Quantitative Productivity Measurements in an HPC Environment

May 26, 2007

Jeremy Kepner, Bob Bond, Andy Funk,
Andy McCabe, Julie Mullen, Albert Reuther

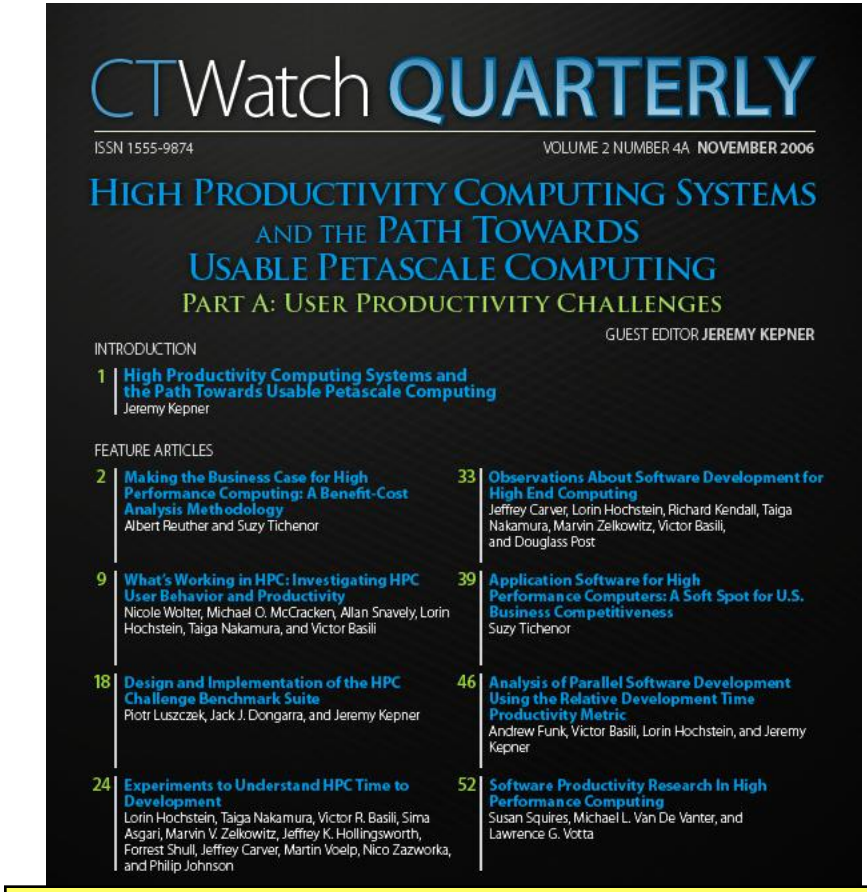
This work is sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Outline

- **Introduction**
- **LLGrid Environment**
- **Results**
- **Summary**

- ***Background***
- ***Notional Productivity***
- ***Formal Productivity***

Background



CTWatch QUARTERLY

ISSN 1555-9874 VOLUME 2 NUMBER 4A **NOVEMBER 2006**

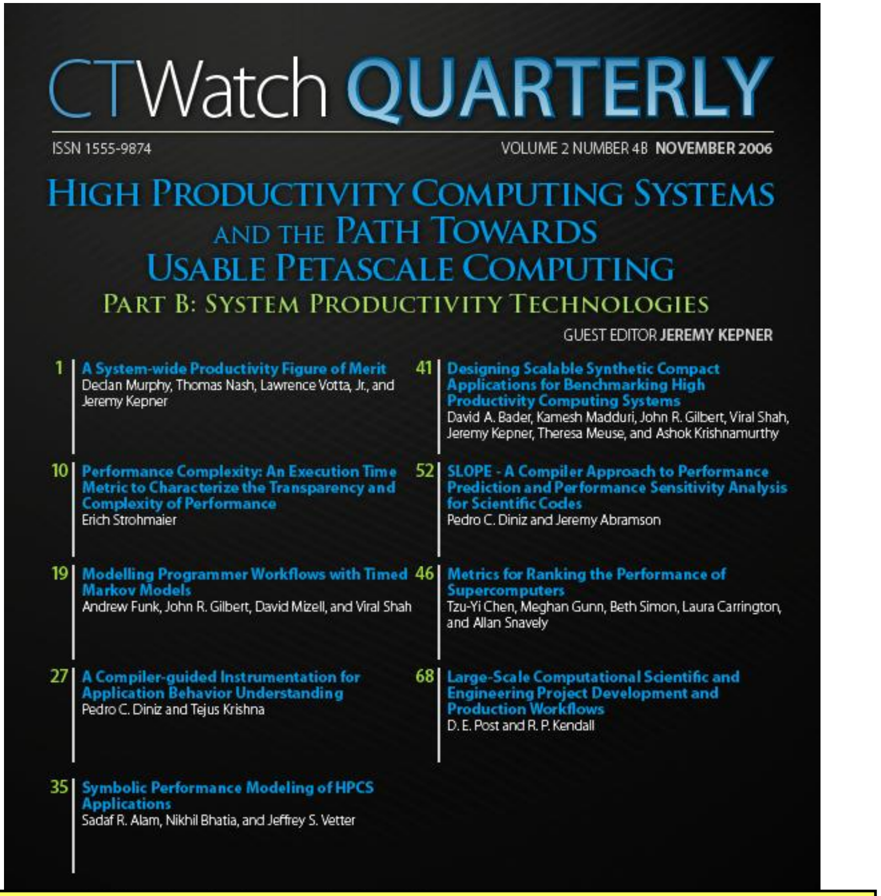
HIGH PRODUCTIVITY COMPUTING SYSTEMS AND THE PATH TOWARDS USABLE PETASCALE COMPUTING

