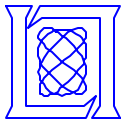
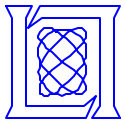


300x Matlab

Dr. Jeremy Kepner
MIT Lincoln Laboratory

September 25, 2002
HPEC Workshop
Lexington, MA

This work is sponsored by the High Performance Computing Modernization Office under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the Department of Defense.



Outline

- **Introduction**
  - *Motivation*
  - *Challenges*
- Approach
- Performance Results
- Future Work and Summary

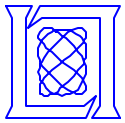
Motivation: DoD Need

- **Cost**

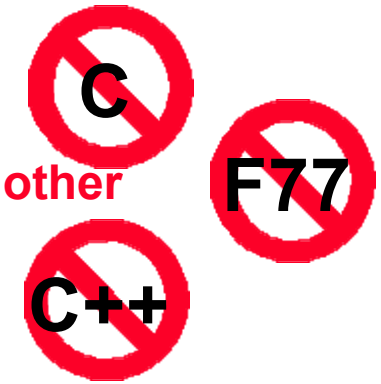
= 4 lines of DoD code

- **DoD has a clear need to rapidly develop, test and deploy new techniques for analyzing sensor data**
  - **Most DoD algorithm development and simulations are done in Matlab**
  - **Sensor analysis systems are implemented in other languages**
  - **Transformation involves years of software development, testing and system integration**

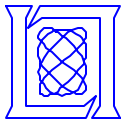
- **MatlabMPI allows any Matlab program to become a high performance parallel program**



Challenges: Why Has This Been Hard?



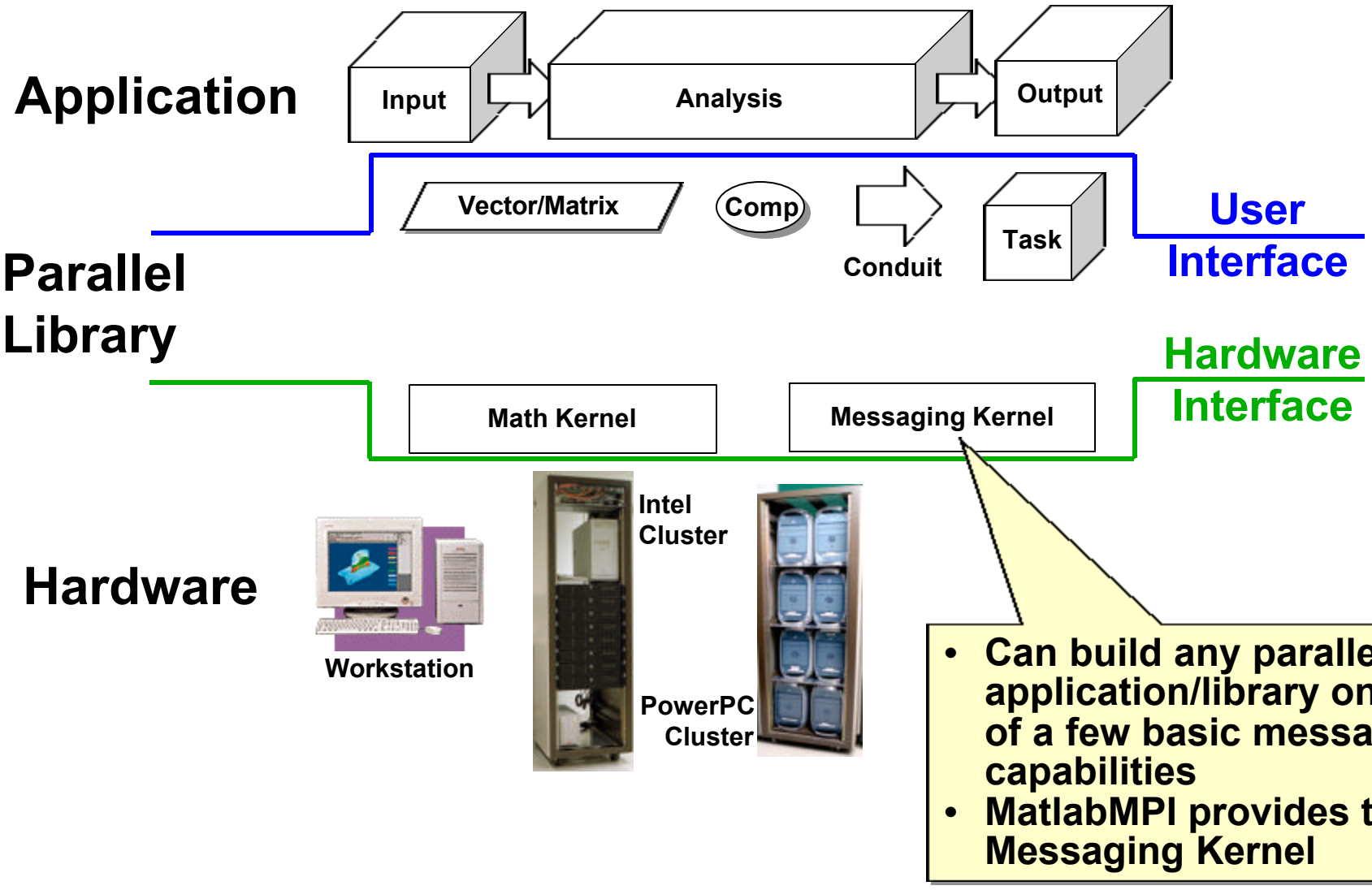
- **Productivity**
  - **Most users will not touch any solution that requires other languages (even cmex)**
- **Portability**
  - **Most users will not use a solution that could potentially make their code non-portable in the future**
- **Performance**
  - **Most users want to do very simple parallelism**
  - **Most programs have long latencies (do not require low latency solutions)**

