IPR2025-00913 (US11,570,034)
IPR2025-00914 (US12,021,679)
IPR2025-00915 (US11,563,621)
IPR2025-00916 (US11,811,582)

Intel Corp. v. Advanced Cluster Systems, Inc.

Andrew S. Ehmke, Haynes and Boone, LLP
Daniel D. Smith, Haynes and Boone, LLP
Attorneys for Petitioner Intel Corporation

DEMONSTRATIVE EXHIBIT - NOT EVIDENCE

Running MATLAB on multiple computers

The Patents

Field

The present disclosure relates to the field of cluster computing generally and to systems and methods for adding cluster computing functionality to a computer program, in particular.

-00913 Proceeding, Ex.1001, 1:15-18

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

For purposes of illustration, some embodiments are described herein in the context of cluster computing with Mathematica software. The present disclosure is not limited to a single software program; the systems and methods can be used with other application software such as, for example, Maple®, MATLAB®, MathCAD®, Apple

-00913 Proceeding, Ex.1001, 4:16-24

Prior Art

Menon

MATLAB is the most popular scientific computing environment available on uniprocessors today. Unfortunately, no such environment is currently available for multiprocessors. MultiMATLAB [1] is a general extension of the MATLAB environment to any distributed memory multiprocessors. This paper presents a new MultiMATLAB system designed to provide high-performance on multiprocessors while maintaining the functionality and usability of the MATLAB environment. This system will enable users

Ex.1005, 1; -00913 Proceeding, Petition, 32

Trefethen

MATLAB®, a commercial product of The MathWorks, Inc., has become one of the principal languages of desktop scientific computing. A system is described that enables one to run MATLAB conveniently on multiple processors. Using short, MATLAB-style commands like Eval, Send, Recv, Bcast, Min, and Sum, the user operating within one MATLAB session can start MATLAB processes on other machines and then pass commands and data between between these various processes in a fashion that maintains MATLAB's traditional user-friendliness. Multi-processor graphics is also supported. The system currently runs under

Ex.1006, 1; -00913 Proceeding, Petition, 23

Discussion Summary

	‘034 Patent IPR2025-00913	‘679 Patent IPR2025-00914	‘621 Patent IPR2025-00915	‘582 Patent IPR2025-00916
“Hardware Processor”	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			X	
“without first passing through the first kernel”			X	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	X	X	X	X
Printed publications	X	X	X	X

Discussion Summary

	'034 Patent IPR2025-00913	'679 Patent IPR2025-00914	'621 Patent IPR2025-00915	'582 Patent IPR2025-00916
"Hardware Processor"	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			X	
"without first passing through the first kernel"			X	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	X	X	X	X
Printed publications	X	X	X	X

‘034 Patent:
Claim language for “hardware processor”

‘034 Patent

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

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

-00913 Proceeding, Ex.1001, 29:52-67

Each processor can contain one or more independent microprocessors

'034 Patent

nications architecture such as a local bus (not shown). A computer system can include one or more processors, and each processor can contain one or more processor cores that are capable of single-threaded execution. Processor cores are generally independent microprocessors, but more than one can be included in a single chip package. Software code designed for single-threaded execution can generally run on one processor core at a time. For example, single-threaded software code typically does not benefit from multiple processor cores in a computer system.

FIG. **2** is a block diagram showing relationships among software modules running on one embodiment of a computer cluster **100**. In the embodiment shown in FIG. **2**, the kernel modules **206***a-e* are designed for single-threaded execution. For example, if each of the processors **112***a*, **112***b*, **122***a*, **122***b*, **132** shown in FIG. **1** includes only one processor core, two kernel modules (for example, kernel modules **206***a*, **206***b*) loaded into the memory **114** of computer system **110** could exploit at least some of the processing bandwidth of the two processors **112***a*, **112***b*. Similarly, two kernel modules **206***c*, **206***d* loaded into the memory **124** of computer system **120** could exploit at least some of the processing bandwidth of the two processors **122***a*, **122***b*. Likewise, the bandwidth of processor **132** of computer system **130** could be utilized by a single instance of a cluster node module **204***e* loaded into the computer system's memory **134**.

Ex.1001, 5:11-37; -00913 Proceeding, Reply at 2.

A processor can be one or more microprocessors

'034 Patent

1. Processors **112***a-b*

In one embodiment, the computer system **110** includes one or more processors **112***a-b*. The processors **112***a-b* can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA® processor, etc. In addition, one or more of the processors

Ex.1001, 7:14-25; -00913 Proceeding, Reply at 4.

A processor can be microprocessors

‘034 Patent

In one embodiment, the computer system **130** includes a processor **132**. The processor **132** can be a general purpose single-core or multi-core microprocessors such as a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, or an ALPHA® processor. In addition, the processor **132** can be

Ex.1001, 9:51-60; -00913 Proceeding, Reply at 5

A ***processor*** can be ***one or more microprocessors***

'034 Patent

computer system can include one or more processors, and each processor can contain one or more processor cores that are capable of single-threaded execution. Processor cores are generally independent microprocessors, but more than one can be included in a single chip package. Software code

Ex.1001, 5:11-16; -00913 Proceeding, Reply at 2

"A processor can contain one or more … independent microprocessors."

In one embodiment, the computer system **110** includes one or more processors **112***a-b*. The processors **112***a-b* can be one or more general purpose single-core or multi-core microprocessors such as, for example, a Pentium® processor, a Pentium® II processor, a Pentium® Pro processor, a Pentium® III processor, Pentium® 4 processor, a Core Duo® processor, a Core 2 Duo® processor, a Xeon® processor, an Itanium® processor, a Pentium® M processor, an x86 processor, an Athlon® processor, an 8051 processor, a MIPS® processor, a PowerPC® processor, an ALPHA®

Ex.1001, 7:15-24; -00913 Proceeding, Reply at 4

"The processors[] can be one or more general purpose single-core or multi-core microprocessors."

In one embodiment, the computer system **130** includes a processor **132**. The processor **132** can be a general purpose single-core or multi-core microprocessors such as a Pentium® processor, a Pentium® II processor, a Pentium® Pro

Ex.1001, 9:51-54; -00913 Proceeding, Reply at 5

"The processor[] can be … single-core or multi-core microprocessors."

-00913 Proceeding, Reply at 6

RS6000 teaches a hardware processor

Petition (citing RS6000)

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

Ex.1007, 1-2; -00913 Proceeding, Petition, 29

RS6000 teaches that each hardware processor has a plurality of cores

Petition (citing RS6000)

> 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*.

Ex.1007, 2, 9; -00913 Proceeding, Petition, 43

Patent Owner Asserts That "Hardware Processor" Must Be Only One Microprocessor

Patent Owner Response

> As Dr. Smith explains, a person of ordinary skill in the art would understand that the claim language—"a hardware processor with a plurality of processing cores"—refers to a single processor package containing multiple cores. (Ex. 2017 ¶¶ 26, 33.) That is the ordinary usage of the term "multi-core processor" at the time of the invention. (*Id.*) It is not how symmetric multi-processing (SMP) is described.

-00913 Proceeding, Patent Owner Response, 15

The ‘034 specification describes a multi-core “single chip package” as just one example of a hardware processor

nications architecture such as a local bus (not shown). A computer system can include one or more processors, and each processor can contain one or more processor cores that are capable of single-threaded execution. Processor cores are generally independent microprocessors, but more than 15 one can be included in a single chip package. Software code

Ex.1001, 5:11-37; -00913 Proceeding, Reply at 2.

- The ‘034 specification also describes “cores” as “units capable of single-threaded execution”

digital signal processor. The total number of processing cores (for example, processing units capable of single-threaded execution) within all processors **112***a-b* in the

Ex.1001, 7:27-29; -00913 Proceeding, Reply, 12-13

The Prior Art Teaches a Multi-core Single Chip Package

- RS6000 uses POWER3 Microprocessors

RS/6000 SP system overview

The IBM RS/6000 SP is IBM's family of scalable, parallel computing solutions. It provides a state-of-the-art parallel computing system and industry-leading application enablers and applications. The RS/6000 SP runs the AIX operating system along with the Parallel System Support Programs (PSSP) system software on the control workstation and all processor nodes. For complete information on SP system software issues, see *Planning Volume 2, Control Workstation and Software Environment*.

The scalable architecture of the SP system, its high-performance communication, PowerPC and POWER3 processors give you the power to handle data-intensive, compute-intensive and I/O-intensive jobs with ease.

Ex.1007, 1; -00913 Proceeding, Petition, 21-22

- The RS/6000 SP Inside Out: POWER3 is a single chip package that contains 8 execution units

POWER3 Microprocessor

The POWER3 is a single chip implemented with 0.25 micron CMOS technology. It operates at 200 MHz clock cycle. The POWER3 design contains a superscalar core that is comprised of eight execution units,

Ex.1010, 57; -00913 Proceeding, Reply at 12-13; -00913 Proceeding, Petition, 21-22

Discussion Summary

	'034 Patent IPR2025-00913	'679 Patent IPR2025-00914	'621 Patent IPR2025-00915	'582 Patent IPR2025-00916
"Hardware Processor"	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			X	
"without first passing through the first kernel"			X	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	X	X	X	X
Printed publications	X	X	X	X

The 1->2->3->1 Sequence of Operation

- 1->2->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, the first single-node kernel configured to interpret user instructions and distribute calls to at least one of a plurality of other nodes for execution; and

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

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

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

-00913 Proceeding, Ex.1001 (claim 1)

- The specification contemplates "instructions" and "scripts"

D. Kernel Module **206**

A kernel module **206** typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language. In

Ex.1001, 23:1-5; -00913 Proceeding, Reply, 19

- The specification contemplates "parallel evaluation" using "for example, … the same operations"

Common Divide-and-Conquer Parallel Evaluation

In one embodiment, the advanced functions module **304** includes functions providing for basic parallelization such as, for example, routines that would perform the same operations on many data elements or inputs, stored on many nodes. These functions can be compared to parallelized

