Interactive, On-Demand Parallel Computing with pMatlab and gridMatlab

Albert Reuther, Nadya Bliss, Robert Bond,
Jeremy Kepner, and Hahn Kim

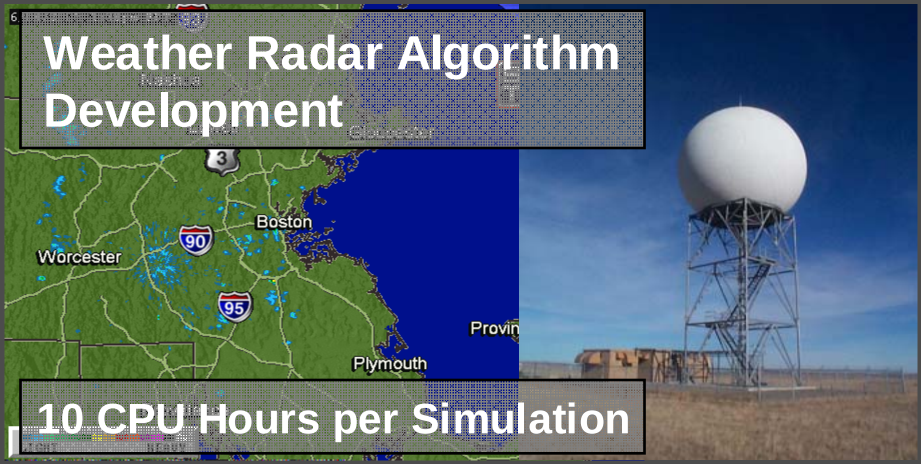
June 15, 2006

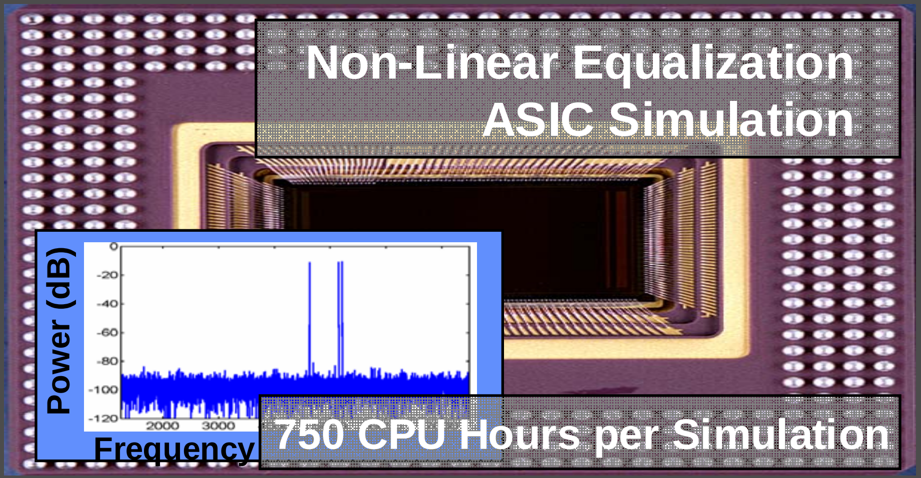
This work is sponsored by the Defense Advanced Research Projects Administration under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

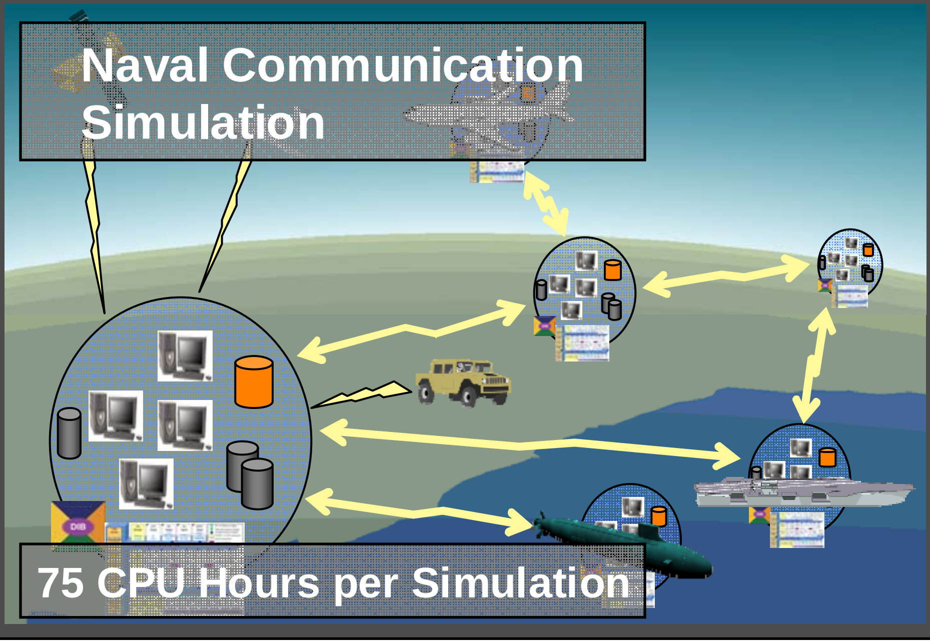
Outline

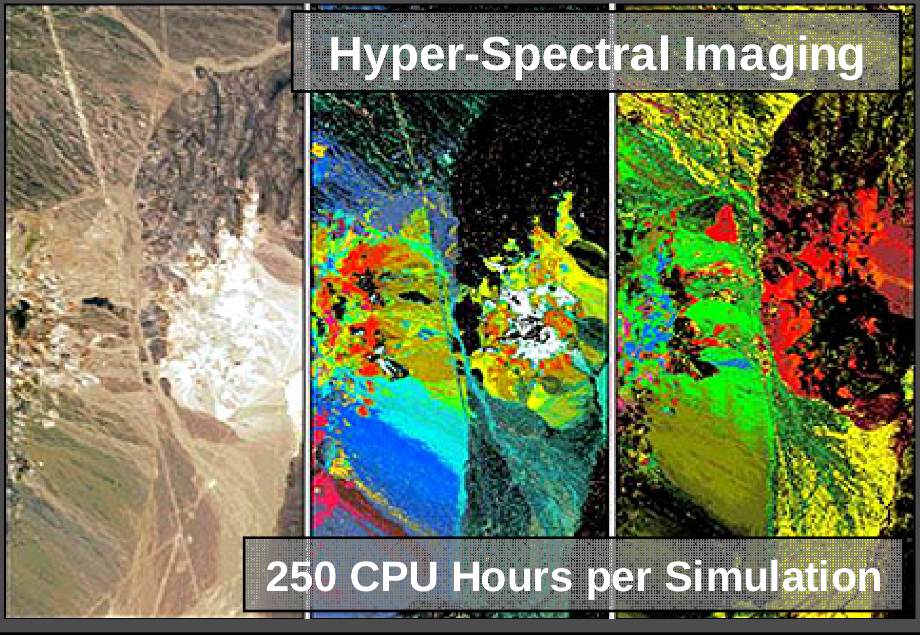
- **Introduction**
  - ***Goals***
  - ***Requirements***
- **Approach**
- **Results**
- **Future Work**
- **Summary**

Typical Applications at MIT Lincoln Laboratory





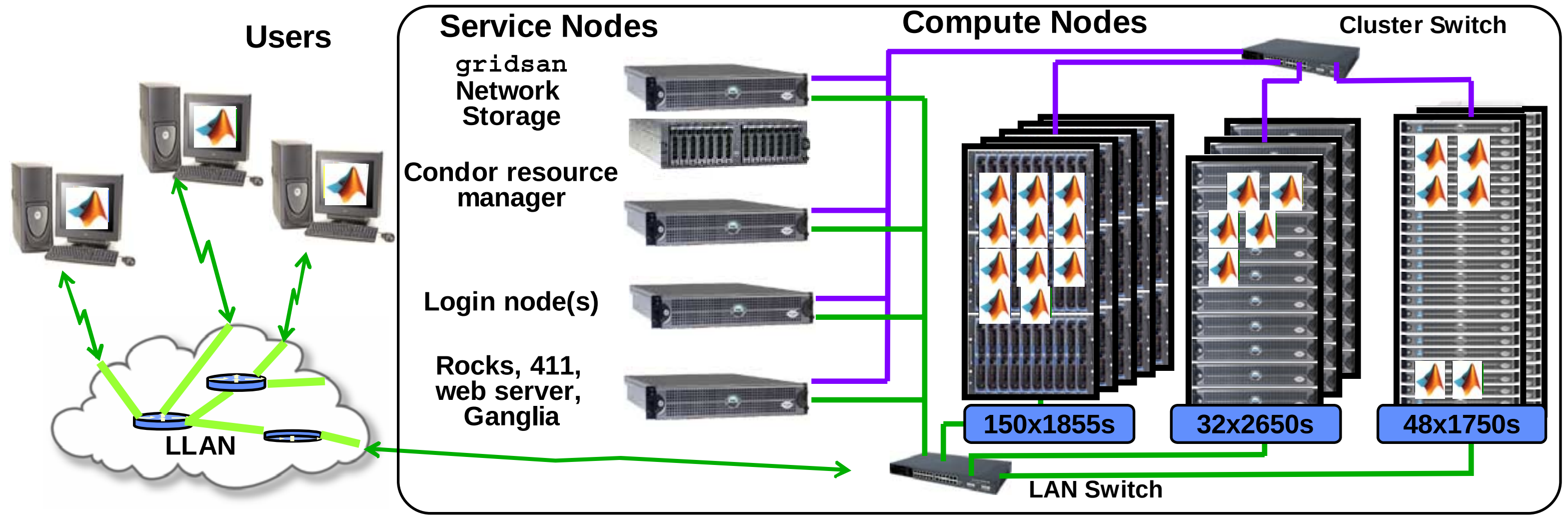




Lincoln Laboratory Grid (LLGrid)

Charter

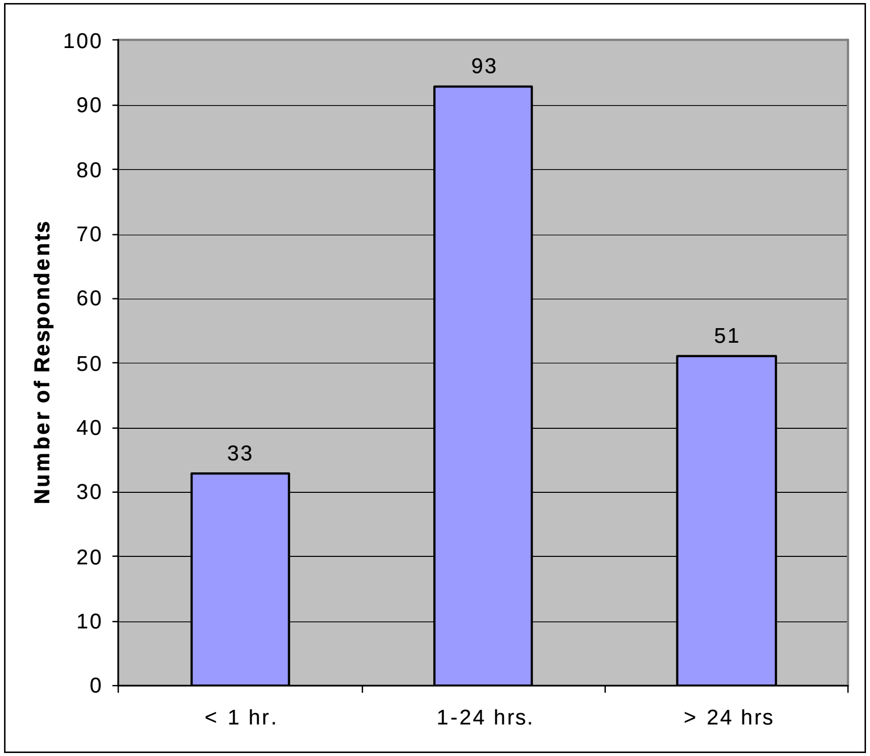
- Enterprise access to high throughput Grid computing (100 Gflops)
- Enterprise access to distributed storage (10 Tbytes)
- Interactive, direct use from the desktop



Goal: ***To provide a grid computing capability that makes it as easy to run parallel programs on a grid as it is to run on own workstation***

- ***Primary initial focus on MATLAB users***

User Requirements

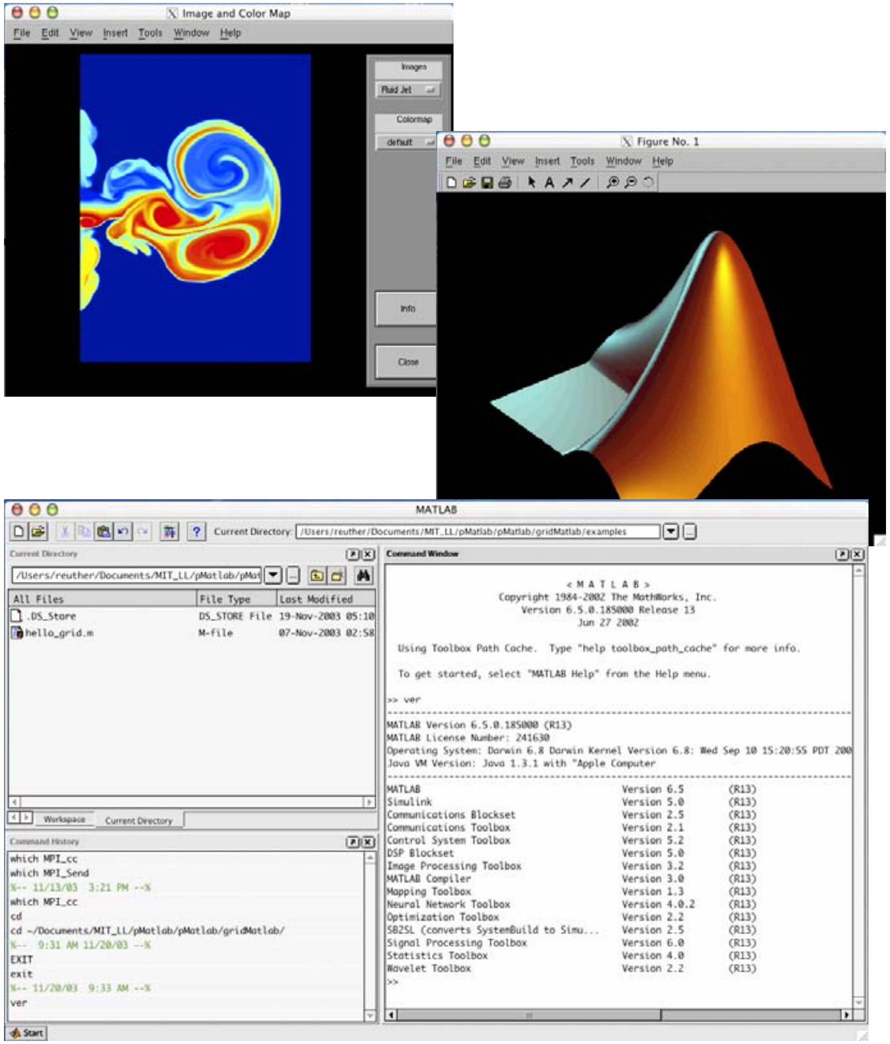


- Conducted survey of Lincoln staff
  - Do you run long jobs?
  - How long do those jobs run (minutes, hours, or days)?
  - Are these jobs unclassified, classified, or both?
- Survey results:
  - 464 respondents
  - 177 answered "Yes" to question on whether they run long jobs
- Lincoln MATLAB users:
  - Engineers and scientists, generally not computer scientists
  - Little experience with batch queues, clusters, or mainframes
  - Solution must be easy to use