PART A: USER PRODUCTIVITY CHALLENGES

GUEST EDITOR **JEREMY KEPNER**

INTRODUCTION

FEATURE ARTICLES



CTWatch QUARTERLY

ISSN 1555-9874 VOLUME 2 NUMBER 4B **NOVEMBER 2006**

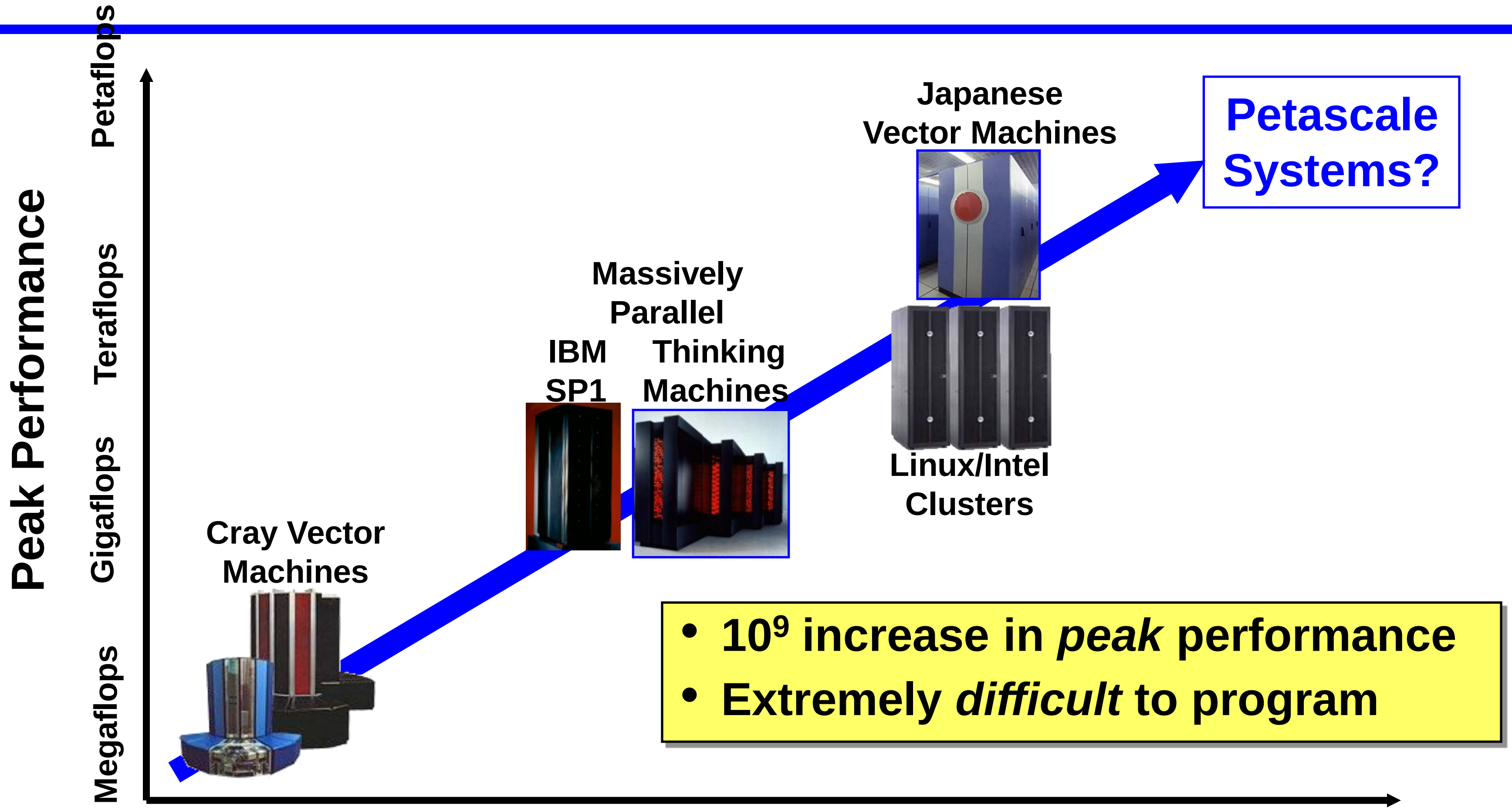
HIGH PRODUCTIVITY COMPUTING SYSTEMS AND THE PATH TOWARDS USABLE PETASCALE COMPUTING

PART B: SYSTEM PRODUCTIVITY TECHNOLOGIES

GUEST EDITOR **JEREMY KEPNER**

"The productivity of HPC users intrinsically deals with some of the brightest people on the planet, solving very complex problems, using the most complex computers in the world. Anyone who truly wants to get insight into such a complex situation must be prepared to invest some time in the endeavor."

Evolution of Supercomputing



- 10^9 increase in *peak* performance
- Extremely *difficult* to program

	1980s	1990s	2000s	2010s
Killer Apps:	Weapons Design, Cryptanalysis	Internet, Biotech	Finance, Animation	Biotech, Entertainment
Processors:	~10	~1000	~10,000	~100,0000
% who can use:	Most →			Very Few

Notional Concept of Productivity (~2000)

- Not sure what it is but know we want it to be better
- "Big Tent" Philosophy
  - Lots of good things to do, pursue them all?
- Focus on:
  - Real (not peak) performance of critical national security applications
  - Programmability: reduce cost and time of developing applications
  - Software portability and system robustness

Formal Definition of Productivity (~2007)

- Productivity is a *very* well defined concept in economics

Productivity = Utility/Cost

- In an HPC Context

$$\Psi \equiv \frac{U}{C} = \frac{U(T)}{C_S + C_O + C_M}$$

ψ = productivity [utility/$]
U = utility [user specified]
T = time to solution [time]
C = total cost [$]

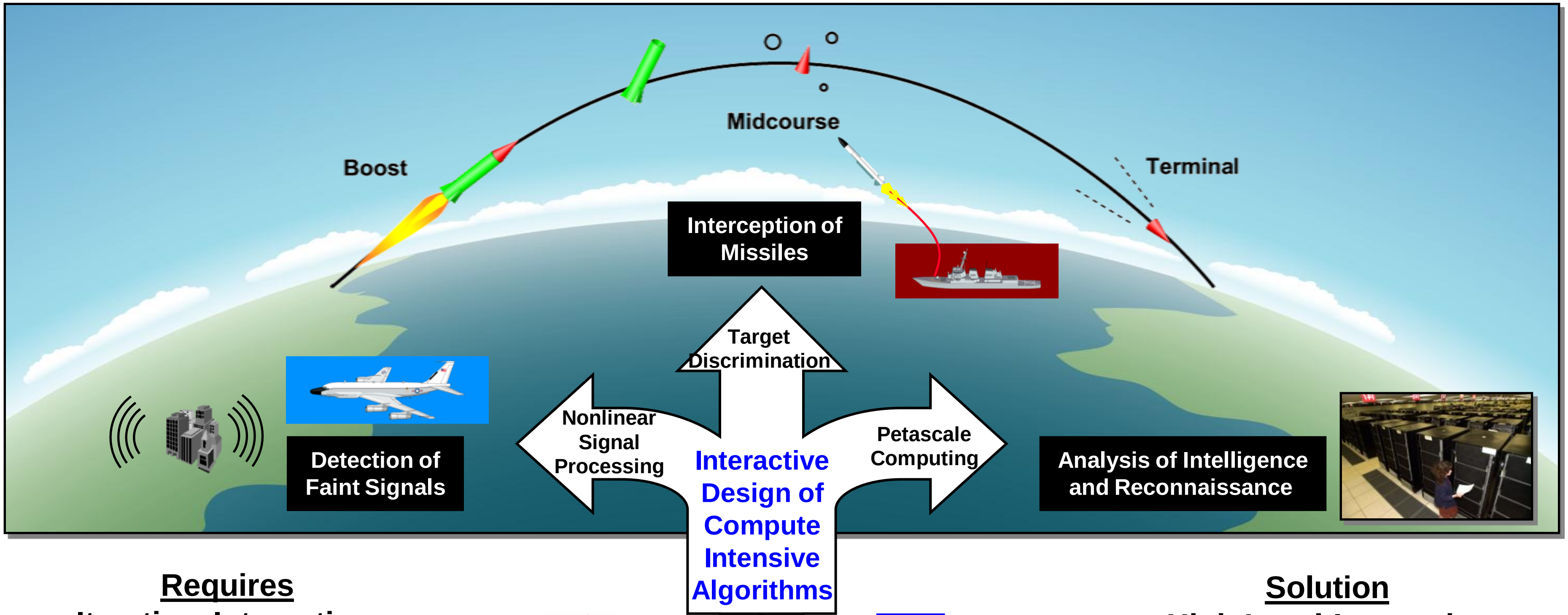
C_S = software cost [$]
C_O = operation cost [$]
C_M = machine cost [$]

- Software costs include time spent by users making codes parallel
- Operating costs include admin time, electric and building costs
- Utility is the stakeholder specific benefit of getting a result
  - Decision Makers
  - Project Managers
  - Users
  - Administrators
  - Service Engineers
  - Operators
  - Vendors/Designers
  - Technology Researchers