Ex.1001, 17:1-6; -00913 Proceeding, Reply, 21

Menon's MultiMATLAB node architecture

Menon

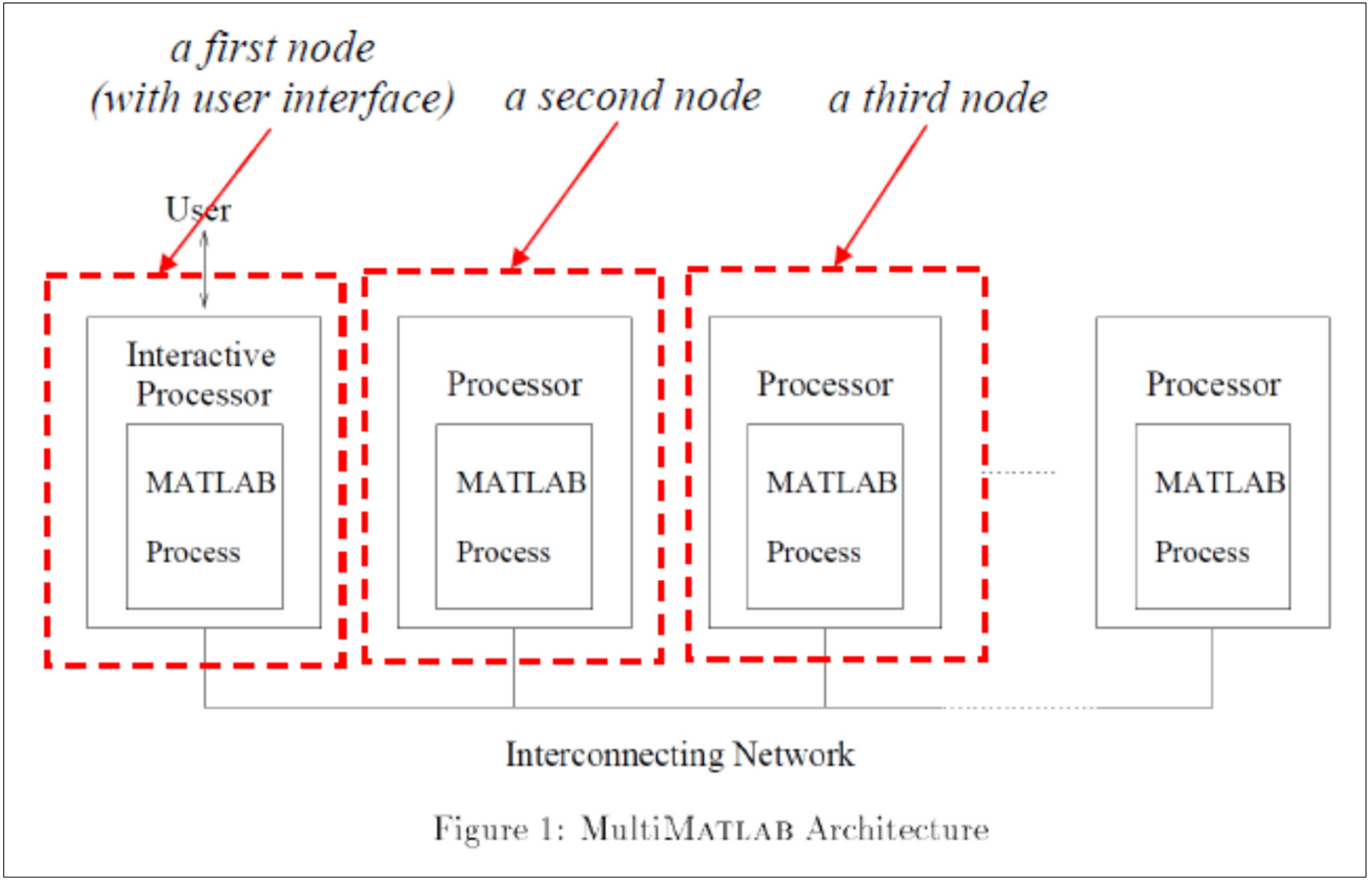


Figure 1: MultiMATLAB Architecture

Ex.1005, 3, 6, FIG.1; -00913 Proceeding, Petition 39-49; Ex.1003, ¶217

Menon describes using a peer-to-peer architecture

Menon

5.2 SPMD/Message Passing

In order to accommodate applications requiring a finer degree of communication, we provide a set of MATLAB message passing routines analogous to those in MPI. In particular, we provide point to point and collective communication MATLAB routines operating directly on MATLAB data structures. We list a subset of these routines in Table 1.

Ex.1005, 9; -00913 Proceeding, Petition, 36

SPMD Communication

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

Ex.1005, 9; -00913 Proceeding, Petition, 37; Ex.1003, ¶189

Menon describes sending data from one process to another

Menon

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

Ex.1005, 9; -00913 Proceeding, Petition, 37; Ex.1003, ¶189

Trefethen's MultiMATLAB example teaches the 1->2->3->1 operation

Trefethen

SPMD programs can be built upon `Send` and `Recv` commands. Typically the program contains `if` and `else` commands that specify different actions for different processes. For example, suppose the m-file `cycle.m` consists of the following program:

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

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

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

Ex.1006, 6; Ex.1003, ¶190; -00913 Proceeding, Petition, 38; Ex.1163, ¶44

Trefethen's MultiMATLAB example teaches the 1->2->3->1 operation

SPMD Communication using Send and Recv

SPMD programs can be built upon `Send` and `Recv` commands. Typically the program contains `if` and `else` commands that specify different actions for different processes. For example, suppose the m-file `cycle.m` consists of the following program:

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

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

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

Ex.1006, 6; Ex.1003, ¶190; -00913 Proceeding, Petition, 38; Ex.1163, ¶44

MultiMATLAB teaches the 1->2->3->1 operation

"1->2"

Trefethen

SPMD programs can be built upon `Send` and `Recv` commands. Typically the program contains `if` and `else` commands that specify different actions for different processes. For example, suppose the m-file `cycle.m` consists of the following program:

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

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

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

Ex.1006, 6; Ex.1003, ¶190; -00913 Proceeding, Petition, 38; Ex.1163, ¶44

Menon

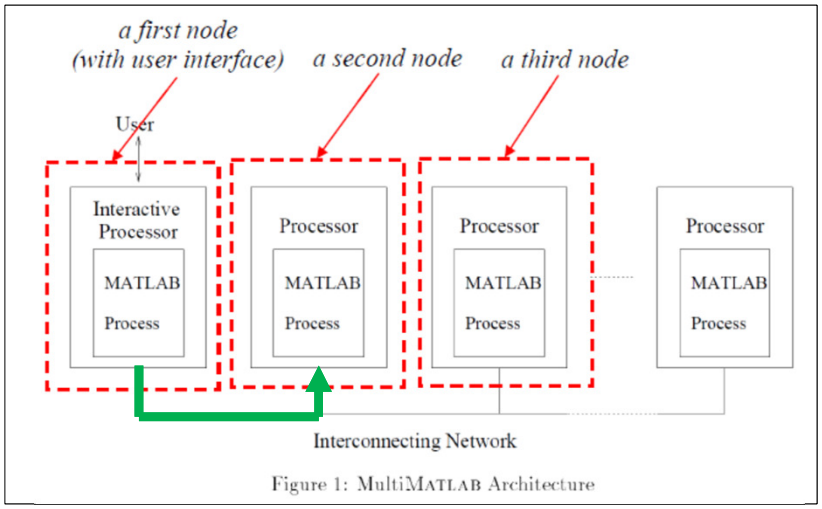


Figure 1: MultiMATLAB Architecture

Ex.1005, 3, 6, FIG.1; -00913 Proceeding, Petition 39-49; Ex.1003, ¶217

MultiMATLAB teaches the 1->2->3->1 operation

"2->3"

Trefethen

SPMD programs can be built upon `Send` and `Recv` commands. Typically the program contains `if` and `else` commands that specify different actions for different processes. For example, suppose the m-file `cycle.m` consists of the following program:

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

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

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

Ex.1006, 6; Ex.1003, ¶190; -00913 Proceeding, Petition, 38; Ex.1163, ¶44

Menon

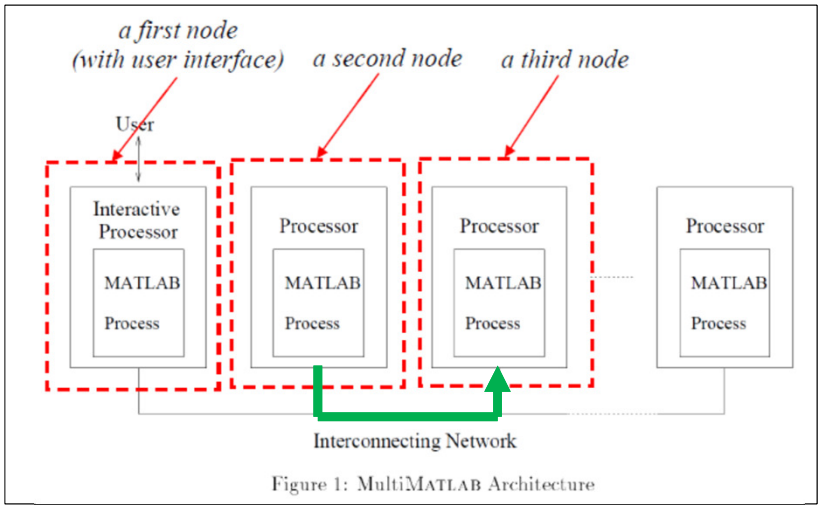


Figure 1: MultiMATLAB Architecture

Ex.1005, 3, 6, FIG.1; -00913 Proceeding, Petition 39-49; Ex.1003, ¶217

MultiMATLAB teaches the 1->2->3->1 operation

"3->1"

Trefethen

SPMD programs can be built upon `Send` and `Recv` commands. Typically the program contains `if` and `else` commands that specify different actions for different processes. For example, suppose the m-file `cycle.m` consists of the following program:

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

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

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

Ex.1006, 6; Ex.1003, ¶190; -00913 Proceeding, Petition, 38; Ex.1163, ¶44

Menon

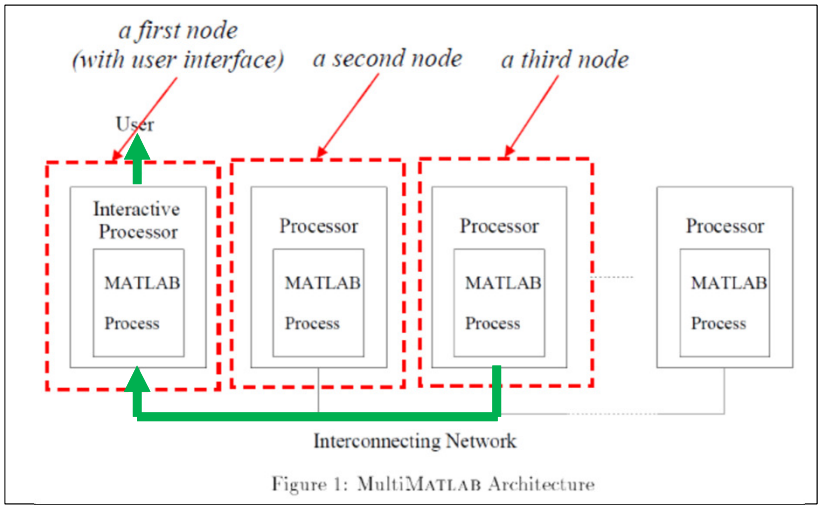


Figure 1: MultiMATLAB Architecture

Ex.1005, 3, 6, FIG.1; -00913 Proceeding, Petition 39-49; Ex.1003, ¶217

MultiMATLAB teaches the 1->2->3->1 operation

- Trefethen teaches the output is sent to the first node and is then plotted and displayed to the user

> The ordering of these numbers is arbitrary, since the processors are not synchronized and output is sent to the master process as soon as it is ready. (It is a good idea to type `Eval('format compact')` at the beginning to keep

Ex.1006, 4; -00913 Proceeding, Ex.1003, ¶221; Petition, 47

> In many applications, the user will find it most convenient to compute on multiple processors but produce plots on the master process, after sending data as necessary. Equally often, however, it may be desirable to produce plots in a distributed fashion that are then sent to the user's screen. This can be particularly useful when one wishes to monitor the progress of computations on several processors graphically.

