SIAM Conference on
Parallel Processing for Scientific Computing
Grand Hyatt Seattle • Seattle, Washington
February 24-26, 2010

Five Important Concepts to Consider When Using Computing High Performance Systems at Scale

Jack Dongarra

University of Tennessee
Oak Ridge National Laboratory
University of Manchester

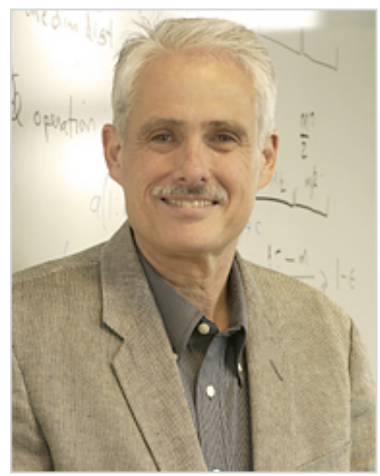
First, Thanks to My Mentors

Brian Smith

Cleve Moler

Jim Wilkinson

Gene Golub

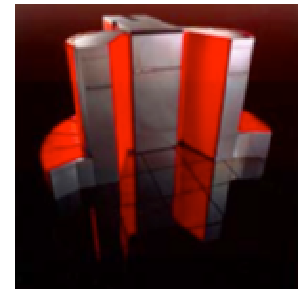


Ken Kennedy

And, Thanks to Many Colleagues

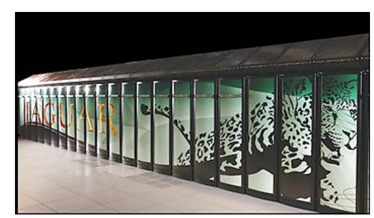
Looking at the Gordon Bell Prize

(Recognize outstanding achievement in high-performance computing applications and encourage development of parallel processing)



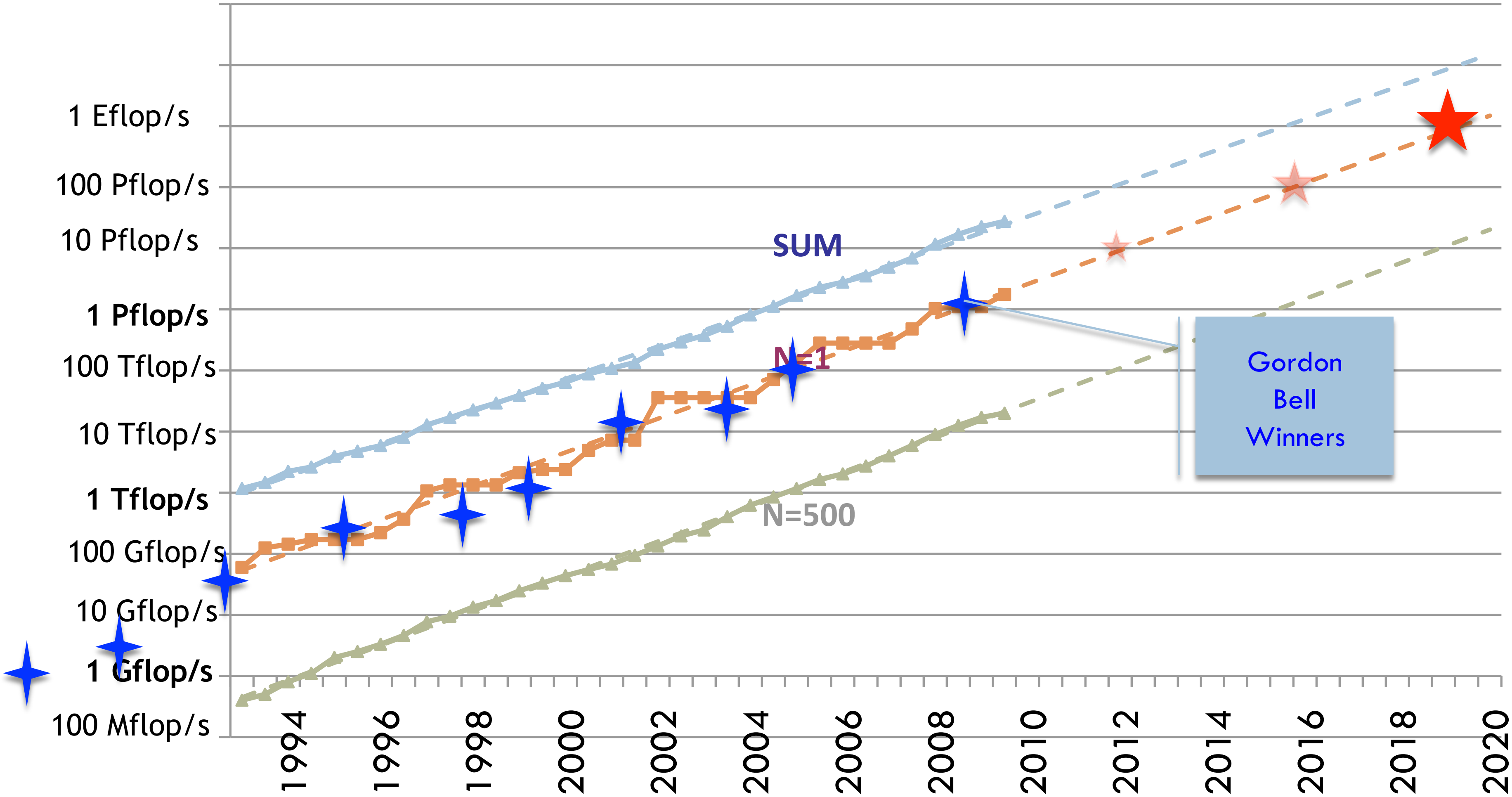
- 1 GFlop/s; 1988; Cray Y-MP; 8 Processors
  - Static finite element analysis

