

# LLgrid: Enabling On-Demand Grid Computing With gridMatlab and pMatlab

*Albert Reuther, Tim Currie, Jeremy Kepner, Hahn G. Kim,  
Andrew McCabe, Michael P. Moore, and Nadya Travinin*

MIT Lincoln Laboratory, Lexington, MA 02420

Phone: 781-981-5699

Email Addresses: {reuther, currie, kepner, hgk, amccabe, moore, [ntj@ll.mit.edu](mailto:ntj@ll.mit.edu)}

May 28, 2004

## Introduction

The concept of grid computing – back-room computational resources that are as accessible and available as the electric grid – has gained momentum [1]. Numerous Grid computing projects such as NetSolve [2] and Legion [3] have provided infrastructure to enable the launching and monitoring of mostly parameter sweep applications. These Grid computing projects demand that users endure a steep learning curve to program and use the system. Also, these systems draw a strong distinction between the users' computers and the grid computing resources: users' work is done on their computers, while grid jobs are executed on grid resources. A computational power grid should have characteristics similar to the electric power grids: always available, ubiquitous throughout the organization, and easy to use.

The goal of the MIT Lincoln Laboratory Grid (LLgrid) project is to develop a On-Demand Grid Computing capability to address these characteristics and use MATLAB® – the dominant programming language for implementing numerical computations, widely used for algorithm development, simulation, data reduction, testing, and system evaluation – as its initial target application [4]. MIT Lincoln Laboratory has over one thousand MATLAB users; nearly two hundred users run very long jobs that could benefit from parallel processing. The LLgrid project has developed three technologies that allow these users to run parallel MATLAB jobs transparently on the LLgrid computational resources:

- *MatlabMPI* for point-to-point messaging;
- *pMatlab* for global array semantics (similar to High Performance Fortran); and
- *gridMatlab* for integrating user's computers into the LLgrid and automatically allocating grid computing resources.

These technologies have combined to create a unique on-demand, interactive Grid Computing experience, whereby running a parallel MATLAB job on LLgrid is identical to running MATLAB on the desktop. Users can use LLgrid from Windows, Linux, Solaris, and Mac OS X computers

with their desktop computer becoming a personal node in the LLgrid thereby establishing a transparent interface between the user's computer and the grid resources. LLgrid is enabling faster algorithm development, prototyping, and validation cycles for Lincoln staff. In addition, the initial creation and setup of user accounts is entirely automated, which minimizes system administration.

## The LLgrid System

A number of components comprise the LLgrid On-Demand Grid Computing system, from the underlying hardware and network to MATLAB and the three Lincoln-developed MATLAB toolboxes: pMatlab, MatlabMPI, and gridMatlab. Figure 1 shows a conceptual depiction of the initial operational system including the LLgrid Alpha Cluster of Red Hat Linux nodes, the network configuration, the users' computers on the central Lincoln local area network (LLAN), and the gridsan network storage, which delivers a common file system to the LLgrid cluster and all of the users.

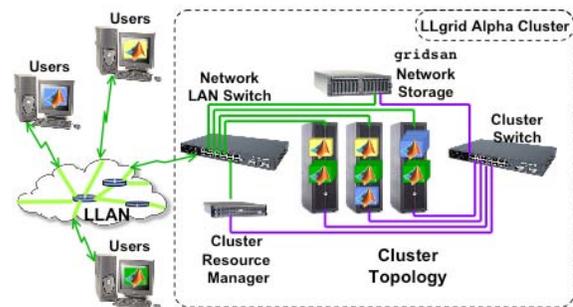


Figure 1: Conceptual diagram of LLgrid System.

The core of the users' experience with the LLgrid system occurs with MATLAB and the pMatlab toolbox, while the MatlabMPI toolbox provides the means for parallel MATLAB processes to communicate with each other and the gridMatlab toolbox provides the interface between MatlabMPI and the LLgrid resources for managing jobs. MatlabMPI [5] consists of a set of MATLAB scripts that implement a subset of MPI, allowing any MATLAB program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely used MPI "look and feel" on top of standard MATLAB file I/O, resulting in a "pure" MATLAB implementation that is exceedingly small (~300 lines of code). Thus, MatlabMPI will run on any combination of computers that MATLAB

# Report Documentation Page

*Form Approved*  
*OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE <b>01 FEB 2005</b>	2. REPORT TYPE <b>N/A</b>	3. DATES COVERED <b>-</b>	
4. TITLE AND SUBTITLE <b>LLgrid: Enabling On-Demand Grid Computing With gridMatlab and pMatlab</b>		5a. CONTRACT NUMBER	
		5b. GRANT NUMBER	
		5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)		5d. PROJECT NUMBER	
		5e. TASK NUMBER	
		5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>MIT Lincoln Laboratory, Lexington, MA 02420</b>		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)	
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release, distribution unlimited</b>			
13. SUPPLEMENTARY NOTES <b>See also ADM001742, HPEC-7 Volume 1, Proceedings of the Eighth Annual High Performance Embedded Computing (HPEC) Workshops, 28-30 September 2004. , The original document contains color images.</b>			
14. ABSTRACT			
15. SUBJECT TERMS			
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>	<b>UU</b>
			18. NUMBER OF PAGES <b>29</b>
			19a. NAME OF RESPONSIBLE PERSON

supports. Overlaid on MatlabMPI, pMatlab combines operator overloading with parallel maps to provide implicit data parallelism and task parallelism [6]. pMatlab allows a MATLAB user to parallelize their program by changing a few lines.

The gridMatlab toolbox transparently integrates the MATLAB on each user’s desktop with shared grid clusters through a cluster resource manager; when a MatlabMPI or pMatlab job is run by the user in his or her MATLAB session, gridMatlab automatically amasses the requested LLgrid computational resources from the shared grid resources to process in parallel with the user’s MATLAB session. (In traditional grid computing the users must submit their jobs to batch job queues where the jobs execution must wait to be executed.) Using MatlabMPI, the underlying common file system (hosted on gridsan) becomes the communication fabric through which each of the parallel MATLAB processes communicate, including the MATLAB on each user’s desk. By integrating the user’s MATLAB session into the set of grid cluster MATLAB sessions working on his or her code, the user receives immediate feedback on the status of his or her job, thereby making parallel MATLAB execution virtually identical to running MATLAB on a single computer.

## Results

As with any distributed, parallel computing system, it is expected to deliver high performance; the LLgrid system meets those expectations. For example, for a hyper-spectral imaging (HSI) analysis application [7], LLgrid is used to classify various characteristics in HSI signal returns using a normal compositional model (NCM). The application is composed of three components: abundance estimates of class at each pixel (up\_abund), NCM class parameter updates (up\_NCM), and NCM log-likelihood computations for given class abundance values (up\_mll). Figure 2 shows the speedup of these three application components, each realizing near linear speedup.