Outline

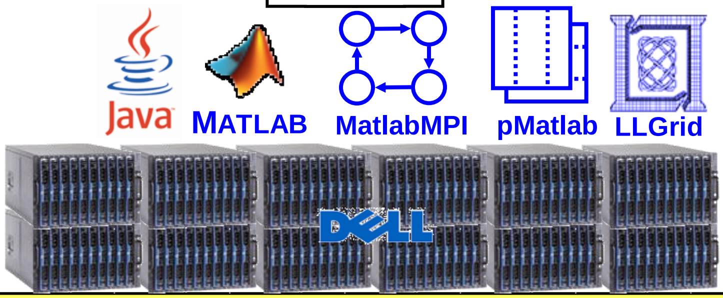
- Introduction
- **LLGrid Environment**
  - *Lincoln Mission*
  - *User Requirements*
  - *ROI Model*
  - *LLGrid Implementation*
- User Impact
- Summary

Lincoln Mission: Rapid Algorithm Development for National Security


Boost
Midcourse
Terminal
Interception of Missiles
Target Discrimination
Nonlinear Signal Processing
Petascale Computing
Detection of Faint Signals
Analysis of Intelligence and Reconnaissance
Interactive Design of Compute Intensive Algorithms

Requires
Iterative, Interactive Development

Requires
~10 Teraflops Computation
~1 Petabyte Virtual Memory

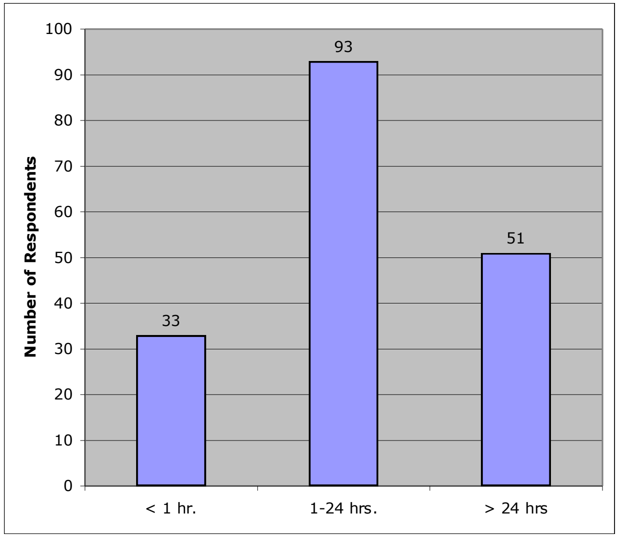


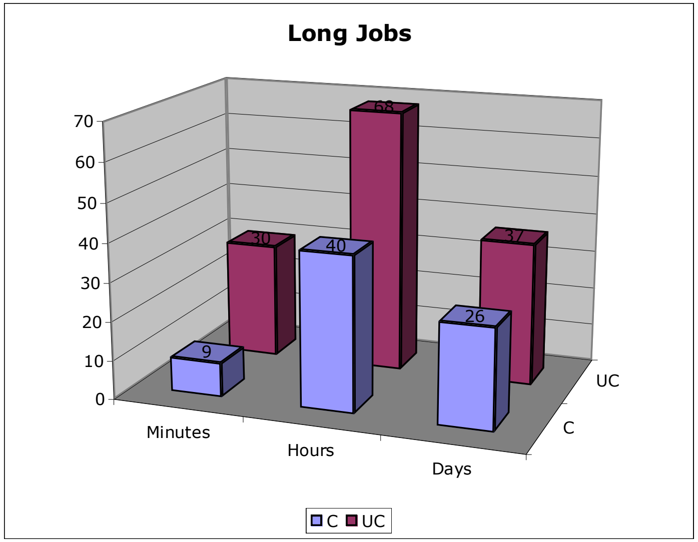
Solution
High Level Interactive Programming Environments

Solution
HPCMP Distributed HPC Project Hardware

- **HPC can significantly accelerate the interactive development and testing of algorithms critical to National Security**

User Requirements Survey





- Conducted survey (03-Nov-03) 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

Measuring Return On Investment

$$\begin{array}{c}\textbf{productivity}\\\textbf{(ROI)}\end{array} = \frac{\left(\textbf{Utility}\right)}{\left(\begin{array}{c}\textbf{Software}\\\textbf{Cost}\end{array}\right) + \left(\begin{array}{c}\textbf{Maintenance}\\\textbf{Cost}\end{array}\right) + \left(\begin{array}{c}\textbf{System}\\\textbf{Cost}\end{array}\right)}$$

Measuring Return On Investment

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

Measuring Return On Investment

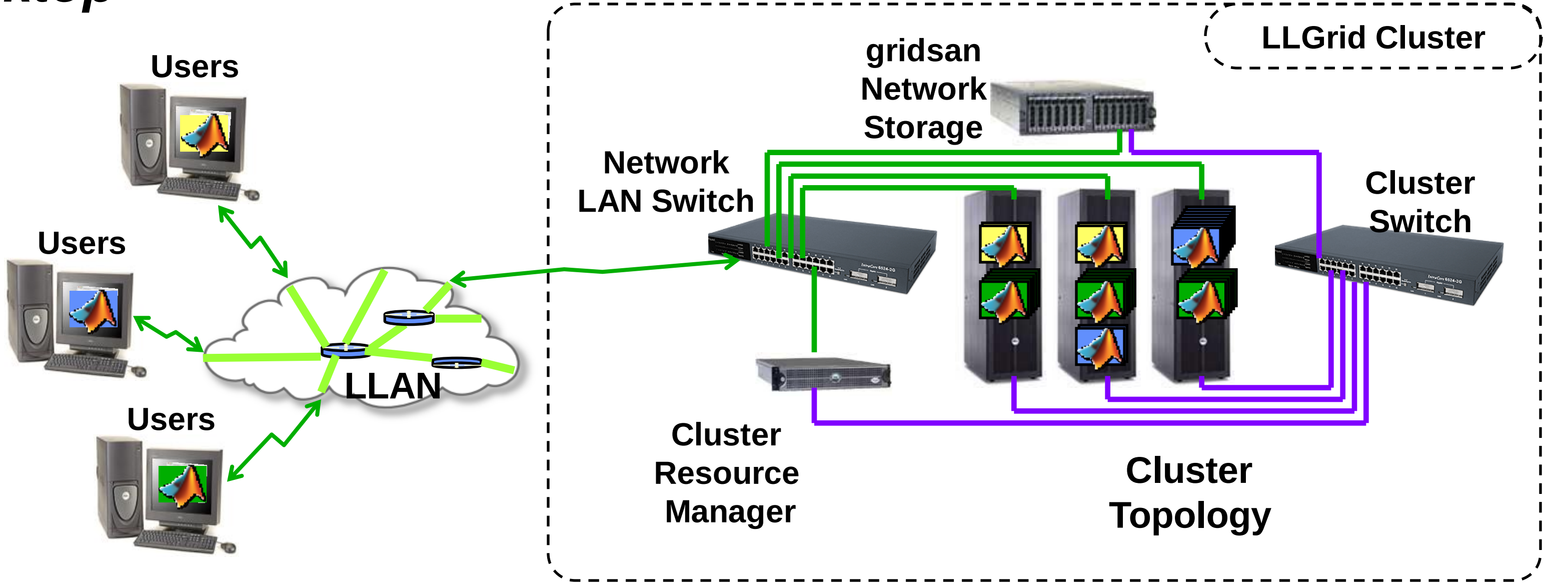
$$\underset{\text{(ROI)}}{\text{productivity}} = \frac{\left(\text{time saved by users on system}\right)}{\left(\text{time to parallelize}\right) + \left(\text{time to train}\right) + \left(\text{time to launch}\right) + \left(\text{time to admin.}\right) + \left(\text{system cost}\right)}$$