- **Many users would like to accelerate jobs <1 hour**
  - **Requires "On Demand" Grid computing**

LLgrid System Requirements

- **Easy to use -**
  - **Using LLgrid should be the same as running a MATLAB job on user's computer**
- **Easy to set up**
  - **First time user setup should be automated and take less than 10 minutes**
- **Compatible**
  - **Windows, Linux, Solaris, and MacOS X**
- **Easily maintainable**
  - **One system administrator**



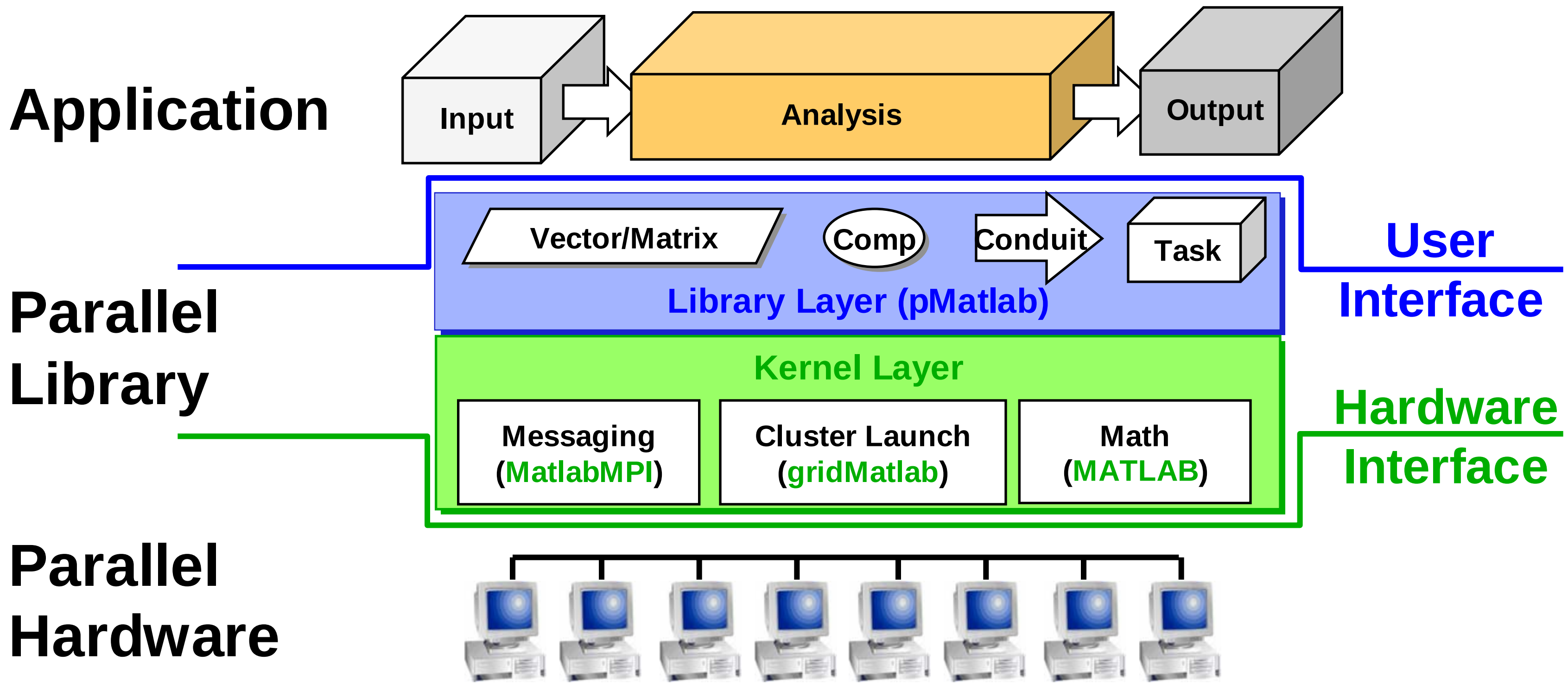
Outline

- Introduction
- **Approach**
  - ***pMatlab Design***
  - *gridMatlab*
- Results
- Future Work
- Summary

Outline

- Introduction
- **Approach**
  - ***pMatlab Design***
  - *pMatlab Examples*
  - *gridMatlab*
- Results
- Future Work
- Summary

Parallel Matlab (pMatlab)



Layered Architecture for parallel computing

- **Kernel layer does single-node math & parallel messaging**
- **Library layer provides a parallel data and computation toolbox to Matlab users**

pMatlab Maps and Distributed Arrays

A processor ***map*** for a numerical array is an ***assignment of blocks of data to processing elements***.

```
mapA = map([2 2], {}, [0:3]);
```

Grid specification together with processor list describe where the data is distributed.

Distribution specification describe how the data is distributed (default is block).

```
A = zeros(4,6, mapA);
```

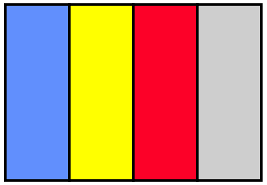
P0	P2
P1	P3

MATLAB constructors are overloaded to take a map as and argument, and return a dmat, a distributed array.

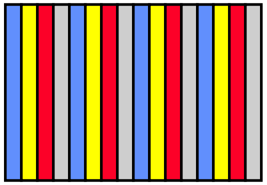
A =

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

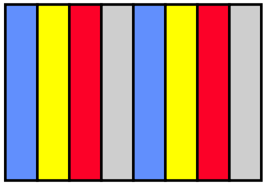
Supported Distributions



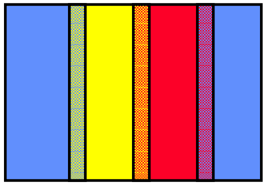
**Block,
in any dimension**



**Cyclic,
in any dimension**



**Block-cyclic,
in any dimension**



**Block-overlap,
in any dimension**

Distribution can be different for each dimension

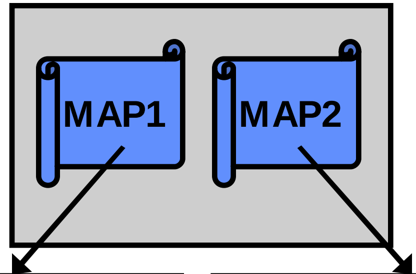
```
mapA = map([1 4],{},[0:3]);
mapB = map([4 1],{},[4:7]);
A = rand(M,N,mapA);
B = zeros(M,N,mapB);
B(:,:) = fft(A);
```

Functions are overloaded for `dmats`. Necessary communication is performed by the library and is abstracted from the user.

While function coverage is not exhaustive, redistribution is supported for any pair of distributions.

Advantages of Maps

Maps are scalable. Changing the number of processors or distribution does not change the application.

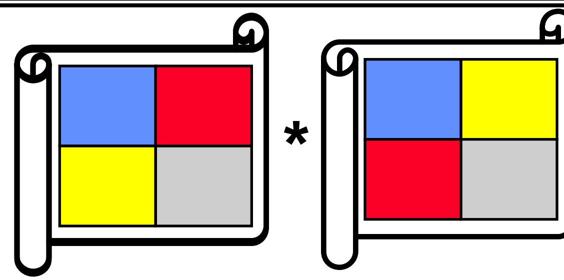



```
%Application
A=rand(M,map<i>);
B=fft(A);
```

```
map1=map([1 Ncpus],{},[0:Ncpus-1]);
```

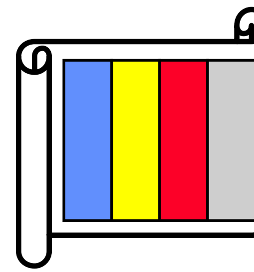
```
map2=map([4 3],{},[0:11]);
```

Maps support different algorithms. Different parallel algorithms have different optimal mappings.

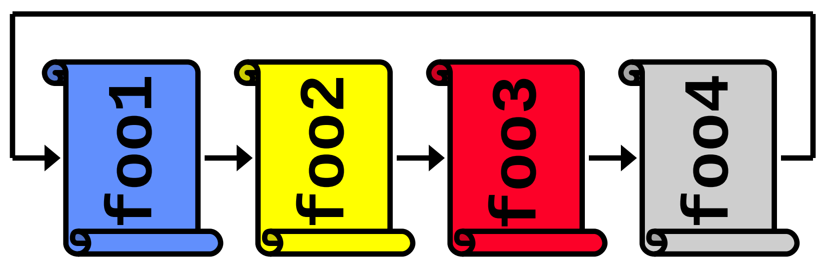


Matrix Multiply

FFT along columns



Maps allow users to set up pipelines in the code (implicit task parallelism).



Different Array Access Styles

- **Implicit global access (recommended for data movement)**

```
Y(:,:) = X;
Y(i,j) = X(k,l);
```

Most elegant; performance issues; accidental communication

- **Implicit local access (not recommended)**

```
[I J] = global_ind(X);
for i=1:length(I)
  for j=1:length(I)
    X_ij = X(I(i),J(I));
  end
end
```

Less elegant; possible performance issues

- **Explicit local access (recommended for computation)**

```
x = local(X);
x(i,j) = 1;
X = put_local(X,x);
```

A little clumsy; guaranteed performance; controlled communication

- **Distributed arrays are very powerful, use them only when necessary**

Outline

- Introduction
- **Approach**
  - *pMatlab Design*
  - ***pMatlab Examples***
  - *gridMatlab*
- Results
- Future Work
- Summary

Parallel Image Processing

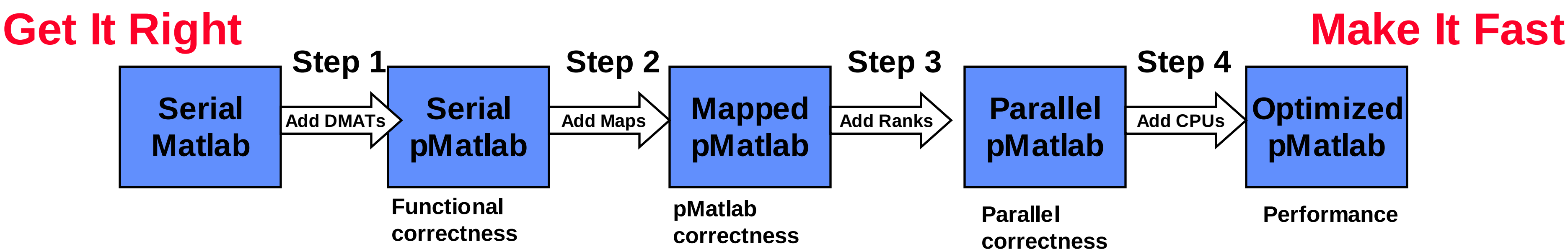
(see pMatlab/examples/pBlurimage.m)

```
mapX = map([Ncpus/2 2],{},[0:Ncpus-1],[N_k M_k]);   % Create map with overlap

X = zeros(N,M,mapX);                                 % Create starting images.

[myI myJ] = global_ind(X);                           % Get local indices.

X_local = local(X);                                  % Get local data.

                                                     % Assign data.
X_local = (myI.' * ones(1,length(myJ))) + (ones(1,length(myI)).' * myJ) );

% Perform convolution.
X_local(1:end-N_k+1,1:end-M_k+1) = conv2(X_local,kernel,'valid');

X = put_local(X,X_local);                            % Put local back in global.

X = synch(X);                                        % Copy overlap.
```

■ Implicitly Parallel Code ■ Required Change

Serial to Parallel in 4 Steps

Well defined process for going from serial to a parallel program



Step 1: Add distributed matrices without maps, verify functional correctness

Step 2: Add maps, run on 1 CPU, verify pMatlab correctness

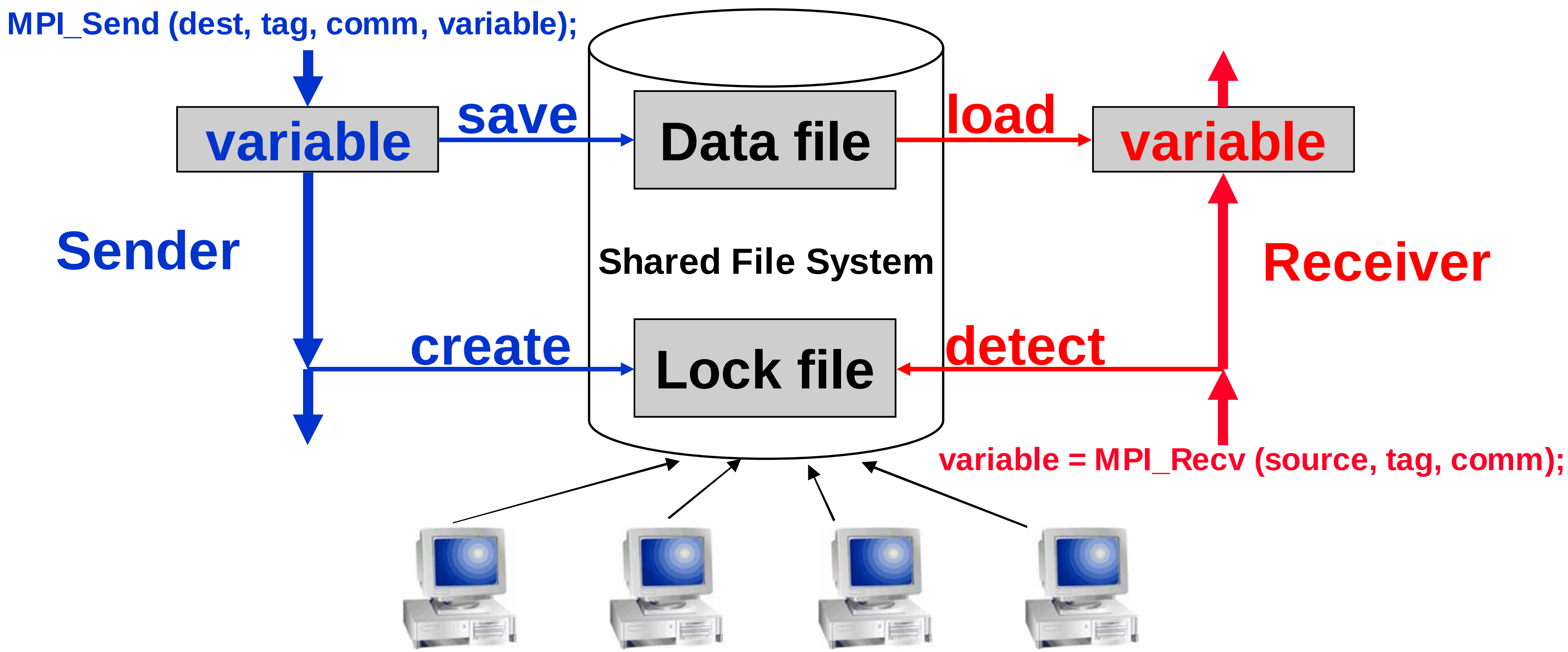
Step 3: Run with more processes (ranks), verify parallel correctness