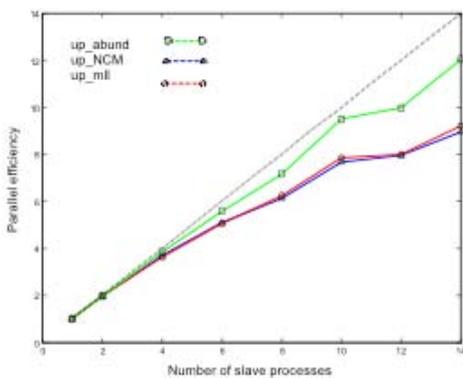


Figure 2: Speedup realized by Hyperspectral Analysis application

When it comes to modeling productivity, we turned to the Darpa IPTO High Productivity Computing Systems (HPCS) program. One of the challenges in the HPCS mission is defining, modeling, and measuring the productivity that a computer system delivers to its users and consequently to the users’ organization. To answer this challenge, the HPCS

Productivity Team has developed a high performance productivity framework and evaluation model [8]. Their model implies a profoundly different way of viewing HPC systems by including the user-associated costs in the model and by viewing innovative hardware as a key aspect to lowering the very high cost of high performance computing software.

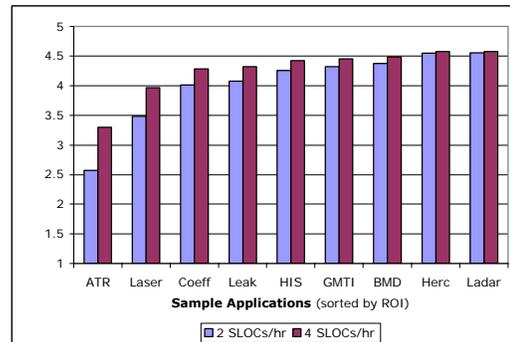


Figure 3: Range of Return of Investment for sample LLgrid applications.

Using the HPCS productivity model, Figure 3 plots two return on investment (ROI) values for the LLgrid system and nine sample applications that are currently being run on the LLgrid system, it shows a potential upper and lower bounds by calculating the ROI for average programming rates of 2 and 4 SLOCs per hour. The graph shows that the ROI range is between 2.6 and 4.6 across the nine sample applications and average programming rates, which means that for every \$1.00 spent on the LLgrid, between \$2.60 and \$4.60 is returned in user time saved.

## References

- [1] I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Second Edition, Morgan-Kaufman, 2004.
- [2] S. Agrawal, J. Dongarra, K. Seymour, and S. Vadhiyar, *Grid Computing – Making the Global Infrastructure a Reality*, F. Berman, G. Fox, and T. Hey (eds.), John Wiley & Sons, Ltd., 2002.
- [3] A. S. Grimshaw and W. A. Wulf, “The Legion vision of a worldwide virtual computer,” *Communications of the ACM*, Vol. 40, No. 1, Jan. 1997.
- [4] MATLAB, The MathWorks, Inc., <http://www.MathWorks.com/products/matlab/>.
- [5] J. Kepner, “Parallel programming with MatlabMPI,” *High Performance Embedded Computing (HPEC) Workshop 2001*, <http://www.ll.mit.edu/hpec/>, 2001.
- [6] J. Kepner and N. Travinin, “Parallel MATLAB: The next generation,” *High Performance Embedded Computing (HPEC) Workshop 2003*, <http://www.ll.mit.edu/hpec/>, 2003.
- [7] D. Stein, “A parallel implementation of the normal compositional model for hyperspectral analysis based on MatlabMPI,” *High Performance Embedded Computing (HPEC) Workshop 2001*, <http://www.ll.mit.edu/hpec/>, 2001.
- [8] J. Kepner, “HPC productivity model synthesis,” accepted for publication in the *International Journal of High Performance Computing Applications*, Vol. 18, No. 4, November 2004.



# **LLGrid: On-Demand Grid Computing with gridMatlab and pMatlab**

**Albert Reuther**

**MIT Lincoln Laboratory**

**29 September 2004**

**This work is sponsored by the Department of the Air Force 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 United States Government.**

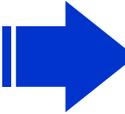
**MIT Lincoln Laboratory**



# LLGrid On-Demand Grid Computing System Agenda

- **Introduction**

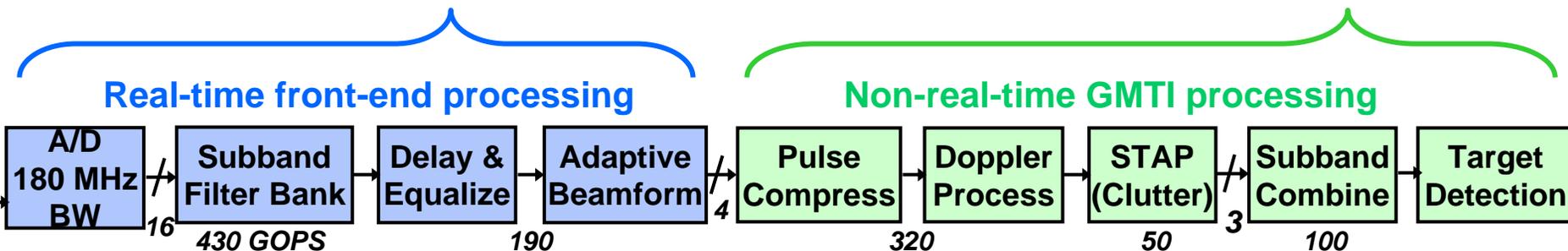
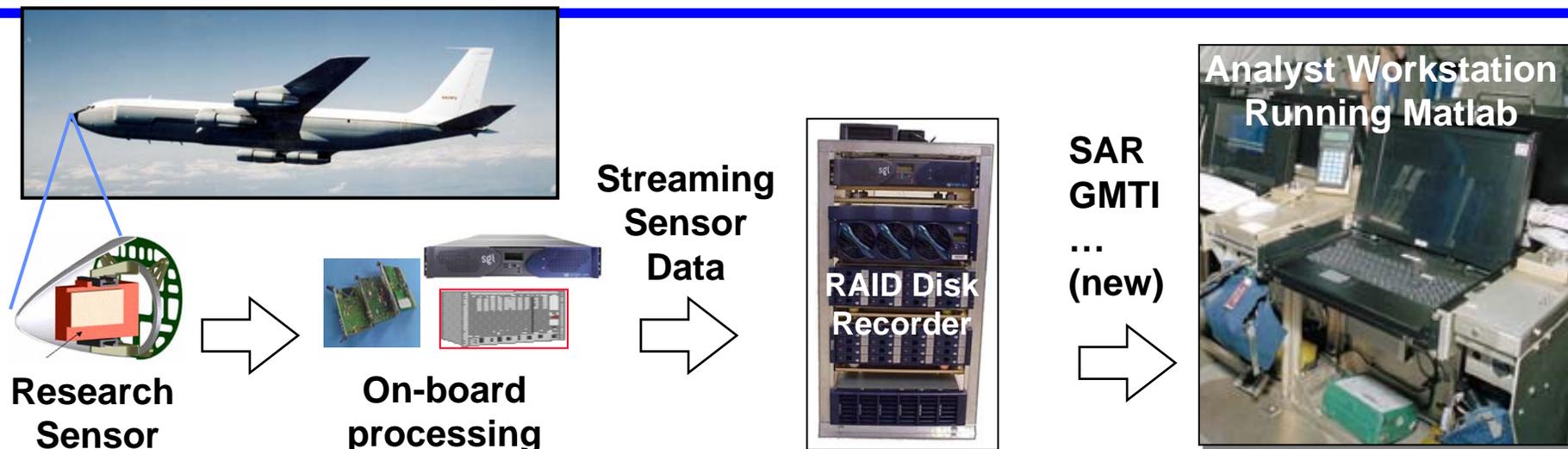
- **LLGrid System**
- **Performance Results**
- **LLGrid Productivity Analysis**
- **Summary**