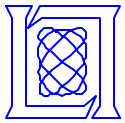


Outline

- Introduction
- **Approach**
  - ***Basic Requirements***
  - ***File I/O based messaging***
- Performance Results
- Future Work and Summary

Modern Parallel Software Layers



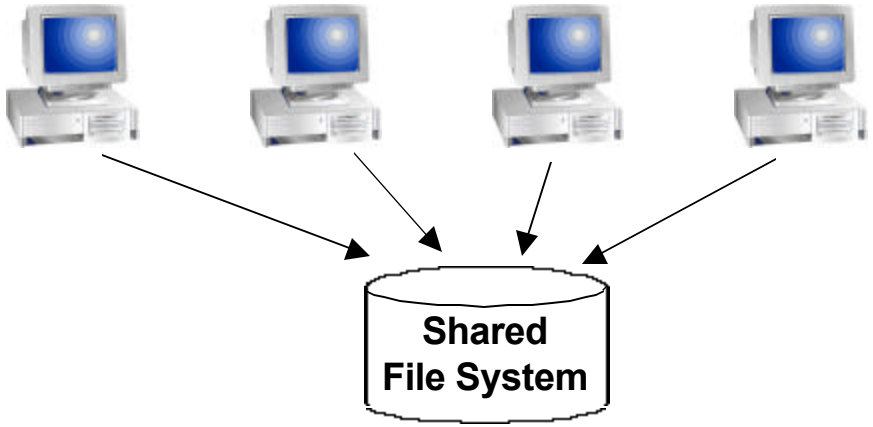


MatlabMPI "Core Lite"

- Parallel computing requires eight capabilities
  - MPI_Run launches a Matlab script on multiple processors
  - MPI_Comm_size returns the number of processors
  - MPI_Comm_rank returns the id of each processor
  - MPI_Send sends Matlab variable(s) to another processor
  - MPI_Recv receives Matlab variable(s) from another processor
  - MPI_Init called at beginning of program
  - MPI_Finalize called at end of program