Step 4: Run with more CPUs, compare performance with Step 2

- **Most user's familiar with Matlab, new to parallel programming**
- **Starting point is serial Matlab program**

MatlabMPI: Point-to-point Communication*

- Any messaging system can be implemented using file I/O
- File I/O provided by Matlab via load and save functions
  - Takes care of complicated buffer packing/unpacking problem
  - Allows basic functions to be implemented in ~250 lines of Matlab code



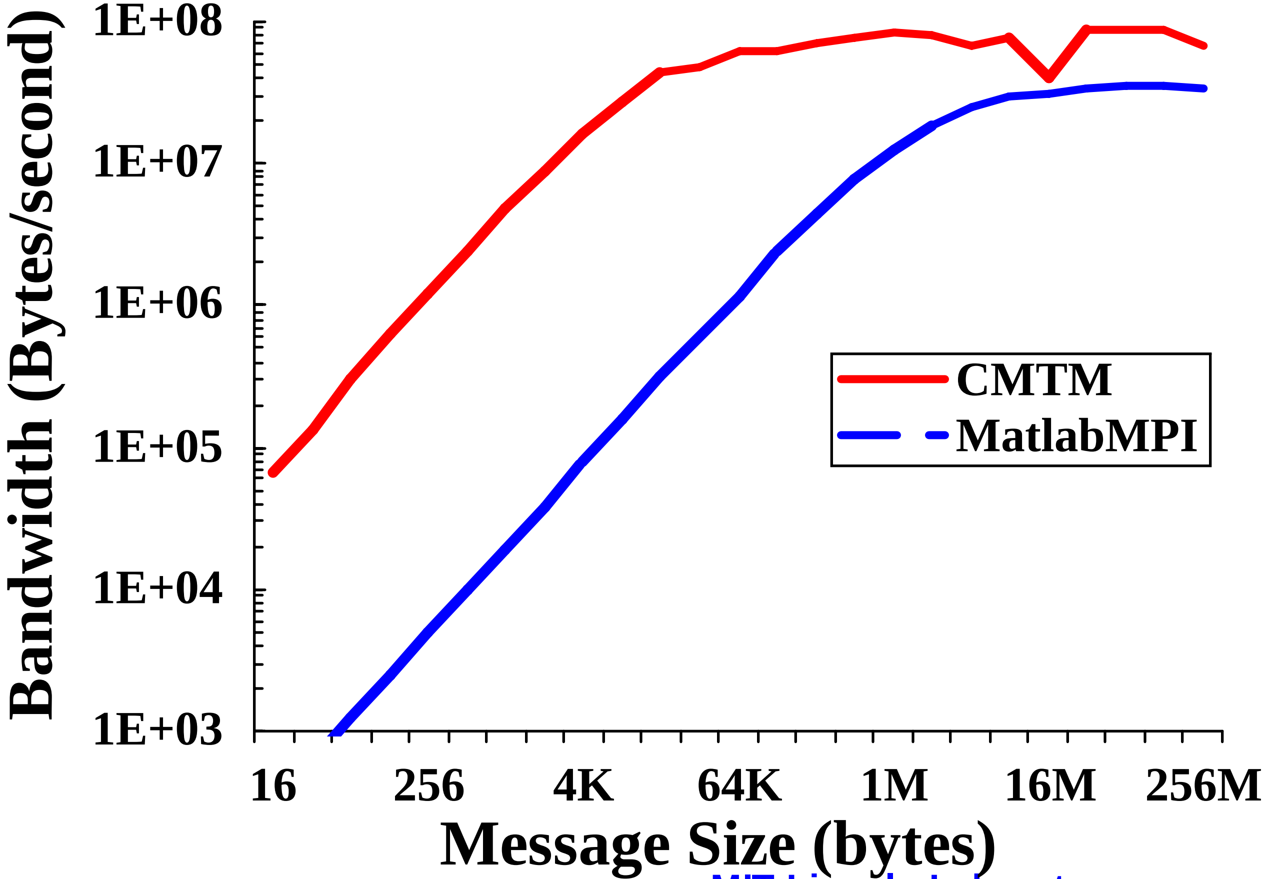
- Sender saves variable in Data file, then creates Lock file
- Receiver detects Lock file, then loads Data file

Unified MPI API Functions*

- Unified subset of functions from MatlabMPI (Lincoln) and CMTM (Cornell)
- Basic set of the most commonly used MPI functions required for global arrays

- MPI_Init
- MPI_Comm_size
- MPI_Comm_rank
- MPI_Send
- MPI_Recv
- MPI_Finalize
- MPI_Abort
- MPI_Bcast
- MPI_Iprobe

- mpirun?



Outline

- Introduction
- **Approach**
  - *pMatlab Design*
  - ***gridMatlab***
- Results
- Future Work
- Summary

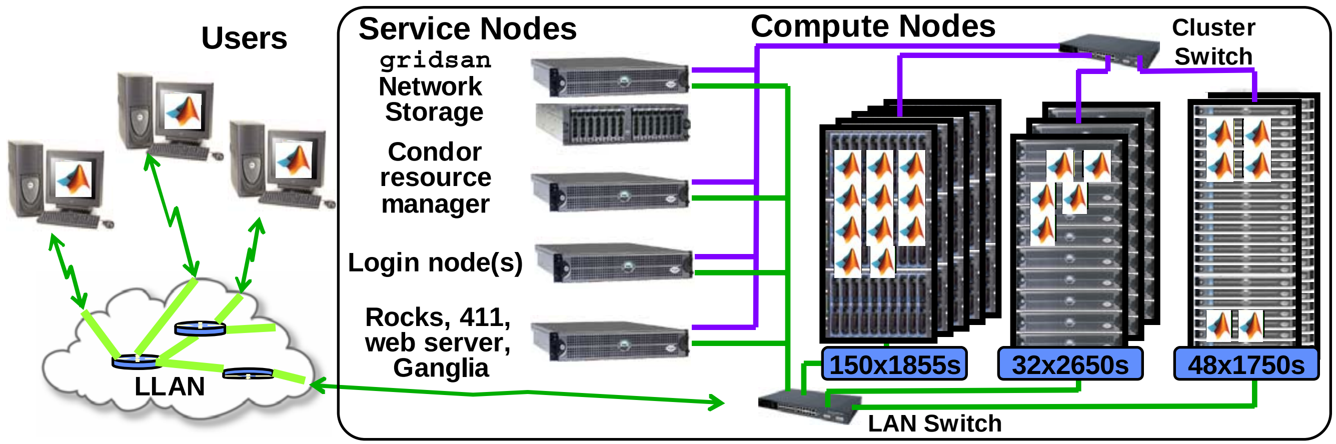
Outline

- Introduction
- **Approach**
  - *pMatlab Design*
  - *pMatlab Examples*
  - ***gridMatlab***
- Results
- Future Work
- Summary

Beta Grid Hardware

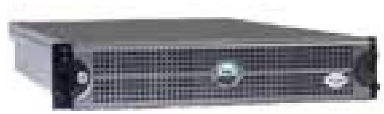
230 Nodes, 460 Processors, 1220 GB RAM

- Commodity Computers
- Commodity OS
- High Availability



Node Descriptions:

32 DELL PowerEdge 2650



- Dual 2.8 GHz Xeon (P4)
- 400 MHz front-side bus
- 4 GB RAM memory
- Two 36 GB SCSI hard drives
- 10/100 Mgmt Ethernet interface
- Two Gig-E Intel interfaces
- Running Red Hat Linux 9

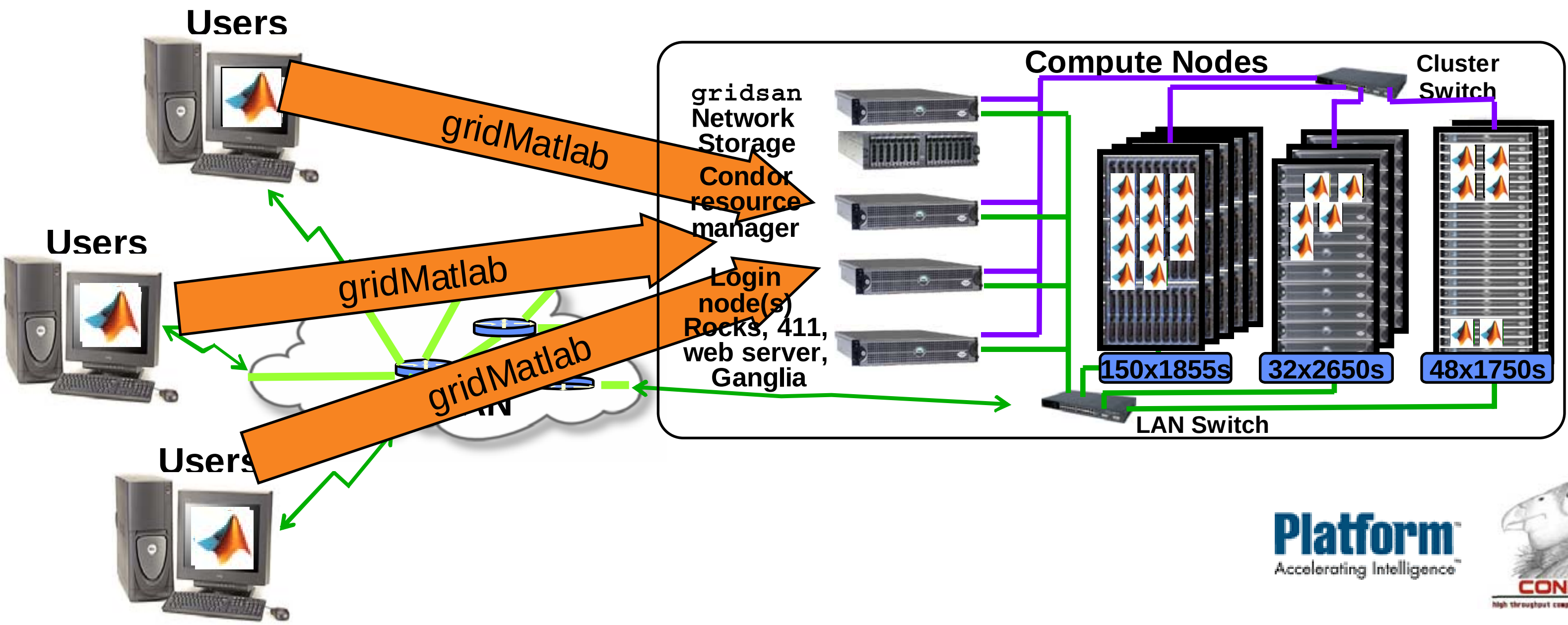
48 DELL PowerEdge 1750

- Dual 3.06 GHz Xeon (P4)
- 533 MHz front-side bus
- 4 GB RAM memory
- Two 36 GB SCSI hard drives
- 10/100 Mgmt Ethernet interface
- Two Gig-E Intel interfaces
- Running Red Hat Linux 9

15 x 10 DELL PowerEdge 1855MC

- Dual 3.2 GHz EM-64T Xeon (P4)
- 800 MHz front-side bus
- 6 GB RAM memory
- Two 144 GB SCSI hard drives
- 10/100 Mgmt Ethernet interface
- Two Gig-E Intel interfaces
- Running Red Hat Linux ES 3

Interactive, On-Demand HPC on LLGrid



GridMatlab adapts pMatlab to a grid environment

- User’s desktop system automatically pulled into the grid when a job is launched
  - full participating member of the grid computation
- Shared network file system as the primary communication interface
- Provides integrated set of gridMatlab services
- Allows interactive computing from the desktop

gridMatlab Functionality*

Job Launch

- Check if enough resources are available
- Build MPI_COMM_WORLD – job environment
- Write Linux launch shell scripts
- Write MATLAB launch scripts
- Write resource manager submit script
- Launch N-1 subjobs on cluster via resource manager
- Record job number
- Hand off to MPI_Rank=0 subjob

Job Abort

- Determine job number
- Issue job abort command via resource manager

Action	OpenPBS	SGE	Condor	LSF
Cluster status	`qstat`	`qstat`	`condor_status`	`bqueues`
Job launch	`qsub`	`qrsh`	`condor_submit`	`lsgrun`
Job abort	`qdel`	`qdel`	`condor_rm`	`bkill`

Users do not have to:

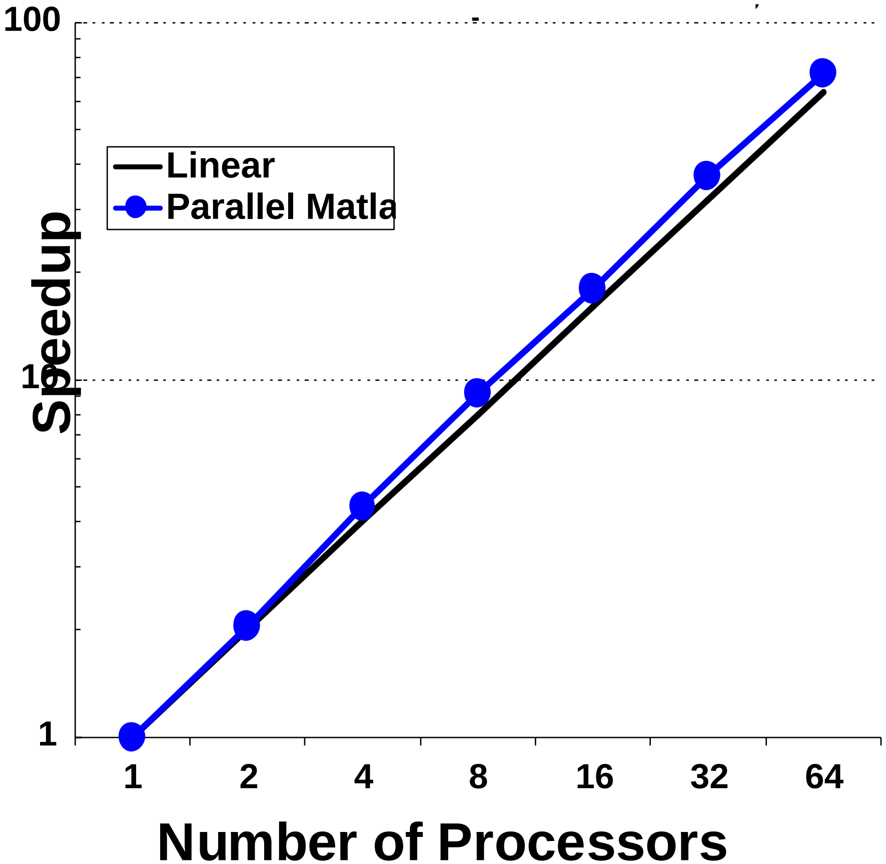
- Log into Linux cluster
- Write batch submit scripts
- Submit resource manager commands