- *Example Application*
- *LLGrid Vision*
- *User Survey*
- *System Requirements*



# Example App: Prototype GMTI & SAR Signal Processing

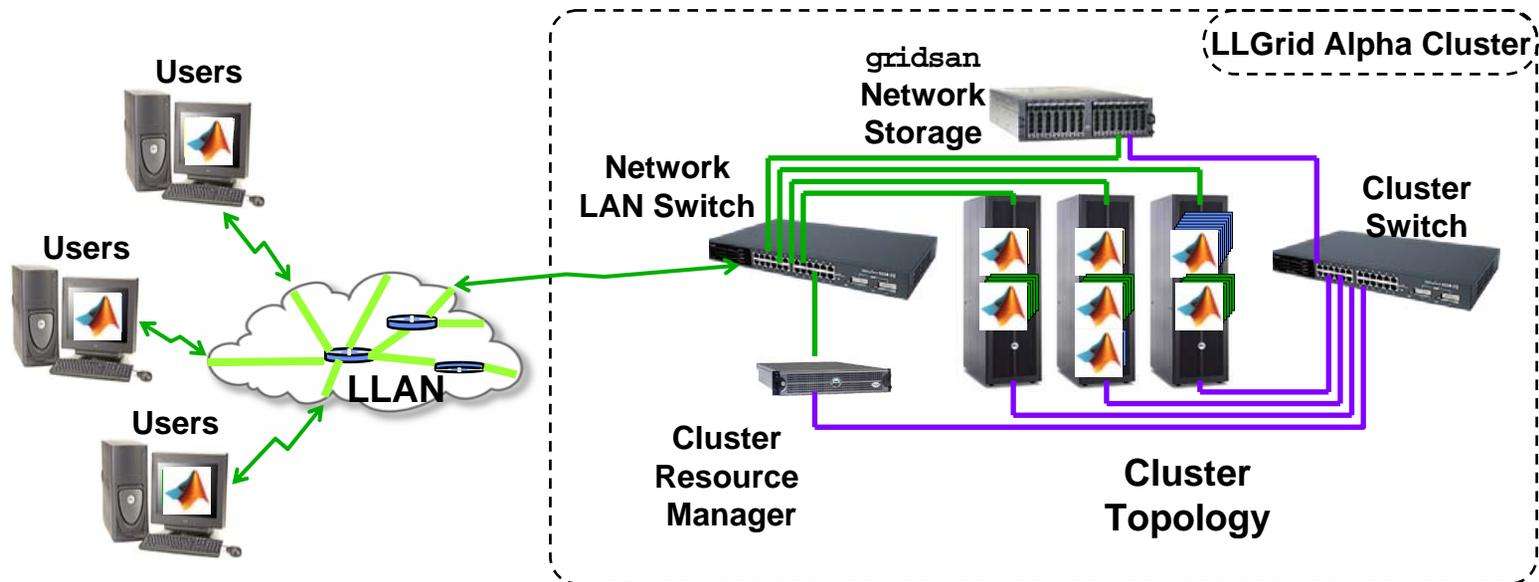


- Airborne research sensor data collected
- Research analysts develop signal processing algorithms in MATLAB® using collected sensor data
- Individual runs can last hours or days on single workstation



# LLGrid

**Goal: 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.**

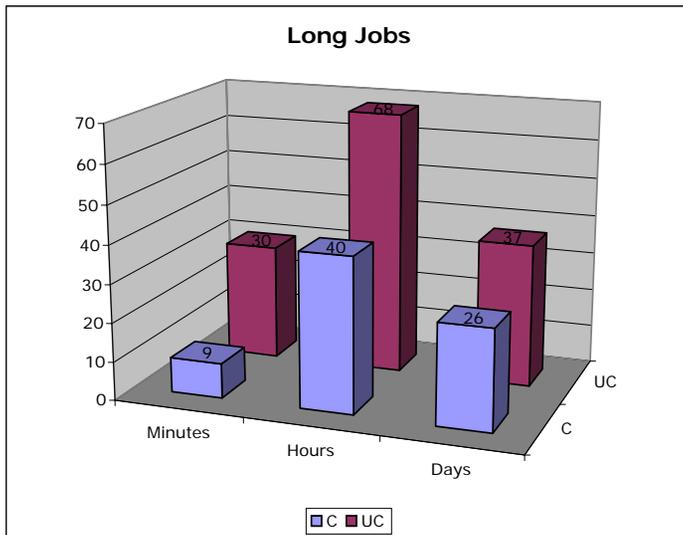
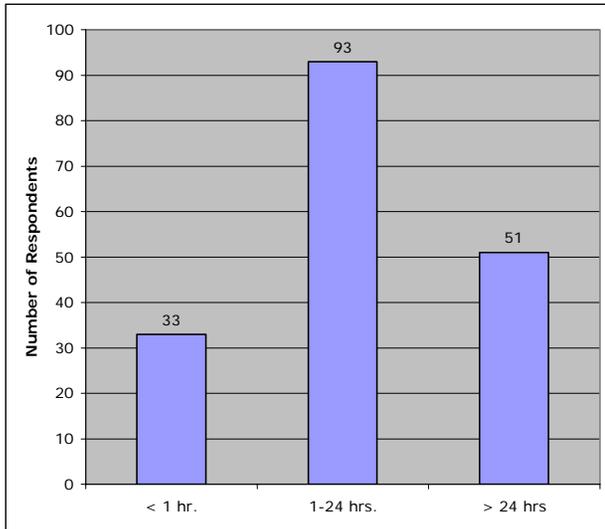