Key Insight: File I/O based messaging

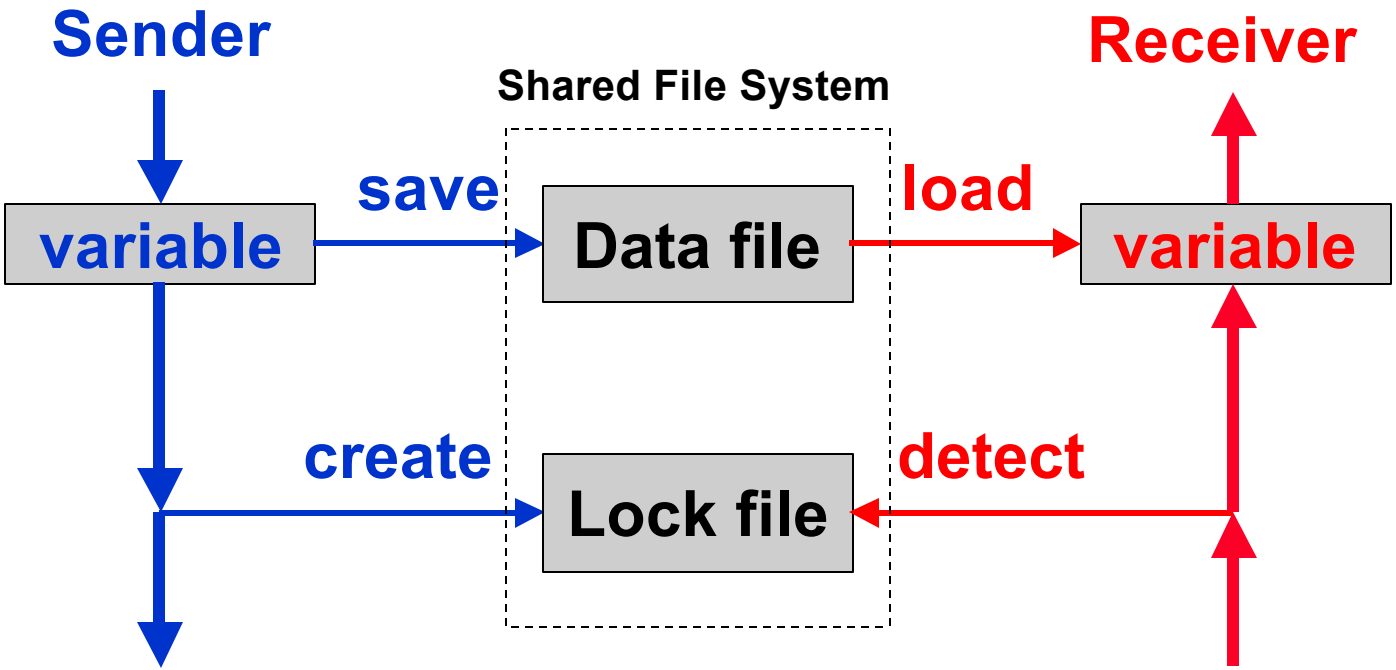
- Any messaging system can be implemented using file I/O



- 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

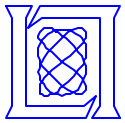
MatlabMPI: Point-to-point Communication

MPI_Send (dest, tag, comm, variable);



variable = MPI_Recv (source, tag, comm);

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



Example: Basic Send and Receive

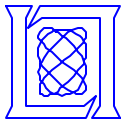
- Initialize
- Get processor ranks

- Execute send
- Execute recieve

- Finalize
- Exit

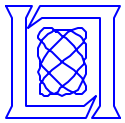
```
MPI_Init;                              % Initialize MPI.
comm = MPI_COMM_WORLD;                 % Create communicator.
comm_size = MPI_Comm_size(comm);       % Get size.
my_rank = MPI_Comm_rank(comm);         % Get rank.
source = 0;                            % Set source.
dest = 1;                              % Set destination.
tag = 1;                               % Set message tag.

if(comm_size == 2)                     % Check size.
 if (my_rank == source)                % If source.
   data = 1:10;                        % Create data.
   MPI_Send(dest,tag,comm,data);       % Send data.
 end
 if (my_rank == dest)                  % If destination.
   data=MPI_Recv(source,tag,comm);     % Receive data.
 end
end

MPI_Finalize;                          % Finalize Matlab MPI.
exit;                                  % Exit Matlab
```

- Uses standard message passing techniques
- Will run anywhere Matlab runs
- Only requires a common file system