Ex.1006, 7; -00913 Proceeding, Ex.1003, ¶227; Petition, 48

MultiMATLAB teaches the 1->2->3->1 operation

"1->2->3->1"

SPMD programs can be built upon `Send` and `Recv` commands. Typically the program contains `if` and `else` commands that specify different actions for different processes. For example, suppose the m-file `cycle.m` consists of the following program:

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

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

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

Ex.1006, 6; -00913 Proceeding, Ex.1003, ¶190; Petition, 38; Ex.1163, ¶44

Petitioner's Supporting Evidence

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTEL CORPORATION,
ADVANCED MICRO DEVICES, INC.,
Petitioners

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

DECLARATION OF CHANDRAJIT L. BAJAJ, PH.D., UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW

Ex.1003

CHANDRAJIT L. BAJAJ

Professor of Computer Sciences,
Computational Applied Mathematics Chair in Visualization,
Director, Center for Computational Visualization,
Institute for Computational Engineering and Sciences
The University of Texas at Austin
ACES 2.324, 201 East 24th Street, Austin, TX 78712

EDUCATION

1980 B.TECH. (Electrical Engineering), Indian Institute of Technology, Delhi
1983 M.S. (Computer Science), Cornell University
1984 Ph.D. (Computer Science), Cornell University

PROFESSIONAL EXPERIENCE

- Assistant Professor of Computer Sciences, Purdue University, 1984-89
- Associate Professor of Computer Sciences, Purdue University, 1989-93
- Visiting Associate Professor of Computer Sciences, Cornell University, 1990-91
- Professor of Computer Sciences, Purdue University, 1993-97
- Director of Image Analysis and Visualization Center, Purdue University, 1996-97
- CAM (Comp. Appd. Math) Chair of Visualization, University of Texas at Austin, 1997- Present
- Professor of Computer Sciences, University of Texas at Austin, 1997- Present
- Director of Computational Visualization Center, University of Texas at Austin, 1997- Present

Ex.1004

Petitioner's Supporting Evidence

- Dr. Bajaj discussed the knowledge of a POSITA

Ex.1011, 21. The "pipeline is one of the simplest and most popular functionality decomposition paradigms." Ex.1011, 21. FIG. 1.6 of Buyya below shows a schematic representation of the data pipeline structure.

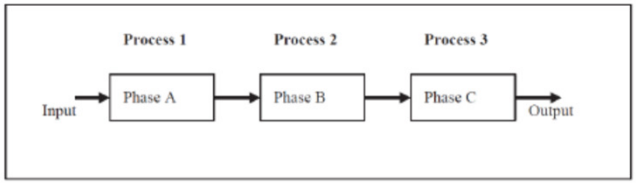


Figure 1.6 Data pipeline structure.

Ex.1011, FIG. 1.6

-00913 Proceeding, Ex.1003, ¶41 (citing Ex.1011); Petition, 9

Ex.1011, 53. FIG. 3.3 of Buyya below shows a peer-to-peer network comprising *n* cluster nodes where "a task communicates with all other tasks." Ex.1011, 53.

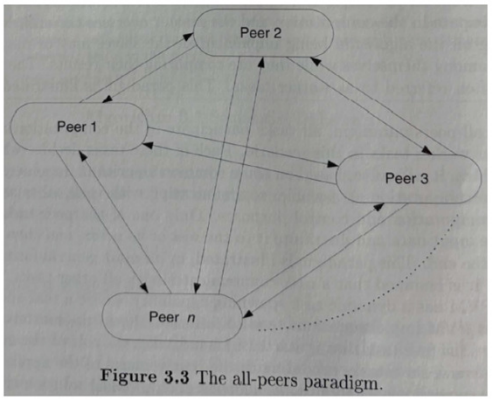
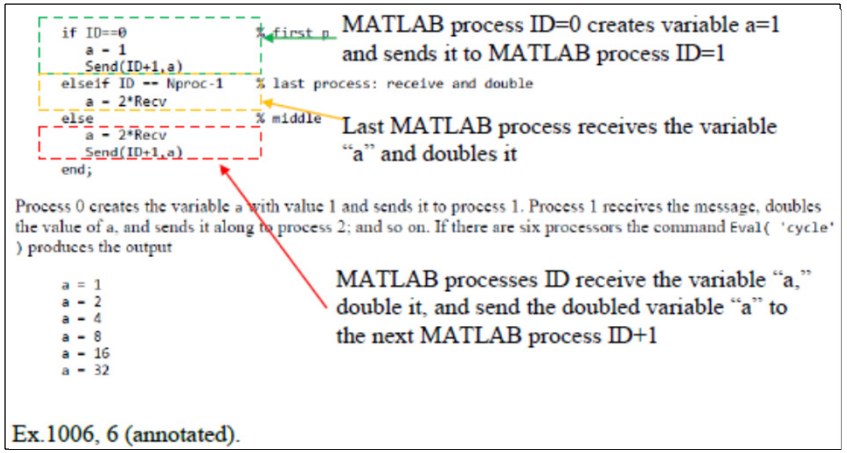


Figure 3.3 The all-peers paradigm.

Ex.1011, FIG. 3.3

-00913 Proceeding, Ex.1003, ¶44 (citing Ex.1011); Petition, 9

- Dr. Bajaj analyzed Menon and Trefethen



Ex.1006, 6 (annotated).

-00913 Proceeding, Ex.1003, ¶45 (citing Ex.1006); Petition, 9

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.

-00913 Proceeding, Ex.1003, ¶114 (citing Ex.1006); Petition, 21

Patent Owner's Evidence

D. Reliance On The Existing Evidentiary Record

Patent Owner does not submit a new expert declaration with this Response because it is not necessary for the Board to resolve the issues presented. The dispositive issues are resolved by the intrinsic record and the asserted references. To

-00913 Proceeding, Patent Owner Response, 8

Patent Owner's Arguments Lack Merit

Argument	Patent Owner's Position
"Master Controlled Dispatch"	process. (Ex. 1005, 3, 6.) That is a master-controlled dispatch model. It does not disclose that a second node is configured to send an intermediate evaluation result to a third node as part of an evaluation chain, much less that the third node is -00913 Proceeding, Patent Owner Response, 35
"Eval-Return / Further Use"	an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return -00913 Proceeding, Patent Owner Response, 24
"Prewritten Code"	29:52–30:4.) Instead, it depicts a prewritten program in which each process performs a fixed action—doubling an input—based solely on its ID. The fact that the behavior is fully scripted only underscores the failure of the Trefethen to disclose the claimed -00913 Proceeding, Patent Owner Response, 36
"Distinct Mathematical Expressions"	The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's -00913 Proceeding, Patent Owner Response, 37

Patent Owner's Arguments Lack Merit

Argument	Patent Owner's Position	Petitioner's Counterargument
"Master Controlled Dispatch"	process. (Ex. 1005, 3, 6.) That is a master-controlled dispatch model. It does not disclose that a second node is configured to send an intermediate evaluation result to a third node as part of an evaluation chain, much less that the third node is -00913 Proceeding, Patent Owner Response, 35	The Send() and Recv() functions send an intermediate result from one node to the next node. (see slides 17-28, 35)
"Eval-Return / Further Use"	an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return -00913 Proceeding, Patent Owner Response, 24	
"Prewritten Code"	29:52–30:4.) Instead, it depicts a prewritten program in which each process performs a fixed action—doubling an input—based solely on its ID. The fact that the behavior is fully scripted only underscores the failure of the Trefethen to disclose the claimed -00913 Proceeding, Patent Owner Response, 36	
"Distinct Mathematical Expressions"	The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's -00913 Proceeding, Patent Owner Response, 37	

Patent Owner's Arguments Lack Merit

Argument	Patent Owner's Position	Petitioner's Counterargument
"Master Controlled Dispatch"	process. (Ex. 1005, 3, 6.) That is a master-controlled dispatch model. It does not disclose that a second node is configured to send an intermediate evaluation result to a third node as part of an evaluation chain, much less that the third node is -00913 Proceeding, Patent Owner Response, 35	The Send() and Recv() functions send an intermediate result from one node to the next node. (see slides 17-28, 35)
"Eval-Return / Further Use"	an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return -00913 Proceeding, Patent Owner Response, 24	Menon and Trefethen teach returning the results to the node with the user interface and then displaying the results graphically. (see slides 25, 36-38)
"Prewritten Code"	29:52–30:4.) Instead, it depicts a prewritten program in which each process performs a fixed action—doubling an input—based solely on its ID. The fact that the behavior is fully scripted only underscores the failure of the Trefethen to disclose the claimed -00913 Proceeding, Patent Owner Response, 36	
"Distinct Mathematical Expressions"	The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's -00913 Proceeding, Patent Owner Response, 37	

Patent Owner's Arguments Lack Merit

Argument	Patent Owner's Position	Petitioner's Counterargument
"Master Controlled Dispatch"	process. (Ex. 1005, 3, 6.) That is a master-controlled dispatch model. It does not disclose that a second node is configured to send an intermediate evaluation result to a third node as part of an evaluation chain, much less that the third node is -00913 Proceeding, Patent Owner Response, 35	The Send() and Recv() functions send an intermediate result from one node to the next node. (see slides 17-28, 35)
"Eval-Return / Further Use"	an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return -00913 Proceeding, Patent Owner Response, 24	Menon and Trefethen teach returning the results to the node with the user interface and then displaying the results graphically. (see slides 25, 36-38)
"Prewritten Code"	29:52–30:4.) Instead, it depicts a prewritten program in which each process performs a fixed action—doubling an input—based solely on its ID. The fact that the behavior is fully scripted only underscores the failure of the Trefethen to disclose the claimed -00913 Proceeding, Patent Owner Response, 36	The specification describes user instructions and scripts. (see slides 16, 39-40)
"Distinct Mathematical Expressions"	The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's -00913 Proceeding, Patent Owner Response, 37	

Patent Owner's Arguments Lack Merit

Argument	Patent Owner's Position	Petitioner's Counterargument
"Master Controlled Dispatch"	process. (Ex. 1005, 3, 6.) That is a master-controlled dispatch model. It does not disclose that a second node is configured to send an intermediate evaluation result to a third node as part of an evaluation chain, much less that the third node is -00913 Proceeding, Patent Owner Response, 35	The Send() and Recv() functions send an intermediate result from one node to the next node. (see slides 17-28, 35)
"Eval-Return / Further Use"	an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return -00913 Proceeding, Patent Owner Response, 24	Menon and Trefethen teach returning the results to the node with the user interface and then displaying the results graphically. (see slides 25, 36-38)
"Prewritten Code"	29:52–30:4.) Instead, it depicts a prewritten program in which each process performs a fixed action—doubling an input—based solely on its ID. The fact that the behavior is fully scripted only underscores the failure of the Trefethen to disclose the claimed -00913 Proceeding, Patent Owner Response, 36	The specification describes user instructions and scripts. (see slides 16, 39-40)
"Distinct Mathematical Expressions"	The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's -00913 Proceeding, Patent Owner Response, 37	The specification describes the nodes as performing the "same operations … stored on many nodes". (see slides 16, 41-43)

Patent Owner's Arguments: "Master-Controlled Dispatch"

Patent Owner Response

> Menon describes a MultiMATLAB architecture in which a user interacts with an "interactive process" and other processes wait for commands from that interactive process. (Ex. 1005, 3, 6.) That is a master-controlled dispatch model. It does not disclose that a second node is configured to send an intermediate evaluation result to a third node as part of an evaluation chain, much less that the third node is

-00913 Proceeding, Patent Owner Response 35

- But, the prior art teaches sending intermediate results from node to node

Menon

SPMD Communication	
Send(pid,data)	Send data from one process to another.
Recv(pid)	Receive data sent from another process.

Ex.1005, 9; -00913 Proceeding, Petition, 37; Ex.1003, ¶189

Trefethen

> 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

Ex.1006, 6; -00913 Proceeding, Ex.1003, ¶190; Petition, 38; Ex.1163, ¶44

Patent Owner's Arguments: "Eval-Return" / "Return … data for further use"

Patent Owner Response

> may be "sent to the user's screen" (*id.* 48; Ex. 1006, 7). Those disclosures describe an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return the result of a second evaluation back to the first node and then to the user interface as claim 1 requires.

