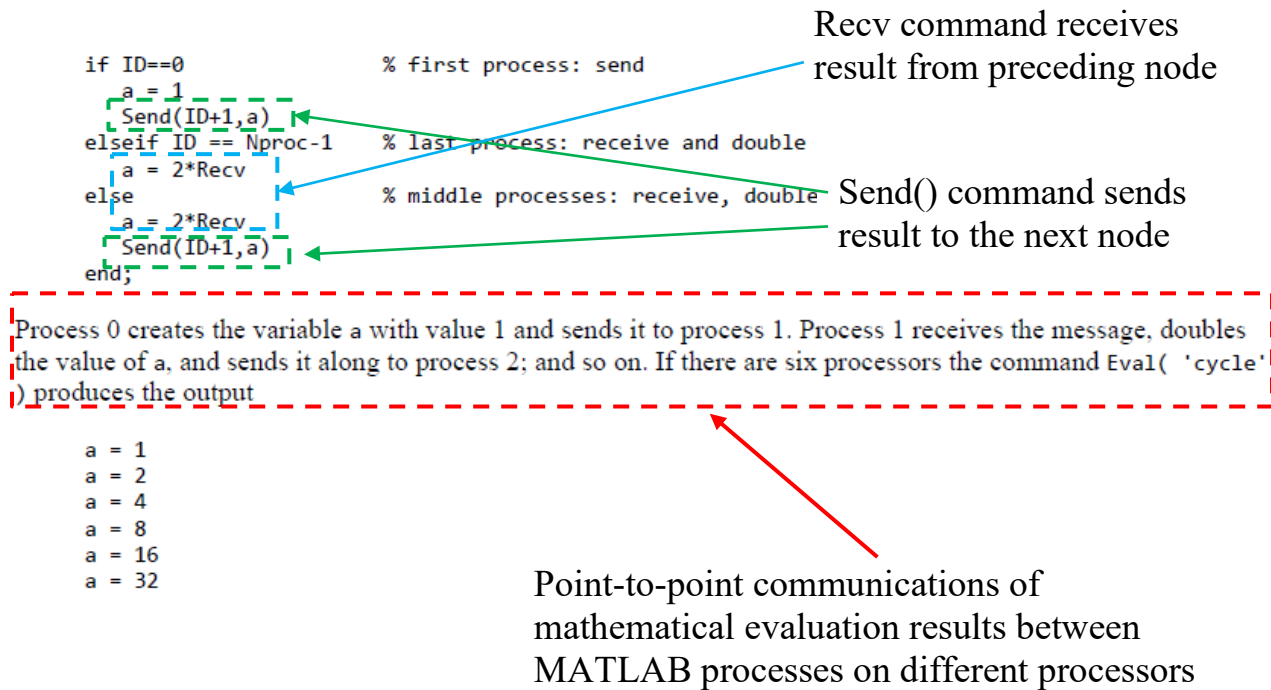
Table 1: Selected commands in the MultiMATLAB system

Commands for point-to-point communication

Ex.1005, Table 1 (annotated); Ex.1003, ¶149

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*.

Trefethen describes a MATLAB process using point-to-point commands to communicate *results of mathematical expression evaluation* to another processor node. Ex.1006, 3-4. Trefethen provides an example script in which a MATLAB process communicates *results of mathematical expression evaluation* directly to a next MATLAB process:



Ex.1006, 6; Ex.1003, ¶190.

While Trefethen’s example uses six processor nodes, it would have been obvious to a POSITA that Trefethen contemplates any number of processor nodes, including three. Ex.1003, ¶190.

Trefethen discloses that MultiMATLAB commands can 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.

RS6000 describes the IBM SP2’s network for connecting the nodes. For example, each node has “five [] adapter slots” that can include “Ethernet,” “Token Ring,” “ATM,” “Wide Area Network (WAN)” and more. Ex.1007, 2.

Additionally, each node has “integrated ethernet with BNC and RJ45 ports.”

Ex.1007, 10. Further, the IBM SP2 includes a “switch” to “provide a message-passing network that connects all processor nodes with a minimum of four paths between every pair of nodes.” Ex.1007, 4.

Accordingly, Menon’s MEX routines that enable each processor node to communicate tasks and data directly with the other nodes over an interconnecting network implemented on an IBM SP2, in view of Trefethen’s teaching that MEX routines enable mathematical expression evaluation results from a preceding MATLAB process to be communicated to a next MATLAB process, further in view of RS6000 teaching that each IBM SP2 node uses Ethernet and other communication adapters to communicate over a message-passing network, renders obvious this limitation. Ex.1003, ¶¶183-193.

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

See [1.1.1], *supra*. Ex.1003, ¶194.

[1.3.1] *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,*

As shown in [1.1.1], Menon and RS6000 disclose a node having a symmetric multiprocessor. Further, as shown in [1.1.3], Menon and RS6000 disclose a processor that accesses address space in a memory to execute MATLAB.

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, ¶199. Accordingly, each node executing MATLAB discloses each node executing a single-node kernel.

Menon discloses “[t]he user interacts directly with one MATLAB process, called the interactive process.” Ex.1005, 3. For example, Menon describes “an intuitive graphical interface” to allow “users to specify geometries, differential equations, and boundary conditions.” Ex.1005, 6.

Accordingly, Menon and RS6000 describe a node having (i) a symmetric multiprocessor that accesses and executes MATLAB stored in memory, and (ii) the code for MATLAB and the user interface stored in the memory of, and accessed by, the node allows users to interact with the node to provide equations, rendering obvious this limitation. Ex.1003, ¶¶195-200.

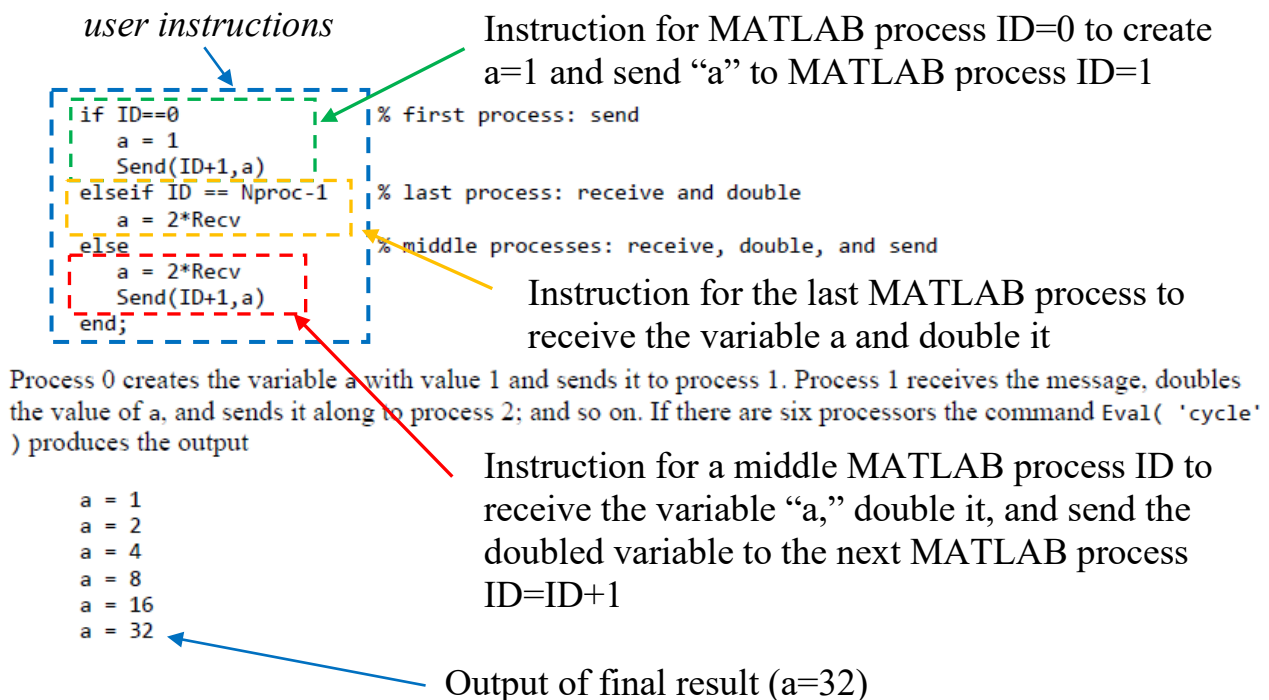
[1.3.2] *the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and*

As shown in [1.1.3], Menon teaches MATLAB as a single-node kernel. Menon teaches an example of a user providing user instructions with “an Eval routine for parallel evaluation.” Ex.1005, 4. The Eval routine is “a MEX Routine” that receives “MATLAB process IDs and a MATLAB command... [and] sends the commands over the underlying communication layer to the processes

corresponding to the specified IDs.” Ex.1005, 4. Menon describes MEX routines, such as the Eval routine, are “run directly from [the] MATLAB” environment. Ex.1005, Abstract, 3. MATLAB is configured to distribute the calls using the MEX routines (*e.g.*, Eval). Ex.1005, Abstract, 3, 4; Ex.1003, ¶202.

Trefethen describes how *user instructions* are interpreted. Ex.1006, 3-7. For example, Trefethen discloses the user instruction “Eval([4 5] , ‘cond(hilb(ID))’).” MATLAB interprets the instruction and calls the Eval MEX routine so that the calls “cond(hilb(ID))” are distributed to the MATLAB processes on processor ID=4 and ID=5 for each to perform. Ex.1006, 4; Ex.1003, ¶203.

As another example, Trefethen discloses the m-file “cycle.m,” which is distributed to the plurality of nodes using Eval. Ex.1006, 6. These user instructions instruct: (i) the interactive MATLAB process (ID=0) to create variable *a* with a value of 1 and send variable *a* to the next MATLAB process, (ii) each middle MATLAB process to receive variable *a* from the immediately preceding MATLAB process, double the value of received variable *a*, and send the result to the next MATLAB process, and (iii) the last MATLAB process to receive variable *a* and double its value. Ex.1006, 6. The *user instructions* cycle.m and the final output from six MATLAB processes are shown below:



Ex.1006, 6 (annotated); Ex.1003, ¶205.

Accordingly, Menon’s teaching that MATLAB is configured to interpret commands and cause 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, ¶¶201-206.

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

As shown in [1.1.1], Menon discloses executing a series of MATLAB processes on 32 IBM SP2 processor nodes.