MatlabMPI Additional Functionality

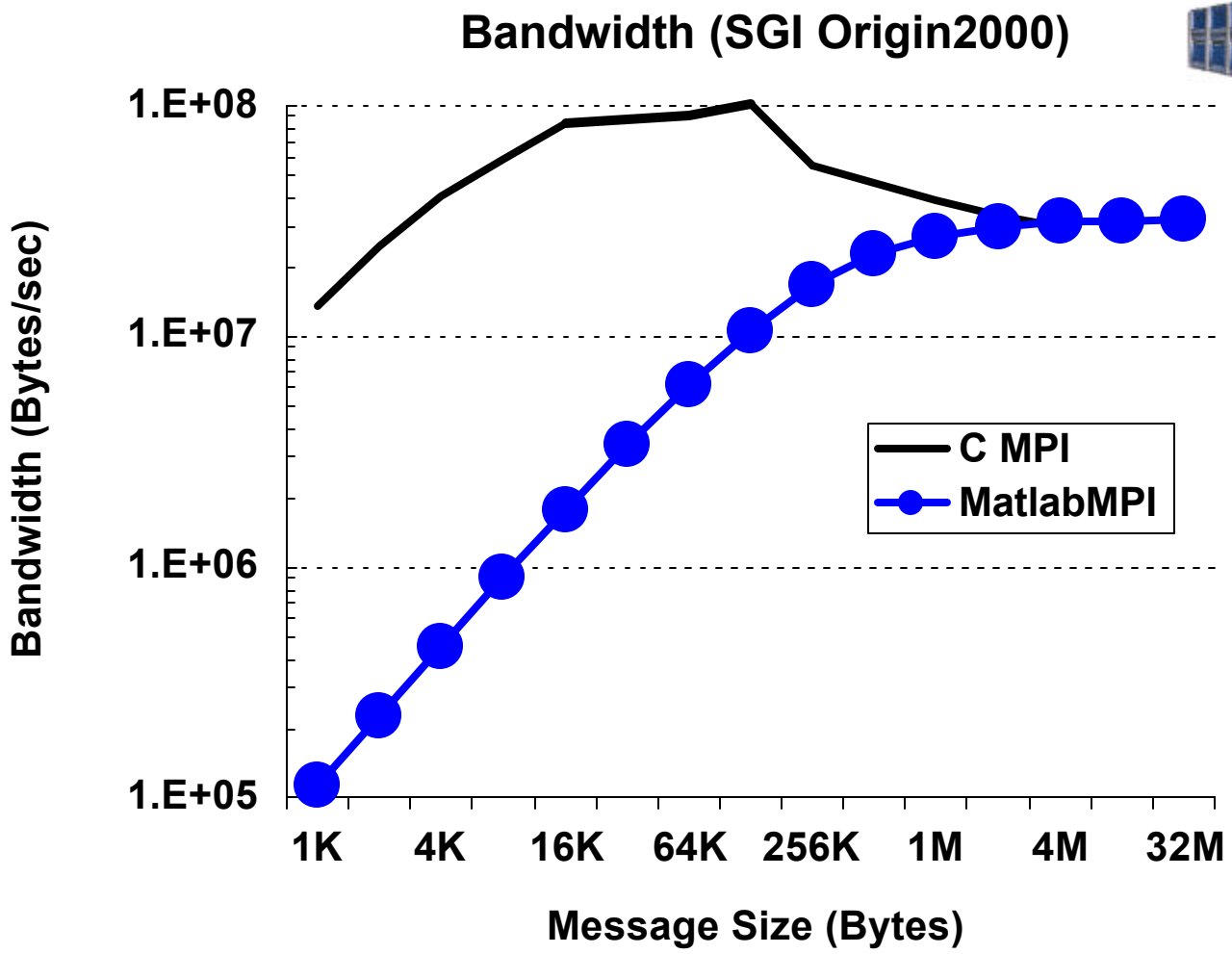
- Important MPI conveniences functions
  - MPI_Abort kills all jobs
  - MPI_Bcast broadcasts a message
    exploits symbolic links to allow for true multi-cast
  - MPI_Probe returns a list of all incoming messages
    allows more dynamic message reading
- MatlabMPI specific functions
  - MatMPI_Delete_all cleans up all files after a run
  - MatMPI_Save_messages toggles deletion of messages
    individual messages can be inspected for debugging
  - MatMPI_Comm_settings user can set MatlabMPI internals
    rsh or ssh, location of Matlab, unix or windows, …
- Other
  - Processor specific directories



Outline

- Introduction
- Approach
- **Performance Results**
  - ***Bandwidth***
  - ***Parallel Speedup***
- Future Work and Summary