-00913 Proceeding, Patent Owner Response, 24

- But, the claim does not recite "further use"

> received result, and communicate the result of the second mathematical expression evaluation to the first node;
> wherein the first node is configured to return the result of the second mathematical expression evaluation to the user interface;

-00913 Proceeding, Ex.1001, 29:65 - 30:3 (claim 1)

Patent Owner's Arguments: "Eval-Return" / "Return … data for further use"

Patent Owner Response

> may be "sent to the user's screen" (*id.* 48; Ex. 1006, 7). Those disclosures describe an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return the result of a second evaluation back to the first node and then to the user interface as claim 1 requires.

-00913 Proceeding, Patent Owner Response, 24

- But, Menon teaches that the processes communicate with each other and that the user interacts with the interactive process

> form individually runs a MATLAB process. Each process is then provided with the ability to communicate with other processes through a communication layer that runs over the parallel platform's interconnection network. The user interacts directly with one MATLAB process, called the interactive process, and operates within that process's MATLAB environment. Other MATLAB processes await commands from the interactive process. These other

Ex.1005, 3; -00913 Proceeding, Petition, 48; Ex.1003, ¶227

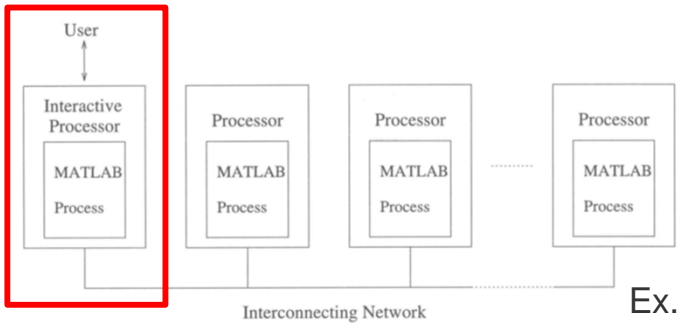


Figure 1: MultiMATLAB Architecture

Ex.1005, 3; -00913 Proceeding, Petition, 37; Ex.1003, ¶238

Patent Owner's Arguments: "Eval-Return" / "Return … data for further use"

Patent Owner Response

> may be "sent to the user's screen" (*id.* 48; Ex. 1006, 7). Those disclosures describe an interactive front end and a display/monitoring behavior. They do not describe any mechanism by which the *value* computed on a remote node is returned to the originating process as data for further use—much less a system configured to return the result of a second evaluation back to the first node and then to the user interface as claim 1 requires.

-00913 Proceeding, Patent Owner Response, 24

- But, Trefethen teaches the output is sent to the first node and is then plotted and displayed to the user

> The ordering of these numbers is arbitrary, since the processors are not synchronized and output is sent to the master process as soon as it is ready. (It is a good idea to type `Eval('format compact')` at the beginning to keep

Ex.1006, 4; -00913 Proceeding, Ex.1003, ¶221; Petition, 47

> In many applications, the user will find it most convenient to compute on multiple processors but produce plots on the master process, after sending data as necessary. Equally often, however, it may be desirable to produce plots in a distributed fashion that are then sent to the user's screen. This can be particularly useful when one wishes to monitor the progress of computations on several processors graphically.

Ex.1006, 7; -00913 Proceeding, Ex.1003, ¶227; Petition, 48

Patent Owner's Arguments: "Prewritten Program"/ "User Authored Code"

Patent Owner Response

> 29:52–30:4.) Instead, it depicts a prewritten program in which each process performs a fixed action—doubling an input—based solely on its ID. The fact that the behavior is fully scripted only underscores the failure of the Trefethen to disclose the claimed order of operations. Trefethen confirms that the system does not transmit evaluation

-00913 Proceeding, Patent Owner Response, 36

- But, the claim language does not exclude "prewritten programs"

> 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;

-00913 Proceeding, Ex.1001, 29:54-58 (claim 1)

- And the specification contemplates "instructions" and "scripts"

> D. Kernel Module **206**
>
> A kernel module **206** typically includes program code for interpreting high-level code, commands, and/or instructions supplied by a user or a script into low-level code, such as, for example, machine language or assembly language. In 5

-00913 Proceeding, Ex.1001, 23:1-5; Reply, 19

Patent Owner's Arguments: "Prewritten Program"/ "User Authored Code"

Patent Owner Response

> 29:52–30:4.) Instead, it depicts a prewritten program in which each process performs a fixed action—doubling an input—based solely on its ID. The fact that the behavior is fully scripted only underscores the failure of the Trefethen to disclose the claimed order of operations. Trefethen confirms that the system does not transmit evaluation

-00913 Proceeding, Patent Owner Response, 36

- But, Patent Owner has argued that "prewritten programs" are in the scope of the claims

> For example, Intel provides "HCCL Demo," which is "a program utilizing HCCL APIs, demonstrating both collective and point to point communication."[60] The source code for HCCL Demo[61] includes the details for using HCCL commands to initialize a HCCL communicator on a HLS-1 system comprising multiple Gaudi HL-205 Mezzanine card nodes, accessing program code for a user interface and for a first single-node kernel, for interpreting user instructions, for distributing calls to at least one of a plurality of other Gaudi HL-205 Mezzanine card nodes for execution, for receiving the result of mathematical expression from other nodes, for executing mathematical expression evaluations, and for communicating a result of mathematical expression evaluation to other nodes and to the user interface.

Patent Owner's Infringement Contentions, Ex. 1022, 46-47;
-00913 Proceeding, Reply, 19-20

Patent Owner's Arguments: "Distinct Mathematical Expression Evaluations"

Patent Owner Response

> The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's

-00913 Proceeding, Patent Owner Response, 37

- But, the claims do not preclude the same operation

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

-00913 Proceeding, Ex.1001, 29:52-67 (claim 1); Reply, 21

Patent Owner's Arguments: "Distinct Mathematical Expression Evaluations"

Patent Owner Response

> The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's

-00913 Proceeding, Patent Owner Response, 37

- But, the specification contemplates "parallel evaluation" using the "same operations"

> Common Divide-and-Conquer Parallel Evaluation
>
> In one embodiment, the advanced functions module **304** includes functions providing for basic parallelization such as, for example, routines that would perform the same operations on many data elements or inputs, stored on many nodes. These functions can be compared to parallelized

-00913 Proceeding, Ex.1001, 17:1-6; Reply, 21

Patent Owner's Arguments: "Distinct Mathematical Expression Evaluations"

Patent Owner Response

> The claim requires two distinct mathematical expression evaluations executed at two different nodes, with the first evaluation result being communicated to the third node and used in the second evaluation. (Ex. 1001, 29:52–30:4.) The Trefethen script is not described as executing a first expression evaluation at node 2 and a second expression evaluation at node 3 using a received result as claimed. It is a single programmed routine that each node executes, with each node performing the same doubling operation on a transmitted variable. (Ex. 1006, 5–6.) Petitioner's