- 1 TFlop/s; 1998; Cray T3E; 1024 Processors
  - Modeling of metallic magnet atoms, using a variation of the locally self-consistent multiple scattering method.
- 1 PFlop/s; 2008; Cray XT5; $1.5x10^5$ Processors
  - Superconductive materials



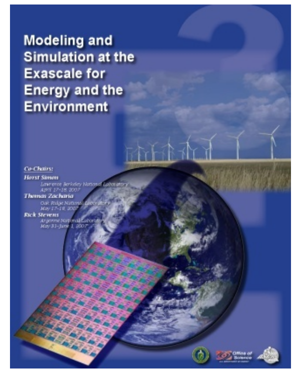
- 1 EFlop/s; ~2018; ?; $1x10^7$ Processors (10^9 threads)

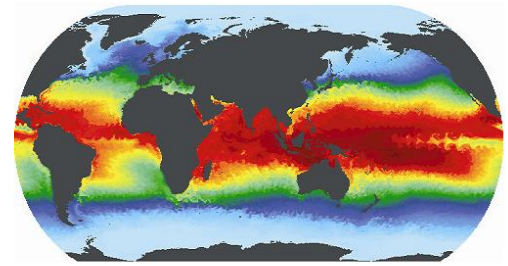
Performance Development in Top500

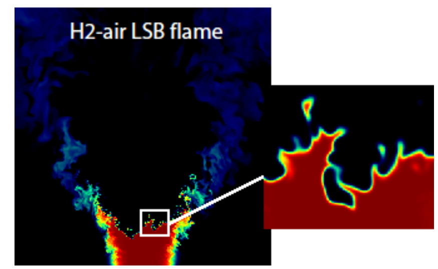


Process for Identifying Exascale Applications and Technology for DOE

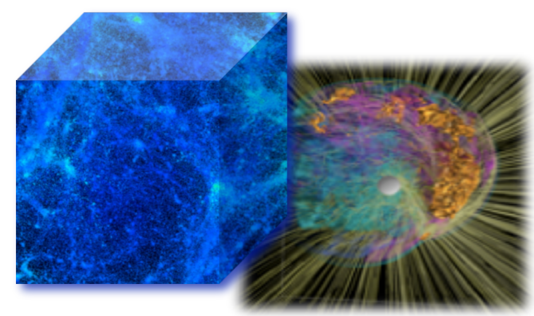
- **Town Hall Meetings April-June 2007**
- **Scientific Grand Challenges Workshops Nov, 2008 - Feb, 2010**
  - Climate Science (11/08),
  - High Energy Physics (12/08),
  - Nuclear Physics (1/09),
  - Fusion Energy (3/09),
  - Nuclear Energy (5/09),
  - Biology (8/09),
  - Material Science and Chemistry (8/09),
  - National Security (10/09)
  - Cross-cutting technologies (2/10)
- **Exascale Steering Committee**
  - "Denver" vendor NDA visits 8/2009
  - SC09 vendor feedback meetings