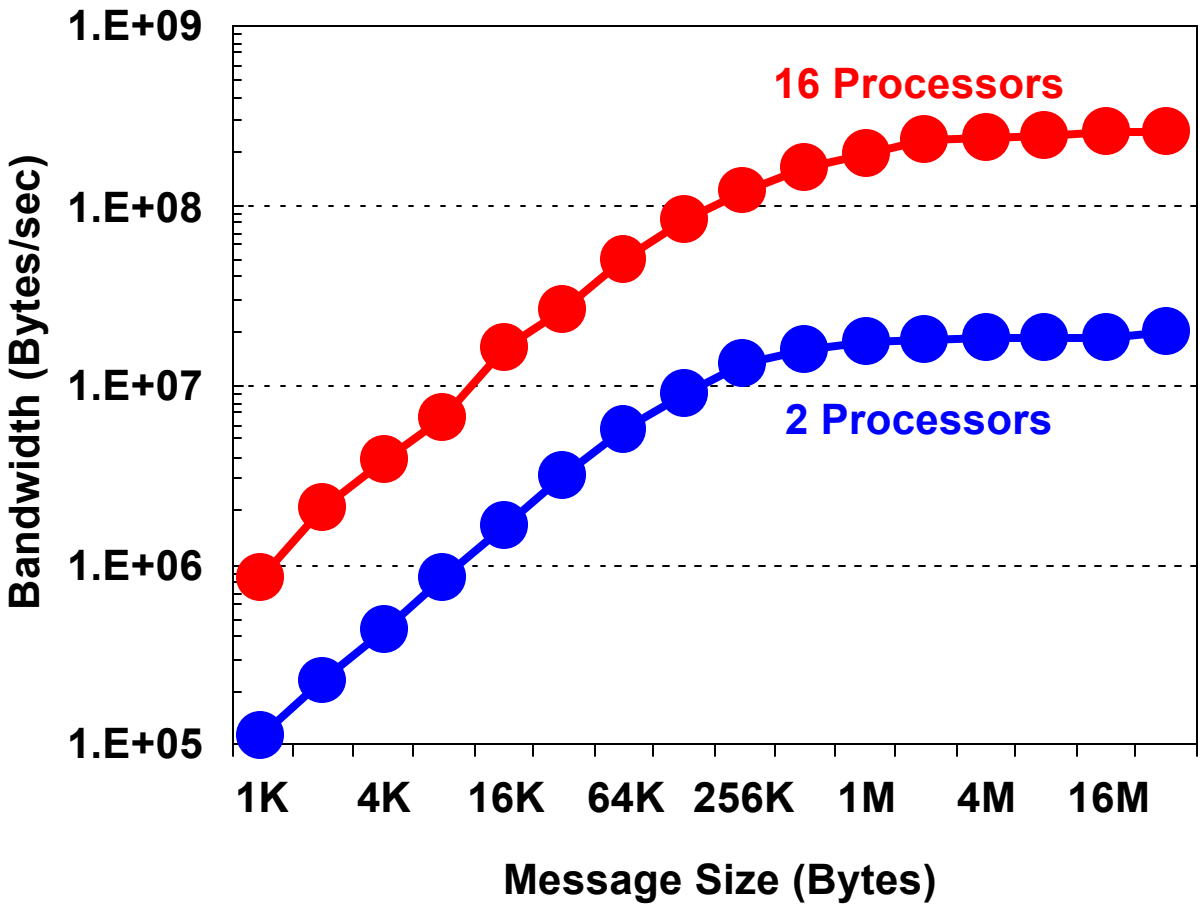
MatlabMPI vs MPI bandwidth



- Bandwidth matches native C MPI at large message size
- Primary difference is latency (35 milliseconds vs. 30 microseconds)