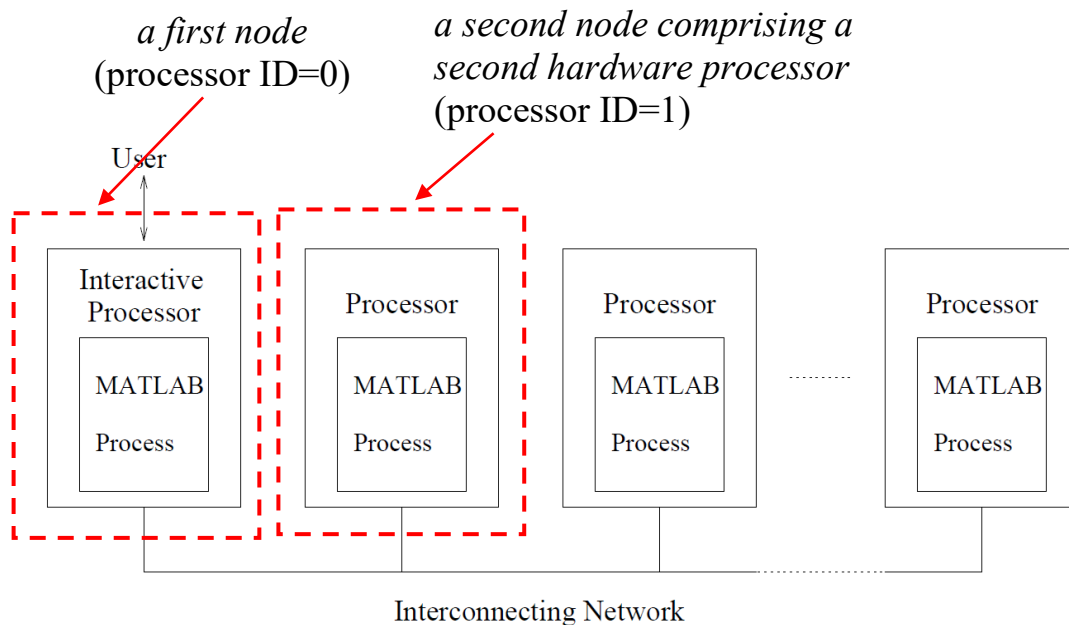


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶168

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

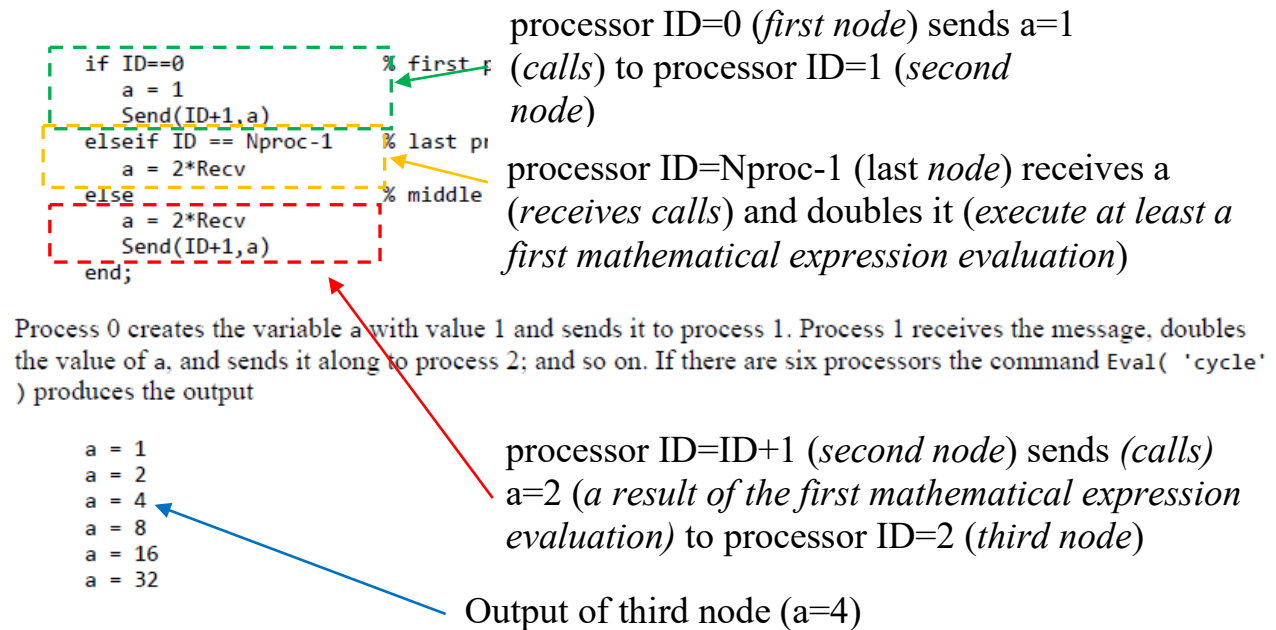
Menon and RS6000 teaching 32 IBM SP2 processor nodes, where each IBM SP2 node has a symmetric multiprocessor having up to sixteen processors (each having a core), renders obvious this limitation. Ex.1003, ¶¶207-210.

[1.4.2] 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 the first mathematical expression evaluation to a third node;

As shown in [1.2], Menon teaches MEX routines having code to implement standard variants of MPI calls to enable point-to-point communication so that the MATLAB processes can communicate tasks and data directly 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 [1.3.2], Trefethen discloses that MultiMATLAB can be configured with user instructions that cause: (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 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.



Ex.1006, 6 (annotated); Ex.1003, ¶214.

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 using 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, ¶¶211-215.

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

Menon describes a third processor node:

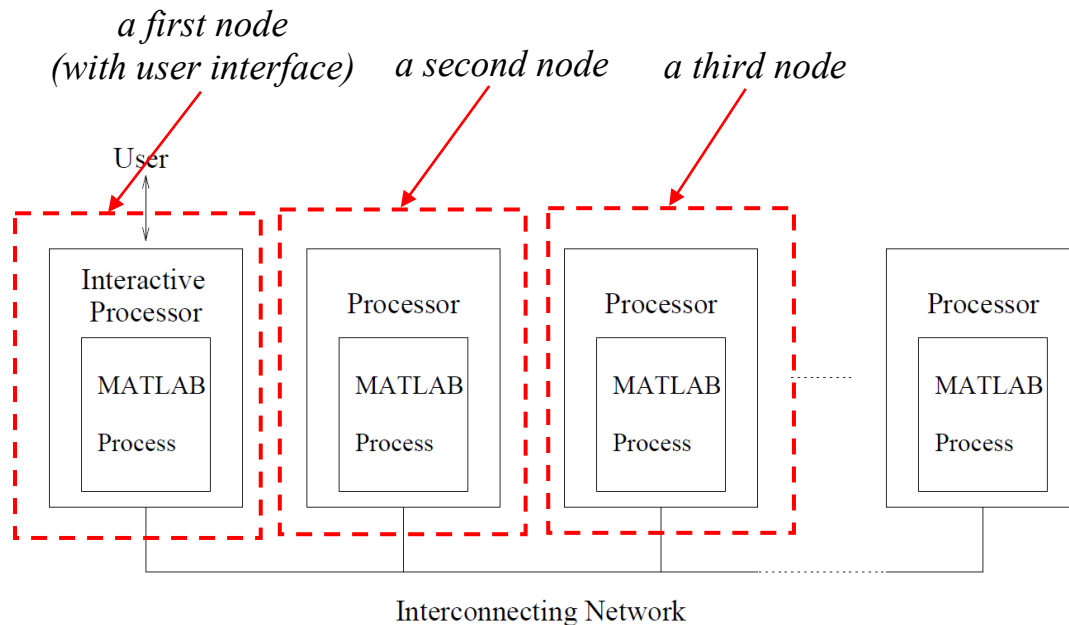


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶177

As shown in limitation [1.4.1], each IBM SP2 node has a symmetric multiprocessor having up to sixteen processor cores. Ex.1007, 9, Ex.1003, ¶¶216-217.

[1.5.2] wherein the third node is configured to receive the result of the first mathematical expression evaluation from the second node, execute at least a second mathematical expression evaluation using the received result, and communicate the result of the second mathematical expression evaluation to the first node;

As shown in [1.4.2], Trefethen teaches the MATLAB process at processor ID=2 (a third node) receives 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 MATLAB process at processor ID=2 uses its hardware processor to double the received variable:

```

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 ← Output of third node (a=4)
a = 8
a = 16
a = 32 ← Output of final result (a=32)

```

Ex.1006, 6 (annotated); Ex.1003, ¶220.

Trefethen further discloses that each MATLAB process transmits its output “to the master process as soon as it is ready.”⁵ Ex.1006, 4. A POSITA would

⁵ 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, ¶221.

recognize Trefethen's "master process" as Menon's interactive MATLAB process that provides the output to the user. Ex.1003, ¶221.

Accordingly, Menon's teaching of a point-to-point message passing 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, ¶¶218-225.

[1.6] wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;

Menon discloses "[t]he user interacts directly with one MATLAB process, called the interactive process." Ex.1005, 3. Trefethen discloses mathematical results are "sent to the user's screen" when "one wishes to monitor the progress of computations on several processors graphically." Ex.1006, 7.

As shown in [1.5.2], when 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 interactive processor to provide the result to the user (output result from third node, "a=4").

Therefore, Menon's interactive MATLAB process in view of Trefethen's use of MATLAB to display output and results on the user's screen render obvious this limitation. Ex.1003, ¶¶226-228.

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

See [1.1.1], *supra*. Ex.1003, ¶229.

[1.7.1] accept user instructions;

As shown in [1.3.1], Menon discloses MultiMATLAB's interactive processor has a MATLAB process that provides a user interface to allow users to input equations.

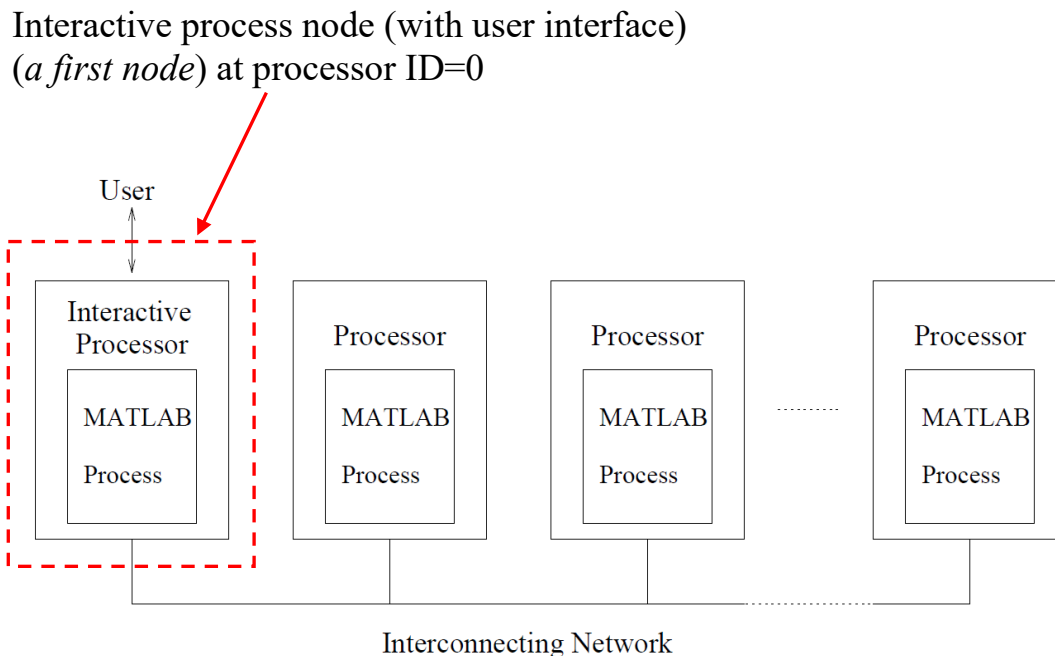


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶231

Menon explains that users can provide “geometries, differential equations, and boundary conditions.” Ex.1005, 6. Further, as discussed in [1.3.2], a user can also provide *user instructions* via an “m-file,” which can contain “dozens of high-level commands.” Ex.1006, 2.