  - Extreme Architecture and Technology Workshop 12/2009
- **International Exascale Software Project**
  - Santa Fe, NM 4/2009; Paris, France 6/2009; Tsukuba, Japan 10/2009; Oxford 4/2010







MISSION IMPERATIVES



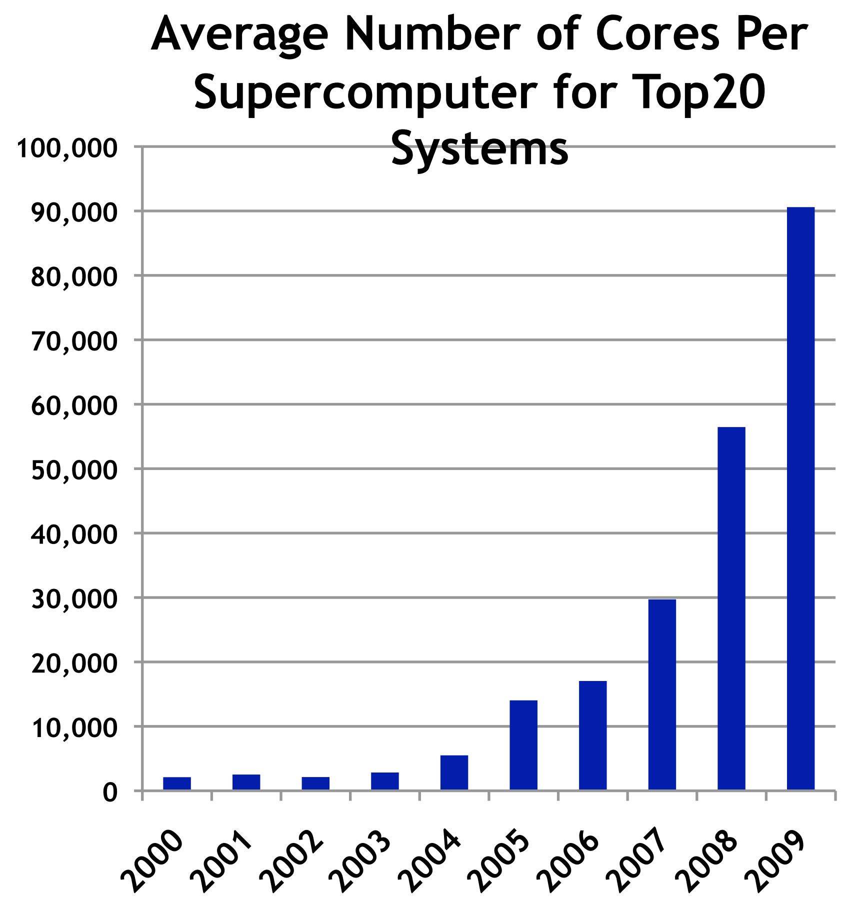
FUNDAMENTAL SCIENCE

Potential System Architecture

Systems	2009
System peak	**2 Pflop/s**
Power	**6 MW**
System memory	0.3 PB
Node performance	125 GF
Node memory BW	25 GB/s
Node concurrency	12
Total Node Interconnect BW	3.5 GB/s
System size (nodes)	18,700
Total concurrency	225,000
Storage	15 PB
IO	0.2 TB
MTTI	days

Factors that Necessitate Redesign of Our Software

- Steepness of the ascent from terascale to petascale to exascale
- Extreme parallelism and hybrid design
  - Preparing for million/billion way parallelism
- Tightening memory/bandwidth bottleneck
  - Limits on power/clock speed implication on multicore
  - Reducing communication will become much more intense
  - Memory per core changes, byte-to-flop ratio will change
- Necessary Fault Tolerance
  - MTTF will drop
  - Checkpoint/restart has limitations