Outline

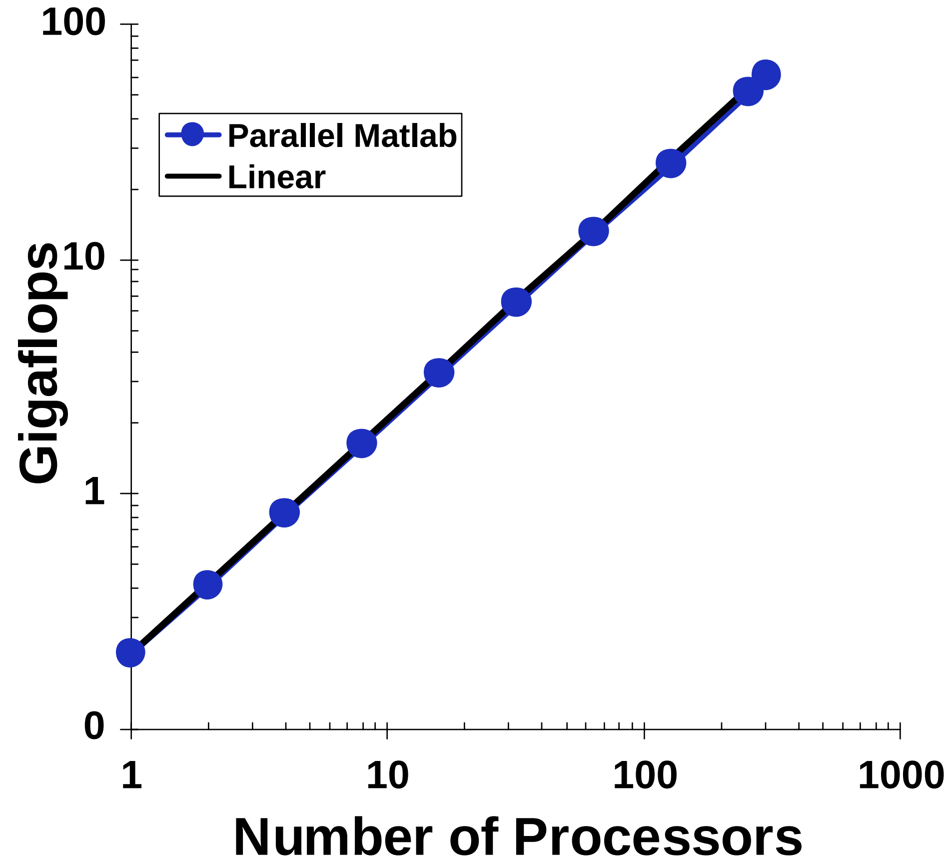
- Introduction
- Approach
- **Results**
  - *Performance Results*
  - *User Statistics*
- Future Work
- Summary

Speedup for Fixed and Scaled Problems

Fixed Problem Size



Scaled Problem Size



- Achieved “classic” super-linear speedup on fixed problem
- Achieved speedup of ~300 on 304 processors on scaled problem

HPCchallenge Benchmark Results

HPCchallenge Benchmark Results: C/MPI vs. pMatlab

	Maximum Problem Size	Execution Performance	Code Size: C/MPI to pMatlab ratio
RandomAccess	Comparable (128x)	Comparable	6x
Top500	pMatlab (86x) C/MPI (83x)	pMatlab (3x) C/MPI (35x)	66x
FFT	Comparable (128x)	Comparable (26x)	35x
STREAM	Comparable (128x)	Comparable (128x)	8x

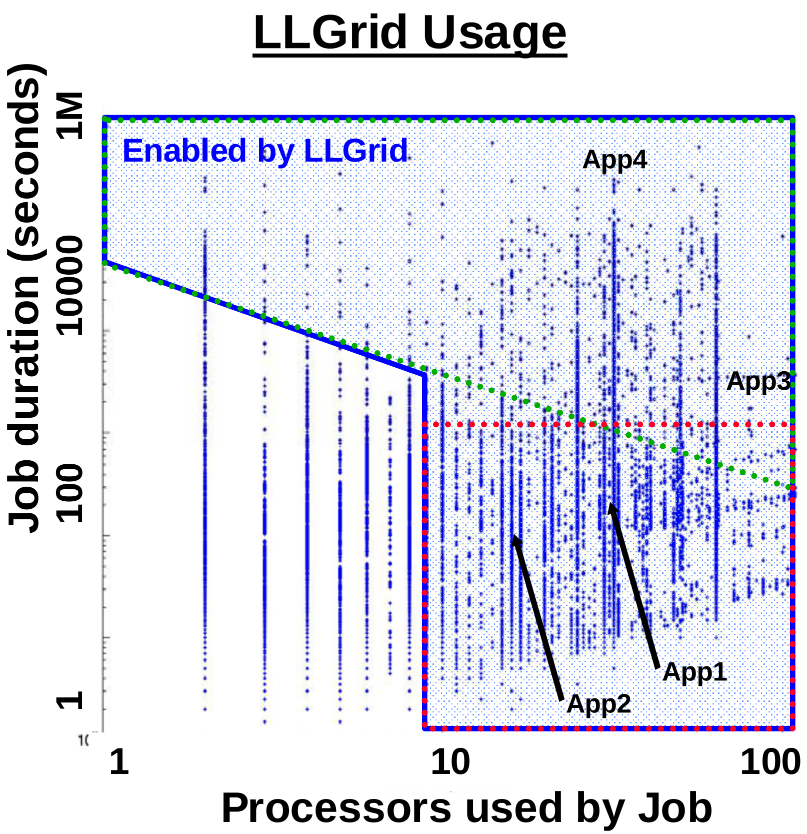
- 64 Dual processors Linux Cluster with Gigabit Ethernet
- Benchmark Results Summary:
  - pMatlab memory scalability comparable to C/MPI on nearly all of HPCchallenge. Allows Matlab users to work on much larger problems.
  - pMatlab execution performance comparable to C/MPI on nearly all of HPCchallenge. Allows Matlab users run their programs much faster.
  - pMatlab code size much smaller. Allows Matlab users to write programs much faster than C/MPI
- pMatlab allows Matlab users to effectively exploit parallel computing, and can achieve performance comparable to C/MPI.

Performance: Time to Parallelize

Important Considerations

Description	Serial Code Dev Time	Time to Parallelize	Applications that Parallelization Enables
Missile & Sensor BMD Sim. (BMD) - Group 38	2000 hours	8 hours	Discrimination simulations Higher fidelity radar simulations
First-principles LADAR Sim. (Ladar) - Group 38	1300 hours	1 hour	Speckle image simulations Aimpoint and discrimination studies
Analytic TOM Leakage Calc. (Leak) - Group 38	40 hours	0.4 hours	More complete parameter space sim.
Hercules Metric TOM Code (Herc) - Group 38	900 hours	0.75 hours	Monte carlo simulations
Coherent laser propagation sim. (Laser) - Group 94	40 hours	1 hour	Reduce simulation run time
Polynomial coefficient approx. (Coeff) - Group 102	700 hours	8 hours	Reduced run-time of algorithm training
Ground motion tracker indicator computation simulator (GMTI) - Group 102	600 hours	3 hours	Reduce evaluation time of larger data sets
Automatic target recognition (ATR) - Group 102	650 hours	40 hours	Ability to consider more target classes Ability to generate more scenarios
Normal Compositional Model for Hyper-spectral Image Analysis (HSI) Group 97	960 hours	6 hours	Larger datasets of images

LLgrid Usage December-03 – May-06



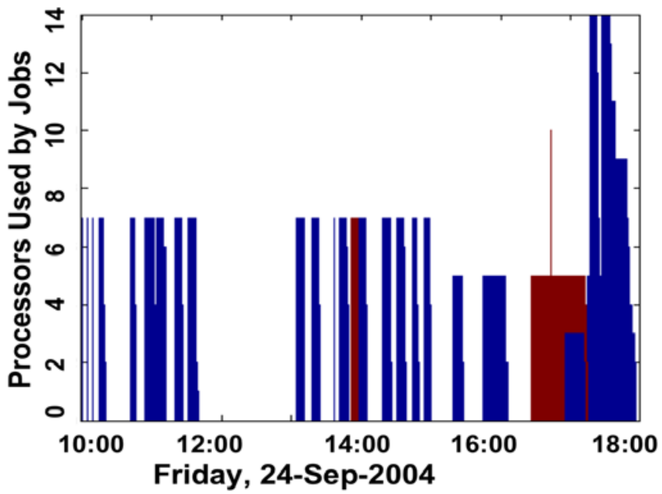
>8 CPU hours - Infeasible on Desktop
>8 CPUs - Requires On-Demand Parallel Computing

40,230 jobs, 24,100 CPU Days
December-03 – May-06

- Allowing Lincoln staff to effectively use parallel computing daily from their desktop
  - Interactive parallel computing
  - 186 CPUs, 110 Users, 19 Groups
- Extending the current space of data analysis and simulations that Lincoln staff can perform
  - Jobs requiring rapid turnaround
    - App1: Weather Radar Algorithm Development
    - App2: Biological Agent Propagation in Subways
  - Jobs requiring many CPU hours
    - App3: Non-Linear Equalization ASIC Simulation
    - App4: Hyper-Spectral Imaging

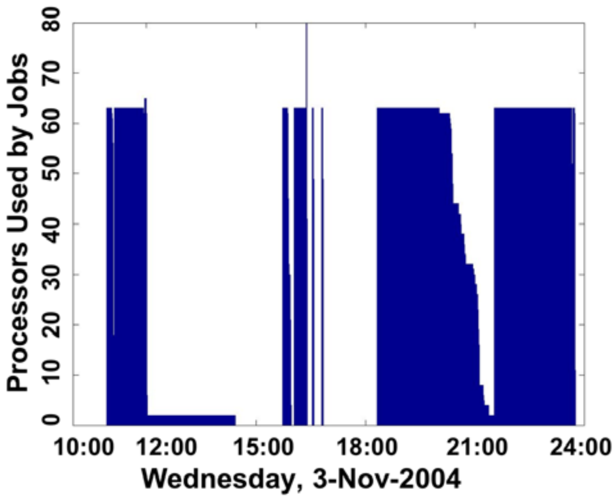
Individuals' Usage Examples

Weather Radar Algorithm Development



- Simulation results direct subsequent algorithm development and parameters
- Many engineering iterations during course of day

Non-Linear Equalization ASIC Simulation



- Post-run processing from overnight run
- Debug runs during day
- Prepare for long overnight runs

Selected Satellite Clusters*

- Sonar Lab
  - pMatlab/MatlabMPI, gridMatlab
- Missile descrimination
  - pMatlab/MatlabMPI
- Laser propagation simulation
  - Rocks, Condor
- LiMIT QuickLook
  - pMatlab/MatlabMPI, KickStart
- Satellite path propagation
  - Condor
- Other
  - Blades, Rocks, pMatlab/MatlabMPI, gridMatlab, Condor
- CEC Simulation
  - Blades, Rocks, pMatlab/MatlabMPI, gridMatlab, Condor
- Other
  - pMatlab/MatlabMPI, Condor

Outline

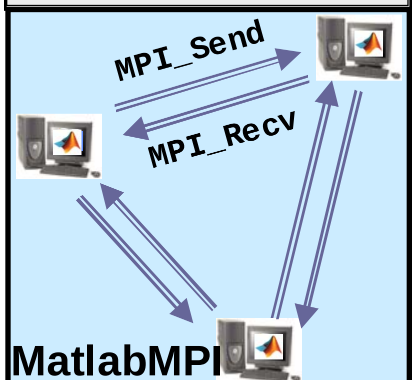
- Introduction
- Approach
- Results
- **Future Work**
  - ***Automatic Mapping***
  - ***Extreme Virtual Memory***
  - ***HPCMO Hardware***
- Summary