LLGrid Implementation Approach

- Focus on accelerating user apps; test tools on benchmarks
- Focus on broad user base
- Keep code changes to a few lines (PGAS)
- Use familiar environment (pMatlab)
- Use hands on expert consulting model
- 100% Interactive (no waiting)
- Build/train admin team first, then scale hardware
- Co-design system and facility

LLGrid Scalable System Architecture

__Goal__: ***To make enterprise wide access to high throughput Grid computing and distributed storage as easy as running on the desktop***

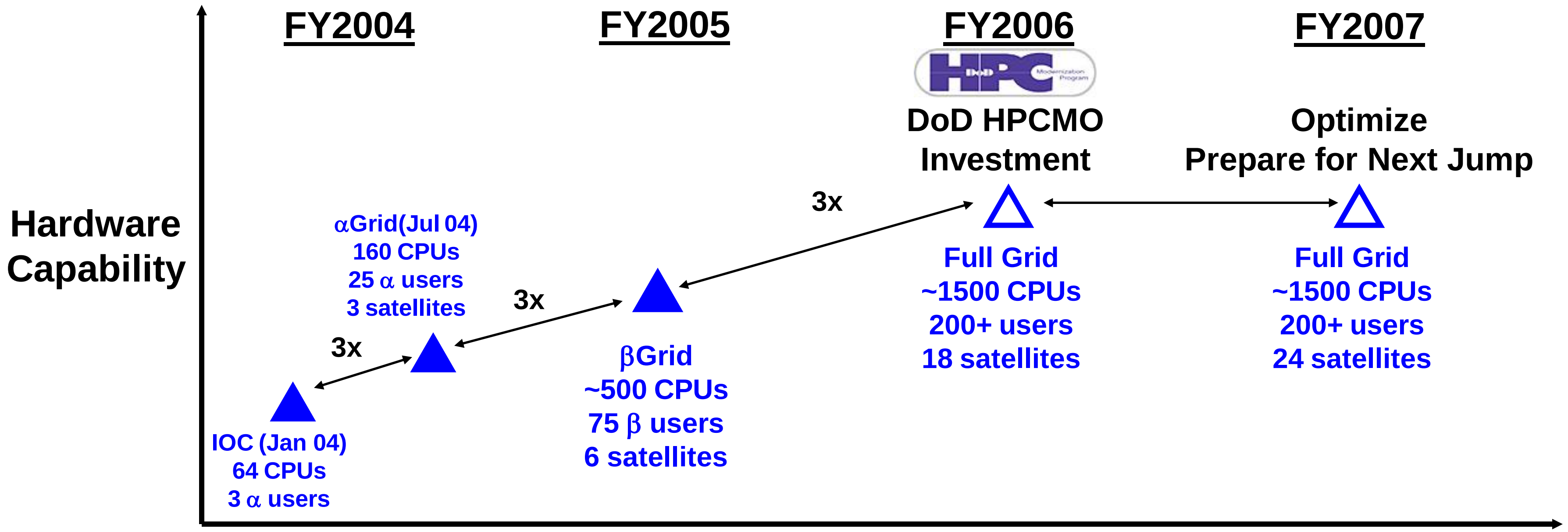


__Scalable__
-Processing
-Data
-Users

__Lab Grid Computing Components__

- High Performance Computers
- High Throughput Storage
- Parallel Processing Libraries

LLGrid Hardware Growth



Hardware Capability

FY2004

FY2005

FY2006

FY2007

DoD HPCMO Investment

Optimize
Prepare for Next Jump

IOC (Jan 04)
64 CPUs
3 α users

3x

αGrid(Jul 04)
160 CPUs
25 α users
3 satellites

3x

βGrid
~500 CPUs
75 β users
6 satellites

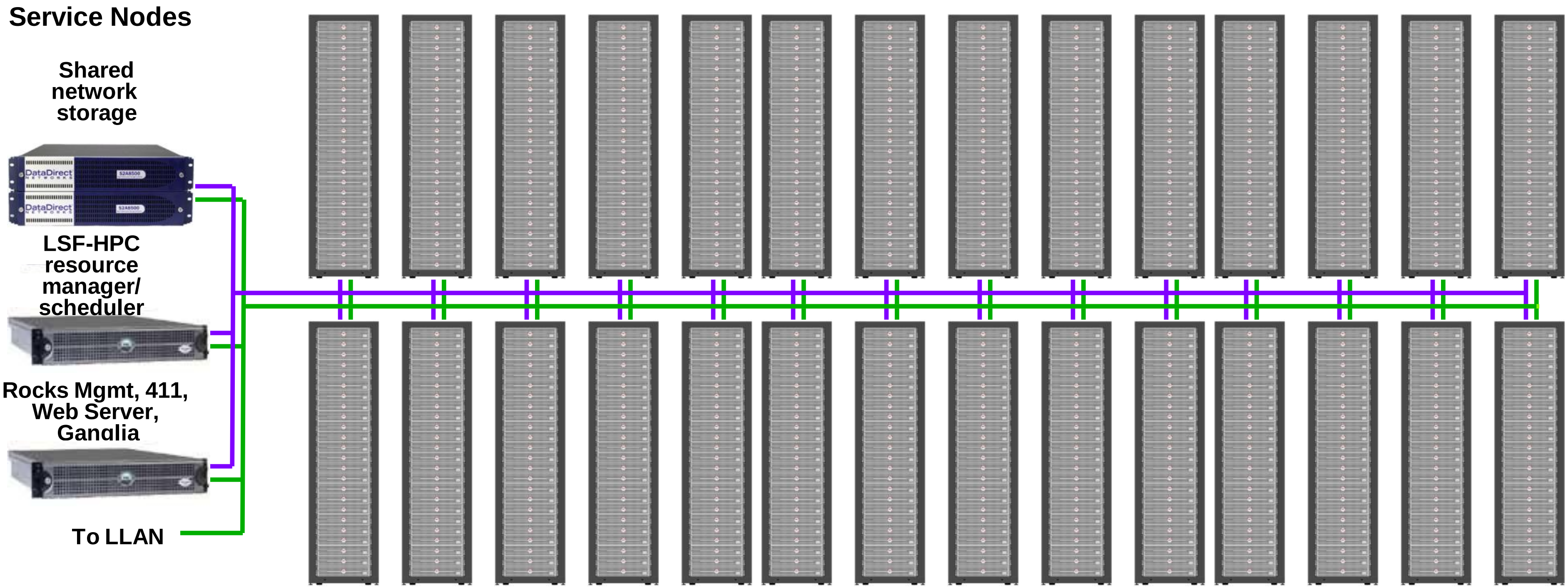
3x

Full Grid
~1500 CPUs
200+ users
18 satellites

Full Grid
~1500 CPUs
200+ users
24 satellites

- Goal: increase hardware while keep staff costs constant
- Approach: built team first, hardware second
- Growing 3X every year for 4 years.
- Current capability is
  - LLGrid ~1500 CPUs ~750 nodes ~40 racks
  - Satellites ~18 x 2 racks ~40 racks
  - Total ~80 racks