Software infrastructure does not exist today

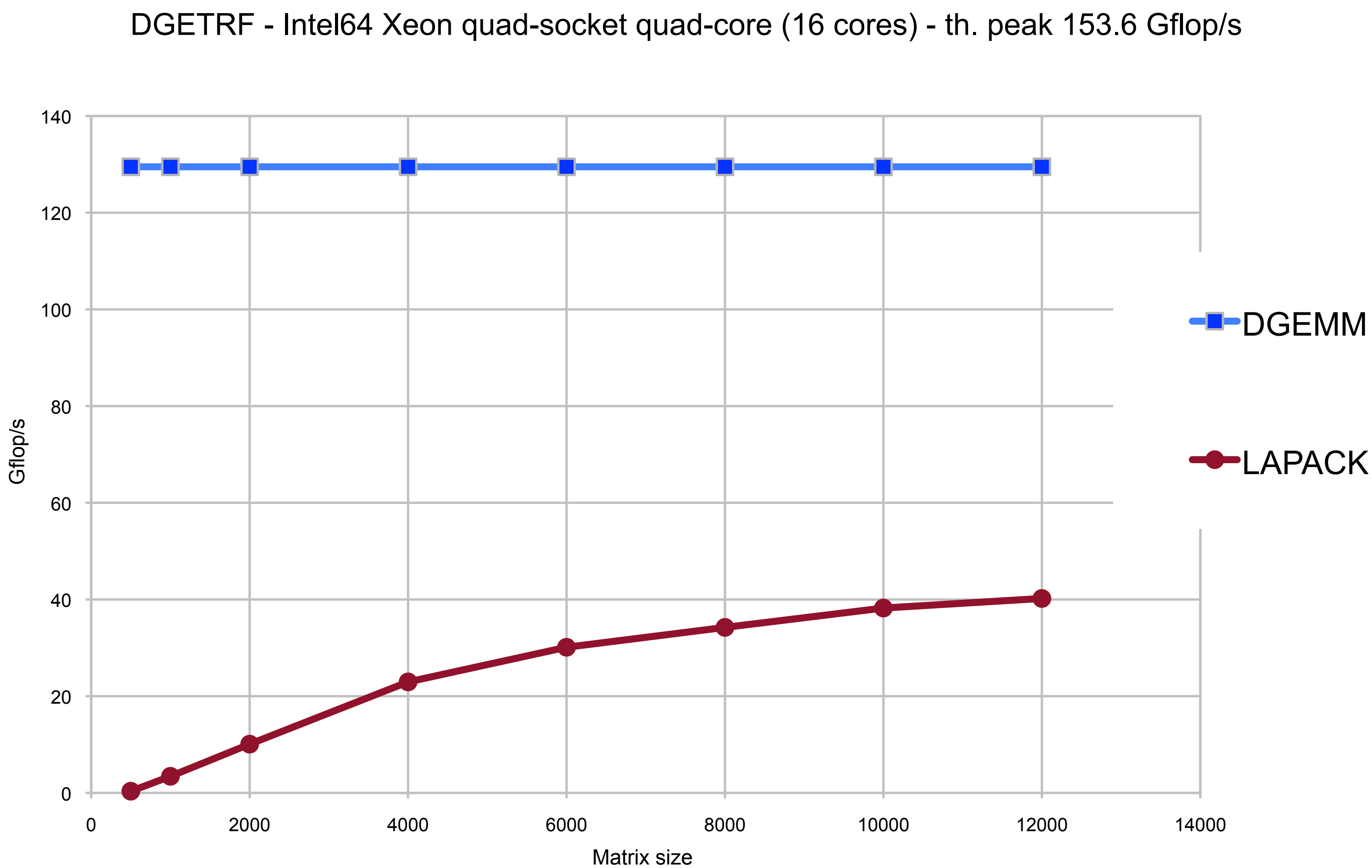
Major Changes to Software

- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with cluster computing and message passing
  - Rethink and rewrite the applications, algorithms, and software
- Numerical libraries for example will change
  - For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this