Evolution of Parallel Programming

EASE OF PROGRAMMING

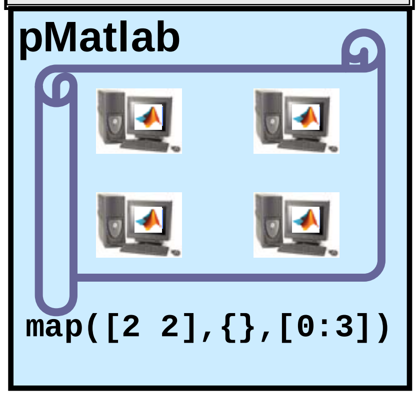
B(:,:)=fft(A)

```
my_rank=MPI_Comm_rank(comm);
if (my_rank==0)|(my_rank==1)|(my_rank==2)|(my_rank==3)
  A_local=rand(M,N/4);end
if (my_rank==4)|(my_rank==5)|(my_rank==6)|(my_rank==7)
  B_local=zeros(M/4,N);end
A_local=fft(A_local);
tag=0;if (my_rank==0)...MPI_Send(4,tag,comm,A_local(1:M/4,:);
elseif (my_rank==4)...B_local(:,1:N/4) = MPI_Recv(0,tag,comm);end
tag = tag+1;if (my_rank==0)...MPI_Send(5,tag,comm,A_local(M/4+1:2M/4,:);
elseif (my_rank==5)...B_local(:,1:N/4) = MPI_Recv(0,tag,comm);end
tag=tag+1;if (my_rank==0)...MPI_Send(6,tag,comm,A_local(2M/4+1:3M/4,:);
elseif (my_rank==6)...B_local(:,1:N/4) = MPI_Recv(0,tag,comm);end
tag=tag+1;if (my_rank==0)...MPI_Send(7,tag,comm,A_local(3M/4+1:M,:);
elseif (my_rank==7)...B_local(:,1:N/4) = MPI_Recv(0,tag,comm);end
tag=tag+1;if (my_rank==1)...MPI_Send(4,tag,comm,A_local(1:M/4,:);
elseif (my_rank==4)...B_local(:,N/4+1:2N/4) = MPI_Recv(1,tag,comm);end
tag=tag+1;if (my_rank==1)...MPI_Send(5,tag,comm,A_local(M/4+1:2M/4,:);
elseif (my_rank==5)...B_local(:,N/4+1:2N/4) = MPI_Recv(1,tag,comm);end
tag=tag+1;if (my_rank==1)...MPI_Send(6,tag,comm,A_local(2M/4+1:3M/4,:);
elseif (my_rank==6)...B_local(:,N/4+1:2N/4) = MPI_Recv(1,tag,comm);end
tag=tag+1;if (my_rank==1)...MPI_Send(7,tag,comm,A_local(3M/4+1:M,:);
elseif (my_rank==7)...B_local(:,N/4+1:2N/4) = MPI_Recv(1,tag,comm);end
tag=tag+1;if (my_rank==2)...MPI_Send(4,tag,comm,A_local(1:M/4,:);
elseif (my_rank==4)...B_local(:,2N/4+1:3N/4) = MPI_Recv(2,tag,comm);end
tag=tag+1;if (my_rank==2)...MPI_Send(5,tag,comm,A_local(M/4+1:2M/4,:);
elseif (my_rank==5)...B_local(:,2N/4+1:3N/4) = MPI_Recv(2,tag,comm);end
tag=tag+1;if (my_rank==2)...MPI_Send(6,tag,comm,A_local(2M/4+1:3M/4,:);
elseif (my_rank==6)...B_local(:,2N/4+1:3N/4) = MPI_Recv(2,tag,comm);end
tag=tag+1;if (my_rank==2)...MPI_Send(7,tag,comm,A_local(3M/4+1:M,:);
elseif (my_rank==7)...B_local(:,2N/4+1:3N/4) = MPI_Recv(2,tag,comm);end
tag=tag+1;if (my_rank==3)...MPI_Send(4,tag,comm,A_local(1:M/4,:);
elseif (my_rank==4)...B_local(:,3N/4+1:N) = MPI_Recv(3,tag,comm);end
tag=tag+1;if (my_rank==3)...MPI_Send(5,tag,comm,A_local(M/4+1:2M/4,:);
elseif (my_rank==5)...B_local(:,3N/4+1:N) = MPI_Recv(3,tag,comm);end
tag=tag+1;if (my_rank==3)...MPI_Send(6,tag,comm,A_local(2M/4+1:3M/4,:);
elseif (my_rank==6)...B_local(:,3N/4+1:N) = MPI_Recv(3,tag,comm);end
tag=tag+1;if (my_rank==3)...MPI_Send(7,tag,comm,A_local(3M/4+1:M,:);
elseif (my_rank==7)...B_local(:,3N/4+1:N) = MPI_Recv(3,tag,comm);end
```



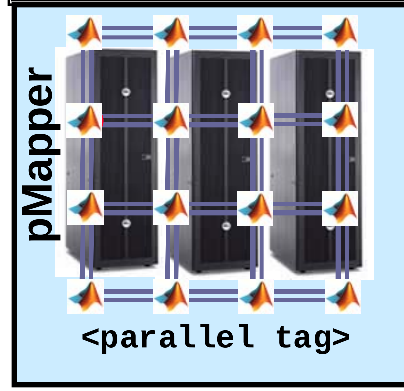
B(:,:)=fft(A)

```
mapA = map([1 4],{},[0:3]);
mapB = map([4 1],{},[4:7]);
A = rand(M,N,mapA);
B = zeros(M,N,mapB);
B(:,:) = fft(A);
```



B(:,:)=fft(A)

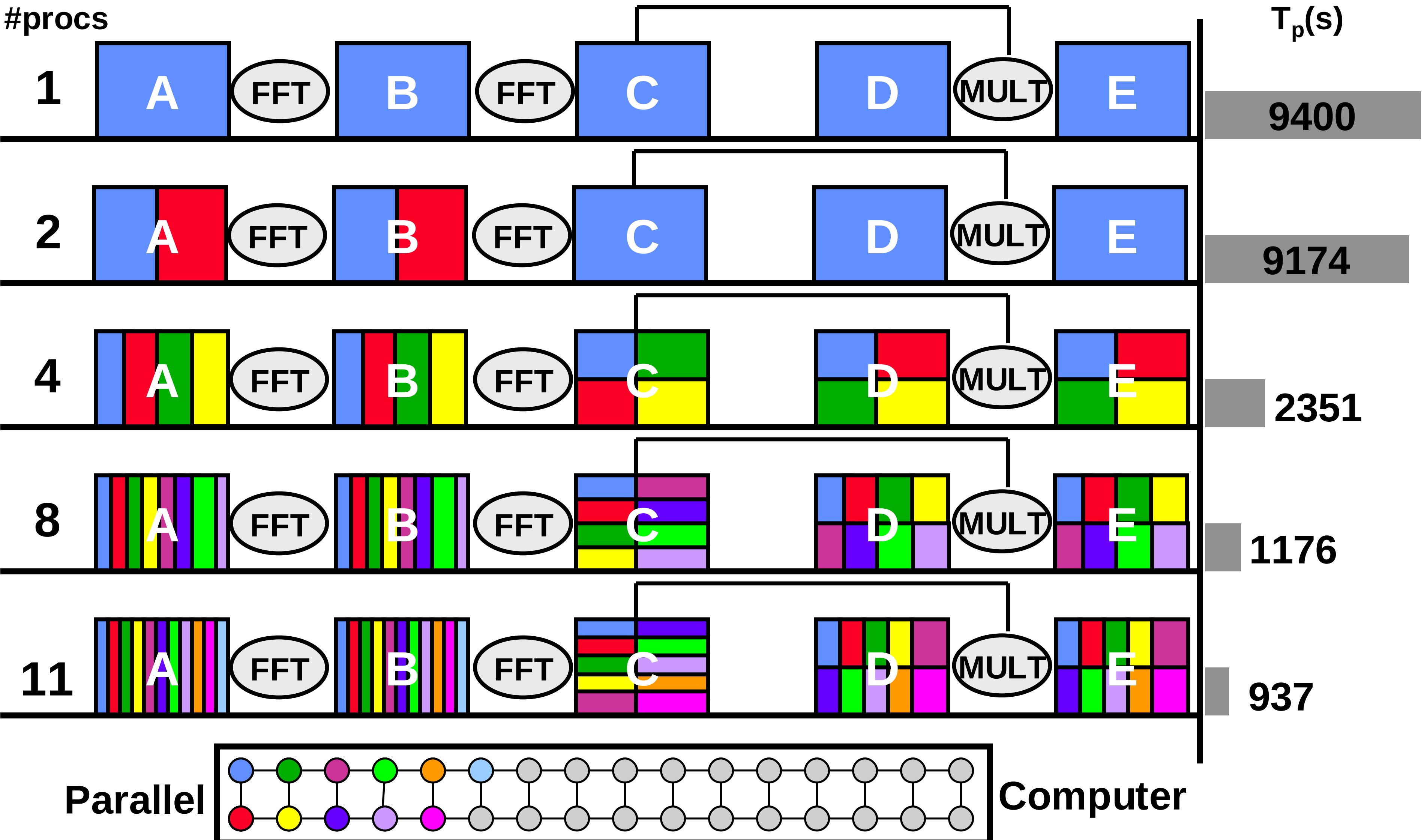
```
A = rand(M,N,p);
B = zeros(M,N,p);
B(:,:) = fft(A);
```



pMapper assumes the user is not a parallel programmer.

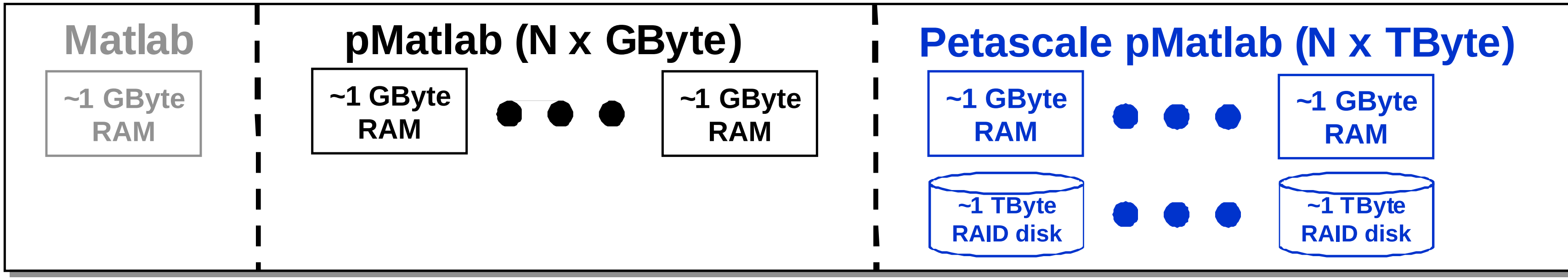
ABSTRACTION

pMapper Automatic Mapping

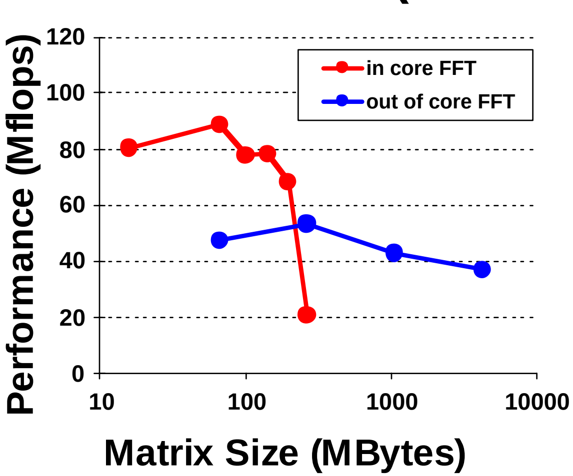


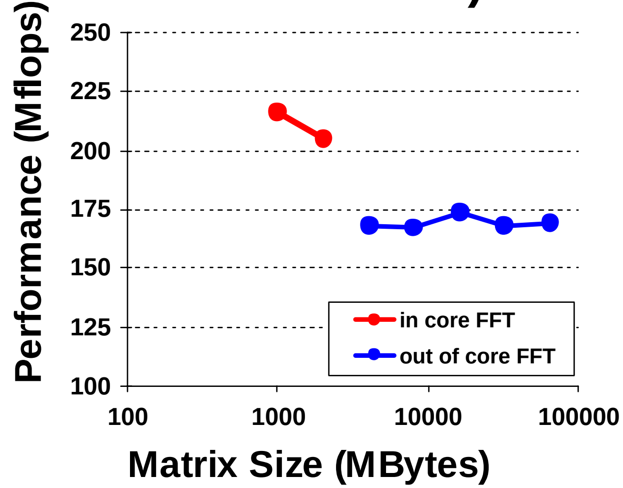
Parallel Out-of-Core

- Allows disk to be used as memory



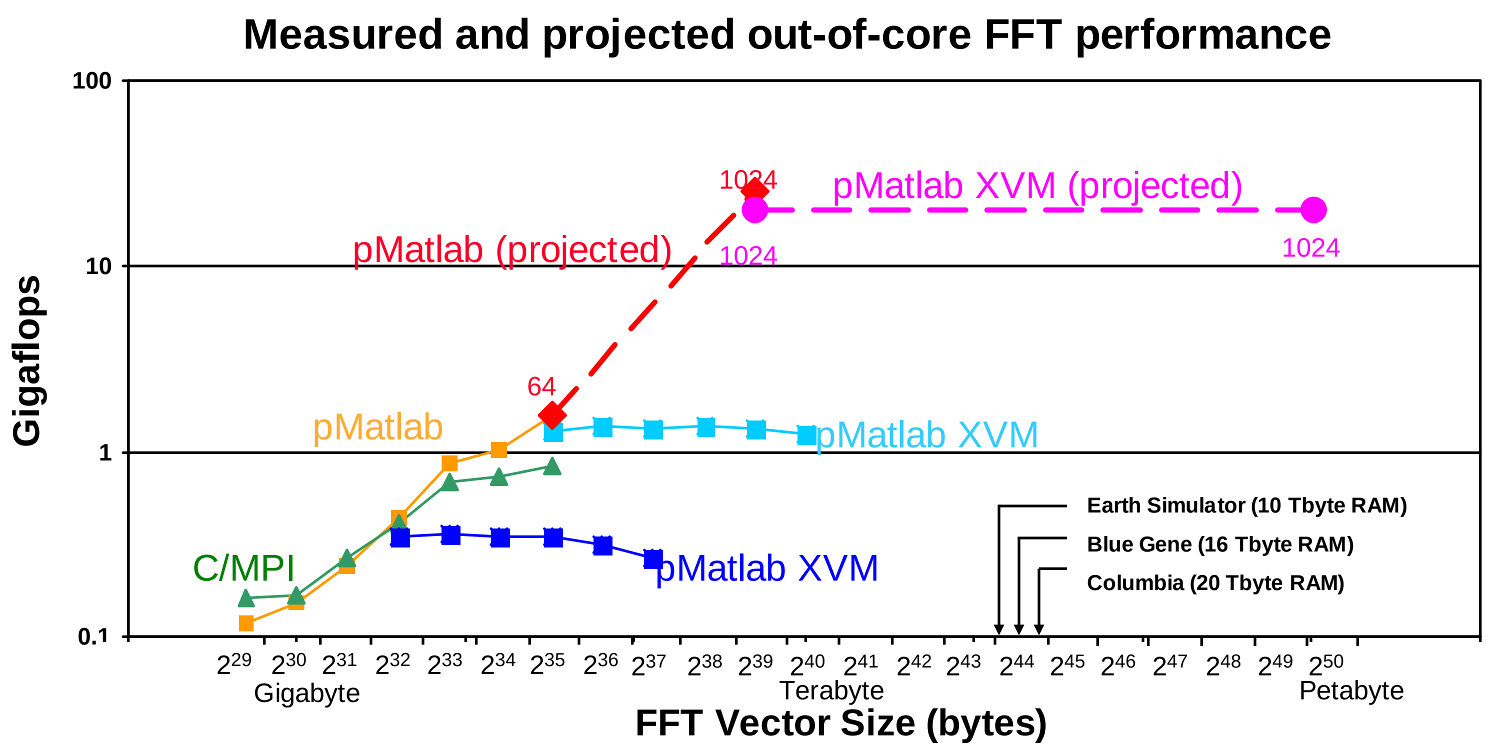
- Hand coded results (workstation and 4 node cluster)