TX-2500: Hardware/Facility Co-Design



Service Nodes

Shared network storage

LSF-HPC resource manager/ scheduler

Rocks Mgmt, 411, Web Server, Ganglia

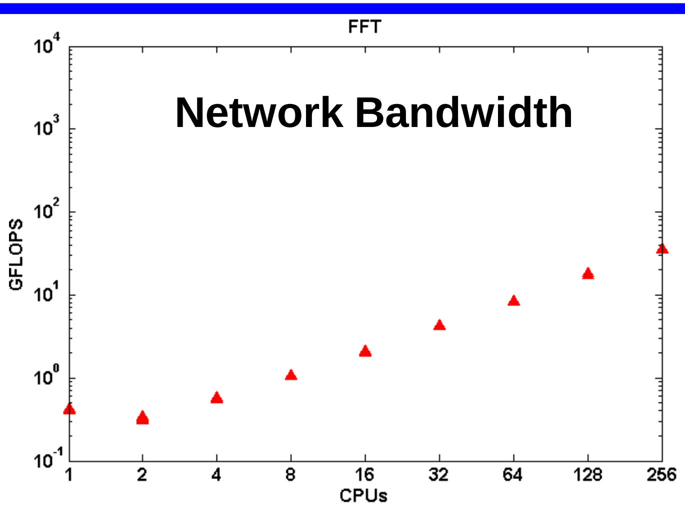
To LLAN

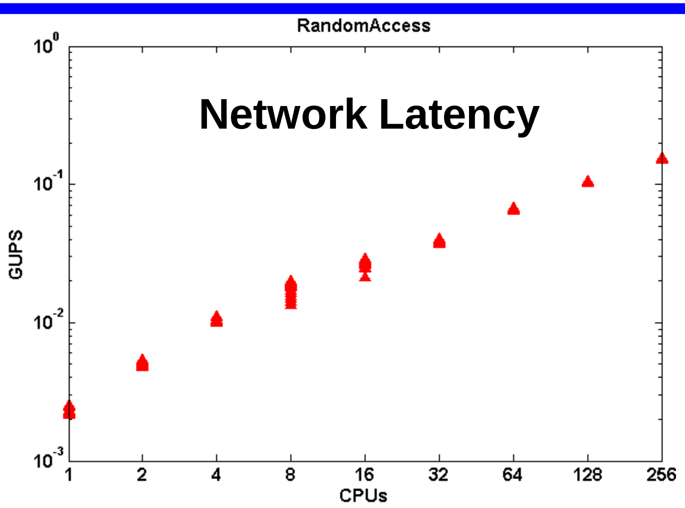
432 DELL PowerEdge 2850

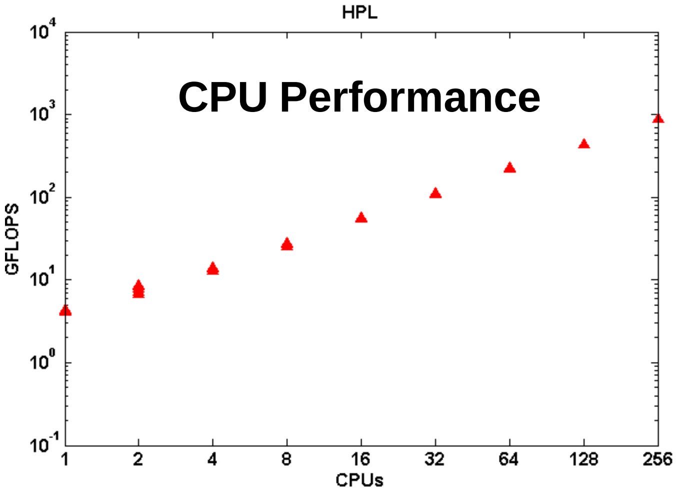
Dual 3.2 GHz EM64-T Xeon (P4)
8 GB RAM memory
Two Gig-E Intel interfaces
Infiniband interface
Six 300-GB disk drives

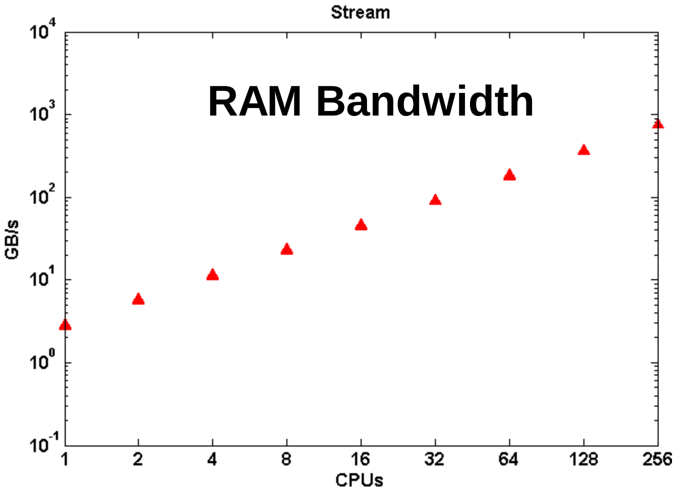
- 432+5 Nodes
- 864+10 CPUs
- 3.4 TB RAM
- 0.78 PB of Disk
- 28 Racks

Effectiveness Testing: HPC Challenge


Network Bandwidth


Network Latency


CPU Performance


RAM Bandwidth

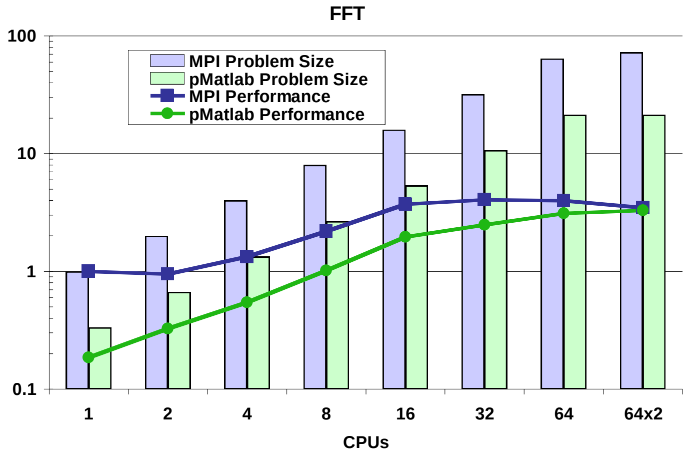
- Rigorously test with actionable benchmarks

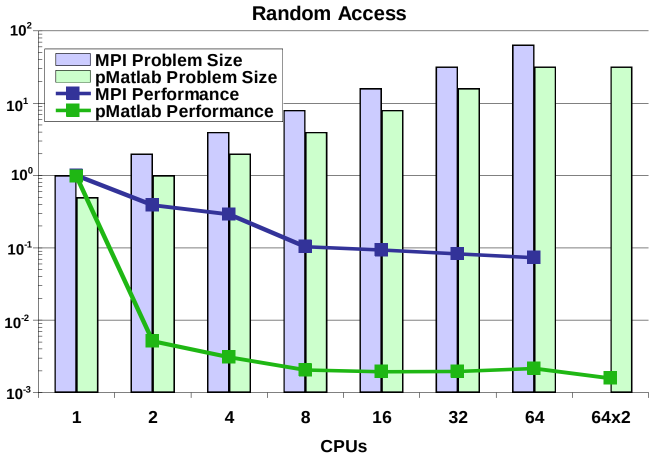
Exploit Benefits of High Level Language + PGAS