## Lab Grid Computing Components

- Enterprise access to high throughput Grid computing
- Enterprise distributed storage



# MATLAB<sup>®</sup> Users Survey

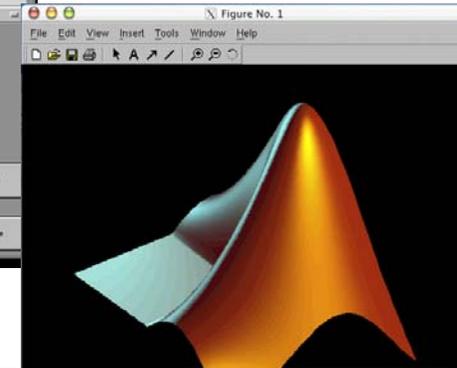
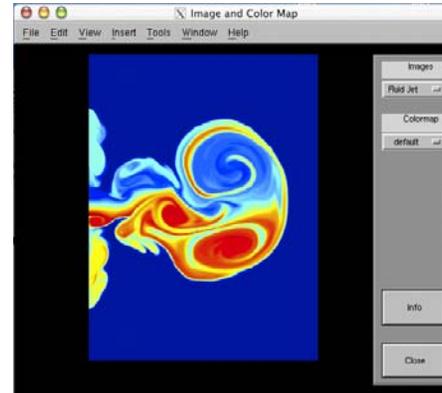


- **Conducted survey of Lab staff**
  - Do you run long MATLAB jobs?
  - How long do those jobs run (minutes, hours, or days)?
  - Are these jobs unclassified, classified, or both?
- **Survey results:**
  - 464 respondents
  - 177 answered “Yes” to question on whether they run long jobs
- **Lincoln MATLAB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



# LLGrid User Requirements

- **Easy to set up**
  - First time user setup should be automated and take less than 10 minutes
- **Easy to use**
  - Using LLGrid should be the same as running a MATLAB job on user's computer
- **Compatible**
  - Windows, Linux, Solaris, and MacOS X
- **High Availability**
- **High Throughput for Medium and Large Jobs**



```
< M A T L A B >
Copyright 1984-2002 The MathWorks, Inc.
Version 6.5.0.185000 release 13
Jun 27 2002

Using Toolbox Path Cache. Type "help toolbox_path_cache" for more info.

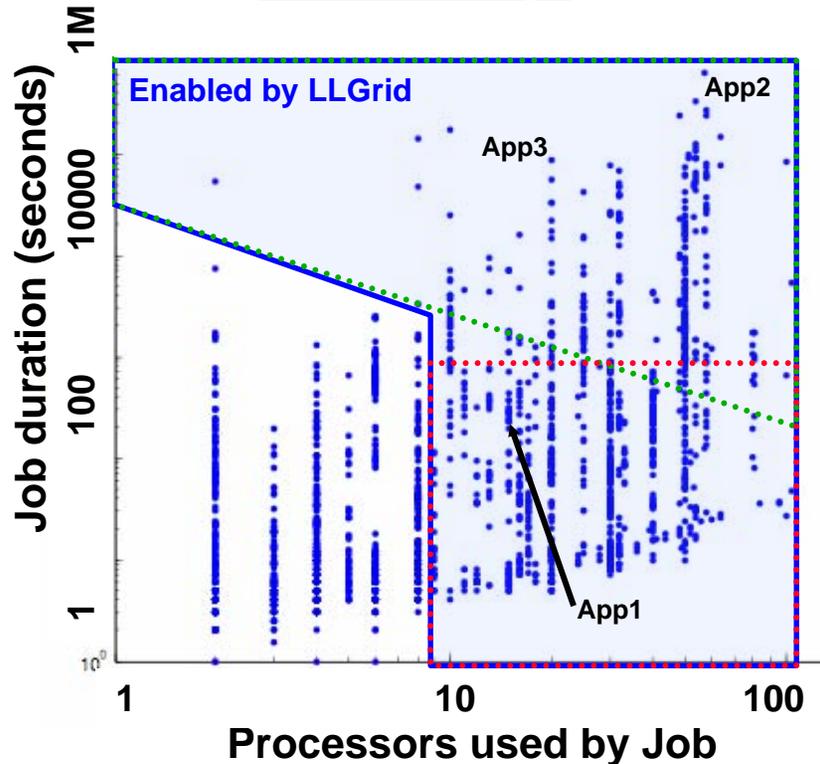
To get started, select "MATLAB Help" from the Help menu.

>> ver
-----
MATLAB Version 6.5.0.185000 (R13)
MATLAB License Number: 241630
Operating System: Darwin 6.8 Darwin Kernel Version 6.8: Wed Sep 10 15:20:55 PDT 2003
Java VM Version: Java 1.3.1 with "Apple Computer"
-----
MATLAB Version 6.5 (R13)
Simulink Version 5.0 (R13)
Communications Blockset Version 2.5 (R13)
Communications Toolbox Version 2.1 (R13)
Control System Toolbox Version 5.2 (R13)
DSP Blockset Version 5.0 (R13)
Image Processing Toolbox Version 3.2 (R13)
MATLAB Compiler Version 3.0 (R13)
Mapping Toolbox Version 1.3 (R13)
Neural Network Toolbox Version 4.0.2 (R13)
Optimization Toolbox Version 2.2 (R13)
SRSZL (Converts SystemBuild to Simulink) Version 2.5 (R13)
Signal Processing Toolbox Version 6.0 (R13)
Statistics Toolbox Version 4.0 (R13)
Wavelet Toolbox Version 2.2 (R13)
>>
```



# LLgrid Usage

## LLGrid Usage



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

3500 jobs, 3600 CPU Days  
December 03 – June 04

- Allowing Lincoln staff to effectively use parallel computing daily from their desktop
  - Interactive parallel computing
  - 160 CPUs, 25 Users, 11 Groups
- Extending the current space of data analysis and simulations that Lincoln staff can perform
  - **Jobs requiring rapid turnaround**  
App1: Weather Radar Signal Processing Algorithms
  - **Jobs requiring many CPU hours**  
App2: Hyperspectral Image Analysis  
App3: Laser Propagation Simulation