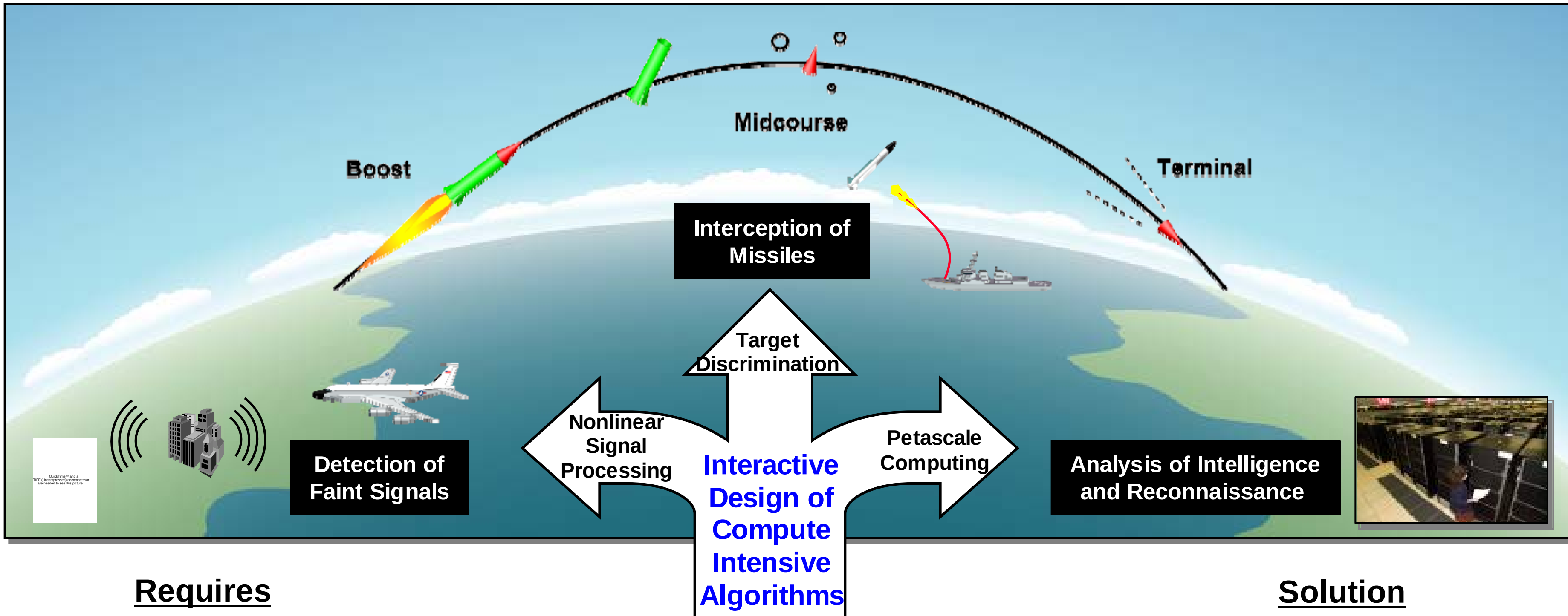
- pMatlab Approach
  - Add level of hierarchy to pMatlab maps; same partitioning semantics
  - Validate on HPCchallenge and other benchmarks

XVM parallel FFT performance



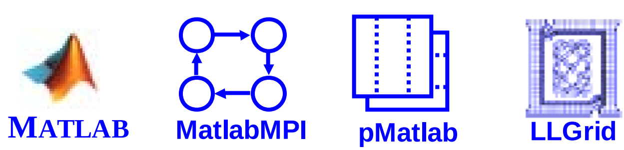
- **Out-of-core extreme virtual memory FFT (pMatlab XVM) scales well to 64 processors and 1 Terabyte of memory**
  - **Good performance relative to C/MPI; 80% efficient relative to in-core**
- **Petabyte FFT calculation should take ~9 days**
- **HPCchallenge and SSCA#1,2,3 should take a similar time**

High Performance Computing Proposal
- Multi-Layered WMD Defense Elements

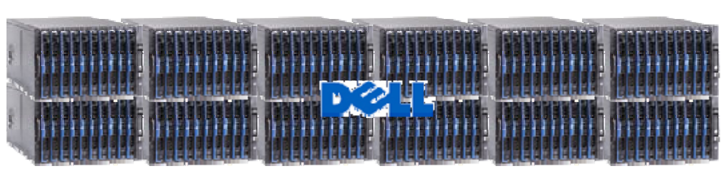


Requires
Iterative, Interactive Development

Solution
High Level Interactive Programming Environments



Requires
~10 Teraflops Computation
~1 Petabyte Virtual Memory



Solution
HPCMP Distributed HPC Project Hardware

Coming in 2006

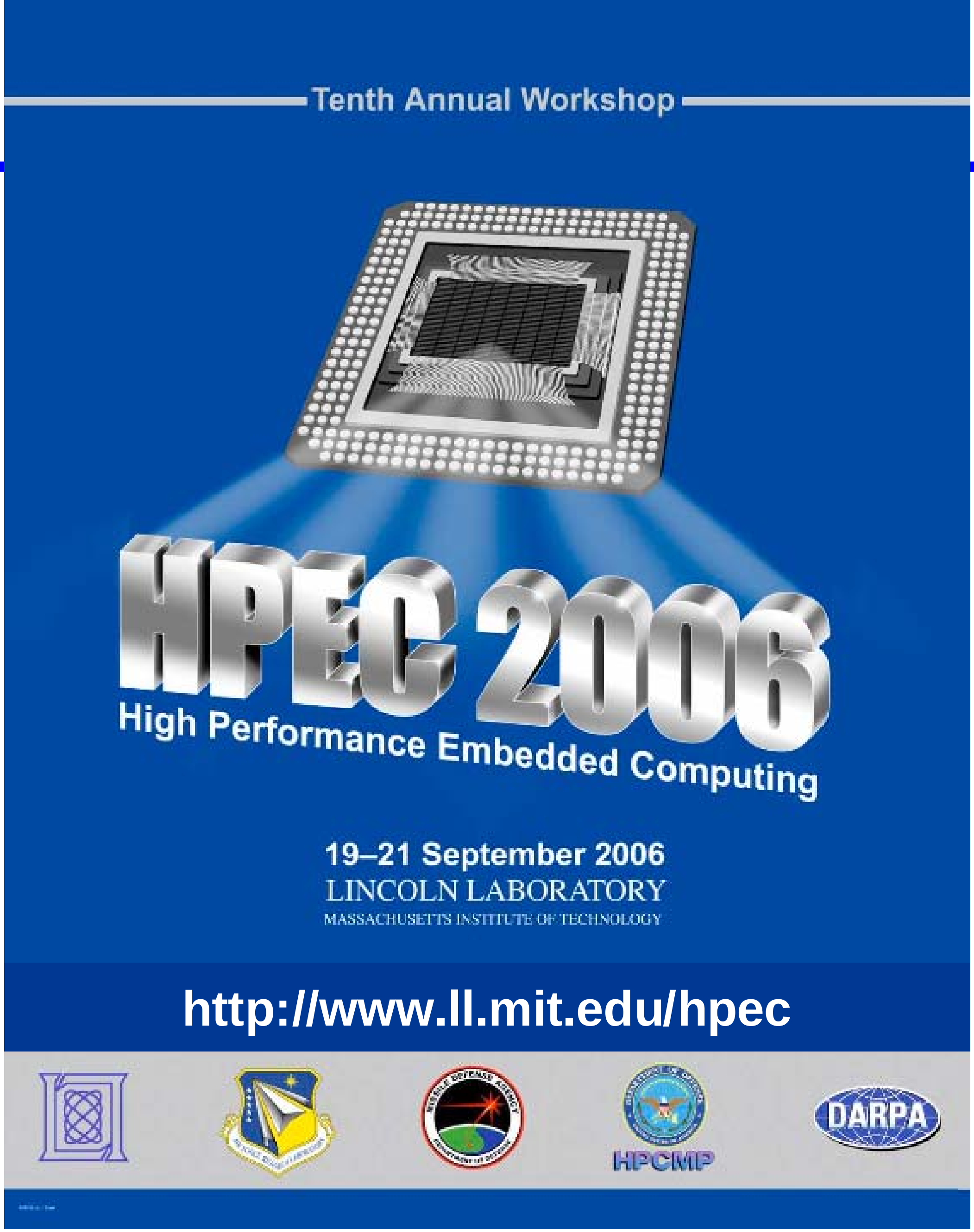
“Parallel Programming in MATLAB®”

by

Jeremy Kepner

SIAM (Society of Industrial and Applied Mathematics) Press series on Programming Environments and Tools

(series editor: Jack Dongarra)


Tenth Annual Workshop
HPEC 2006
High Performance Embedded Computing
19–21 September 2006
LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
http://www.ll.mit.edu/hpec
HPCMP
DARPA

Summary

- Goal: build a parallel Matlab system that is as easy to use as Matlab on the desktop
- Many daily users running jobs they couldn't run before
  - gridMatlab connects desktop computer to cluster
  - LLGrid allows account creation to first parallel job in <10 minutes
- Parallel Matlab has two main constructs:
  - Maps
  - Distributed arrays
- Parallel Matlab performance has been compared with C/MPI implementations of HPCchallenge
  - Memory and performance scalability is comparable on most benchmarks
  - Code is 6x to 60x smaller
- MathWorks has provided outstanding access to its products, design process, software engineers

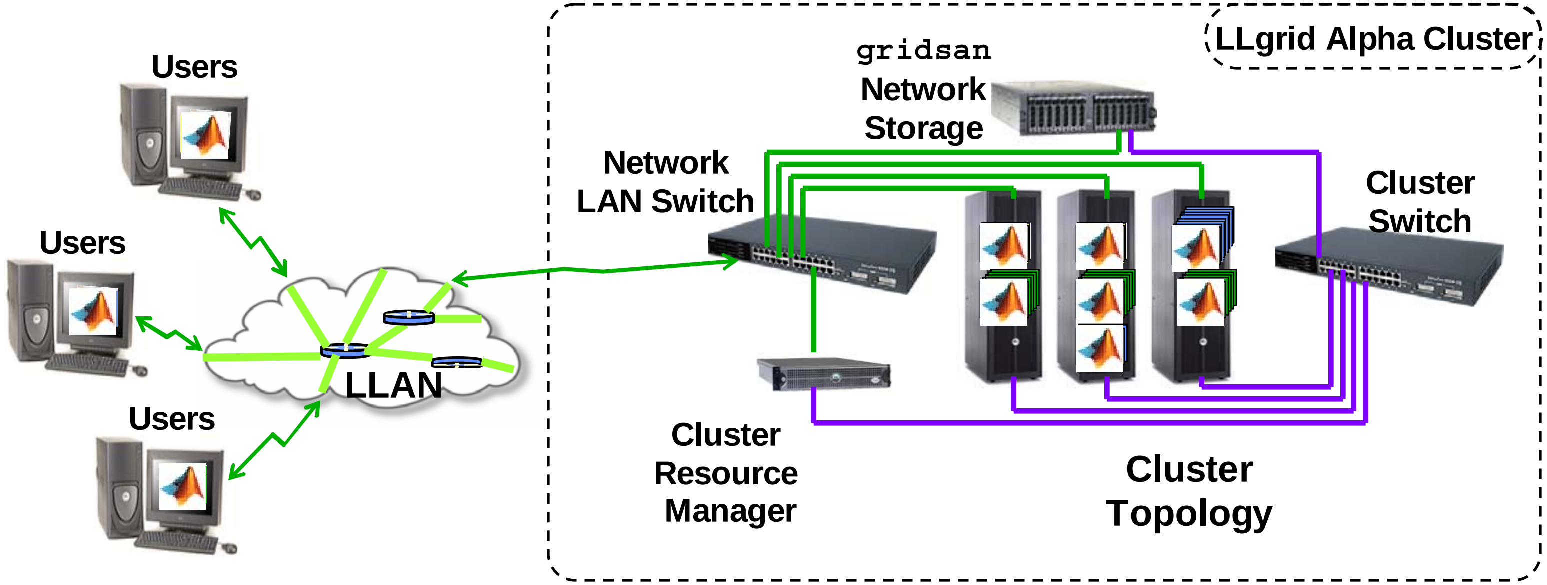
Summary

- **Goal: build a parallel Matlab system that is as easy to use as Matlab on the desktop**
- **LLGrid allows account creation to first parallel job in <10 minutes**
  - **gridMatlab connects desktop computer to cluster**
  - **Many daily users running jobs they couldn't run before**
- **Parallel Matlab has been tested on deployed systems**
  - **Allows in flight analysis of data**
- **Parallel Matlab performance has been compared with C/MPI implementations of HPCchallenge**
  - **Memory and performance scalability is comparable on most benchmarks**
  - **Code is 6x to 60x smaller**
- **MathWorks has provided outstanding access to its produces, design process, software engineers**

Backup Slides

Goal

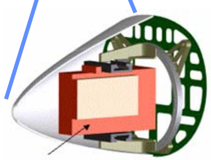
<u>Goal</u>: *To develop a grid computing capability that makes it as easy to run parallel Matlab programs on grid as it is to run Matlab on own workstation.*



Lab Grid Computing Components

- **Enterprise access to high throughput Grid computing**
- **Enterprise distributed storage**
- ***Real-time grid signal processing***

Example App: Prototype GMTI & SAR Signal Processing

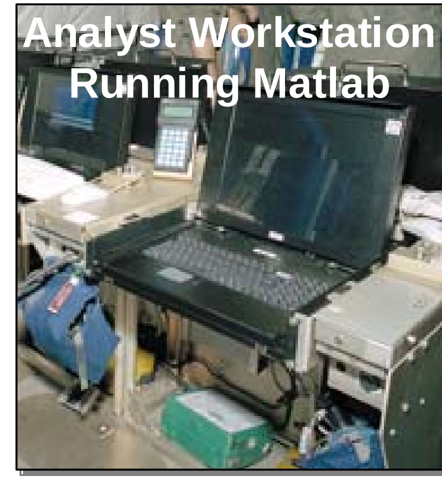


Research Sensor

On-board processing

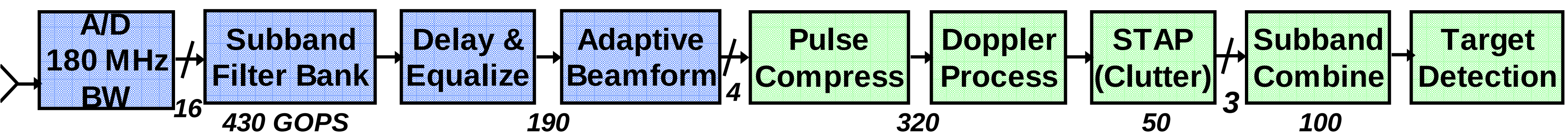
Streaming Sensor Data

SAR
GMTI
…
(new)