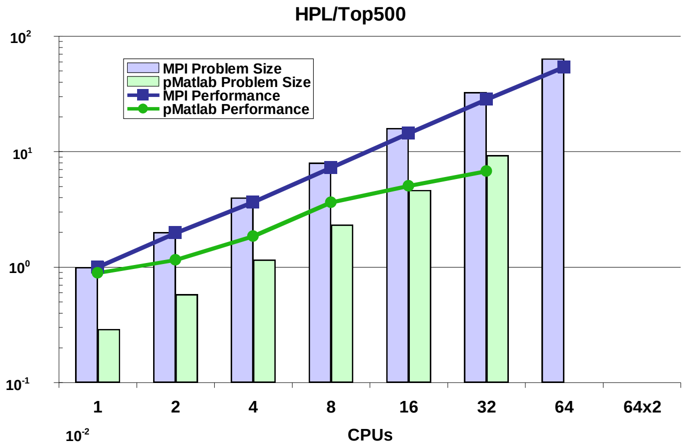
Technology	UPC	F2008	GA++	PVL	VSIPL	PVTOL	Titanium	StarP	pMatlab	DCT	Chapel	X10	Fortress
Organization	Std Body	Std Body	DOE PNNL	Lincoln	Std Body	Lincoln	UC Berkeley	ISC	Lincoln	Math-works	Cray	IBM	Sun
Sponsor	DoD	DOE SC	DOE	Navy	DoD HPCMP		DOE, NSF	DoD	DARPA		DARPA	DARPA	DARPA
Type	Lang Ext	Lang Ext	Library	Library	Library	Library	New Lang	Library	Library	Library	New Lang	New Lang	New Lang
Base Lang	C	Fortran	C++	C++	C++	C++	Java	Matlab	Matlab	Matlab	ZPL	Java	HPF
Precursors		CAF		STAPL, POOMA	PVL, POOMA	VSIPL++, pMatlab		pMatlab	PVL, StarP	pMatlab, StarP			
Real Apps	2001	2001	1998	2000	2004	~2007		2002	2003	2005			
Data Parallel	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Block-cyclic	1D		ND blk	2D	2D	Y	ND	2D	4D	1D	ND	ND	
Atomic			Y									Y	Y
Threads	Y		Y								Y	Y	Y
Task Parallel			Y	Y	Y	Y	Y		Y		Y	Y	
Pipelines			Y	Y		Y			Y				
Hier. arrays						Y	Y		Y		Y	Y	Y
Automap				Y		Y			Y				
Sparse							?	Y	Y	Y	Y	?	?
FPGA IO					Y	Y							

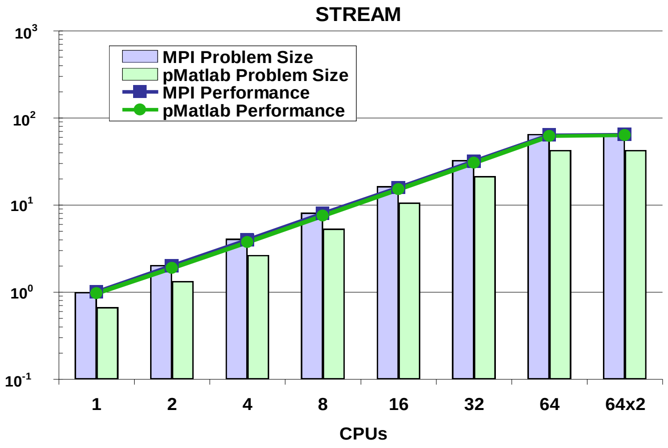
- **PGAS + high level environments is a "no brainer"; widely implemented; enables complex programs; makes simple programs trivial (even on clusters); community has settled on a common set of features**
  - **Data parallelism, block cyclic data distributions, atomic sections, threads, task parallelism, pipeline constructs, hierarchical arrays, and sparse arrays**

pMatlab HPC Challenge on LLGrid









- **Tested pMatlab against HPC Challenge benchmarks to verify performance and properly manage user expectations**

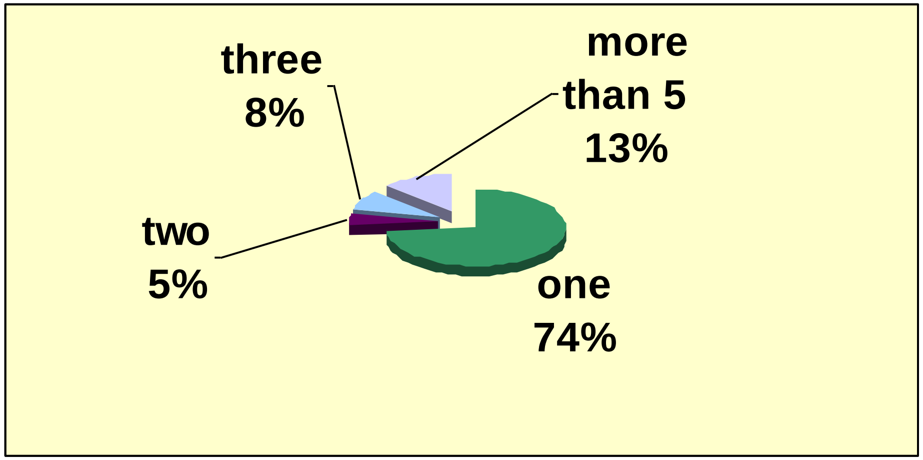
Outline

- Introduction
- LLGrid Environment
- **Results**
  - *User Response*
  - *Usage Statistics*
  - *ROI Calculation*
- Summary

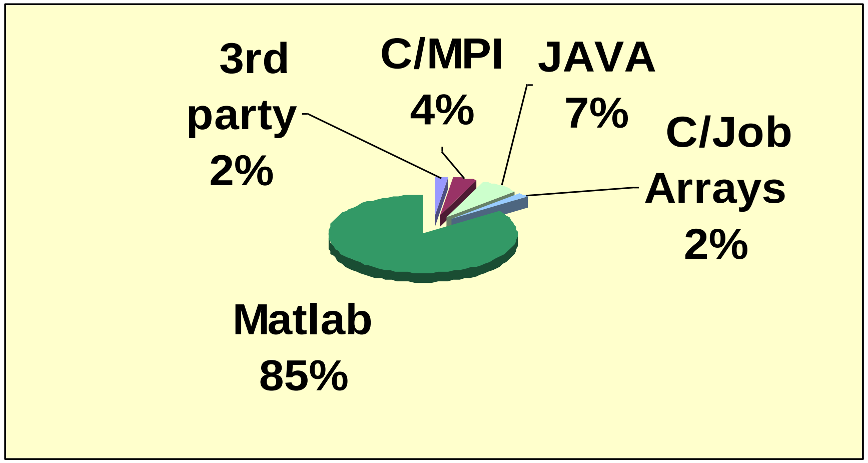
User Time to Parallelize

Project	Serial Code Dev Time	Time to Parallelize
1	2000 hours	8 hours
2	1300 hours	1 hour
3	40 hours	0.4 hours
4	900 hours	0.75 hours
5	40 hours	1 hour
6	700 hours	8 hours
7	600 hours	3 hours
8	650 hours	40 hours
0	960 hours	6 hours