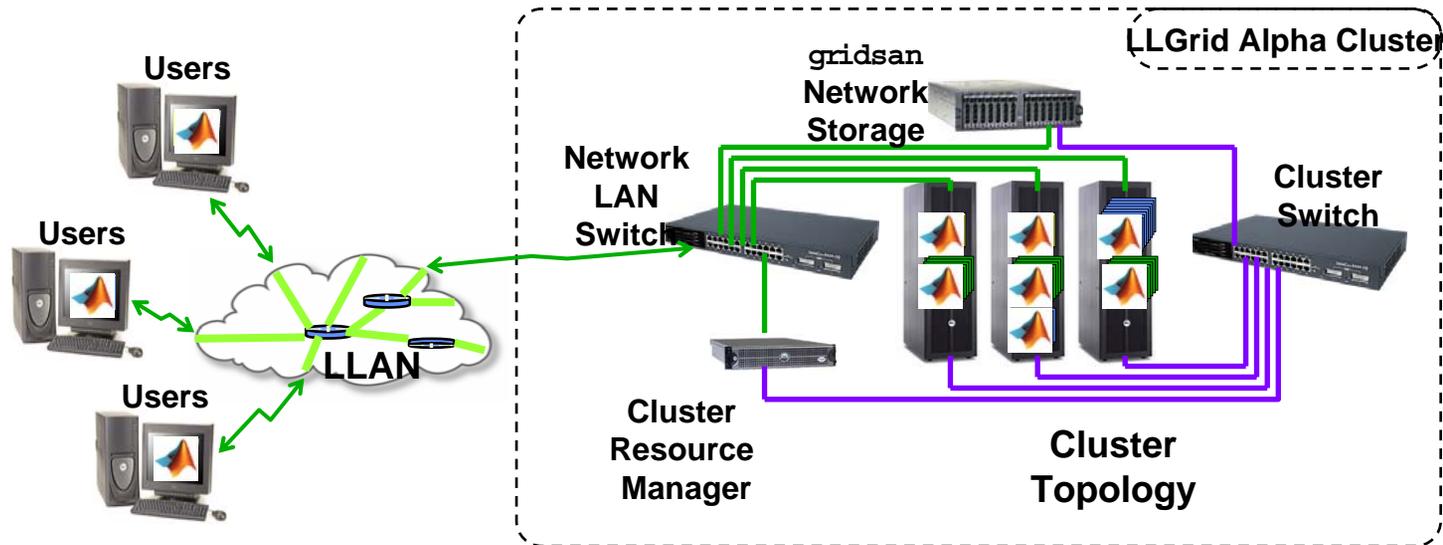


# LLGrid On-Demand Grid Computing System Agenda

- Introduction
  - **LLGrid System**
  - Performance Results
  - LLGrid Productivity Analysis
  - Summary
- 
- *Overview*
  - *Hardware*
  - *Management Scripts*
  - *MatlabMPI*
  - *pMatlab*
  - *gridMatlab*



# LLGrid Alpha Cluster



## Key Innovations:

**pMatlab - Global array semantics for parallel MATLAB**

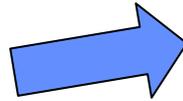
**gridMatlab - User's computer is transparently included into LLGrid**

**- User never logs into LLGrid (only mounts file system)**



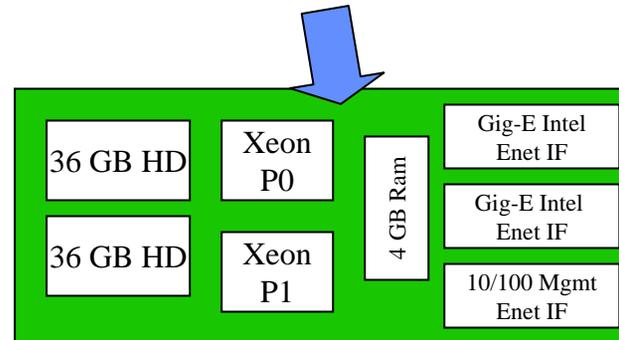
# Alpha Grid Hardware

80 Nodes + Head Node - 160+2 Processors, 320 GB RAM



Nodes:

**DELL** 2650 &  
**DELL** 1750



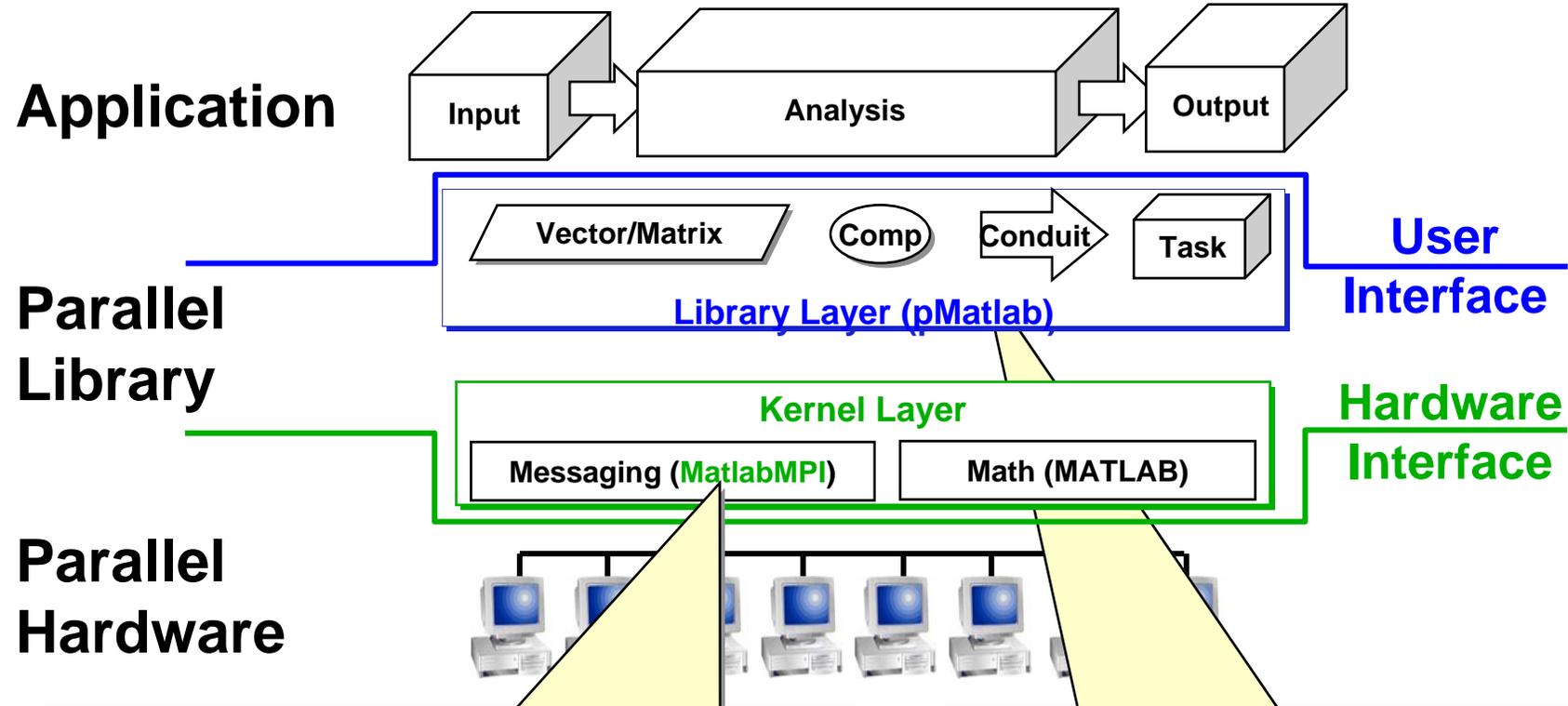
- Dual 2.8 & 3.06 GHz Xeon (P4)
- 400 & 533 MHz front-side bus
- 4 GB RAM memory
- Two 36 GB SCSI hard drives
- 10/100 Mgmt Ethernet interface
- Two Gig-E Intel interfaces
- Running Red Hat Linux