-00913 Proceeding, Patent Owner Response, 37

- But, the prior art performs two distinct mathematical expression evaluations

> 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
```

Process 1's expression evaluation is: 1 x 2
Process 2's expression evaluation is: 2 x 2

Ex.1006, 6; -00913 Proceeding, Ex.1163, ¶¶57-58, 61; Reply, 21

Discussion Summary

	‘034 Patent IPR2025-00913	‘679 Patent IPR2025-00914	‘621 Patent IPR2025-00915	‘582 Patent IPR2025-00916
“Hardware Processor”	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			**X**	
“without first passing through the first kernel”			X	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	X	X	X	X
Printed publications	X	X	X	X

‘621 Patent:
Claim language for “reside on the same die”

‘621 Patent

20. The computer cluster of claim **13**, wherein the first processor and the second processor reside on the same die.

-00915 Proceeding, Ex.1001, 31:47-48

25. The computer cluster of claim **24**, wherein the first processor and the second processor reside on the same die.

-00915 Proceeding, Ex.1001, 32:37-38

RS6000 renders obvious "reside on the same die"

RS6000

Processor requirements and options

375 MHz POWER3 SMP High Nodes require a minimum of four 375 MHz, 630FP processors mounted on one card. You can order up to three additional four-processor cards (**F/C 4350**) to configure the node with a

Ex.1007, 10; -00915 Proceeding, Petition, 97

Petition

Second, RS6000 teaches that the IBM SP2 node "require[s] a minimum of four 375 MHz, 630FP processors **mounted on one card**." Ex.1007, 10. It was well-known to POSITAs that one implementation for multiple CPUs on a single card is multiple processor cores on the same die. *See, e.g.* Ex.1050, 1:12-13 ("Process technology scaling has enabled many-core super-scalar microprocessors, e.g., there may be 256 processor cores on a single die."); Ex.1051, 10:5-30 ("A semiconductor die, comprising: a first processing core and a second processing core"). Combining multiple processors onto a single die is a matter of obvious engineering choice. *In re Larson*, 340 F.2d 965, 968 ("the use of one piece construction … would be merely a matter of obvious engineering choice").

-00915 Proceeding, Petition, 97-98; Ex.1003, ¶¶383-386

Patent Owner seeks to limit "reside on the same die" to unspecified semiconductor fabrication techniques

Patent Owner Response

> gap. (Pet. 96–97; Ex. 1003 ¶¶ 383–386.) But the contemporaneous RS6000 documentation describes multiple discrete processor chips mounted on a card, not two processors fabricated on a single die, and nowhere suggests such on-die integration. (Ex. 1007, 11–12.)

-00915 Proceeding, Patent Owner Response, 37

- Claims 20 and 25 are the only mention of "reside on the same die" in the entire specification. There is no description of *any* fabrication techniques.

> **20**. The computer cluster of claim **13**, wherein the first processor and the second processor reside on the same die.

-00915 Proceeding, Ex.1001, 31:47-48

Patent Owner's Evidence

D. Reliance On The Existing Evidentiary Record

Patent Owner does not submit a new expert declaration with this Response because it is not necessary for the Board to resolve the issues presented. The dispositive issues are resolved by the intrinsic record and the asserted references. To

-00915 Proceeding, Patent Owner Response, 10

Discussion Summary

	‘034 Patent IPR2025-00913	‘679 Patent IPR2025-00914	‘621 Patent IPR2025-00915	‘582 Patent IPR2025-00916
“Hardware Processor”	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			X	
“without first passing through the first kernel”			**X**	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	X	X	X	X
Printed publications	X	X	X	X

‘621 Patent:
Claim language for “without first passing through first kernel” / “logically positioned between”

‘621 Patent

a first kernel residing in the at least one computer-readable medium, said first kernel configured to translate commands into code for execution on the first processor;

a first cluster node module residing in the at least one computer-readable medium, said first cluster node module configured to:

receive first commands from a user interface without the first commands first passing through the first kernel, and

after receiving the first commands from the user interface, send second commands to the first kernel;

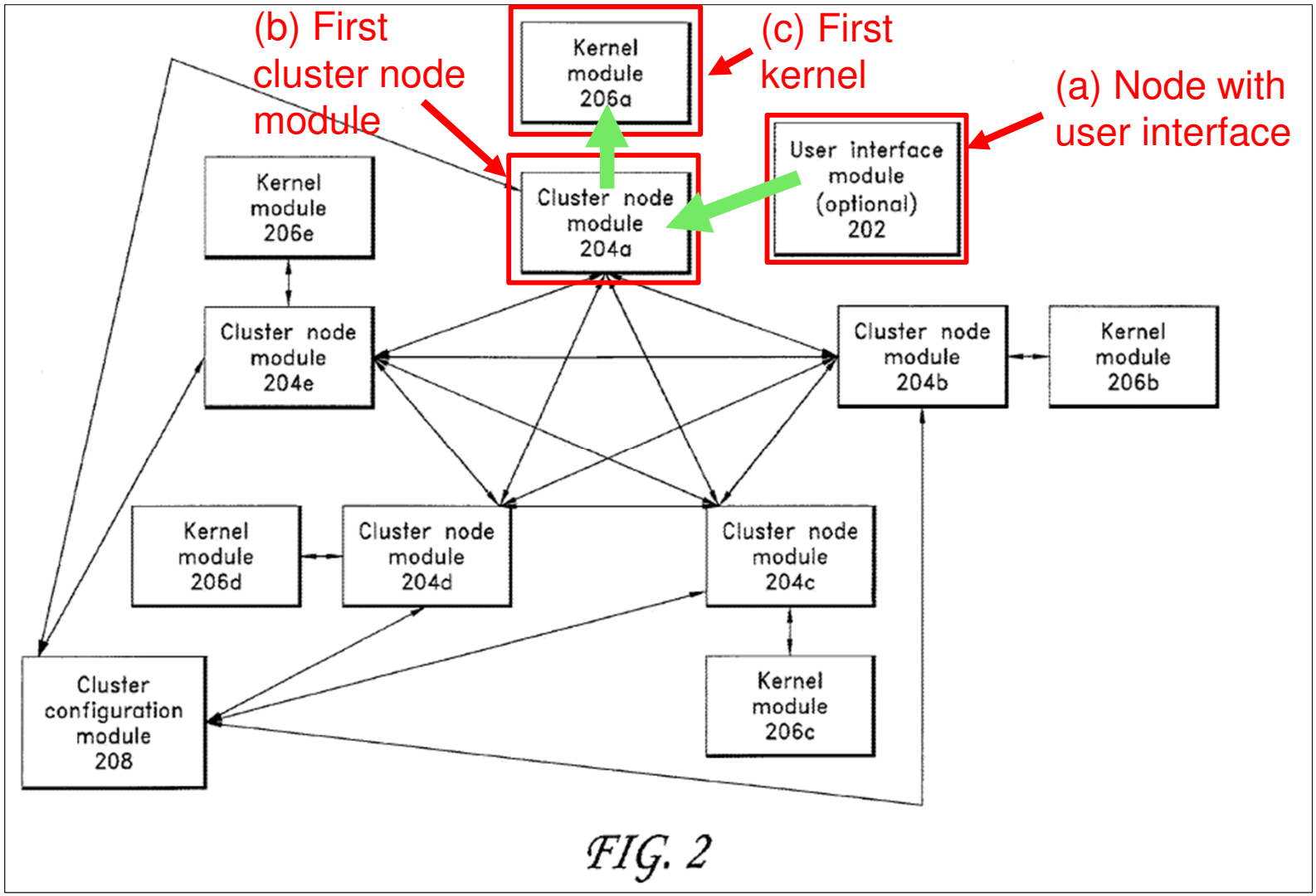
-00915 Proceeding, Ex.1001, 29:26-36

2. The computer cluster of claim 1, wherein the first cluster node module is logically positioned between the user interface and the first kernel.

-00915 Proceeding, Ex.1001, 29:63-65

‘621 Patent:
“without first passing through first kernel” / “logically positioned between”

‘621 Patent



-00915 Proceeding, Ex.1001, FIG. 2; Petition, 11

Menon's first cluster node module receives the commands without first passing through the first kernel and is positioned between

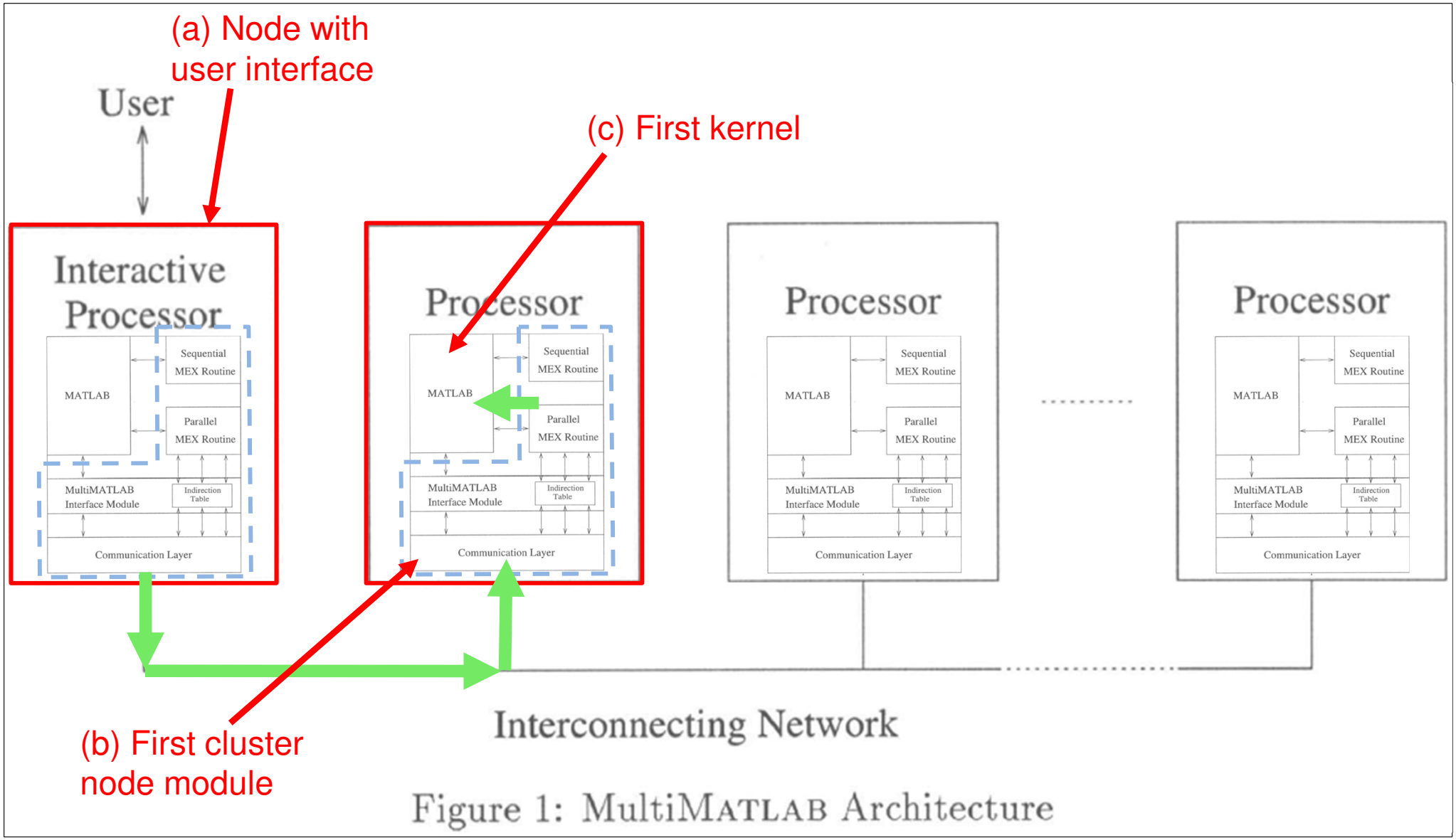
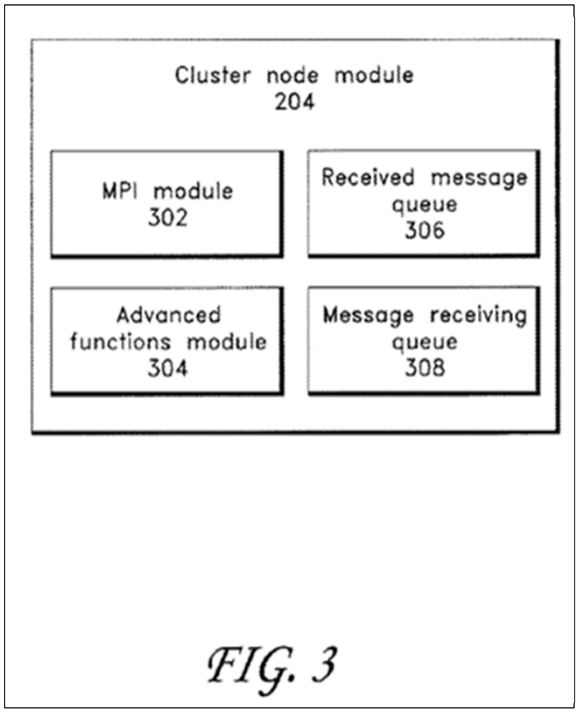


Figure 1: MultiMATLAB Architecture

Ex.1005, FIGs. 1 and 2; -00915 Proceeding, Petition, 41; Ex.1003, ¶212

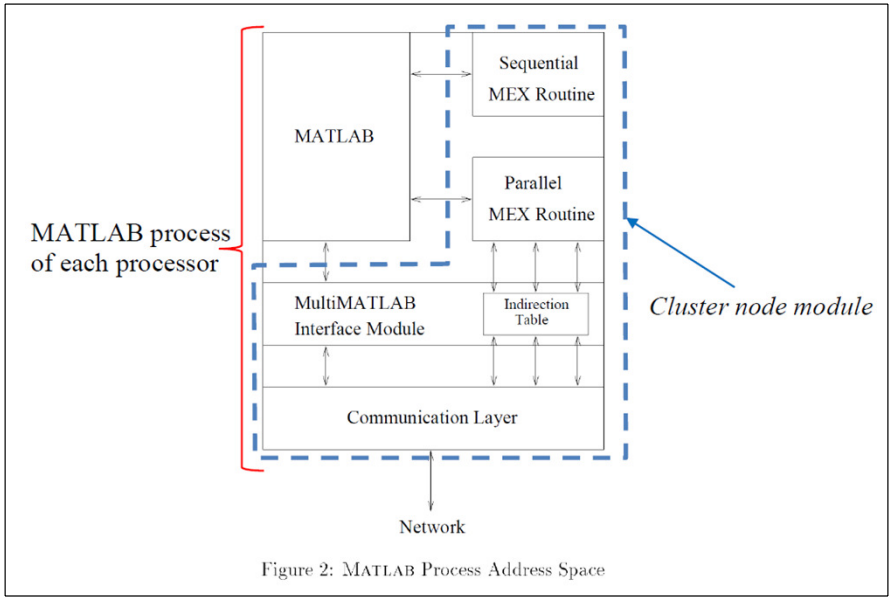
‘621 Patent:
“cluster node module” is a set of components

‘621 Patent



-00915 Proceeding, Ex.1001, FIG. 3; Petition, 39

Menon



-00915 Proceeding, Ex.1005, Fig. 2; Ex.1003, ¶204; Petition, 38

The key component in this architecture is the MultiMATLAB interface module. The interface module, shown in Figure 2, is responsible for initializing an underlying communication layer, such as MPI [2], PVM [3], BLACS [4], or any other package available on the platform, and exposing it to the rest of the system.

MEX routines, also shown in the Figure 2, are executables originally written in C or Fortran that may be run directly from MATLAB. All parallel functionality in the MultiMATLAB system is provided through MEX routines. As a simple example, the MultiMATLAB sys-

-00915 Proceeding, Ex.1005, 3; Petition, 17-19

interactive process and executes them in its own MATLAB environment. All parallel MEX routines access the underlying communication layer and, hence, the network, through the MultiMATLAB interface module.

-00915 Proceeding, Ex.1005, 4; Petition, 17-19

Menon teaches (i) a user interface and (ii) a first node

Menon

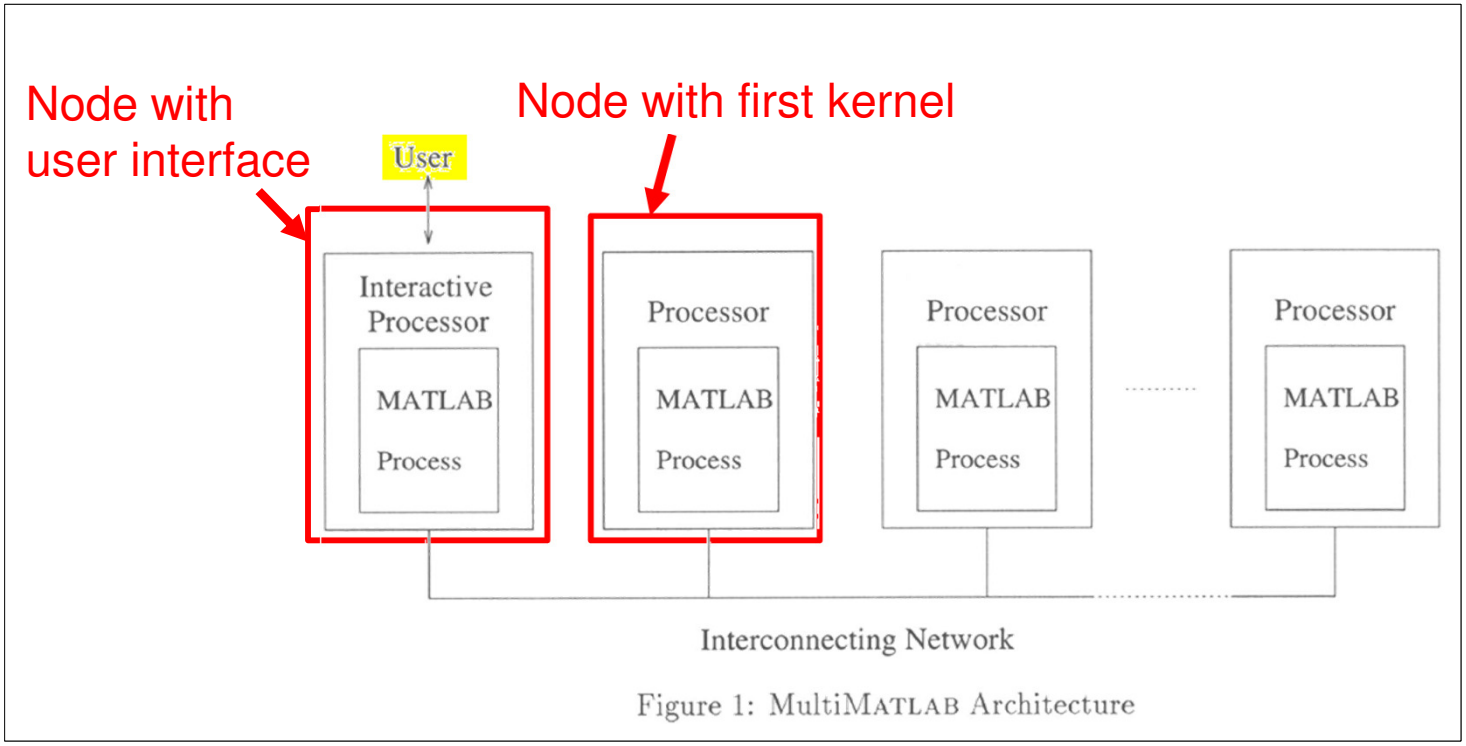


Figure 1: MultiMATLAB Architecture

-00915 Proceeding, Ex.1005, 3; Petition, 30

Menon