Results from LLGrid Feedback Interviews

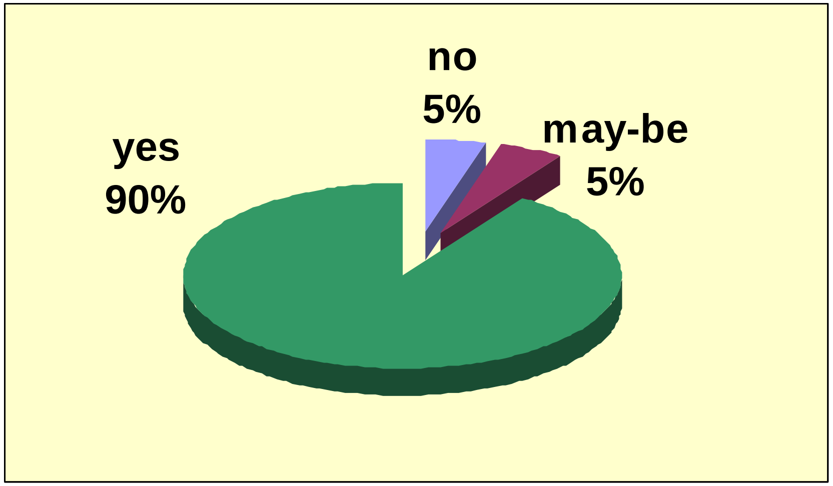


Number of Projects on LLGrid



Languages Used on LLGrid

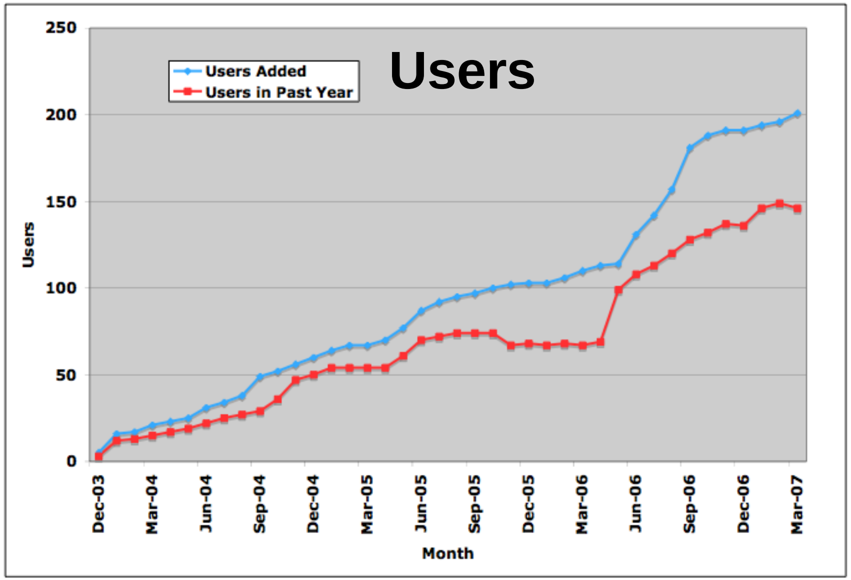
- Used meeting as a chance to probe any problems or issues users might have encountered
- 54 of 70 active unclassified users responded (~81%)
- 13 have not used LLGrid – 100% of these have not used it because of changes in their project or project goals
- Results are from 46 users across the Laboratory