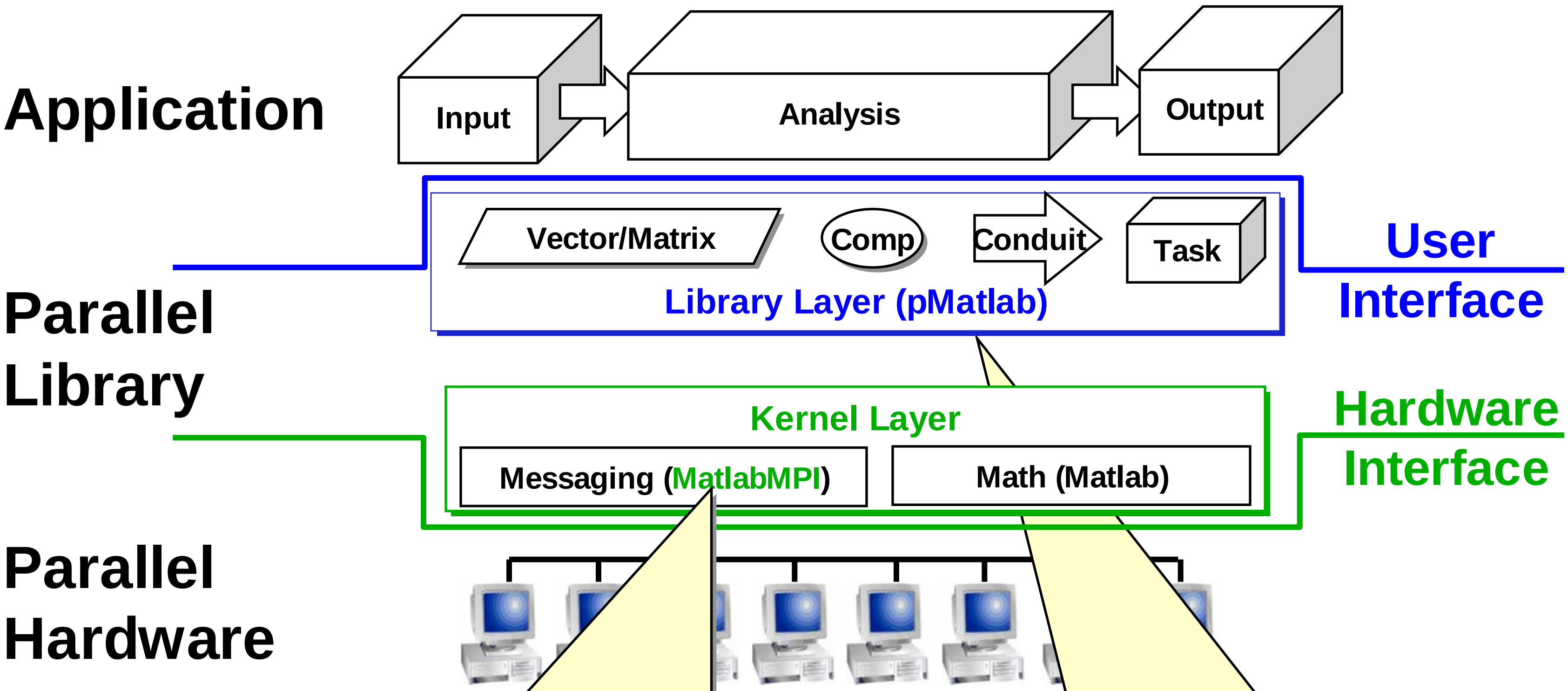
Real-time front-end processing

Non-real-time GMTI processing



- Airborne research sensor data collected
- Research analysts develop signal processing algorithms in MATLAB® using collected sensor data
- Individual runs can last hours or days on single workstation

MatlabMPI & pMatlab Software Layers



- Can build a parallel library with a few messaging primitives
- MatlabMPI provides this messaging capability:

```
MPI_Send(dest,comm,tag,X);
X = MPI_Recv(source,comm,tag);
```

- Can build applications with a few parallel structures and functions
- pMatlab provides parallel arrays and functions

```
X = ones(n,mapX);
Y = zeros(n,mapY);
Y(:,:) = fft(X);
```

pMatlab Support Functions

synch: synchronize the data in the distributed matrix.

agg: aggregates the parts of a distributed matrix on the leader processor.

agg_all: aggregates the parts of a distributed matrix on all processors in the map of the distributed matrix

global_block_range: returns the ranges of global indices local to the current processor.

global_block_ranges: returns the ranges of global indices for all processors in the map of distributed array D on all processors in communication scope.

global_ind: returns the global indices local to the current processor.

global_inds: returns the global indices for all processors in the map of distributed array D.

global_range: returns the ranges of global indices local to the current processor.

global_ranges: returns the ranges of global indices for all processors in the map of distributed array D.

local: returns the local part of the distributed array.

put_local: assigns new data to the local part of the distributed array.

grid: returns the processor grid onto which the distributed array is mapped.

inmap: checks if a processor is in the map.

Implementation Support (Lebak Levels)

Distribution Data Support Level

- L0 Distribution of data is not supported [not a parallel implementation]
- L1 One dimension of data may be block distributed
- L2 Two dimensions of data may be block distributed
- L3 Any and all dimensions of data may be block distributed
- L4 Any and all dimensions of data may be block or cyclicly distributed.

Note: Support for data distribution is assumed to include support for overlap in any distributed dimension

Distributed Operation Support Levels

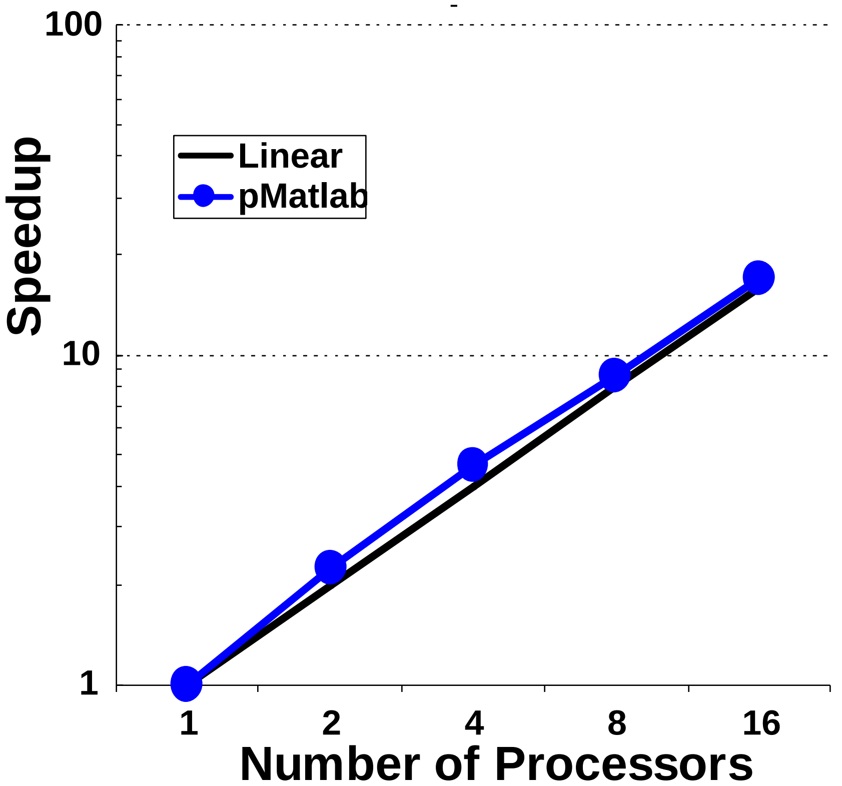
- L0 No distributed operations supported [not a parallel implementation]
- L1 Distributed assignment, get, and put operations, and support for obtaining data and indices of local data from a distributed object.
- L2 Distributed operation support (the implementation must state which operations those are)

- DataL4/OpL1 as been successfully implemented many times
- DataL1/OpL2 may be possible but has not yet been demonstrated
  - Semantic ambiguity between serial, replicated and distributed data
  - Optimal algorithms depend on distribution and array sizes

Clutter Simulation Example

(see pMatlab/examples/ClutterSim.m)

Fixed Problem Size (Linux Cluster)

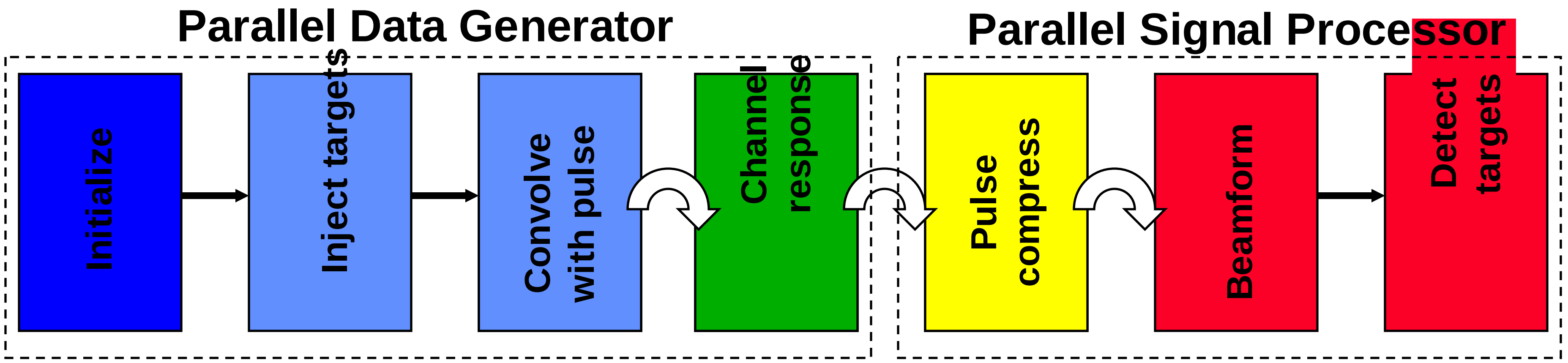



```
PARALLEL = 1;
mapX = 1;  mapY = 1;
% Initialize
% Map X to first half and Y to second half.
if (PARALLEL)
  pMatlab_Init; Ncpus=comm_vars.comm_size;
  mapX=map([1 Ncpus/2],{},[1:Ncpus/2])
  mapY=map([Ncpus/2 1],{},[Ncpus/2+1:Ncpus]);
end

% Create arrays.
X = complex(rand(N,M,mapX),rand(N,M,mapX));
Y = complex(zeros(N,M,mapY);

% Initialize coefficents
coefs = ...
weights = ...

% Parallel filter + corner turn.
Y(:,:) = conv2(coefs,X);
% Parallel matrix multiply.
Y(:,:) = weights*Y;

% Finalize pMATLAB and exit.
if (PARALLEL) pMatlab_Finalize;
```

- **Achieved “classic” super-linear speedup on fixed problem**
- **Serial and Parallel code “identical”**

Eight Stage Simulator Pipeline

(see pMatlab/examples/GeneratorProcessor.m)



Example Processor Distribution

- 0, 1
- 2, 3
- 4, 5
- 6, 7
- all

Matlab Map Code

```
map3 = map([2 1], {}, 0:1);
map2 = map([1 2], {}, 2:3);
map1 = map([2 1], {}, 4:5);
map0 = map([1 2], {}, 6:7);
```

- Goal: create simulated data and use to test signal processing
- parallelize all stages; requires 3 “corner turns”
- pMatlab allows serial and parallel code to be nearly identical
- Easy to change parallel mapping; set map=1 to get serial code

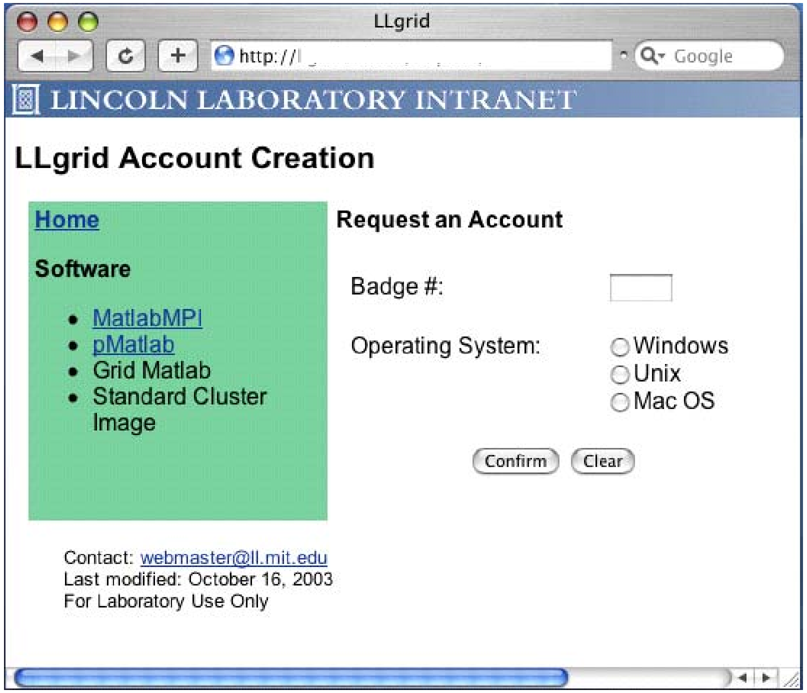
pMatlab Code

(see pMatlab/examples/GeneratorProcessor.m)

```
pMATLAB_Init; SetParameters; SetMaps;          %Initialize.
Xrand = 0.01*squeeze(complex(rand(Ns,Nb, map0),rand(Ns,Nb, map0)));
X0 = squeeze(complex(zeros(Ns,Nb, map0)));
X1 = squeeze(complex(zeros(Ns,Nb, map1)));
X2 = squeeze(complex(zeros(Ns,Nc, map2)));
X3 = squeeze(complex(zeros(Ns,Nc, map3)));
X4 = squeeze(complex(zeros(Ns,Nb, map3)));
...
for i_time=1:NUM_TIME                          % Loop over time steps.

  X0(:,:) = Xrand;                             % Initialize data
  for i_target=1:NUM_TARGETS
    [i_s i_c] = targets(i_time,i_target,:);
    X0(i_s,i_c) = 1;                           % Insert targets.
  end
  X1(:,:) = conv2(X0,pulse_shape,'same');      % Convolve and corner turn.
  X2(:,:) = X1*steering_vectors;               % Channelize and corner turn.
  X3(:,:) = conv2(X2,kernel,'same');           % Pulse compress and corner turn.
  X4(:,:) = X3*steering_vectors';              % Beamform.
  [i_range,i_beam] = find(abs(X4) > DET);      % Detect targets.
end
pMATLAB_Finalize;                              % Finalize.
```