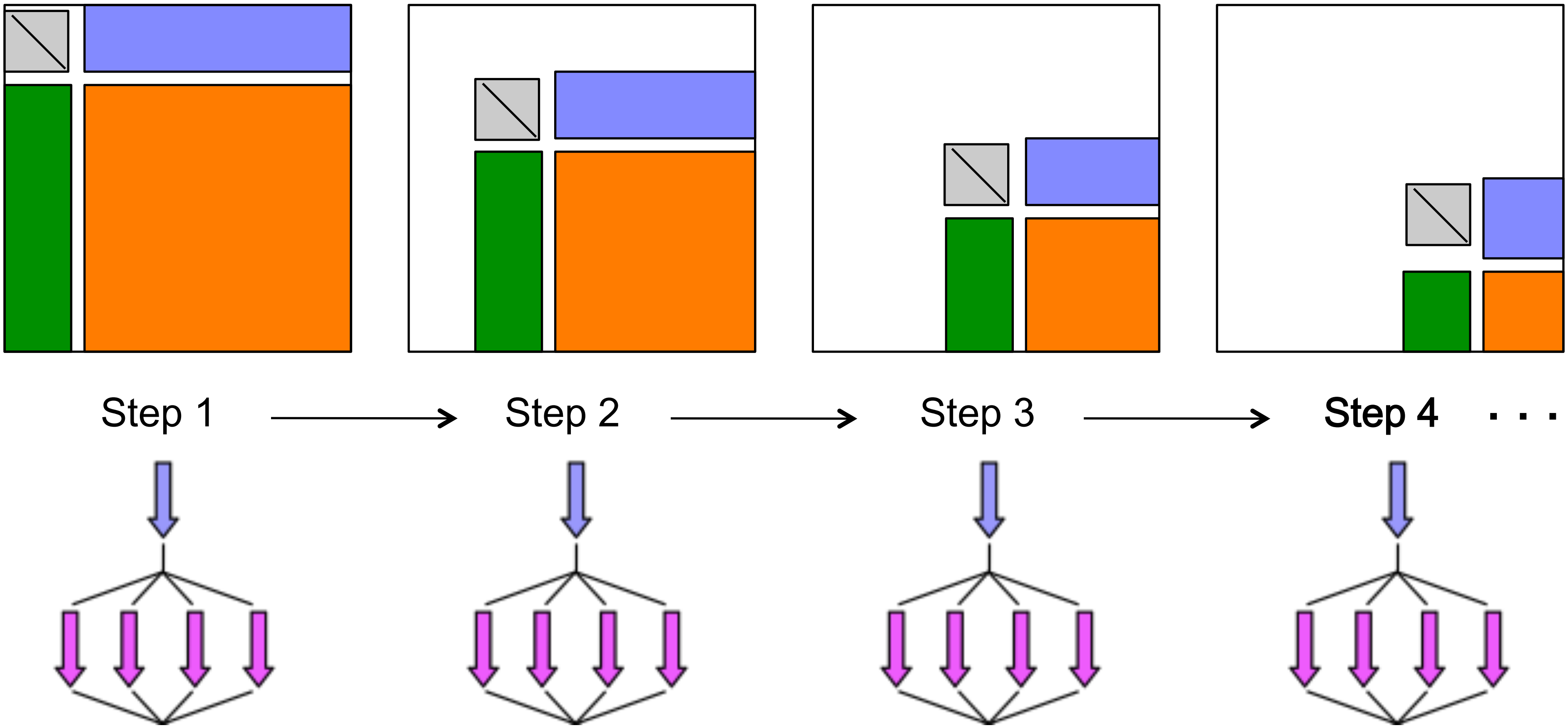
Five Important Features to Consider When Computing at Scale

- Effective Use of Many-Core and Hybrid architectures
  - Break fork-join parallelism
  - Dynamic Data Driven Execution
  - Block Data Layout
- Exploiting Mixed Precision in the Algorithms
  - Single Precision is 2X faster than Double Precision
  - With GP-GPUs 10x
  - Power saving issues
- Self Adapting / Auto Tuning of Software
  - Too hard to do by hand
- Fault Tolerant Algorithms
  - With 1,000,000's of cores things will fail
- Communication Reducing Algorithms
  - For dense computations from O(n *log* p) to O(*log* p) communications
  - Asynchronous iterations
  - GMRES k-step compute ($x, Ax, A^2x, \ldots A^kx$)