- Commodity Hardware
- Commodity OS
- High Availability





# 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 an application with a few parallel structures and functions
- **pMatlab** provides Global Array Semantic via parallel arrays and functions

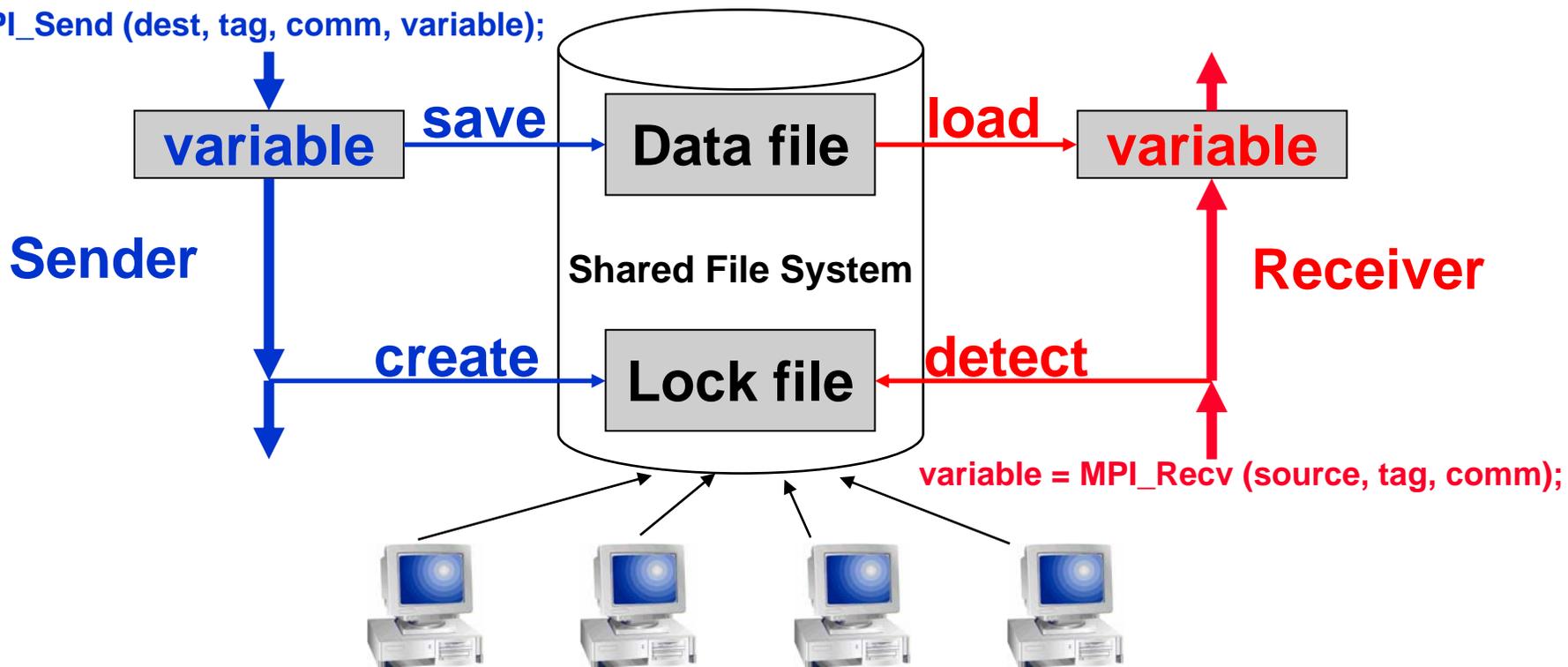
```
X = ones(n, mapX);  
Y = zeros(n, mapY);  
Y(:, :) = fft(X);
```



# 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**

`MPI_Send (dest, tag, comm, variable);`



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





# LLGrid Account Creation

LLgrid Account Creation

Request an Account

Badge #:

Operating System:  Windows  Unix  Mac OS

Contact: [webmaster@ll.mit.edu](mailto:webmaster@ll.mit.edu)  
Last modified: October 16, 2003  
For Laboratory Use Only

## 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



# Account Setup Steps

## Typical Supercomputing Site Setup

- Account application/renewal [months]
- Resource discovery [hours]
- Resource allocation application/renewal [months]
- Explicit file upload/download (usually ftp) [minutes]
- Batch queue configuration [hours]
- Batch queue scripting [hours]
- Differences between control vs. compute nodes [hours]
- Secondary storage configuration [minutes]
- Secondary storage scripting [minutes]
- Interactive requesting mechanism [days]
- Debugging of example programs [days]
- Documentation system [hours]
- Machine node names [hours]
- GUI launch mechanism [minutes]
- Avoiding user contention [years]

## LLGrid Account Setup [minutes]

- 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



# MathWorks Distributed MATLAB (DML)

**“Dear MATLAB user,**

**“This is an invitation to participate in an upcoming Beta Test for the Distributed MATLAB product. This will be available on the following platforms, Win 2000, Win NT, WIN XP, and Linux.**

**“The goal of this first release of Distributed MATLAB is to address the requirements of coarse-grain applications, in which the same MATLAB algorithm is executed in remote MATLAB sessions on different data sets without communication or data exchange between sessions.”**

**– From DML beta email**

**Lincoln has installed DML and is testing it to determine how it integrates with LLGrid technologies.**



# LLGrid On-Demand Grid Computing System Agenda

---

- Introduction
- LLGrid System
- Performance Results
- LLGrid Productivity Analysis
- Summary