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, ¶¶230-234.

[1.7.2] *after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and*

As discussed in [1.3.2] and [1.7.1], *user instructions* can be provided via m-files that specify different actions for different MATLAB processes.

As discussed in [1.2], Menon, Trefethen, and RS6000 render obvious a communications network interface for the nodes to communicate with each other.

Menon describes how the interactive processor (e.g., ID=0) sends commands to the other MATLAB processes, stating that the Eval routine is “a MEX Routine” that receives “MATLAB process IDs and a MATLAB command... [and] sends the

commands over the underlying communication layer to the processes

corresponding to the specified IDs.” Ex.1005, 4.

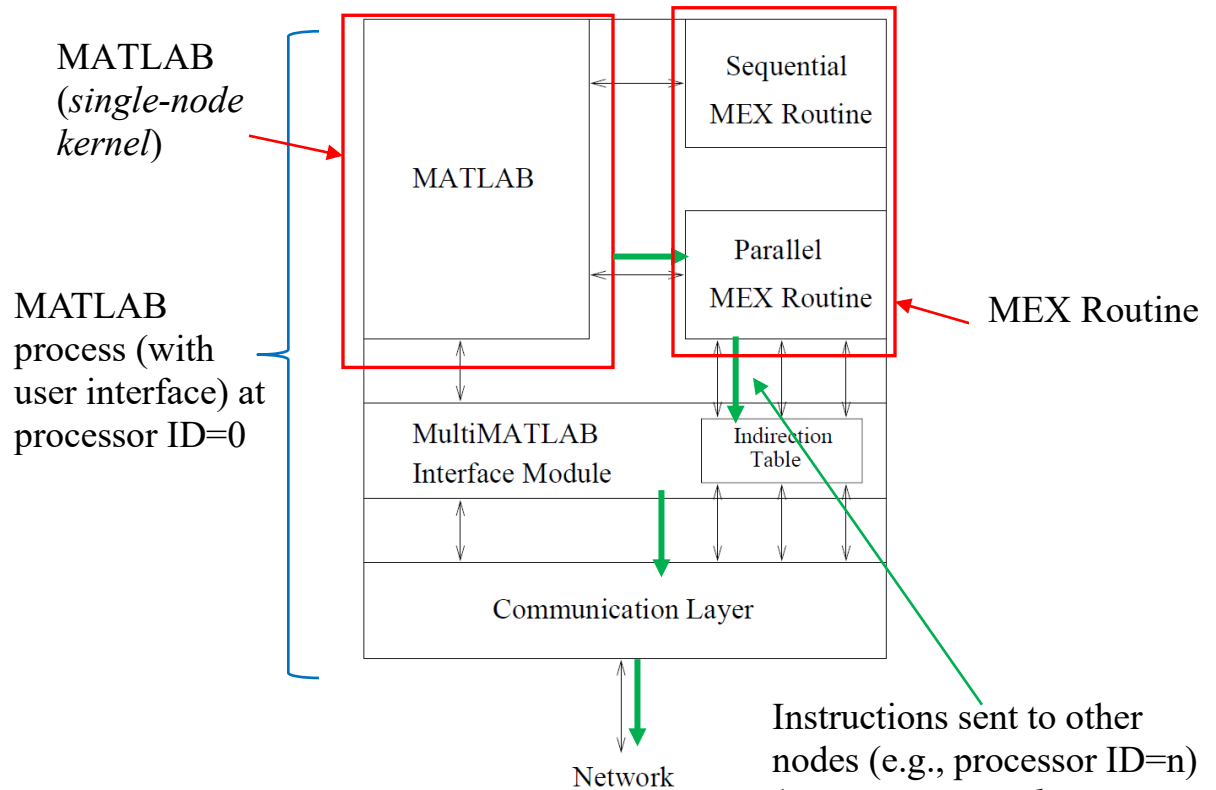


Figure 2: MATLAB Process Architecture
Instructions sent to other nodes (e.g., processor ID=n) (communicate at least some of the user instructions using the communications network interface)

Ex.1005, FIG. 2 (annotated); Ex.1003, ¶238

Accordingly, Menon in view of Trefethen teaches that (i) the interactive processor receives user instructions (including IDs identifying MATLAB processes in MultiMATLAB) and then (ii) after receiving those instructions, the instructions are communicated over the communication layer to the other MATLAB processes corresponding to the IDs using MEX routines so that the

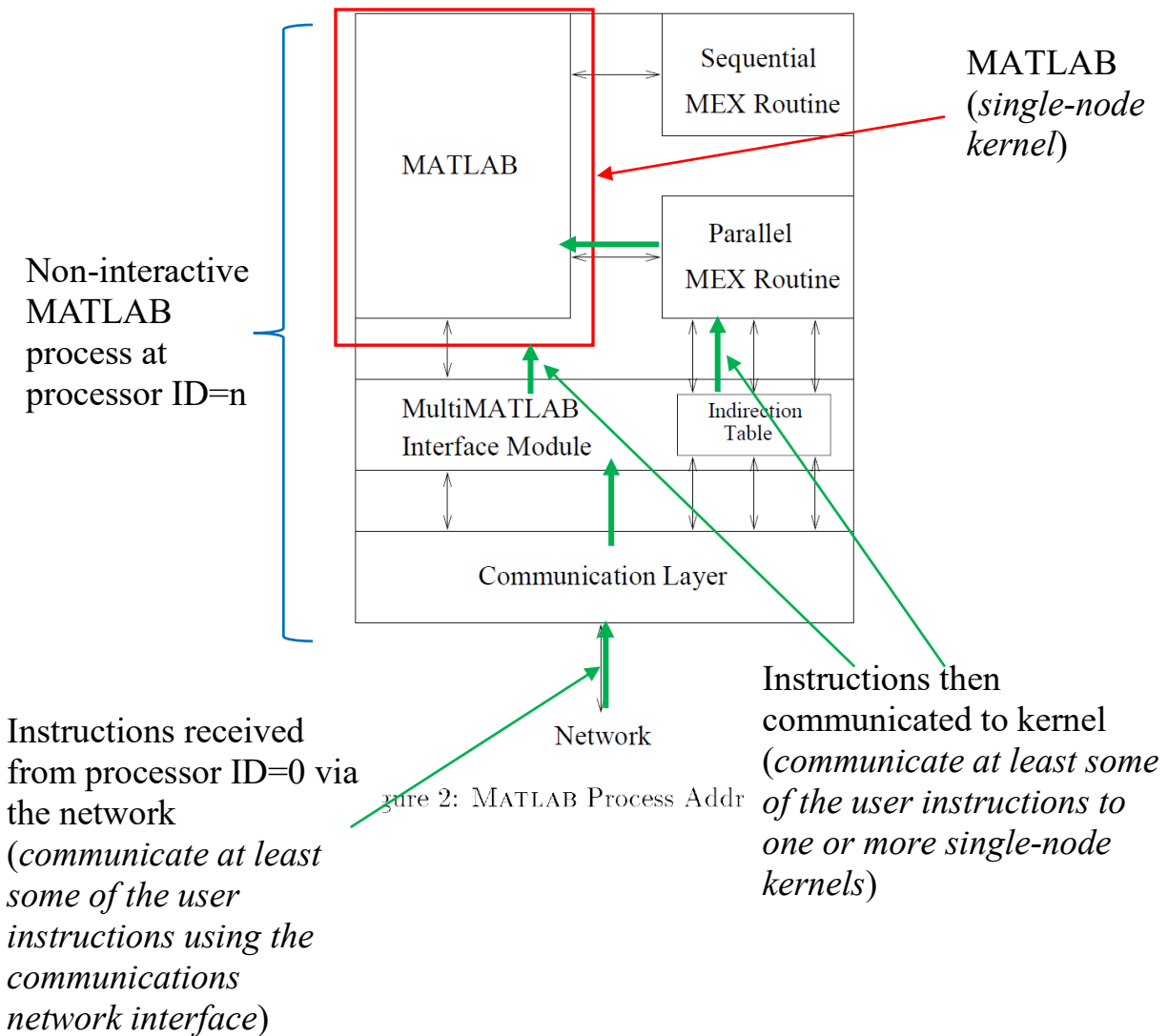
MATLAB processes can communicate tasks and data directly with each other over the IBM SP2's interconnection network, thereby rendering obvious this limitation.

Ex.1003, ¶¶235-239.

[1.7.3] *after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels.*

Trefethen provides examples showing that commands specified in the *user instructions* are processed by the respective MATLAB processes identified by the IDs in the user instructions. Ex.1006, 3-7. For example, as discussed in [1.3.2], the user provides *user instructions*, such as “Eval([4 5] , ‘cond(hilb(ID))’)”, to the interactive MATLAB process. Then, in accordance with that instruction, the calls “cond(hilb(ID))” (*at least some of the user instructions*) 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, ¶242.



Ex.1005, FIG. 2 (annotated); Ex.1003, ¶242

Accordingly, Menon and Trefethen teach that the MEX routines on the interactive processor communicate commands over the interconnection network implemented on an IBM SP2, and once received by each non-interactive node, the commands are communicated to that node's MATLAB (*single-node kernel*) for processing, thereby rendering obvious this limitation. Ex.1003, ¶¶240-243.

Claim 2

[2.0] *The computer cluster of claim 1, wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface.*

As discussed in [1.2], Menon and RS6000 teach the nodes communicating with each other over the IBM SP2's message-passing network.

RS6000 further describes that each node has “five [] adapter slots” that can include “Ethernet,” “Token Ring,” “ATM”, “Wide Area Network (WAN)” and more. Ex.1007, 2.

Accordingly, Menon and Trefethen teach MultiMATLAB on an IBM SP2 in view of RS6000 teaching the IBM SP2 having at least a local area network (LAN) and wide area network (WAN), rendering obvious this limitation. Ex.1003, ¶¶244-245.

Claim 3

[3.0] *The computer cluster of claim 1, wherein each of the nodes comprises one or more cluster node modules.*

Menon discloses that each node includes a set of components to provide interconnection with other modules: “MEX Routines,” “MultiMATLAB interface module,” “indirection table” and “communication layer” (collectively, the *cluster node module*). Ex.1005, 3, 5. “MEX routines” provide “[a]ll parallel functionality in the MultiMATLAB system.” Ex.1005, 3. “The interface module ... is

responsible for initializing an underlying communication layer, such as MPI ..., and exposing it to the rest of the system.” Ex.1005, 3.