■ Implicitly Parallel Code   ■ Required Change

LLGrid Account Creation



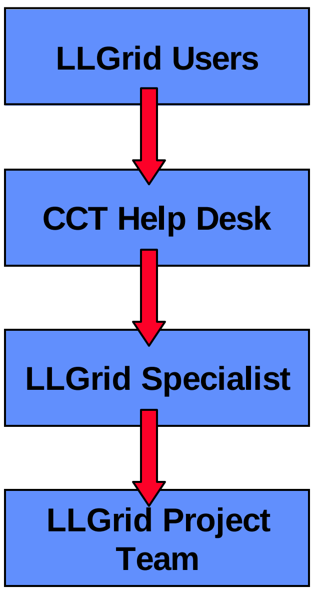
LLGrid Account Setup

- Go to Account Request web page; Type Badge #, Click "Create Account"
- Account is created and mounted on user's computer
- Get User Setup Script
- Run User Setup Script
- User runs sample job

• **Account Creation Script** (Run on LLGrid)	– Creates account on gridsan – Creates NFS & SaMBa mount points – Creates cross-mount communication directories
• **User Setup Script** (Run on User's Computer)	– Mounts gridsan – Creates SSH keys for grid resource access – Links to MatlabMPI, pMatlab, & gridMatlab source toolboxes – Links to MatlabMPI, pMatlab, & gridMatlab example scripts

Help Desk Integration

Moving Towards a Three-Tier Support Structure



- **LLGrid Users and LLGrid Team are Trained to Use …**

  grid-help@ll.mit.edu
  End User Support Mailing List

- **Hiring LLGrid Specialist**
- **Identifying Tasks That Help Desk Can Perform**
- **Escalate Users Requests**

Web Interest

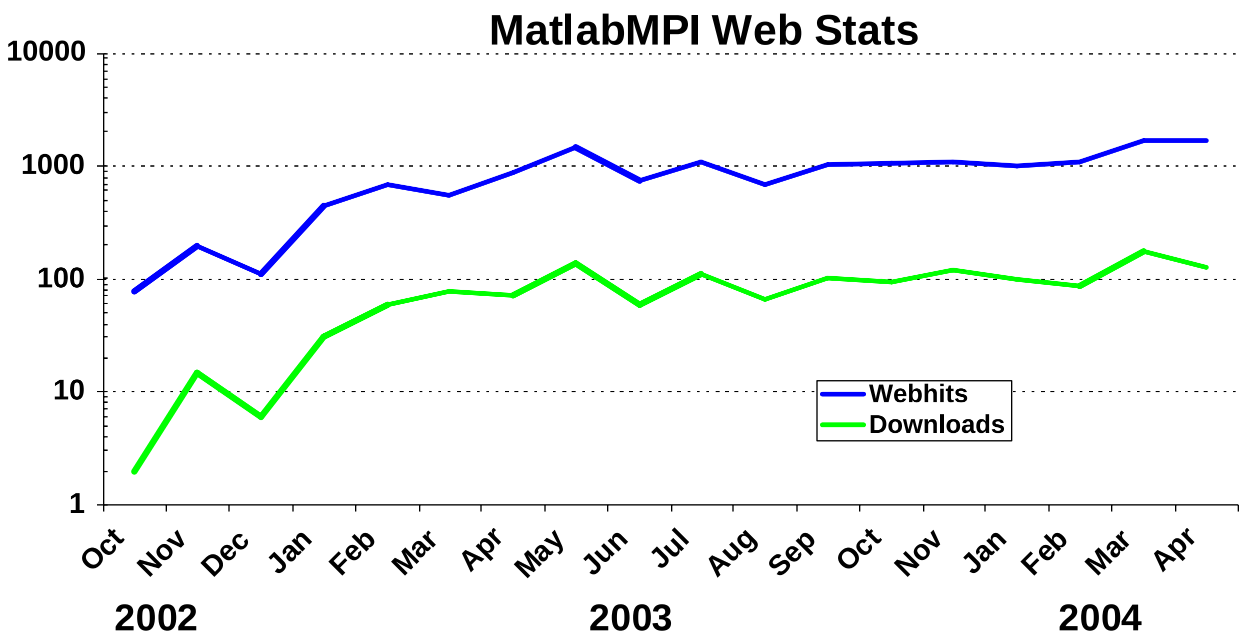

HPC wire

The global publication of record for High Performance Computing / April 16, 2004: Vol. 13, No. 15

News Briefs - Software:

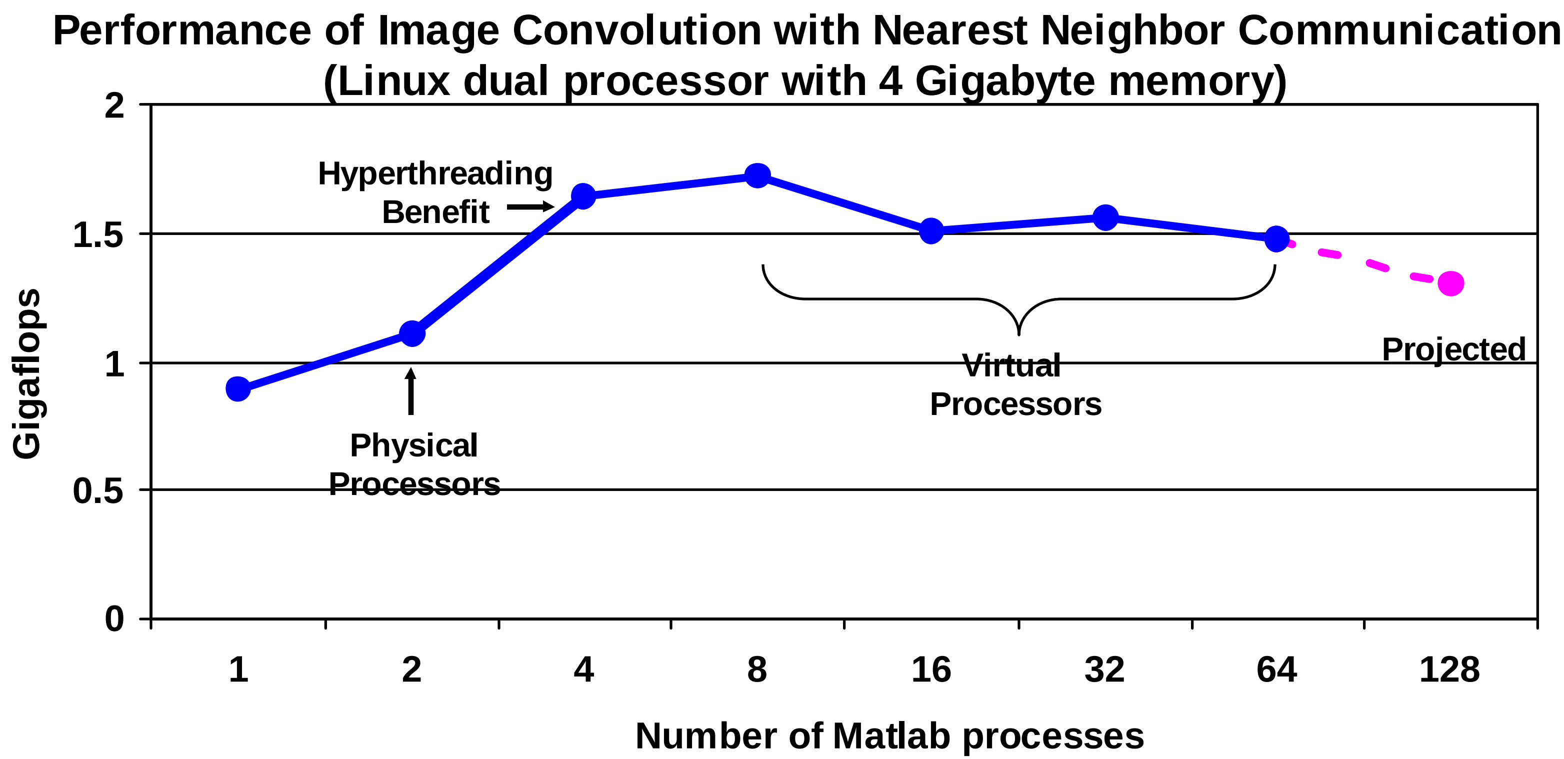
MIT Lincoln Lab, Ohio Researchers Deploy MatlabMPI Solution

For decades, high performance computing (HPC) researchers have struggled with low level programming environments to exploit parallel computers.



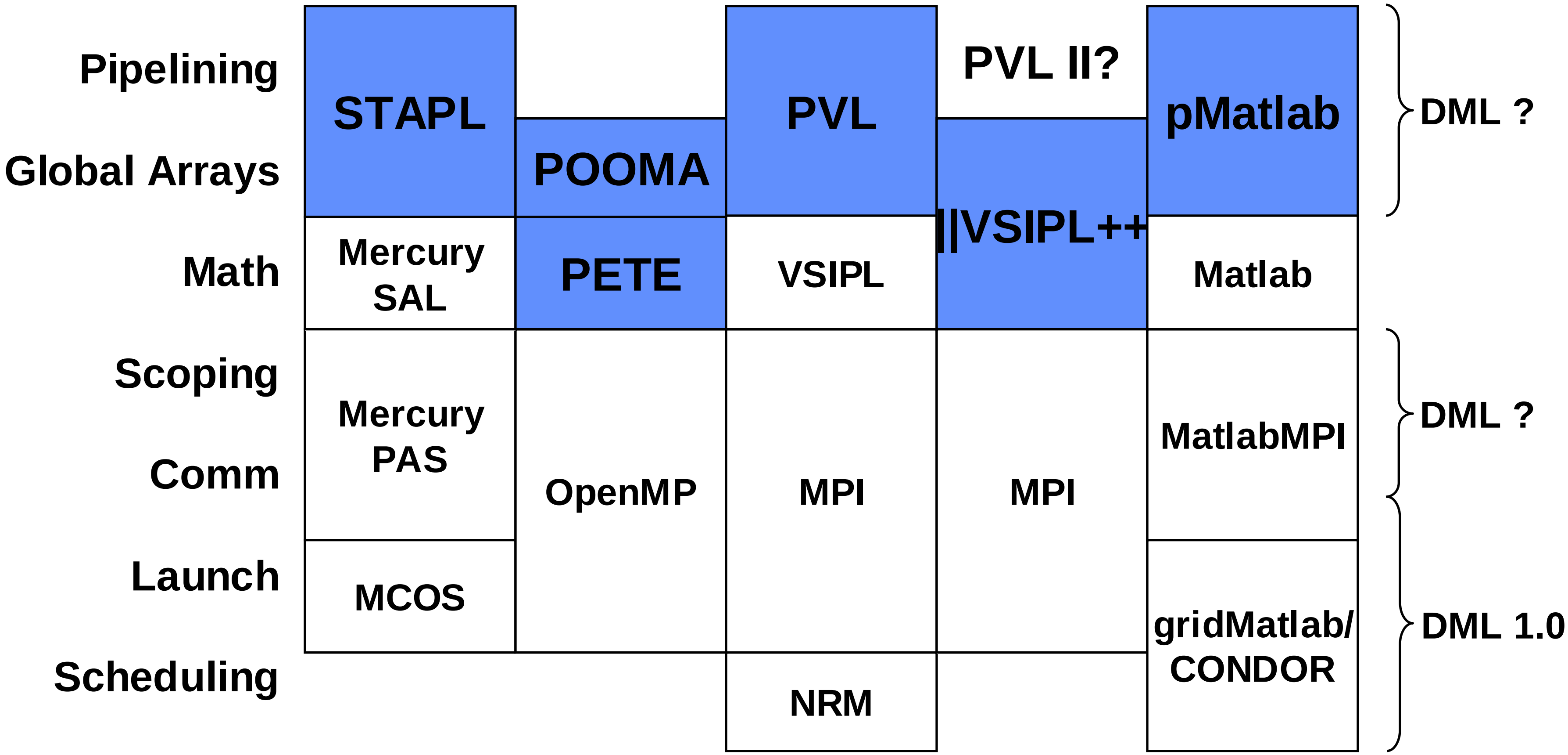
- **Hundreds of MatlabMPI users worldwide?**

Virtual Processor Performance



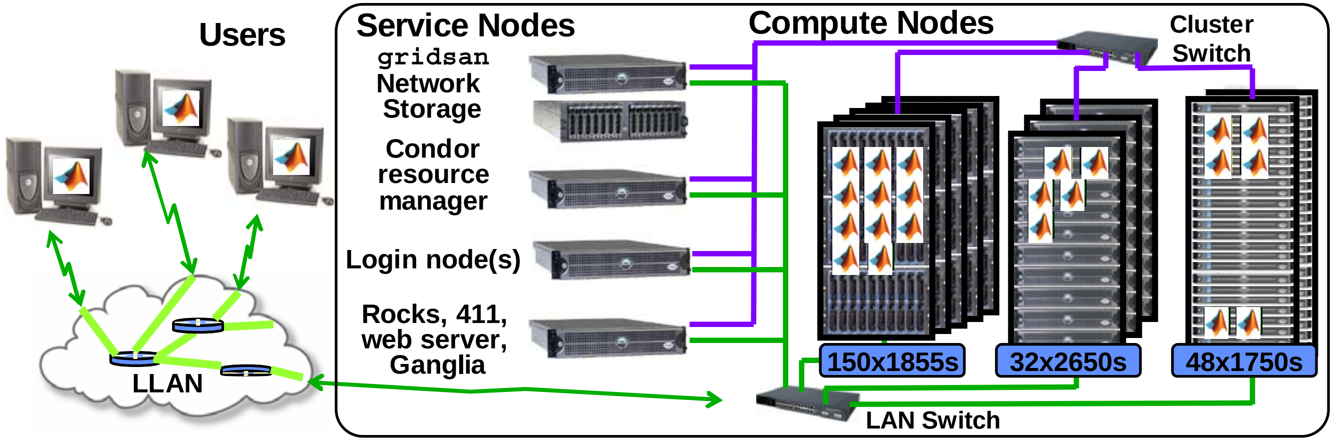
- **Can simulate 64+ processors on dual processor system**
  - Initial performance benefit from hyperfthreading
  - Small performance hit
- **pMatlab XVM and MatlabMPI provides necessary**
  - Very small per process working set; highly asynchronous messaging
- **Should be able simulate 64,000 processors on 512 node system**

Layered Architecture



- The “correct” layered architecture for parallel libraries is probably the principal achievement of HPC software research of the 1990s
- Mathworks DML is the first step in this ladder

Summary



- Many different signal processing applications at Lincoln
- LLGrid System: commodity hardware, pMatlab, gridMatlab
- Enabling Interactive, On-Demand HPC
- 90 users, 17,040 CPU days of CPU time
- Scaling up to 1024 CPU system in the future
- Releasing pMatlab to open source: http://www.ll.mit.edu/pMatlab/