In the MultiMATLAB architecture, illustrated in Figure 1, each processor in a parallel platform individually runs a MATLAB process. Each process is then provided with the ability to communicate with other processes through a communication layer that runs over the parallel platform's interconnection network. The user interacts directly with one MATLAB process, called the interactive process, and operates within that process's MATLAB environment. Other MATLAB processes await commands from the interactive process. These other processes are used either by explicitly sending MATLAB commands to them or by executing routines that, in turn, use them.

-00915 Proceeding, Ex.1005, 3; Petition, 17

Discussion Summary

	‘034 Patent IPR2025-00913	‘679 Patent IPR2025-00914	‘621 Patent IPR2025-00915	‘582 Patent IPR2025-00916
“Hardware Processor”	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			X	
“without first passing through the first kernel”			X	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	X	X	X	X
Printed publications	X	X	X	X

‘582 Patent:
“cluster configuration module”

‘582 Patent claim 4

4. The computer cluster of claim **1**, further comprising a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.

-00916 Proceeding, Ex.1001, 30:1-4

‘582 Patent specification

B. Cluster Configuration Module **208**

Cluster configuration module **208** includes program code for initializing a plurality of cluster node modules to add cluster computing support to computer systems **110**, **120**,

-00916 Proceeding, Ex.1001, 22:4-6

Menon uses IBM's Parallel Operating Environment to start the processes

Menon

> In this implementation, all MATLAB processes are started via POE, IBM's Parallel Operating Environment. 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. All other processes wait for commands from the interactive MATLAB process.

Ex.1005, 6; -00916 Proceeding, Petition, 112-113

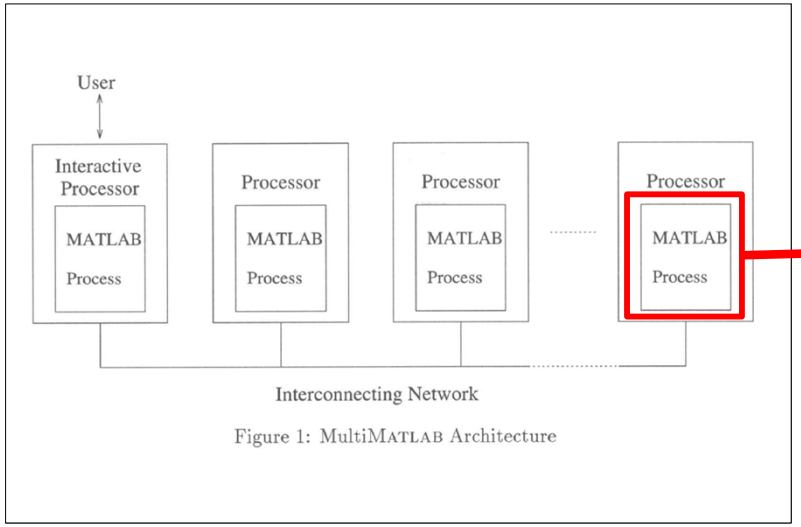


Figure 1: MultiMATLAB Architecture

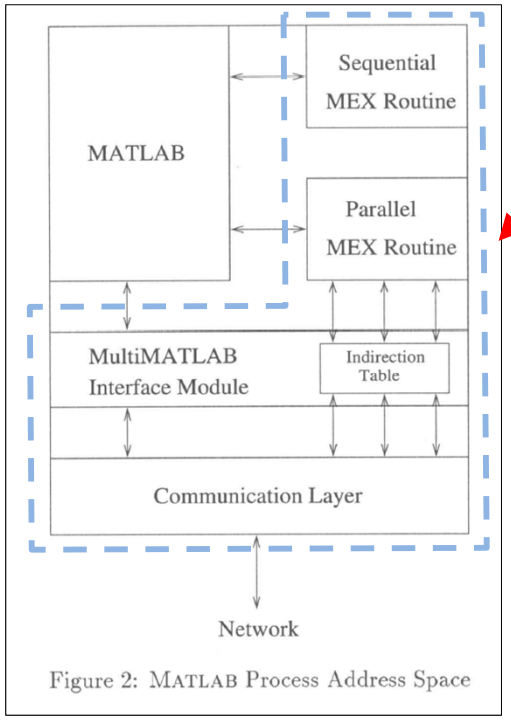


Figure 2: MATLAB Process Address Space

Cluster node module

Ex.1005, 3-4; -00916 Proceeding, Petition, 35,41

POE initializes the environment and loads the program on each node

POEref

When 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. The Partition Manager also passes the option list to each remote node. If your program is in a shared file system, the Partition Manager loads a copy of it on each node. If your program is in a private file system, you will have already manually copied your executable to the nodes using the **mcp** command (or the PSSP commands **dsh** or **pcp**) as described in "Step 2: Copy files to individual nodes" on page 10. If you are using the dynamic message passing interface, the appropriate communication subsystem library implementation (IP or US) is automatically loaded at this time.

Ex.1008, 28; -00916 Proceeding, Petition, 113

Patent Owner's Arguments: "cluster configuration module"

Patent Owner Response

> Claim 4 of the '582 Patent adds a specific structural requirement to the system recited in Claim 1: it requires "a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module." (Ex. 1001, 30:1-4.) This limitation is not satisfied by any mechanism that merely launches processes or allocates hardware resources. It requires an internal module of the claimed system—i.e., a component of the cluster computing architecture itself—that performs initialization of cluster node modules so that they can operate together in the claimed environment.

-00916 Proceeding, Patent Owner Response, 19

- But, the claim recites nothing about an "internal module"

> **4**. The computer cluster of claim **1**, further comprising a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module.

-00916 Proceeding, Ex.1001, 30:1-4

Patent Owner's Arguments: "cluster configuration module"

Patent Owner Response

> Claim 4 of the '582 Patent adds a specific structural requirement to the system recited in Claim 1: it requires "a cluster configuration module that initializes at least one of the first cluster node module, the second cluster node module, or the third cluster node module." (Ex. 1001, 30:1-4.) This limitation is not satisfied by any mechanism that merely launches processes or allocates hardware resources. It requires an internal module of the claimed system—i.e., a component of the cluster computing architecture itself—that performs initialization of cluster node modules so that they can operate together in the claimed environment.

-00916 Proceeding, Patent Owner Response, 19

- But, the specification says the cluster configuration module can be "on a remote computer system"

> with one of the processors 112*a*, 112*b*, 122*a*, 122*b*, 132. A cluster configuration module stored on one or more of the computer systems **110**, **120**, **130** or on a remote computer system, for example, can establish communication with the

-00916 Proceeding, Ex.1001, 11:36-39; Reply, 11

Patent Owner's Evidence

D. Reliance On The Existing Evidentiary Record

Patent Owner does not submit a new expert declaration with this Response because it is not necessary for the Board to resolve the issues presented. The dispositive issues are resolved by the intrinsic record and the asserted references. To

-00916 Proceeding, Patent Owner Response, 9

Discussion Summary

	‘034 Patent IPR2025-00913	‘679 Patent IPR2025-00914	‘621 Patent IPR2025-00915	‘582 Patent IPR2025-00916
“Hardware Processor”	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			X	
“without first passing through the first kernel”			X	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	**X**	**X**	**X**	**X**
Printed publications	X	X	X	X

Obvious to combine Menon and Trefethen

- Ex.1005 (Menon) and Ex.1006 (Trefethen) were written by the same authors

Menon

MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing

Vijay Menon
Department of Computer Science
Cornell University
Ithaca, NY 14853
vsm@cs.cornell.edu

Anne E. Trefethen
Cornell Theory Center
Cornell University
Ithaca, NY 14853
anne@tc.cornell.edu

Ex.1005,1; -00913 Proceeding, Petition, 23

Trefethen

MultiMATLAB:
MATLAB on Multiple Processors

Anne E. Trefethen
Cornell Theory Center
aet@tc.cornell.edu
http://www.tc.cornell.edu/~anne

Vijay S. Menon
Computer Science Department, Cornell University
vsm@cs.cornell.edu
http://www.cs.cornell.edu/Info/People/vsm

Ex.1006,1; -00913 Proceeding, Petition, 23

Obvious to combine Menon and Trefethen

- Menon (Ex.1005) expressly cites to Trefethen (Ex.1006)

Menon

In [1], Trefethen, et al. present MultiMATLAB, a general extension of the MATLAB environment to distributed memory multiprocessors, as a first step in this direction. In particular, we demonstrate that MultiMATLAB achieves the following goals.

- **Familiarity:** The original MATLAB environment is preserved. The MultiMATLAB system is simply an extension of the MATLAB environment. The system does not alter the way users interact with MATLAB; it merely provides new functionality.