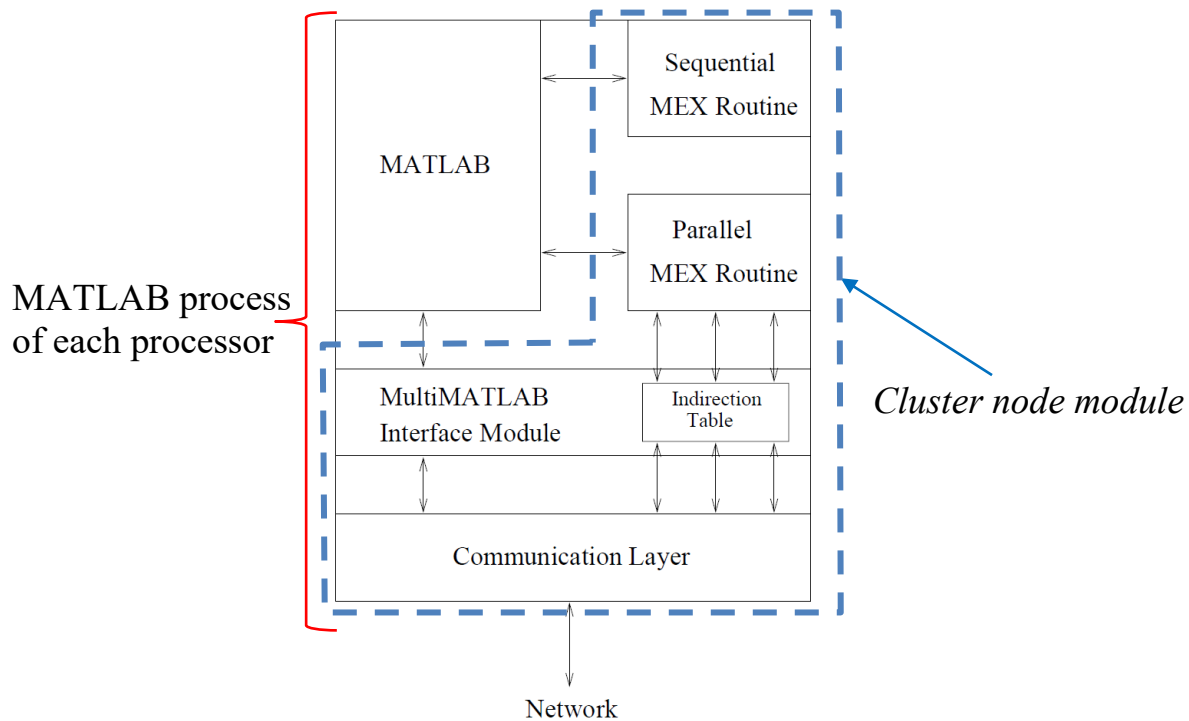
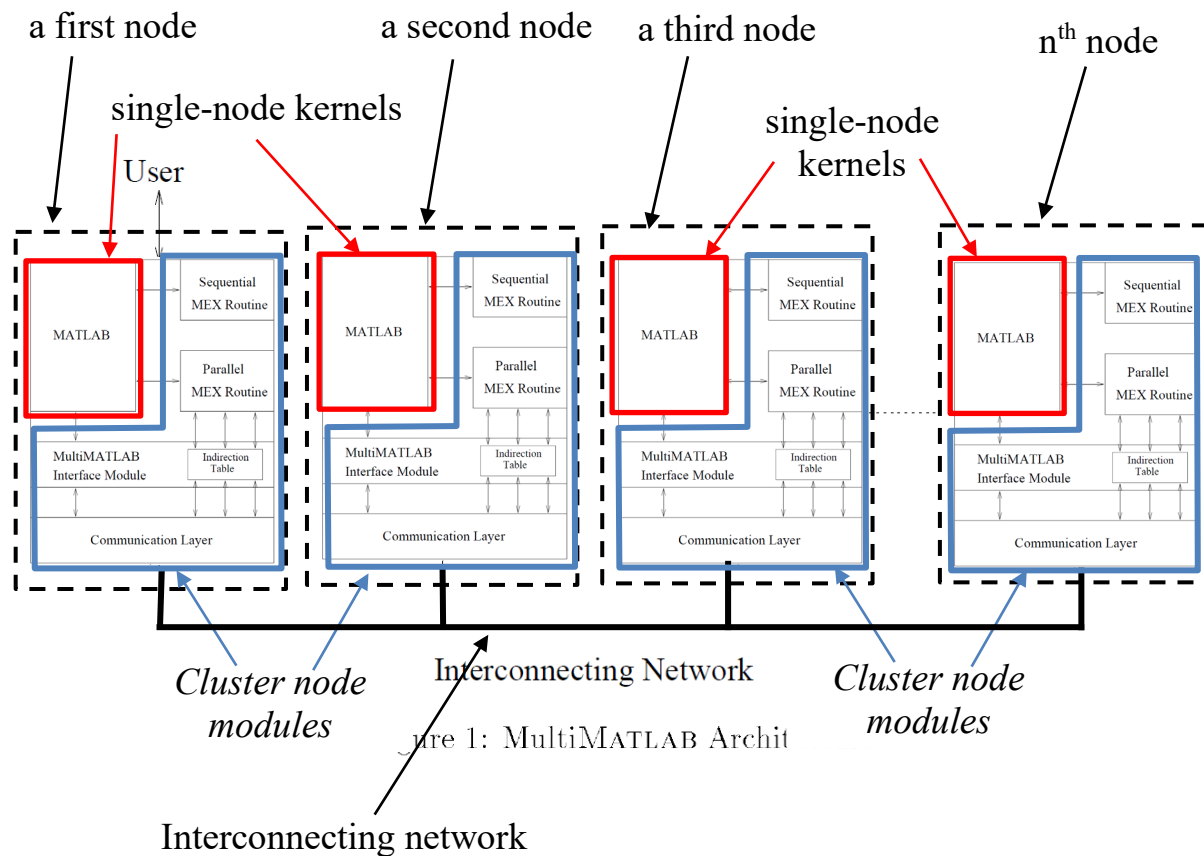


Figure 2: MATLAB Process Address Space

Ex.1005, FIG. 2 (annotated); Ex.1003, ¶249

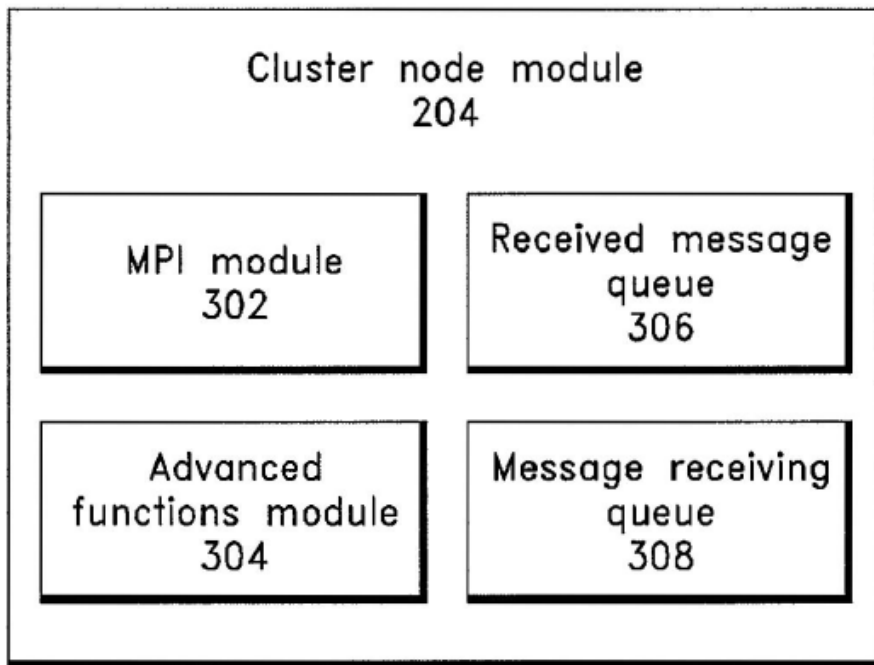
Menon’s Figure 1 and Figure 2, combined into a single figure as shown below, illustrate the relationship among the communication components (MEX routines, interface module, indirection table, and the communication layer) and the

single-node kernel for each MultiMATLAB node.



Ex.1005, FIGs. 1 and 2 (combined and annotated); Ex.1003, ¶250

Correspondingly, the '034 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 node having a set of components to provide interconnection with other modules (MEX routines, MultiMATLAB interface module, indirection table, and communication layer) (*cluster node module*), rendering obvious this limitation. Ex.1003, ¶¶246-251.

Claim 4

[4.0] *The computer cluster of claim 1, wherein each single-node kernel is stored in a non-transitory computer-readable medium and configured to accept and execute a request.*

As shown in [1.1.3], each processor node stores MATLAB (*single-node kernel*) in memory address space on disk drives or memory cards (*computer-readable medium*).

As shown in [1.7.2], Menon and Trefethen disclose that each MATLAB process waits for commands and executes routines based on those commands when received. Ex.1005, 3-4.

Accordingly, Menon teaches that each node has a single-node kernel (MATLAB) that (i) is stored in memory and executed by a processor and (ii) awaits MATLAB commands (*requests*) to arrive and then executes those commands in its own MATLAB environment, rendering obvious this limitation. Ex.1003, ¶¶252-257.

Claim 5

[5.0] *The computer cluster of claim 4, wherein each of the nodes comprises one or more cluster node modules, wherein each of the cluster node modules comprises instructions stored in a non-transitory computer-readable medium, and wherein the instructions, when executed by the hardware processor, cause the cluster node module to communicate with the single-node kernel and with one or more other cluster node modules.*

As shown in [3.0], each node has a set of components for handling communications with other processors (MEX routines, a MultiMATLAB interface module, indirection table and communication layer) (*cluster node module*).

As shown in [1.1.3], the address space on each processor node is a disk drive or a memory card (*computer-readable medium*) that is accessed by the node. A POSITA would recognize that the set of components for handling communications with other processors (MEX routines, MultiMATLAB interface module,

indirection table, and communication layer (*cluster node module*), is software (*computer instructions*) stored in memory for execution by the processor.

Ex.1005, 4.

Menon describes that its cluster node module (MEX routines, a MultiMATLAB interface module, indirection table and communication layer) facilitates communication with the single-node kernel (MATLAB) and across the network. Ex.1005, 3-4 (“[a]ll parallel MEX routines access the underlying communication layer and, hence, the network, through the MultiMATLAB interface module”). Further, Menon’s Figure 1 and Figure 2, combined into a single figure as shown below, illustrate the communication between MATLAB (*single-node kernel*) and the MultiMATLAB *cluster node module*.