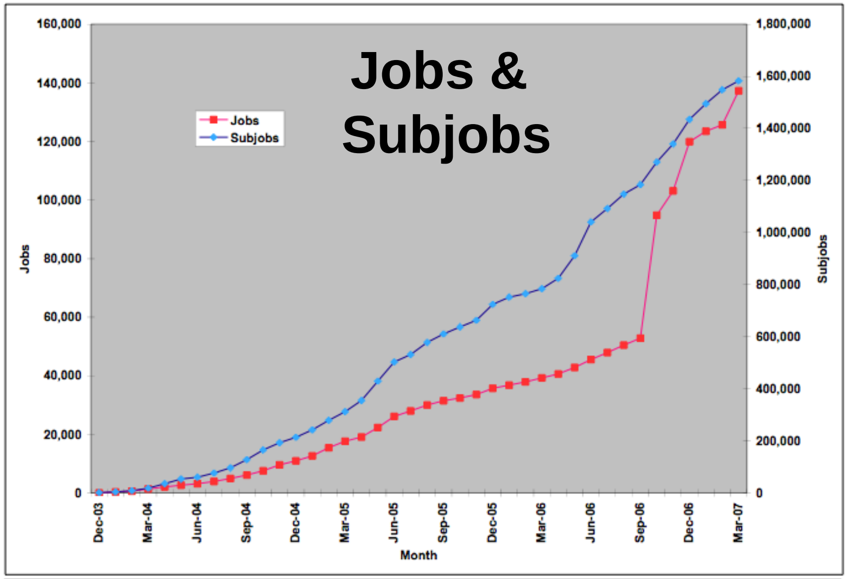


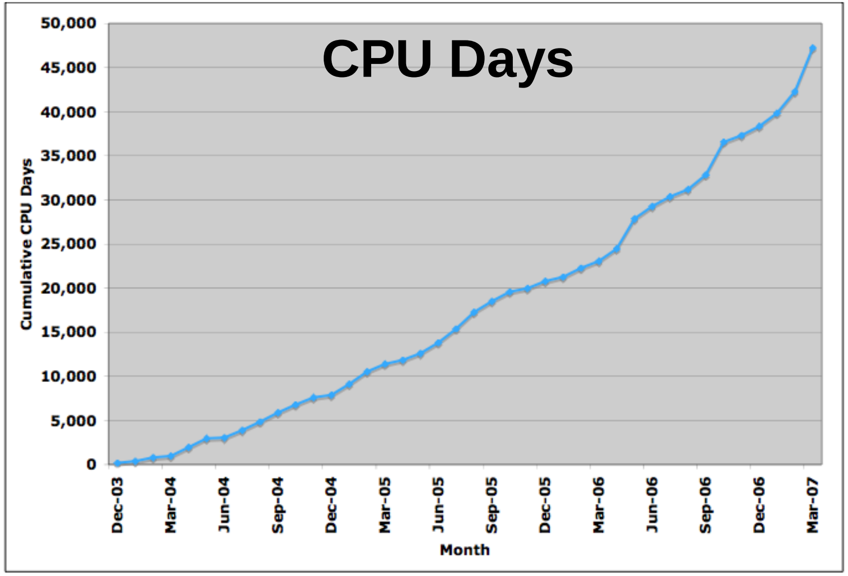
Recommend LLGrid

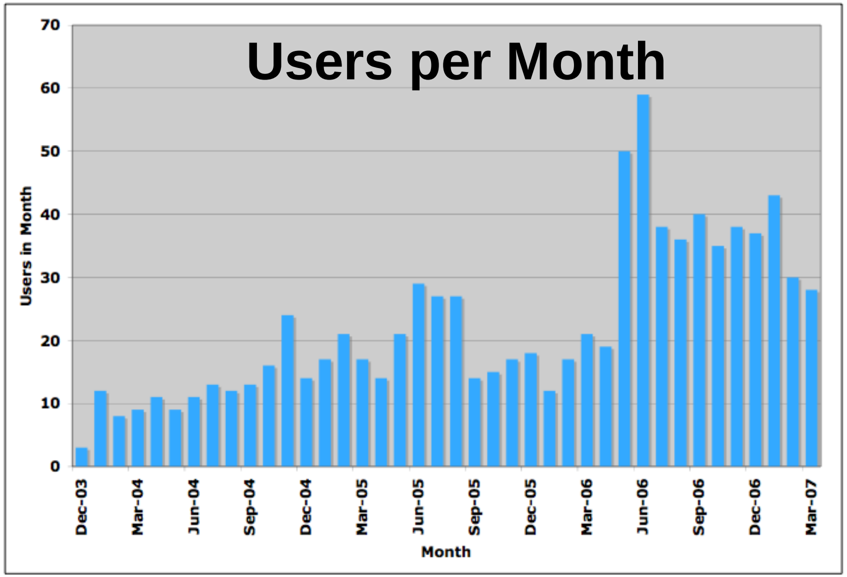
Cumulative Statistics per Month

December-03 to March-07

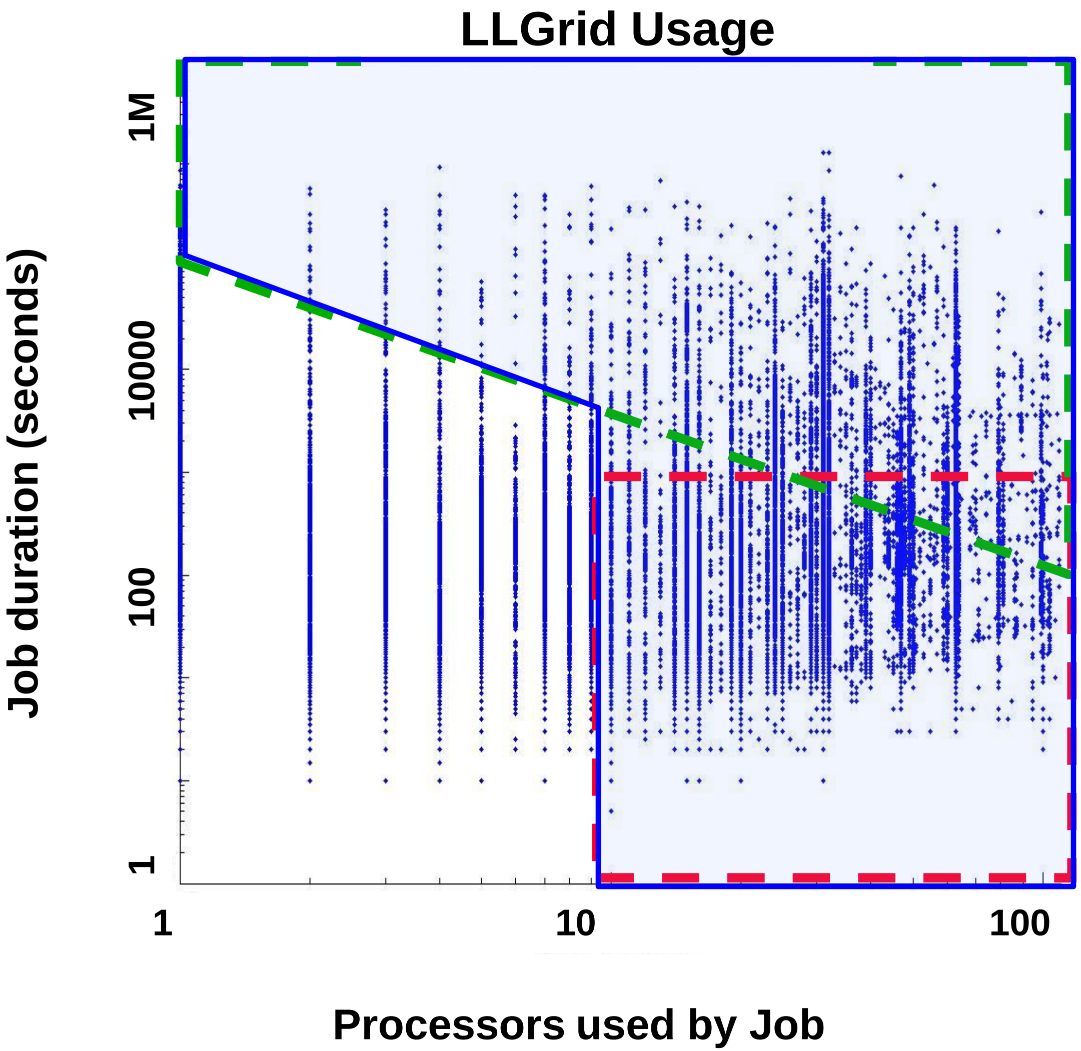

Users


Jobs & Subjobs


CPU Days


Users per Month

LLgrid Usage
December 2003 – March 2007



Statistics

- 186-280 CPUs
- 204 Users
- 29 Groups + campus
- 137,600 Jobs
- 47,700 CPU Days

>8 CPU hours - Infeasible on Desktop

>8 CPUs - Requires On-Demand Parallel Computing

Measuring Return On Investment

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

Production LLgrid model assumptions

- <VARIABLE> users Lab-wide
- <VARIABLE> simultaneous jobs
- Average <VARIABLE> CPUs per job
- 2 SLOCs per hou
- 1000 SLOCs perr simulation * Lab-wide users
- 1.0% time-to-parallelize overhead
- Training time - 4 hours * Lab-wide users
- <VARIABLE> parallel job launches
- 10 seconds to launch
- <VARIABLE> sys-admins
- <VARIABLE> CPUs @ $<VARIABLE> per node

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

$$or \quad * \left(\text{Average \# of CPUs per job}\right)$$

$$or \quad * \left(\log_2\left(\text{Average \# of CPUs per job}\right)\right)$$

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

$$\left(\text{time to train}\right) = \left(\text{User labor rate}\right) * \left(\text{Total \# of users}\right) * \left(\text{Time to train a user}\right)$$

$$\left(\text{time to launch}\right) = \left(\text{User labor rate}\right) * \left(\text{Number of launches}\right) * \left(\text{Time to launch}\right)$$

$$\left(\text{time to admin.}\right) = \left(\text{Admin. labor rate}\right) * \left(\text{Number of admins}\right) * \left(\text{Admin time}\right)$$

$$\left(\text{system cost}\right) = \left(\text{User labor rate}\right) * \left(\text{Time-value of system}\right)$$

Measuring Return On Investment

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

Production LLgrid model comparisons

Parameters	*Past 3.1 Years*	*Past Year*	*Next Year*
Lab-wide Users	201	201	251
New Users in Latest Year	81	81	50
Active Users in Latest Year	146	146	175
Simultaneous Jobs	10	16	25
Avg. CPUs per Job	16	32	64
Total Job Launches	137,588	95,525	125,000
Number of System Administrators	4	4	4
Nodes in System	592	592	852
New Nodes in System (Latest Year)	442	442	280
Benefit/Cost (Linear: CPUs)	24.10	43.05	144.62
Benefit/Cost (Logarithmic: $\log_2$(CPUs))	4.17	4.66	9.40

Summary

	Total	Interactive		Virtual
	CPUs	CPUs	RAM	Memory
LLGrid	1500	1500	6 TB	1 PB
Top500 Rank	~75	~1	~75	~1

- **LLGrid would be ~75 on worldwide Top500 rank**
- **LLGrid is the worlds largest interactive system**
- **LLGrid is the worlds largest parallel Matlab system**
- **LLGrid is the worlds largest virtual memory system**
- **Lincoln has a higher fraction of its workforce using parallel computing than any organization in the world**
  - **20% of staff have accounts (<5% is typical across the country)**
  - **Active accounts are 60% of total (~10% is typical)**

- **By taking a ROI focused approach Lincoln has quickly developed a leadership capability in its area**