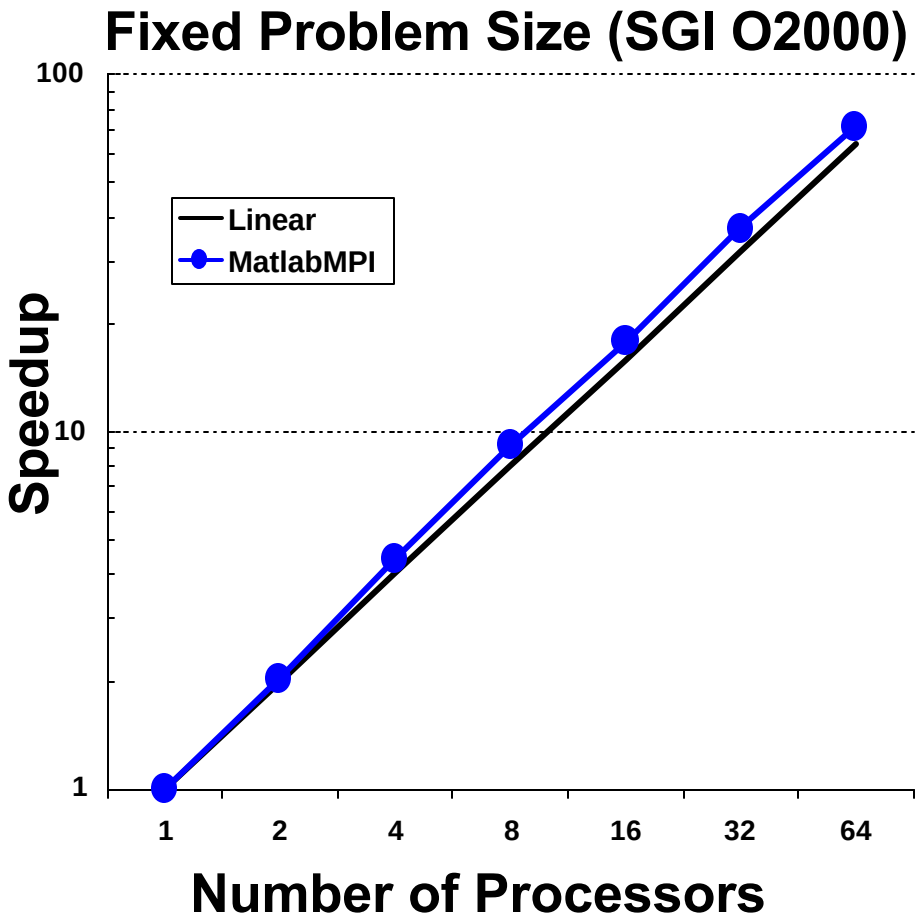
MatlabMPI bandwidth scalability

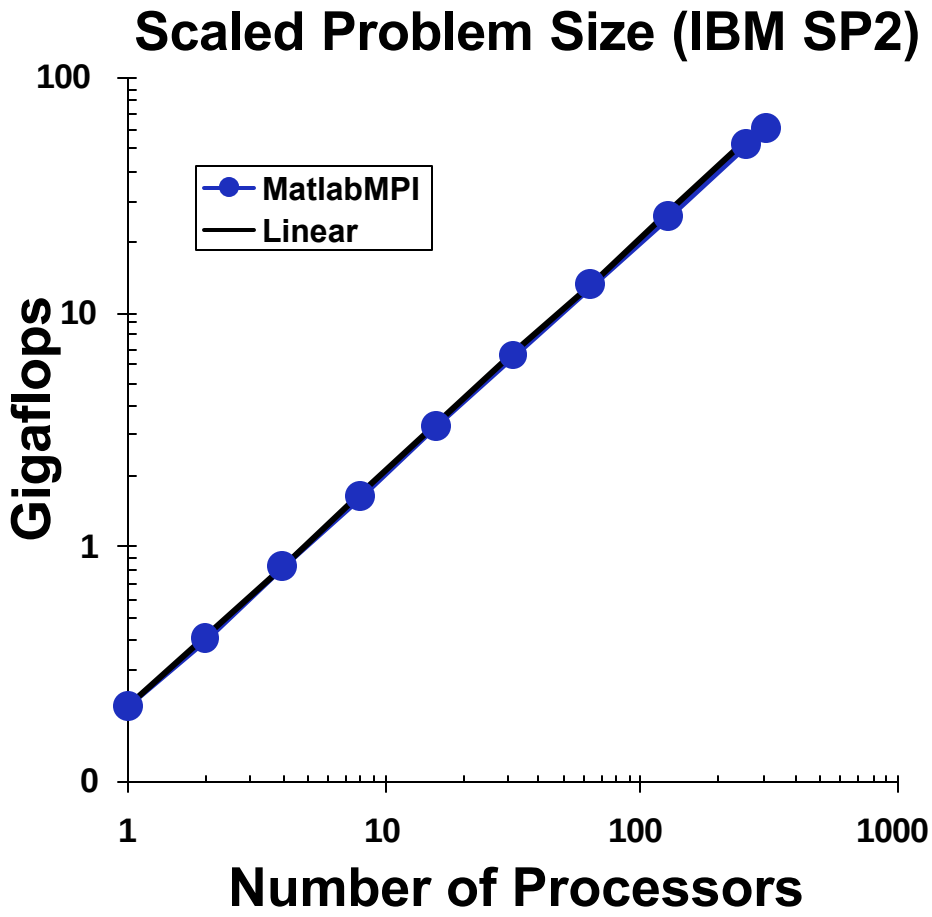
Linux w/Gigabit Ethernet



- Bandwidth scales to multiple processors
- Cross mounting eliminates bottlenecks