# Performance: Time to Parallelize

## Important Considerations

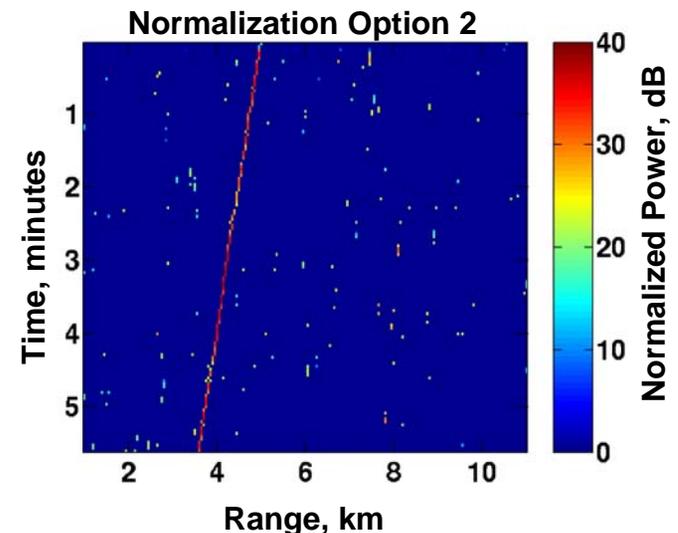
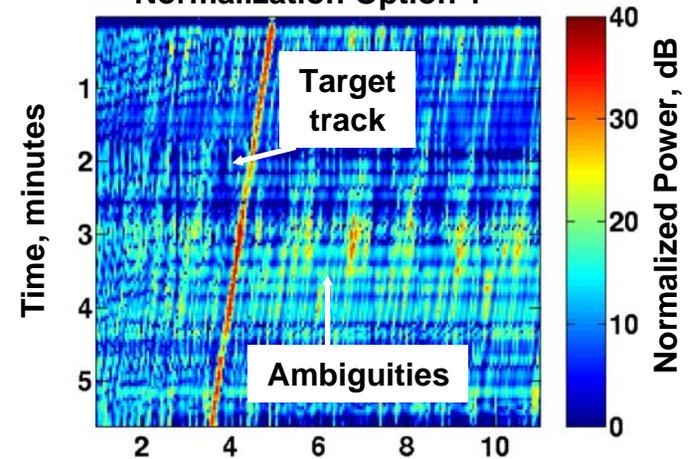
Description	Serial Code Time	Time to Parallelize	Applications that Parallelization Enables
Missile & Sensor BMD Sim. (BMD)	2000 hours	8 hours	Discrimination simulations Higher fidelity radar simulations
First-principles LADAR Sim. (Ladar)	1300 hours	1 hour	Speckle image simulations Aimpoint and discrimination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



# pMatlab Application to 3D Spatial Normalization

- A Lincoln group is developing normalization algorithms for 3D matched-field (MFP) beamformers
- Sponsored by DARPA-ATO under Robust Passive Sonar program
- Large search space (  $O(1e7)$  cells ) makes normalizer evaluation on processed data difficult
- pMatlab code enabled rapid algorithm development and parameter selection
  - **> 20x** speedup by exploiting parallelism across frequency on nodes of Linux cluster
  - Development time was ~1 day

Simulated data:  
Matched field output at target depth, bearing  
Normalization Option 1





# LLGrid On-Demand Grid Computing System Agenda

---

- Introduction
- LLGrid System
- Performance Results
- **LLGrid Productivity Analysis**
- Summary



# LLGrid Productivity Analysis for ROI\*

$$\text{productivity (ROI)} = \frac{\text{Utility}}{\left[ \text{Software Cost} \right] + \left[ \text{Maintenance Cost} \right] + \left[ \text{System Cost} \right]}$$



# LLGrid Productivity Analysis for ROI\*

$$\text{productivity (ROI)} = \frac{\left( \text{time saved by users on system} \right)}{\left( \text{time to parallelize} \right) + \left( \text{time to train} \right) + \left( \text{time to launch} \right) + \left( \text{time to admin.} \right) + \left( \text{system cost} \right)}$$



# LLGrid Productivity Analysis for ROI

$$\text{productivity (ROI)} = \frac{\text{time saved by users on system}}{\text{time to parallelize} + \text{time to train} + \text{time to launch} + \text{time to admin.} + \text{system cost}}$$

## Production LLGrid model assumptions

- 200 users Lab-wide
- 20 simultaneous jobs
- Average 10 CPUs per job
- 2 SLOCs per hour
- 1000 SLOCs per simulation \* Lab-wide users
- 1.0% time-to-parallelize overhead
- Training time - 4 hours \* Lab-wide users
- 10,000 parallel job launches
- 10 seconds to launch
- One sys-admin ≈ 2000 hours
- 200 CPUs @ \$5k per node ≈ 5000 hours

$$\left( \begin{array}{l} \text{time saved} \\ \text{by users on} \\ \text{system} \end{array} \right) = \left( \begin{array}{l} \text{User} \\ \text{salary} \end{array} \right) * \left( \begin{array}{l} \text{Total time} \\ \text{system is} \\ \text{in use} \end{array} \right) * \left( \begin{array}{l} \text{Average} \\ \text{number of} \\ \text{users} \end{array} \right) * \left( 1 - \frac{1}{\left( \begin{array}{l} \text{Average \# of} \\ \text{CPUs per job} \end{array} \right)} \right)$$

$$\left( \begin{array}{l} \text{time to} \\ \text{parallelize} \end{array} \right) = \left( \begin{array}{l} \text{User} \\ \text{salary} \end{array} \right) * \left( \begin{array}{l} \text{Total} \\ \text{\# of} \\ \text{users} \end{array} \right) * \left( \begin{array}{l} \text{Prog.} \\ \text{rate} \end{array} \right) * \left( \begin{array}{l} \text{Average} \\ \text{lines of} \\ \text{code} \end{array} \right) * \left( \frac{1}{\left( \begin{array}{l} \text{Cost for} \\ \text{parallel} \end{array} \right)} - 1 \right)$$

$$\left( \begin{array}{l} \text{time to} \\ \text{train} \end{array} \right) = \left( \begin{array}{l} \text{User} \\ \text{salary} \end{array} \right) * \left( \begin{array}{l} \text{Total} \\ \text{\# of} \\ \text{users} \end{array} \right) * \left( \begin{array}{l} \text{Time to} \\ \text{train a} \\ \text{user} \end{array} \right)$$

$$\left( \begin{array}{l} \text{time to} \\ \text{launch} \end{array} \right) = \left( \begin{array}{l} \text{User} \\ \text{salary} \end{array} \right) * \left( \begin{array}{l} \text{Number of} \\ \text{launches} \end{array} \right) * \left( \begin{array}{l} \text{Time to} \\ \text{launch} \end{array} \right)$$

$$\left( \begin{array}{l} \text{time to} \\ \text{admin.} \end{array} \right) = \left( \begin{array}{l} \text{Admin.} \\ \text{salary} \end{array} \right) * \left( \begin{array}{l} \text{Number of} \\ \text{admins} \end{array} \right) * \left( \begin{array}{l} \text{Admin} \\ \text{time} \end{array} \right)$$

$$\left( \begin{array}{l} \text{system} \\ \text{cost} \end{array} \right) = \left( \begin{array}{l} \text{User} \\ \text{salary} \end{array} \right) * \left( \begin{array}{l} \text{Time-value} \\ \text{of system} \end{array} \right)$$