- **Interactiveness:** Users are provided interactive access to all processors in a multiprocessor within the MATLAB environment. This permits steering of parallel computation.
- **Programmability:** Users are given the tools to write parallel code purely in the MATLAB language. Moreover, parallel and distributed applications can be developed interactively in the MATLAB environment.
- **Portability:** The system should be easily portable onto any parallel platform that allows MATLAB to run on each processor. This includes any network of workstations supporting MATLAB. Since most modern fine-grain multiprocessors are built using commodity processors, it should also include most modern parallel platforms.

Ex.1005, 2; Ex.1003, ¶71; -00913 Proceeding, Petition, 22

References

[1] A. E. Trefethen, V. S. Menon, C. C. Chang, G. J. Czajkowski, C. Myers, and L. N. Trefethen. MultiMATLAB: MATLAB on multiple processors. Technical Report 96-239, Cornell Theory Center, 1996.
http://www.cs.cornell.edu/Info/People/lnt/multimatlab.html.

Ex.1005, 16; Ex.1003, ¶71; -00913 Proceeding, Petition, 22

Obvious to combine Menon and Trefethen

- Trefethen describes how the MultiMATLAB system enables MATLAB to run on multiple computers.

Trefethen

The vision of the MultiMATLAB project has been to bridge this gap. Think of a user who finds him- or herself computing happily in MATLAB, but frustrated by the time it takes to rerun the program for six different boundary conditions, or a dozen different parameter choices, or a hundred different initial guesses. Such a user's problems might be solved by a system that makes it convenient to spawn MATLAB processes on multiple processors of a parallel computer or a network of workstations or PCs. In many cases the needs for communication between the processors are rather small. Convenience of spreading the problem across machines and collecting the results numerically or graphically is paramount.

The MultiMATLAB project is exploring one approach for making this kind of computing possible. We do not at the outset aim for fine-grained parallelism or for peak performance of the kind needed for the grand challenge problems of computational science. Instead, following the philosophy that has made MATLAB so successful, we require reasonable efficiency but put the premium on ease of use. A key principle is that MATLAB itself -- not a home-grown facsimile, which would have little chance of keeping up with the ever-expanding features of the commercial product -- must be run on multiple processors. Our vision is that a user must be able to learn enough in five minutes to become intrigued by the system and begin to use it.

2. Using MultiMATLAB

2.1. Start, Nproc, Eval, ID

Each MultiMATLAB command begins with an initial upper-case letter. We illustrate how the system is used before describing its implementation.

Ex.1006, 3; Ex.1003, ¶71; -00913 Proceeding, Petition, 20

Obvious to combine Menon and Trefethen

- Menon describes enhancing Trefethen's MultiMATLAB system "to achieve two further goals."

Menon

In [1], Trefethen, et al. present MultiMATLAB, a general extension of the MATLAB environment to distributed memory multiprocessors, as a first step in this direction. In particular, we demonstrate that MultiMATLAB achieves the following goals.

- **Familiarity:** The original MATLAB environment is preserved. The MultiMATLAB system is simply an extension of the MATLAB environment. The system does not alter the way users interact with MATLAB; it merely provides new functionality.
- **Interactiveness:** Users are provided interactive access to all processors in a multiprocessor within the MATLAB environment. This permits steering of parallel computation.
- **Programmability:** Users are given the tools to write parallel code purely in the MATLAB language. Moreover, parallel and distributed applications can be developed interactively in the MATLAB environment.
- **Portability:** The system should be easily portable onto any parallel platform that allows MATLAB to run on each processor. This includes any network of workstations supporting MATLAB. Since most modern fine-grain multiprocessors are built using commodity processors, it should also include most modern parallel platforms.

In this paper, we present a new MultiMATLAB architecture redesigned to achieve two further goals.

- **Performance:** The MultiMATLAB architecture now provides direct access from MATLAB to vendor-optimized communication libraries. Users should be able to replace computationally intensive routines within sequential MATLAB code with high-performance parallel routines and expect performance gains.
- **Extensibility:** Users or other third parties may extend the environment with new high-performance parallel routines written in C or Fortran.

Ex.1005, 2; Ex.1003, ¶71; -00913 Proceeding, Petition, 22

Discussion Summary

	‘034 Patent IPR2025-00913	‘679 Patent IPR2025-00914	‘621 Patent IPR2025-00915	‘582 Patent IPR2025-00916
“Hardware Processor”	X			
1->2->3->1 operation	X	X	X	X
Reside on a die			X	
“without first passing through the first kernel”			X	
Cluster configuration module				X
Obvious to combine Menon/Trefethen	X	X	X	X
Printed publications	**X**	**X**	**X**	**X**

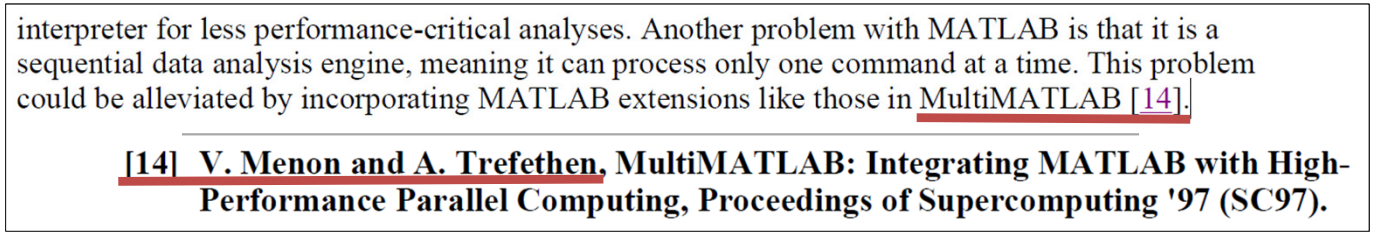
Menon qualifies as a printed publication

- Published and distributed at the 1997 ACM/IEEE Conference on Supercomputing 1997.

Ex.1026; Ex.1027; -00913 Proceeding, Reply, 27

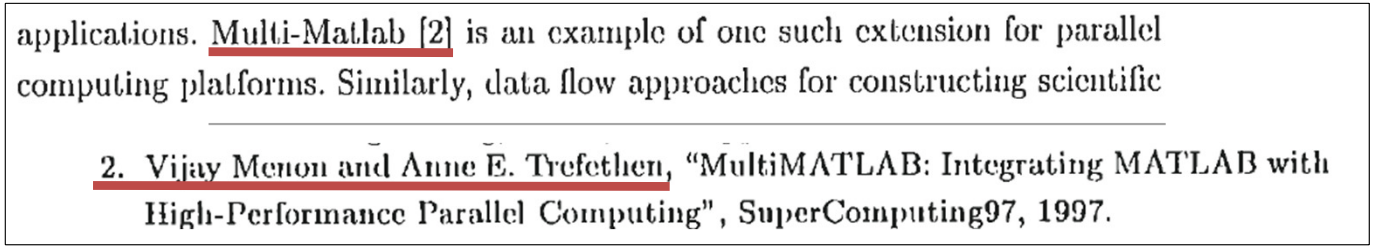
- Published and distributed on Cornell University's web at least as early as 2001.

Ex.1023; Ex.1029; -00913 Proceeding, Reply, 27

- Disseminated and located by interested persons:
  - 
interpreter for less performance-critical analyses. Another problem with MATLAB is that it is a sequential data analysis engine, meaning it can process only one command at a time. This problem could be alleviated by incorporating MATLAB extensions like those in MultiMATLAB [14].

[14] V. Menon and A. Trefethen, MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing, Proceedings of Supercomputing '97 (SC97).

Ex.1094, 5, 12; -00913 Proceeding, Reply, 28

  - 
applications. Multi-Matlab [2] is an example of one such extension for parallel computing platforms. Similarly, data flow approaches for constructing scientific

2. Vijay Menon and Anne E. Trefethen, "MultiMATLAB: Integrating MATLAB with High-Performance Parallel Computing", SuperComputing97, 1997.

Ex.1095, 7, 19, -00913 Proceeding, Reply, 28

- Cataloged in libraries
  - German National Library of Science and Technology
  - Academia Sinica Library
  - Internet Archive