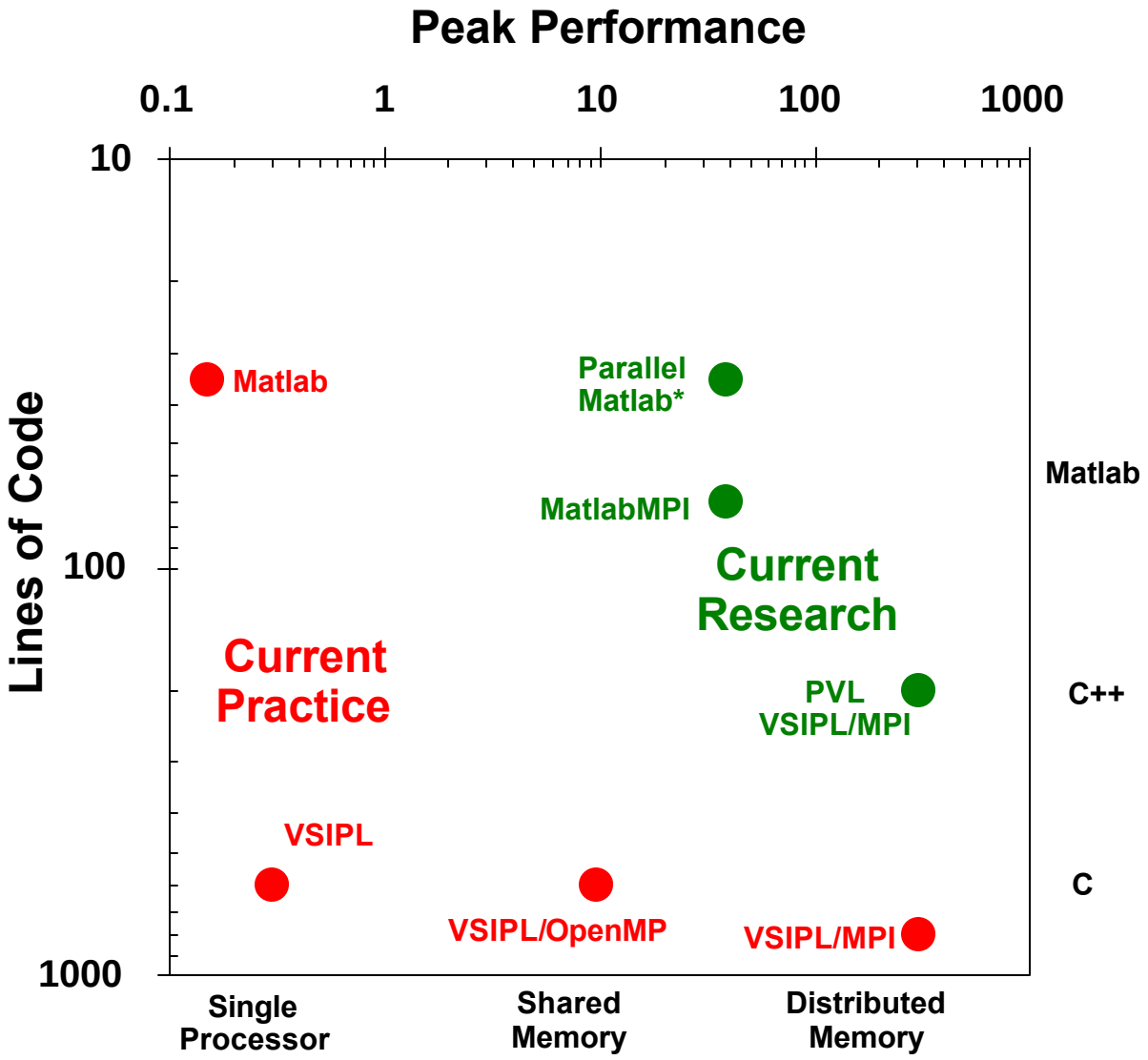
Image Filtering Parallel Performance



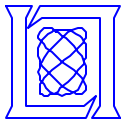


- Achieved "classic" super-linear speedup on fixed problem
- Achieved speedup of ~300 on 304 processors on scaled problem

Productivity vs. Performance



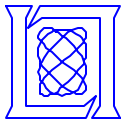
- Programmed image filtering several ways
  - Matlab
  - VSIPL
  - VSIPL/OpenMPI
  - VSIPL/MPI
  - PVL
  - MatlabMPI
- MatlabMPI provides
  - high productivity
  - high performance



Current MatlabMPI deployment

- Lincoln Signal processing (7.8 on 8 cpus, 9.4 on 8 duals)
- Lincoln Radar simulation (7.5 on 8 cpus, 11.5 on 8 duals)
- Lincoln Hyperspectral Imaging (~3 on 3 cpus)
- MIT LCS Beowulf (11 Gflops on 9 duals)
- MIT AI Lab Machine Vision
- OSU EM Simulations
- ARL SAR Image Enhancement
- Wash U Hearing Aid Simulations
- So. Ill. Benchmarking
- JHU Digital Beamforming
- ISL Radar simulation
- URI Heart modelling