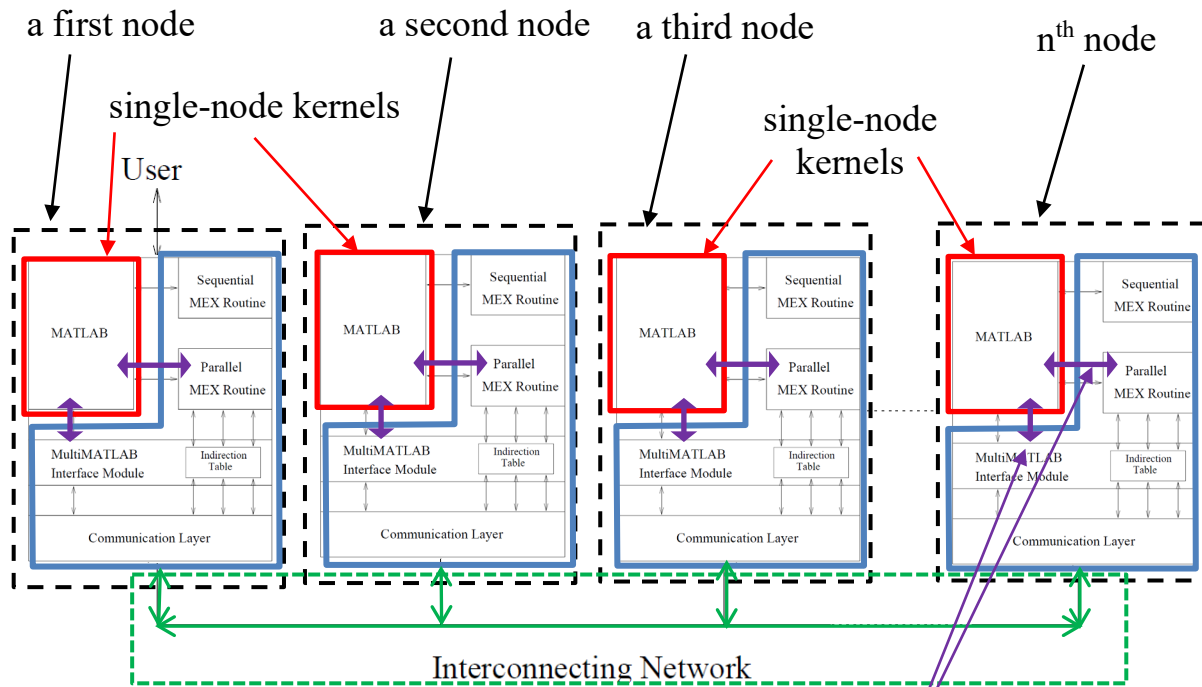


Figure 1: MultiMATLAB Architecture

*Cluster node modules
communicating to and
from each other*

*Single-node kernel
communicating to and
from cluster node module*

Ex.1005, FIGs. 1 and 2 (combined and annotated); Ex.1003, ¶261

Accordingly, Menon describes a set of components (MEX routines, MultiMATLAB interface module, an indirection table and a communication layer) that (i) are stored in memory and executed by the processor, (ii) communicate with MATLAB (*single-node kernel*), and (iii) communicate with the communications components of the other nodes (*one or more other cluster node modules*), rendering obvious this limitation. Ex.1003, ¶¶258-262.

Claim 6

[6.0] *The computer cluster of claim 5, wherein the plurality of cluster node modules act as a cluster.*

As discussed in [5.0], Menon discloses each MATLAB process has a set of components (MEX routines, a MultiMATLAB interface module, indirection table and communication layer) (*the plurality of cluster node modules*) to communicate with the other nodes.

Menon describes that MultiMATLAB is a “parallel platform” implemented on an IBM SP2 to execute “parallel programs on ... a networked cluster.”

Ex.1005, 3, 6; Ex.1008, 1. That is, the MATLAB processes of MultiMATLAB executing commands in parallel are a group of processor nodes communicating with each other to accomplish a mathematical computation as if they were a single computer (*act as a cluster*), rendering obvious this limitation. Ex.1005, 6; Ex.1008, 1; Ex.1003, ¶¶263-271.

Claim 7

[7.0] *The computer cluster of claim 5, wherein the one or more cluster node modules communicate with one another to act as a cluster.*

As shown in [6.0], Menon describes that MultiMATLAB allows a user to solve a mathematical problem by enabling MATLAB to use MEX Routines to communicate with other MATLAB processes so that the group of processor nodes

accomplish the mathematical computation as if they were a single computer.

Ex.1003, ¶¶272-275.

Claim 8

[8.0] *The computer cluster of claim 7, wherein the computer cluster includes the user interface.*

As discussed in [6.0]-[7.0], Menon describes MultiMATLAB as a parallel program on a networked cluster. As discussed in [1.3.1], Menon discloses a graphical interface to allow the user to directly interact with the interactive processor node. Ex.1005, 6.

Accordingly, Menon's MultiMATLAB computer cluster with a graphical interface renders obvious this limitation. Ex.1003, ¶¶276-279.

Claim 9

[9.0] *The computer cluster of claim 5, wherein each cluster node module accepts instructions from the user interface and interprets one or more of the instructions.*

As described in [1.3.1], the user interacts directly with interactive processor to specify user instructions. From the user interface, the user can “directly” run the MEX routines of the cluster node module. Ex.1005, 3; *see also* Ex.1006, 3, 6.

As discussed in [1.7.2] and [1.7.3], the user instructions (tasks and data) are interpreted and communicated directly to, and received by, the other nodes.

As discussed in [5.0], each MATLAB process has a cluster node module for handling communications with other nodes.

Menon describes that the cluster node module interprets uses the “indirection table,” to map (*interpret*) the MEX routines to “MPI calls.” Ex.1005, 6.

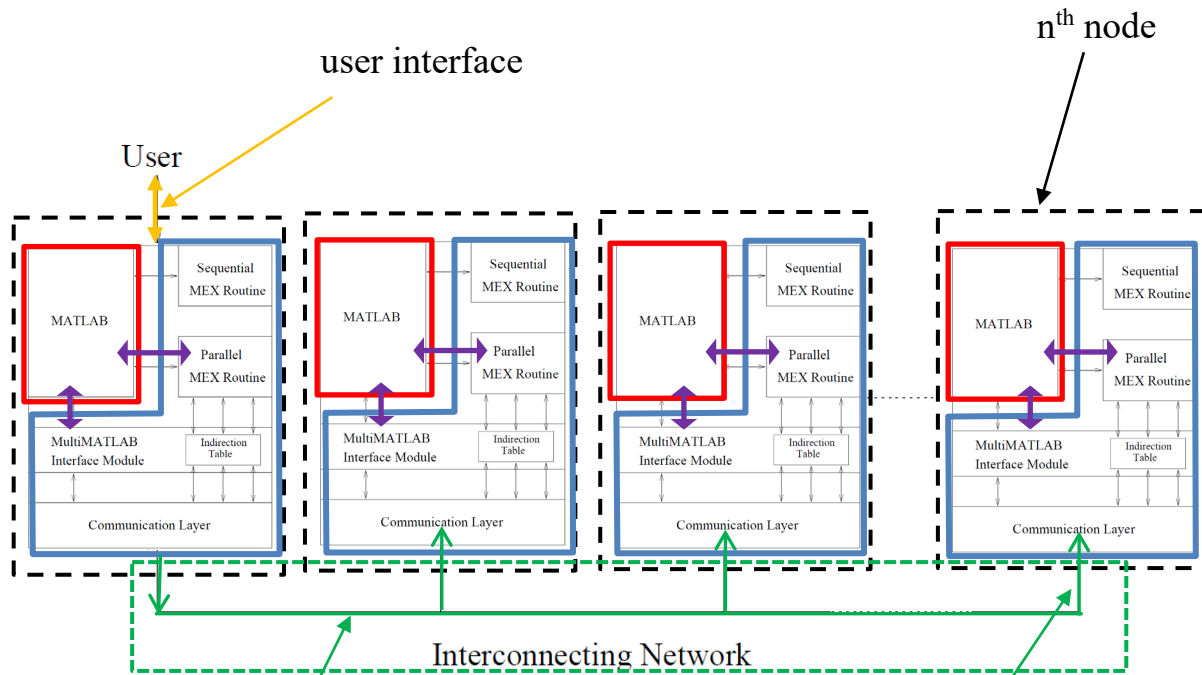


Figure 1: MultiMATLAB Architecture

Instructions transmitted to other nodes using the communication layer

Instructions received from interactive processor via the network

Ex.1005, FIGs. 1 and 2 (combined and annotated); Ex.1003, ¶282

Accordingly, Menon teaches that (i) the user enters instructions to directly run the MEX routines of the cluster node module, (ii) the cluster node module accepts the instructions and interprets the instructions to determine the destination of the instructions, (iii) each of the non-interactive MATLAB processes accepts the

instructions via its cluster node module, and (iv) the cluster node module maps the MPI calls to the table offset, rendering obvious this limitation. Ex.1003, ¶¶280-285.

Claim 10

[10.0] *The computer cluster of claim 1, wherein the cluster initialization process comprises establishing communication among two or more of the nodes.*

As shown in [1.1.2], IBM's POE initializes the MATLAB processes on each node.

POEref provides additional details of Menon's cluster initialization process. POEref describes that POE "ensures that each API initializes properly," including "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 using POE to start MATLAB processes on each node in view of POEref's disclosure that the initialization process ensures initialization of MPI and communication among the processor nodes renders obvious this limitation. Ex.1003, ¶¶286-289.

Claim 11

[11.0] *The computer cluster of claim 1, wherein the cluster initialization process comprises configuring access by one or more of the nodes to a computer-readable medium comprising program code for the single-node kernel.*

As shown in [1.1.3], each node has MATLAB in an address space.

As shown in [1.1.2], IBM's POE initializes the MATLAB processes on each node.

POEref discloses that, during initialization, POE "allocates [] nodes to the job." Ex.1008, 54. Accordingly, a POSITA would understand that when a job (which, in the context of Menon, corresponds to the single-node kernel) is allocated to a node (i.e., during the initialization process), the node is configured to access the computer-readable medium having the program code for the job. Ex.1003, ¶293.

Accordingly, Menon's teaching of using POE to initialize the nodes that each contain a MATLAB process address space having MATLAB, in view of POEref's teaching that nodes are allocated to a job during initialization of the job, renders obvious this limitation. Ex.1003, ¶¶290-294.

Claim 12

[12.0] *The computer cluster of claim 11, wherein the cluster initialization process comprises establishing message-passing support among the nodes in the cluster.*

As shown in [1.1.2], IBM's POE initializes the MATLAB processes on each node. MultiMATLAB is based upon a "version of MPI specifically optimized for the SP2." Ex.1005, 5.