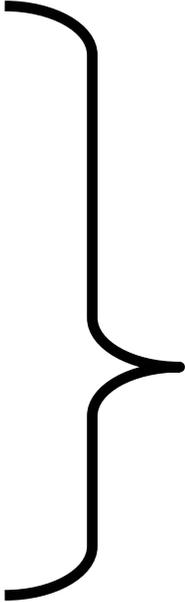


# LLGrid Productivity Analysis for ROI

$$\text{productivity (ROI)} = \frac{\text{time saved by users on system}}{\text{time to parallelize} + \text{time to train} + \text{time to launch} + \text{time to admin.} + \text{system cost}}$$

## Production LLGrid model assumptions

- 200 users Lab-wide
- 20 simultaneous jobs
- Average 10 CPUs per job
- 2 SLOCs per hour
- 1000 SLOCs per simulation \* Lab-wide users
- 1.0% time-to-parallelize overhead
- Training time - 4 hours \* Lab-wide users
- 10,000 parallel job launches
- 10 seconds to launch
- One sys-admin ≈ 2000 hours
- 200 CPUs @ \$5k per node ≈ 5000 hours



$$\text{ROI}_{\text{expected}} = \frac{36,000}{1000+27.8+2000+5000}$$

$$\text{ROI}_{\text{expected}}^* \approx 4.5$$

Steady state with full LLGrid

\* Mileage may vary



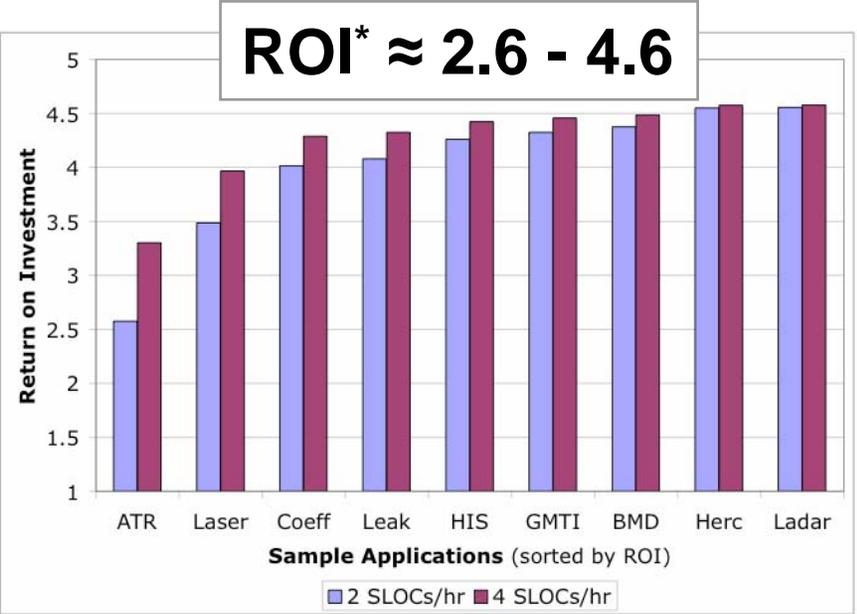
# LLGrid Productivity Analysis for ROI

time saved by users on system

$$\text{productivity (ROI)} = \frac{\text{time saved by users on system}}{\text{time to parallelize} + \text{time to train} + \text{time to launch} + \text{time to admin.} + \text{system cost}}$$

## Production LLGrid model assumptions

- 200 users Lab-wide
- 20 simultaneous jobs
- Average 10 CPUs per job
- **2-4 SLOCs per hour**
- 1000 SLOCs per simulation \* Lab-wide users
- **Measured time-to-parallelize overhead**
- Training time - 4 hours \* Lab-wide users
- 10,000 parallel job launches
- 10 seconds to launch
- One sys-admin ≈ 2000 hours
- 200 CPUs @ \$5k per node ≈ 5000 hours



\* Varying Mileage



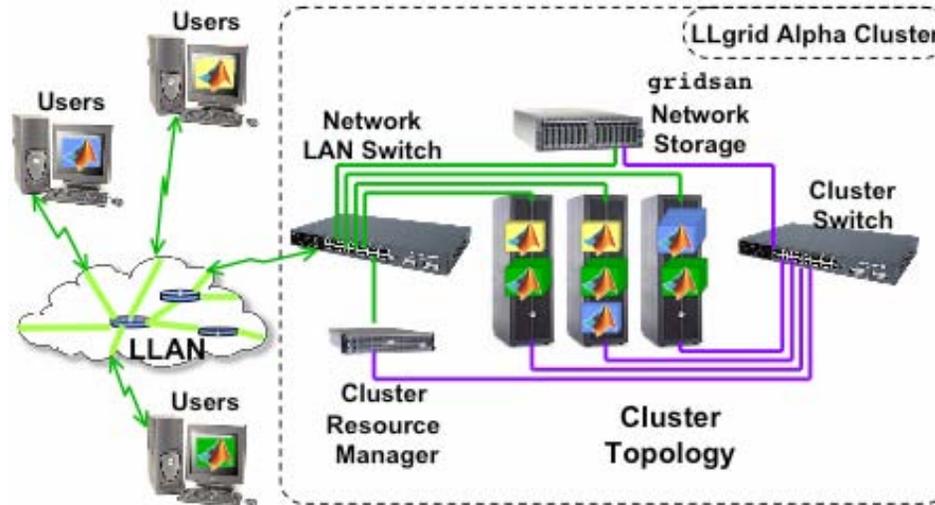
# LLGrid On-Demand Grid Computing System Agenda

---

- Introduction
- LLGrid System
- Performance Results
- LLGrid Productivity Analysis
- Summary



# Summary



- Easy to set up
- Easy to use
- User's computer transparently becomes part of LLGrid
- High throughput computation system
- 25 alpha users, expecting 200 users Lab-wide
- Computing jobs they could not do before
- 3600 CPU days of computer time in 8 months
- LLGrid Productivity Analysis - ROI  $\approx$  4.5