- Rapidly growing MatlabMPI user base
- Web release may create hundreds of users

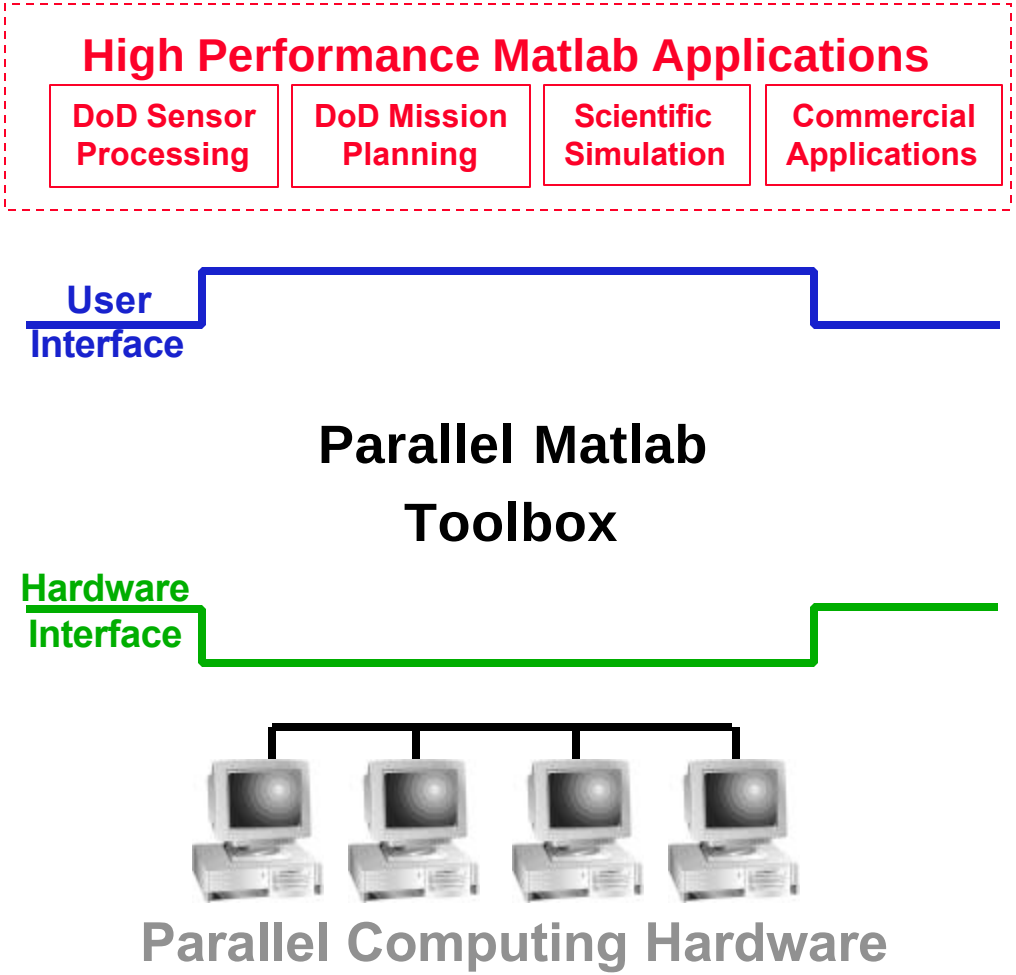


Outline

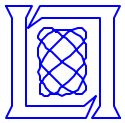
- Introduction
- Approach
- Performance Results
- **Future Work and Summary**

Future Work: Parallel Matlab Toolbox

- **Parallel Matlab need has been identified**
  - HPCMO (OSU)
- **Required user interface has been demonstrated**
  - Matlab*P (MIT/LCS)
  - PVL (MIT/LL)
- **Required hardware interface has been demonstrated**
  - MatlabMPI (MIT/LL)

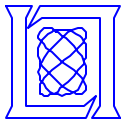


- **Allows parallel programs with no additional lines of code**



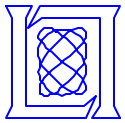
Future Work: Scalable Perception

- **Data explosion**
  - Advanced perception techniques must process vast (and rapidly growing) amounts of sensor data
- **Component scaling**
  - Research has given us high-performance algorithms and architectures for sensor data processing,
  - But these systems are not modular, reflective, or radically reconfigurable to meet new goals in real time
- **Scalable perception**
  - will require a framework that couples the computationally daunting problems of real-time multimodal perception to the infrastructure of modern high-performance computing, algorithms, and systems.
- **Such a framework must exploit:**
  - High-level languages
  - Graphical / linear algebra duality
  - Scalable architectures and networks



Summary

- MatlabMPI has the basic functions necessary for parallel programming
  - Size, rank, send, receive, launch
  - Enables complex applications or libraries
- Performance can match native MPI at large message sizes
- Demonstrated scaling into hundreds of processors
- Demonstrated productivity
- Available on HPCMO systems
- Available on Web



Acknowledgements

- **Support**
  - Charlie Holland DUSD(S&T) and John Grosh OSD
  - Bob Bond and Ken Senne (Lincoln)

- **Collaborators**
  - Stan Ahalt and John Nehrbass (Ohio St.)
  - Alan Edelman, John Gilbert and Ron Choy (MIT LCS)

- **Lincoln Applications**
  - Gil Raz, Ryan Haney and Dan Drake
  - Nick Pulsone and Andy Heckerling
  - David Stein

- **Centers**
  - Maui High Performance Computing Center
  - Boston University

http://www.ll.mit.edu/MatlabMPI