As shown in [10.0], POEref discloses that, during the initialization, POE ensures that MPI is initialized properly and that each node is connected. Ex.1008, 3.

Accordingly, Menon describes using POE to launch the MATLAB processes that use MPI, in view of POEref describing that POE ensures that the MPI communication is initialized properly so that each node can communicate, renders obvious this limitation. Ex.1003, ¶¶295-298.

Claim 13

[13.0] The computer cluster of claim 11, wherein the cluster initialization process comprises launching cluster node modules by the cluster.

As shown in [1.1.2], IBM's POE initializes the MATLAB processes on each node. As shown in [3.0], each MATLAB process includes a cluster node module.

Accordingly, Menon teaches that the MATLAB process on each processor node is started by IBM's POE, which results in MATLAB process on each node, which includes a cluster node module, being launched, rendering obvious this limitation. Ex.1003, ¶¶299-301.

Claim 14

[14.0] The computer cluster of claim 13, wherein the cluster initialization process comprises assigning a processor identification number to each of the cluster node modules.

As shown in [1.1.2], IBM's POE initializes the MATLAB processes on each node. Menon further explains that “[f]or 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.

As shown in [3.0], each MATLAB process includes a MultiMATLAB cluster node module.

Accordingly, Menon assigning an identification number to each of the MATLAB processes, which includes a cluster node module, when the processes are started, renders obvious this limitation. Ex.1003, ¶¶302-305.

Claim 15

[15.0] *The computer cluster of claim 1, wherein the cluster initialization process comprises:*

See [1.1.2], supra. Ex.1003, ¶306.

[15.1] *launching cluster node modules by the cluster; and*

See [13.0], supra. Ex.1003, ¶307.

[15.2] *after launching the cluster node modules, configuring access by one or more of the nodes to a non-transitory computer-readable medium comprising program code for the single-node kernel.*

As discussed in [13.0], Menon teaches launching cluster node modules using POE.

As discussed in [11.0], POEref discloses that when a user starts a job with POE, POE “allocates [] nodes to the job.” Ex.1008, 54. Accordingly, a POSITA would understand that, when a job is allocated to a node during the initialization

process, the node is configured to access the computer-readable medium having the program code for the job. Ex.1003, ¶309.

Moreover, a POSITA would have recognized there are only two options available for timing the configuring and the launching steps with respect to each other, i.e., one before or after the other, rendering both options obvious. *See Uber Techs., Inc. v. X One, Inc.*, 957 F.3d 1334, 1341 (Fed. Cir. 2020) (finding that “[e]ven if the two proposed solutions would have required different implementation,” the fact that the proposed solutions are “two known, finite, predictable solutions for solving the same problem” renders them obvious). Ex.1003, ¶¶308-311.

Claim 16

[16.0] *The computer cluster of claim 1, wherein the cluster initialization process comprises:*

See [1.1.2], *supra*. Ex.1003, ¶312.

[16.1] *launching cluster node modules by the cluster;*

See [13.0], *supra*.. Ex.1003, ¶313.

[16.2] *after launching the cluster node modules, establishing communication among two or more of the nodes; and*

Limitation [16.2] is rendered obvious for the reasons presented for [10.0] and [13.0]. Ex.1003, ¶¶314-317. Further, a POSITA would have recognized there are only two predictable options available for timing the establishing and the

launching steps with respect to each other, i.e., one before or after the other, rendering both options obvious. *Id.* Ex.1003, ¶¶317.

[16.3] *after establishing communication among two or more of the nodes, assigning a processor identification number to each of the cluster node modules.*

Limitation [16.3] is rendered obvious for the reasons presented for [10.0] and [14.0]. Ex.1003, ¶¶318-320. Further, a POSITA would have found it obvious to assign a processor identification number to each of the cluster node modules after establishing communication among two or more of the nodes because POEref teaches that the user enters the “poe” command to establish communication among and with the nodes. Ex. 1008, 30. After the poe command, then the nodes are identified by the node id. Ex.1008, 31. Ex.1003, ¶¶320.

Claim 17

[17.0] *The computer cluster of claim 1, wherein one or more of the nodes are configured to communicate at least some of the user instructions.*

See [1.7.2], *supra*. Ex.1003, ¶321.

Claim 18

[18.0] *The computer cluster of claim 17, wherein one or more of the nodes are configured to communicate at least some of the user instructions to one or more single-node kernels.*

See [1.7.3], *supra*. Ex.1003, ¶322.

Claim 19

[19.0] *The computer cluster of claim 1, wherein one or more of the nodes are configured to accept user instructions via one or more of the nodes.*

See [1.7.2]-[1.7.3] and [1.1.2], supra. Ex.1003, ¶¶323-325.

Claim 20

[20.0] *The computer cluster of claim 19, wherein one or more of the nodes are configured to communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other.*

See [1.7.2], supra. Ex.1003, ¶326.

Claim 21

[21.0] *The computer cluster of claim 1, wherein one or more of the nodes are configured to transmit at least some of the user instructions that originate from a user interface.*

See [1.3.1] and [1.7.2], supra. Ex.1003, ¶¶327-329.

Claim 22

[22.0] *The computer cluster of claim 1, wherein one or more of the nodes are configured to parallelize at least some of the user instructions before communicating at least some of the user instructions to one or more single-node kernels.*

As shown in [1.2], Menon teaches MEX routines having code to implement standard variants of MPI calls to enable point-to-point communication so that the MATLAB processes can communicate tasks and data directly with each other. As shown in [1.7.3], Menon's interactive processor node communicates user instructions to the other nodes using the communication layer.

Menon describes that the interactive MATLAB processor uses “parallel MEX Routines” to provide the “necessary message passing and data distribution.” Ex.1005, 10. An example is “an Eval routine for parallel evaluation.” Ex.1005, 4. The parallel MEX Routines “distribute data from the interactive MATLAB process among all processes” because “[a]ll parallel MEX Routines access the underlying communication layer and, hence, the network.” Ex.1005, 10, 4.

A POSITA would recognize that Menon’s use of MPI calls with its parallel MEX routines is parallelizing the user instructions. Ex.1008, 1-2 (explaining that a developer using MPI to run parallel tasks is called “parallelizing the application.”); Ex.1003, ¶334.

Accordingly, Menon’s interactive processor that receives an equation, uses the parallel MEX Routines in connection with MPI calls to distribute tasks and data to each of the other MATLAB processes (and corresponding MEX Routines) renders obvious this limitation. Ex.1003, ¶¶330-335.

Claim 23

[23.0] *The computer cluster of claim 1, wherein one or more of the nodes are configured to:*

See [1.7.0], *supra*. Ex.1003, ¶336.

[23.1] *accept user instructions;*

See [1.7.1], *supra*. Ex.1003, ¶337.

[23.2] *after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and*

See [1.7.2], supra. Ex.1003, ¶338.

[23.4] *after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels.*

See [1.7.3], supra. Ex.1003, ¶339.

Claim 24

[24.0] *The computer cluster of claim 1, wherein the plurality of nodes are configured to communicate with one another to interpret and translate commands for execution by a plurality of single-node kernels.*

As discussed in [3.0], Menon discloses that each node has a cluster node module for communicating with the other nodes.

As discussed in [1.2], Menon teaches MEX routines having code to implement standard variants of MPI calls to enable point-to-point communication so that the MATLAB processes can communicate tasks and data directly with each other.

As discussed in [9.0], Menon discloses that the MultiMATLAB interface module acts as an interpretation layer that maps MPI calls to the indirection table so tasks and data can to be provided to MATLAB for computation.

Accordingly, Menon teaches that (i) each processor node uses a communication layer that uses MPI calls to enable the processors to communicate

with one another and (ii) the MultiMATLAB interface module on each processor maps those MPI calls to the appropriate table offset (*interprets and translate commands*) to expose all functions and data to MATLAB (*single-node kernel*) for execution, rendering obvious this limitation. Ex.1003, ¶¶340-344.

Claim 25

[25.0] *A computer cluster comprising:*

See [1.0], *supra*. Ex.1003, ¶345.

[25.1.0] *a plurality of nodes:*

See [1.1.0], *supra*. Ex.1003, ¶346.

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

See [1.1.2], *supra*. Ex.1003, ¶¶347.

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

See [10.0], *supra*. Ex.1003, ¶348.

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

Menon describes one of the MATLAB processes as an interactive MATLAB process with which a user interacts directly. Ex.1005, 3, 6.

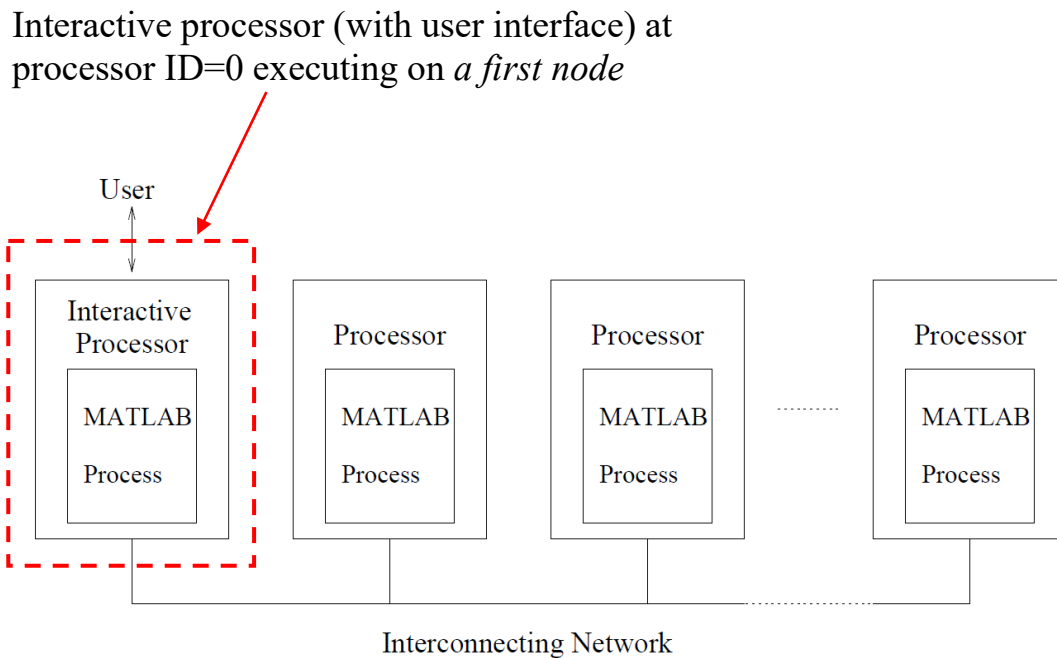


Figure 1: MultiMATLAB Architecture

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶350

Menon discloses “an intuitive graphical interface allowing users to specify” equations. Ex.1005, 6.

Trefethen describes that a user can use an “m-file” to provide instructions. Ex.1006, 5. A POSITA would recognize 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 receives user instructions, in view of Trefethen’s teaching of a user inputting an m-file into

MATLAB that is used to assign different actions to different processes, renders obvious this limitation. Ex.1003, ¶¶349-355.