Ex.1086, ¶¶33-53, -00913 Proceeding, Reply, 28-29

Trefethen qualifies as a printed publication

- Published and distributed on Cornell University's web at least as early as 1996.

Ex.1023; -00913 Proceeding, Reply, 29

- Disseminated and located by interested persons:
  - A similar facility has been made available through the Matlab environment [40]. This uses multiple processes rather than threads and so incurs the overhead of process communication. One

    [40] Anne E. Trefethen, Vijay S. Menon, Chi-Chao Chang, Grzegorz J. Czajkowski, and Chris Myers. Multimatlab: Matlab on multiple processors. URL http://www.cs.cornell.edu/Info/People/lnt/multimatlab.html, May 1996.

  - tions connected on a local area network [10]. MultiMATLAB, developed at Cornell University, initially targeted conventional parallel computers, such as the IBM SP-2, but can also be executed on networks of workstations [11].

    [11] Trefethen, A. E., Menon, V. S., Chang, C.-C., Czajkowski, G. J., Myers, C., and Trefethen, L. N., "MultiMATLAB: MATLAB on Multiple Processors," Technical Report 96–239, Cornell Theory Center, Ithaca, NY (1996).

  - set of tools are interpreted MATLAB clones such as SCILAB and OCTAVE. Another class of tools proposes parallel extensions to MATLAB such as message passing in MultiMatlab [12] or a client-server mechanism in MathServer [8] or IPSC-

    12. Anne E. Trefethen, Vijay S. Menon, Chi-Chao Chang, Grzegorz J. Czajkowski, Chirs Myers, and Lloyd N. Trefethen. MultiMATLAB: MATLAB on multiple processors. Technical Report 239, Cornel Theory Center, 1996.

Ex.1102, 23, 137; -00913 Proceeding, Reply, 29-30

Ex.1104, 2,7; -00913 Proceeding, Reply, 30

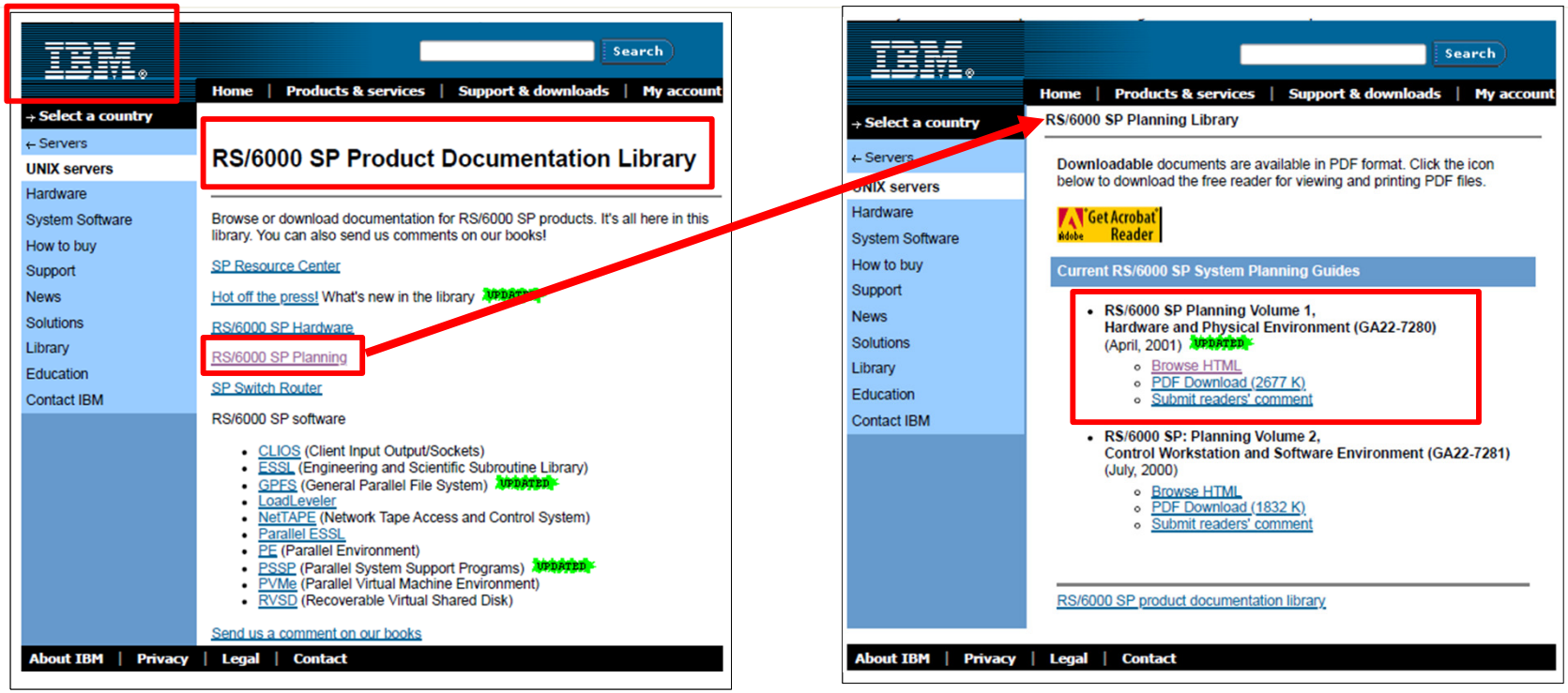
Ex.1107, 9, 21; -00913 Proceeding, Reply, 30

- Cataloged in libraries
  - Washington State University Library
  - Internet Archive

Ex.1086, ¶¶76-85, -00913 Proceeding, Reply, 30

RS6000 qualifies as a printed publication

- Published and distributed in IBM's online document library at least as early as 2002.



Ex.1038, 8-9; -00913 Proceeding, Reply, 31

- Cataloged in libraries
  - Internet Archive

Ex.1086, ¶¶112-119, -00913 Proceeding, Reply, 31

POEref qualifies as a printed publication

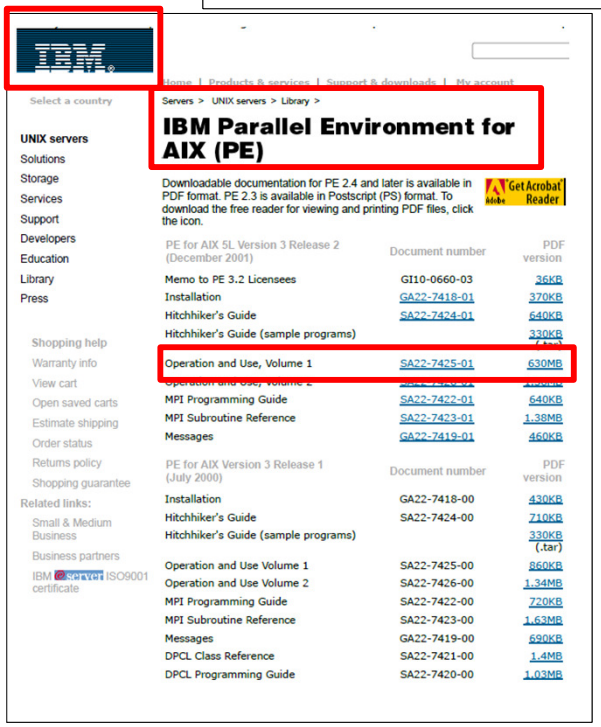
- Published and distributed in IBM's online document library at least as early as 2001, as declared by an IBM representative.

➢

3. The document titled "Operation and Use, Volume 1 Using the Parallel Operating Environment," as shown at https://publibfp.boulder.ibm.com/epubs/pdf/a2274251.pdf?mhsrc=ibmsearch_a&mhq=sa22-7425, was publicly available from IBM as of December 7, 2001.

Ex.1047, 1; -00913 Proceeding, Reply, 32

➢

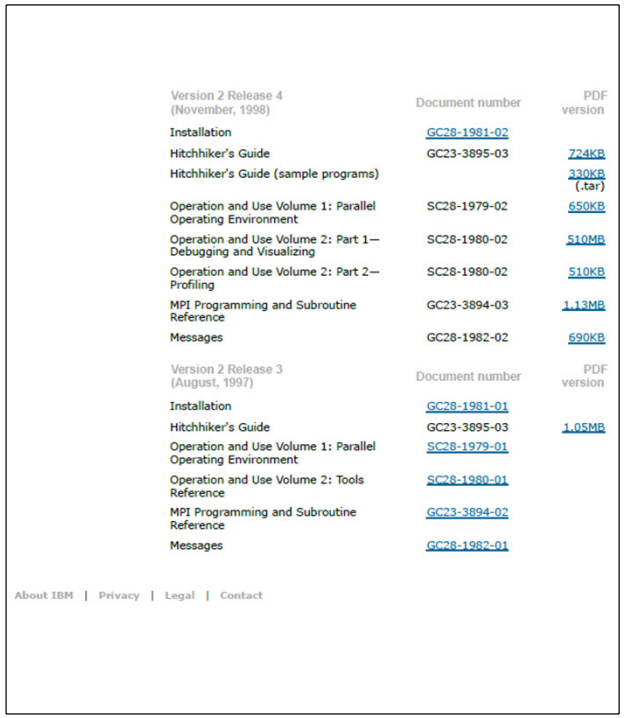


Servers > UNIX servers > Library >

IBM Parallel Environment for AIX (PE)

Downloadable documentation for PE 2.4 and later is available in PDF format. PE 2.3 is available in Postscript (PS) format. To download the free reader for viewing and printing PDF files, click the icon.

PE for AIX 5L Version 3 Release 2 (December 2001)	Document number	PDF version
Memo to PE 3.2 Licensees	GI10-0660-03	26KB
Installation	GA22-7418-01	370KB
Hitchhiker's Guide	SA22-7424-01	640KB
Hitchhiker's Guide (sample programs)		330KB (.tar)
Operation and Use, Volume 1	SA22-7425-01	630MB
MPI Programming Guide	SA22-7422-01	640KB
MPI Subroutine Reference	SA22-7423-01	1.38MB
Messages	GA22-7419-01	460KB
PE for AIX Version 3 Release 1 (July 2000)	**Document number**	**PDF version**
Installation	GA22-7418-00	430KB
Hitchhiker's Guide	SA22-7424-00	710KB
Hitchhiker's Guide (sample programs)		330KB (.tar)
Operation and Use Volume 1	SA22-7425-00	860KB
Operation and Use Volume 2	SA22-7426-00	1.34MB
MPI Programming Guide	SA22-7422-00	720KB
MPI Subroutine Reference	SA22-7423-00	1.63MB
Messages	GA22-7419-00	690KB
DPCL Class Reference	SA22-7421-00	1.4MB
DPCL Programming Guide	SA22-7420-00	1.03MB



Version 2 Release 4 (November, 1998)	Document number	PDF version
Installation	GC28-1981-02	
Hitchhiker's Guide	GC23-3895-03	724KB
Hitchhiker's Guide (sample programs)		330KB (.tar)
Operation and Use Volume 1: Parallel Operating Environment	SC28-1979-02	650KB
Operation and Use Volume 2: Part 1—Debugging and Visualizing	SC28-1980-02	510MB
Operation and Use Volume 2: Part 2—Profiling	SC28-1980-02	510KB
MPI Programming and Subroutine Reference	GC23-3894-03	1.13MB
Messages	GC28-1982-02	690KB
Version 2 Release 3 (August, 1997)	**Document number**	**PDF version**
Installation	GC28-1981-01	
Hitchhiker's Guide	GC23-3895-03	1.05MB
Operation and Use Volume 1: Parallel Operating Environment	SC28-1979-01	
Operation and Use Volume 2: Tools Reference	SC28-1980-01	
MPI Programming and Subroutine Reference	GC23-3894-02	
Messages	GC28-1982-01	

Ex.1039, 8-9; -00913 Proceeding, Reply, 32

- Disseminated and located by interested persons:

➢

tives provided by two popular MPI implementations: the public domain MPICH [2] implementation on a Linux cluster, and the proprietary IBM implementation on an IBM SP2 [3]. (2) To propose and evaluate usage strategies for these prim-

3. Parallel Environment (PE) for AIX V3R2.0: Operation and Use, Vol. 1. December, 2001.

Ex.1114, 2, 10; -00913 Proceeding, Reply, 32

MPIref qualifies as a printed publication

- Published and distributed by the University of Tennessee and Oak Ridge National Library in 1994.

> The University of Tennessee and Oak Ridge National Laboratory made the draft available by anonymous FTP mail servers and were instrumental in distributing the document.

Ex.1017, 7; -00913 Proceeding, Reply, 33

- Disseminated and located by interested persons:
  - > The standard Message Passing transfer protocols ([MPIF 94], [TMC 93]), and more recent Active Message transfer primitives ([von Eicken *et al.* 92], [Liu & Culler 94], [Brewer & Kuszmaul 94]) allow low-level specification of asynchronous execution, but
    >
    > [MPIF 94] MPIF. MPI: A Message-Passing Interface Standard, May 1994.

    Ex.1122, 21, 116; -00913 Proceeding, Reply, 33-34
  - > • **a typed message passing library (currently NXLIB [26] and MPI [19]), and**
    >
    > **[19] MPI. MPI: A Message Passing Interface Standard. Tech. rep., Message Passing Interface Forum, May 1994.**

    Ex.1124, 3, 10; -00913 Proceeding, Reply, 34
- Cataloged in libraries
  - National Science Library of Canada
  - University of Edinburgh
  - Internet Archive

Ex.1086, ¶¶153-166, -00913 Proceeding, Reply, 34

IPR2025-00913 (US11,570,034)
IPR2025-00914 (US12,021,679)
IPR2025-00915 (US11,563,621)
IPR2025-00916 (US11,811,582)

Intel Corp. v. Advanced Cluster Systems, Inc.

Andrew S. Ehmke, Haynes and Boone, LLP
Daniel D. Smith, Haynes and Boone, LLP
Attorneys for Petitioner Intel Corporation