[25.2] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other using asynchronous calls;

As discussed in [1.2], Menon and RS6000 teach the nodes communicating with each other over the IBM SP2's message-passing network.

Trefethen details how nodes communicate mathematical results. 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, ¶357.

Trefethen further discloses that MultiMATLAB commands can 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 that enable the nodes to use point-to-point communication over the IBM SP2’s interconnecting network, further in view of Trefethen’s teaching that mathematical expression evaluation results are asynchronously communicated from a first MATLAB process on a first processor to a second MATLAB process on a second processor, renders obvious this limitation. Ex.1003, ¶¶356-359.

[25.3] 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;

See [1.1.3], *supra*. Ex.1003, ¶360.

[25.4.0] wherein the plurality of nodes comprises:

See [25.1.0], *supra*. Ex.1003, ¶361.

[25.4.1] 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,

See [1.3.1], *supra*. Ex.1003, ¶362.

[25.4.2] 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

See [1.3.2], *supra*. Ex.1003, ¶363.

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

See [1.4.1], supra. Ex.1003, ¶364.

[25.5.2] *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;*

See [1.4.2], supra. Ex.1003, ¶365.

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

See [1.5.1], supra. Ex.1003, ¶366.

[25.6.2] *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;*

See [1.5.2], supra. Ex.1003, ¶367.

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

See [1.6], supra. Ex.1003, ¶368.

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

See [25.1.0], supra. Ex.1003, ¶369.

[25.8.1] *accept user instructions;*

See [1.7.1], supra. Ex.1003, ¶370.

[25.8.2] *after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and*

See [1.7.2], supra. Ex.1003, ¶371.

[25.8.3] *after communicating at least some of the user instruction using the communications network interface, communicate at least some of the user instructions to one or more since-node kernels.*

Petitioners have treated “since-node kernels” as “single-node kernels.”

Limitation [25.8.3] is rendered obvious for the reasons presented for [1.7.3].

Ex.1003, ¶¶372-373.

Claim 27

[27.0] *The computer cluster of claim 25, wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface.*

See [2.0], supra. Ex.1003, ¶374.

Claim 28

[28.0] *A computer cluster comprising:*

See [1.0] and [25.0], supra. Ex.1003, ¶375.

[28.1.1] *a plurality of nodes,*

See [1.1.1] and [25.1.1], supra. Ex.1003, ¶376.

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

See [1.1.2] and [25.1.2], supra. Ex.1003, ¶377.

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

See [10.0], *supra*. Ex.1003, ¶378.

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

See [25.1.3], *supra*. Ex.1003, ¶379.

[28.2] a communications network interface for the nodes to communicate results of mathematical expression evaluation with each other,

See [25.2], *supra*. Ex.1003, ¶380.

[28.3] wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface;

See [27.0], *supra*. Ex.1003, ¶381.

[28.4] 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

See [1.1.3], *supra*. Ex.1003, ¶382.

[28.5.0] wherein the plurality of nodes comprises:

See [1.1.1], *supra*. Ex.1003, ¶383.

[28.5.1] 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,

See [1.3.1], *supra*. Ex.1003, ¶384.

[28.5.2] *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*

See [1.3.2], supra. Ex.1003, ¶385.

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

See [1.4.1], supra. Ex.1003, ¶386.

[28.6.2] *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;*

See [1.4.2] and [25.5.2], supra. Ex.1003, ¶387.

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

See [1.5.1] and [25.6.1], supra. Ex.1003, ¶388.

[28.7.2] *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*

See [1.5.2] and [25.6.2], supra. Ex.1003, ¶389.

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

See [25.7], supra. Ex.1003, ¶390.

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

See [28.1.0], supra. Ex.1003, ¶391.

[28.9.1] *accept user instructions;*

See [1.7.1] and [25.8.1], supra. Ex.1003, ¶392.

[28.9.2] *after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and*

See [1.7.2] and [25.8.2], *supra*. Ex.1003, ¶393.

[28.9.3] *after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to one or more single-node kernels.*

See [1.7.3] and [25.8.3], *supra*. Ex.1003, ¶394.

Claim 30

[30.0] *A computer cluster node for evaluating expressions in parallel with other computer cluster nodes, the computer cluster node comprising:*⁶

Menon teaches MultiMATLAB implemented on the IBM SP2 to execute “parallel programs on ... a networked cluster.” Ex.1005, 6; Ex.1008, 1. That is, the MATLAB processes of MultiMATLAB executing commands in parallel are a group of processor nodes communicating with each other to accomplish a

⁶ In addition to the arguments presented here, “*computer cluster node*” lacks patentable weight because it does not confer any structure on the claimed “*computer cluster 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.”).

mathematical computation as if they were a single computer. Ex.1005, 3, 6;

Ex.1008, 1.⁷

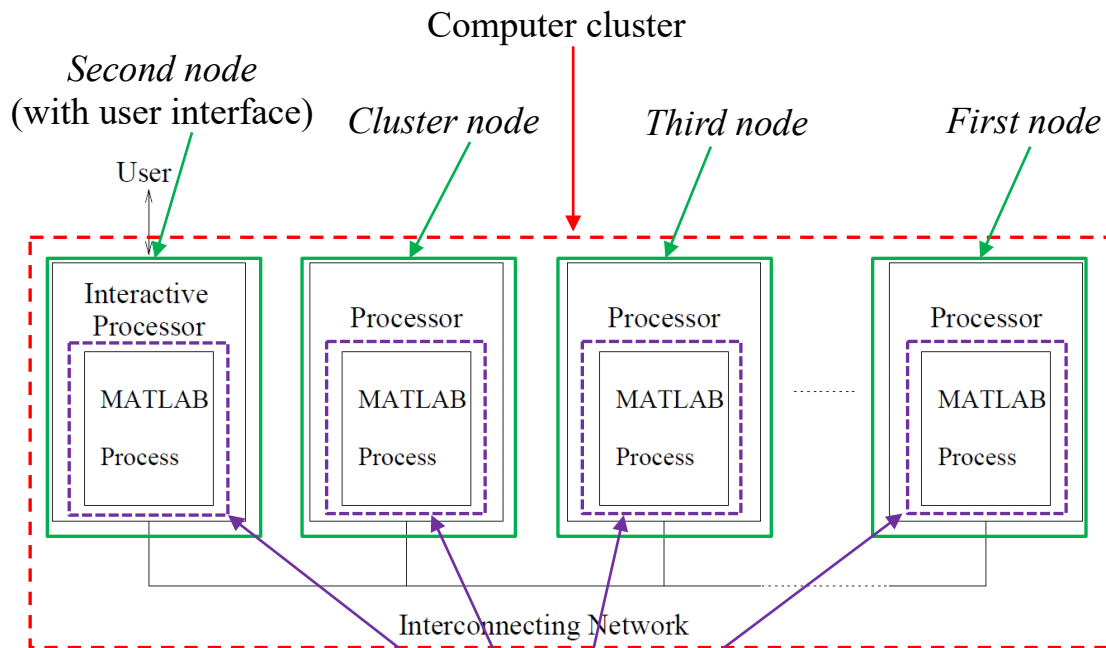


Figure 1: MultiMATLAB Architecture

Each processor node has a MATLAB process

Ex.1005, FIG. 1 (annotated); Ex.1003, ¶397

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

⁷ The functionality recited for the “first,” “second,” and “third” nodes of claim 30 is different than the functionality recited for the similarly named “first,” “second,” and “third” nodes of claim 1.

Accordingly, Menon's processor nodes that evaluate mathematic expressions in parallel on an IBM SP2 multiprocessor system, renders obvious this limitation. Ex.1003, ¶¶395-399.

[30.1.1] 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,

See [1.3.1], [1.3.2], and [1.6], *supra*. Ex.1003, ¶¶—400-415.

[30.1.2] wherein the hardware processor comprises multiple processor cores;

See [1.4.1], *supra* (each node has a symmetric multiprocessor). Ex.1003, ¶416.

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

As discussed in [1.1.2], “all MATLAB process are started via POE, IBM's Parallel Operating Environment.” Ex.1005, 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.

Menon further describes that a set of components (as discussed in [3.0]) on each processor node is responsible for the underlying communication layer with other processors (*user connection interface*). Ex.1005, 3.

Accordingly, Menon teaches a user connection interface that receives the initialization command and process id, rendering obvious this limitation. Ex.1003, ¶¶417-421.

[30.3] *a communications network interface to communicate results of evaluation with other computer cluster nodes using a peer-to-peer architecture; and*

See [1.2], *supra*. Ex.1003, ¶¶422-431.

[30.4] *wherein the communications network interface comprises at least one of: a local bus, a local area network (LAN), a wide area network (WAN), a storage area network (SAN), a TCP/IP interface over Ethernet, or a wireless network interface; and*

See [28.3], *supra*. Ex.1003, ¶432.

[30.5.1] *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;*⁸

As discussed in [1.3.0] and [1.3.1], 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.

⁸ 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.

As discussed in [1.3.2], Menon describes an interactive processor node (*second 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*), (iii) the interactive processor (*second node*) has a user interface, (iv) 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 for execution, rendering obvious this limitation. Ex.1003, ¶¶433-440.

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

See [1.4.1]-[1.4.2], *supra*. Ex.1003, ¶¶441-443.

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.”).

[30.5.3.1] *communicate a result of the first mathematical expression evaluation to a third node comprising a third hardware processor with a plurality of processing cores,*

See [1.4.2] and [1.5.1], *supra*. Ex.1003, ¶¶444-451.

[30.5.3.2] *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;*

See [1.5.1]-[1.5.2], *supra*. 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 30). 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, ¶¶452-456.

[30.6] *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 [30.2] (and [3.0]), Menon discloses the set of components as the *user connection interface*. Menon teaches that the MultiMATLAB interface module in the set of components is responsible for the underlying communication layer with other processors. Ex.1005, 3.

As discussed in [1.5.2], 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” (a component of the user connection interface), which is responsible for the underlying communication layer, in view of Trefethen’s teaching that each MATLAB process transmits its mathematical results back to the processor with the user interface, renders obvious this limitation. Ex.1003, ¶¶457-461.

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

See [30.0], *supra*. Ex.1003, ¶462.

[30.7.1] *accept user instructions;*

See [1.7.1], *supra*. Ex.1003, ¶463.

[30.7.2] *after accepting user instructions, communicate at least some of the user instructions using the communications network interface for the nodes to communicate with each other; and*

See [1.7.2], *supra*. Ex.1003, ¶¶464.

[30.7.3] *after communicating at least some of the user instructions using the communications network interface, communicate at least some of the user instructions to the single-node kernel.*

See [1.7.3], *supra*. Ex.1003, ¶¶465.

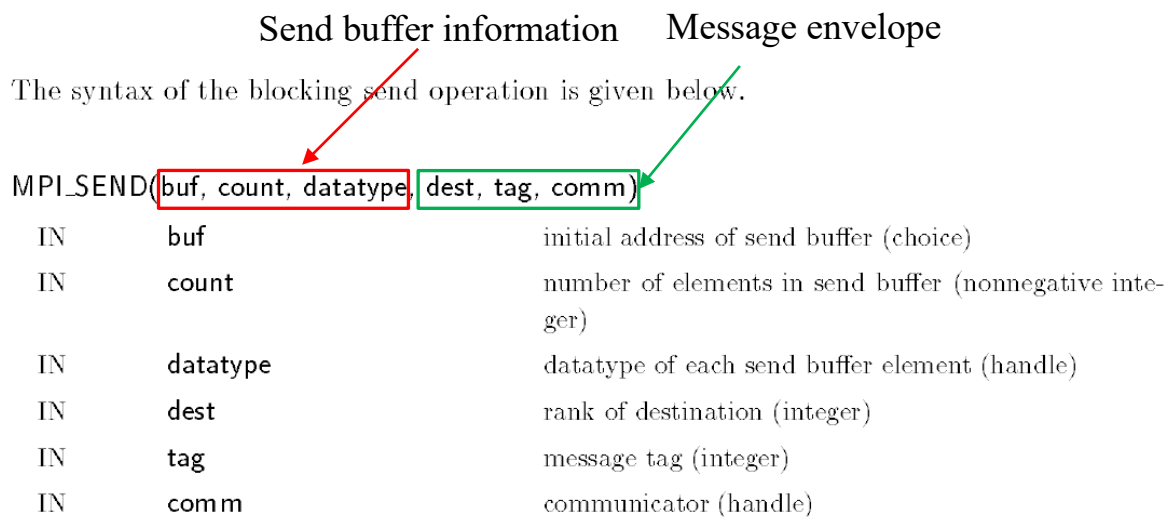
D. Ground 2: Claims 26 and 29 are obvious over Menon in view of Trefethen, RS6000, and POEref, further in view of MPIref

9. 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.” Ex.1017, ii. MPI allows the MATLAB processes of MultiMATLAB 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**.” Ex.1017, 15-16. MPIref also discloses a “destination” field to specify the “message destination.” Ex.1017, 18. (emphases original).



Ex.1017, 16 (annotated); Ex.1003, ¶85.

MPIref discloses that processes receive a message “with the **receive** operation MPI_RECV,” where “the message data is stored into the **receive buffer**.” Ex.1017, 16 (emphases original). MPIref also discloses a “source” field. Ex.1017, 19.

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). Ex.1003, ¶¶126-128.

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 '034 patent and Menon, namely, the “the field of cluster computing” and is reasonably pertinent to the problem of the '034 patent, namely, executing a single program on multiple nodes. Ex.1001, 1:14-15; Ex.1003, ¶¶467-469.

b) *Motivation to combine*

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”). Ex.1003, ¶¶470-471.

Menon discloses using MPI as the “underlying communication substrate.” Ex.1005, 3, 5. Further, Menon describes that a POSITA would look to use the MPI standard 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; Ex.1003, ¶470.

The combination MPIref with Menon would have been obvious because Menon and MPIref together represent a combination of known prior art elements (Menon describing an implementation of MultiMATLAB using MPI as its communication layer and MPIref describing “all the technical features” of MPI) to yield the predictable result of an implementation of MultiMATLAB utilizing the features of MPI disclosed by MPIref. Ex.1005, 3-10; Ex.1017, ii; Ex.1003, ¶472.

The results 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 MATLAB processes of MultiMATLAB, and MPIref provided a “practical, portable, efficient and flexible standard for message passing.” Ex.1005, 3; Ex.1017, 2. Ex.1003, ¶473.

Claim 26

[26.0] *The computer cluster of claim 25, wherein the asynchronous calls comprise a first command to create a first packet containing:*

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

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

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

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

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

As discussed in [3.0], Menon discloses that each node includes a set of components to communicate with other modules: “MEX Routines,” “MultiMATLAB interface module,” “indirection table” and “communication layer” (collectively, the *local cluster node module*). Ex.1005, 3, 5.

As discussed in [1.2], Menon describes 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.

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 MPI_SEND 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) to forward the data to the destination node.

Accordingly, Menon's description of Send command, 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, ¶¶474-503.

Claim 29

[29.0] *The computer cluster of claim 28, wherein each of the single-node kernels are configured to call a send command involving creating a packet containing:*

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

[29.2] *where the expression should be sent;*

[29.3] *wherein, upon reception of the send command by a local cluster node module associated with the single-node kernel, the local cluster node module is*

configured to decode the packet and forward a payload to the cluster node module specified in the packet;

[29.4] wherein each of the single-node kernels is configured to call a receipt command involving creating a packet specifying which message to test for completion and to then wait for a reply expression to evaluate;

[29.5] wherein, upon reception of the receipt command by the local cluster node module associated with the single-node kernel, the local cluster node module is configured to decode the packet and use a message specifier to search for any matching expressions listed as completed in a received message queue;

[29.6] wherein, upon determining that such a completed expression is found, the local cluster node module is configured to send the completed expression to a local single-node kernel associated with the cluster node module in response to the receipt command, and

[29.7] wherein each of the single-node kernels is configured to receive the completed expression and to update variables used by the single-node kernel during evaluation of expressions.

As discussed in claim 28, Menon and MPI describe how the Send() command creates a packet of information (pid value and data value) that is decoded by the MEX routine and mapped to the corresponding MPI call for forwarding to the destination node.

Menon also that 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 describes how the MPI_RECV call uses the information

(i.e., the pid value in the `recv()` command) to wait for and receive the incoming data. Ex.1017, 15-21.

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. Correspondingly, MPIref discloses that the `MPI_TEST` call uses a request handle (message specifier) to find the corresponding message to determine if the operation identified by the request (such as a send or receive) is complete. Ex.1017, 40.

As discussed in [1.3.0]-[1.5.2], Menon and Trefethen describe how MultiMATLAB can be configured using user instructions that cause: (i) process ID=0 to create a variable “a” with a value and send the variable to the next MATLAB process, and (ii) the middle MATLAB processes to wait to receive variable “a” from the previous MATLAB process, double the value of received variable “a”, and send the result to the next MATLAB process. Ex.1006, 6. Ex.1003, ¶¶504-544.

XII. SOTERA STIPULATION

Petitioners each have 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 '034 patent that was raised or could have reasonably been raised in this IPR. Ex.1042; Ex.1046.

XIII. CONCLUSION

Petitioners have established a reasonable likelihood that the challenged claims are unpatentable.

Respectfully submitted,

Dated: April 29, 2025
HAYNES AND BOONE, LLP
2801 N. Harwood Street, Suite 2300
Dallas, Texas 75201
Customer No. 27683

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

XIV. MANDATORY NOTICES

A. Real party-in-interest

Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioners certify that the real parties-in-interest are Intel Corporation and Advanced Micro Devices, Inc.

B. Related matters

Pursuant to 37 C.F.R. § 42.8(b)(2), to the best knowledge of Petitioners, the '034 patent is or was involved in the following cases:

Case Caption	Number	Court	Filed	Status
<i>Advanced Cluster Systems, Inc. v. Intel Corporation</i>	7:24-cv-00245	W.D. Tex.	9/26/24	Pending
<i>Advanced Cluster Systems, Inc. v. Advanced Micro Devices, Inc.</i>	7:24-cv-00244	W.D. Tex.	9/26/24	Pending

To the best knowledge of Petitioners, patents related to the '034 patent are or were involved in the following cases:

<i>Advanced Cluster Systems, Inc. v. NVIDIA Corporation</i>	1:19-cv-02032	D. DE	10/28/19	Dismissed
<i>NVIDIA Corporation v. Advanced Cluster Systems, Inc.</i>	IPR2021-00019	PTAB	10/6/20	Institution Denied
<i>NVIDIA Corporation v. Advanced Cluster Systems, Inc.</i>	IPR2021-00020	PTAB	10/10/20	Institution Denied

<i>Intel Corp. v. Advanced Cluster Systems, Inc.</i>	IPR2025-00794	PTAB	3/28/25	Pending
<i>Intel Corp. v. Advanced Cluster Systems, Inc.</i>	IPR2025-00795	PTAB	3/28/25	Pending
<i>Advanced Micro Devices, Inc. v. Advanced Cluster Systems, Inc.</i>	IPR2025-00862	PTAB	4/16/25	Pending
<i>Advanced Micro Devices, Inc. v. Advanced Cluster Systems, Inc.</i>	IPR2025-00863	PTAB	4/16/25	Pending

C. Lead and back-up counsel and service information

Lead Counsel

Andrew S. Ehmke
HAYNES AND BOONE, LLP
2801 N. Harwood Street, Suite 2300
Dallas, TX 75201

Phone: 214-651-5116
Fax: 214-200-0853
andy.ehmke.ipr@haynesboone.com
USPTO Reg. No. 50,271

Back-up Counsel

David L. McCombs
HAYNES AND BOONE, LLP
2801 N. Harwood Street, Suite 2300
Dallas, TX 75201

Phone: 972-651-5533
Fax: 214-200-0853
david.mccombs.ipr@haynesboone.com
USPTO Reg. No. 32,271

Dan Smith
HAYNES AND BOONE, LLP
2801 N. Harwood Street, Suite 2300
Dallas, TX 75201

Phone: 972-739-8634
Fax: 214-200-0853
dan.smith.ipr@haynesboone.com
USPTO Reg. No. 71,278

Dagim Tilahun
HAYNES AND BOONE, LLP
888 16th Street NW, Suite 300
Washington, DC 20006-4103

Phone: 202-654-4582
Fax: 214-200-0853
dagim.tilahun.ipr@haynesboone.com
USPTO Reg. No. 75,722

Brian E. Ferguson
WINSTON & STRAWN LLP
1901 L Street, N.W.
Washington, D.C. 20036

Phone: 202-282-5200
beferguson@winston.com
AMD-IPR@winston.com
USPTO Reg. No. 36,801

Chaoxuan Charles Liu
WINSTON & STRAWN LLP
2121 North Pearl Street, Ste. 900
Dallas, TX 75201

Phone: 214-453-6500
USPTO Reg. No. 76,616
ccliu@winston.com

James Kappos
WINSTON & STRAWN LLP
2121 North Pearl Street, Ste. 900
Dallas, TX 75201

Phone: 214-453-6500
USPTO Reg. No. 78,587
jkappos@winston.com

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

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioners hereby certify, 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 13,998. 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: April 29, 2025

/Andrew S. Ehmke/

Andrew S. Ehmke

Lead Counsel

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 April 29, 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. 11,570,034;
Petitioners' Powers of Attorney;
Petitioners' Exhibit List; and
Exhibits 1001-1049.

Persons served Knobbe, Martens, Olson & Bear, LLP
2040 Main Street, Fourteenth Floor
Irvine, CA 92614

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