LAPACK LU - Intel64 - 16 cores

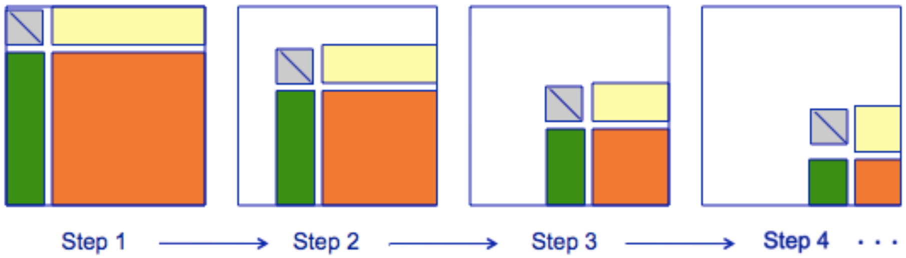


LAPACK LU

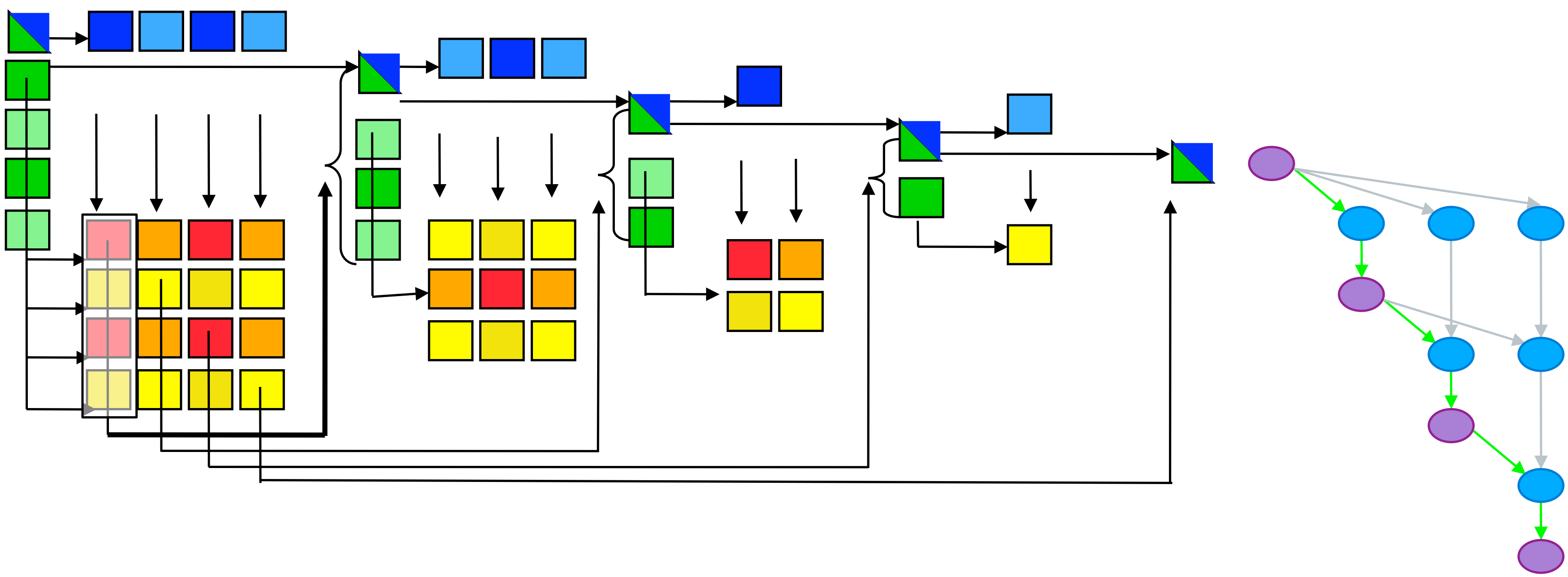


- **Fork-join, bulk synchronous processing**

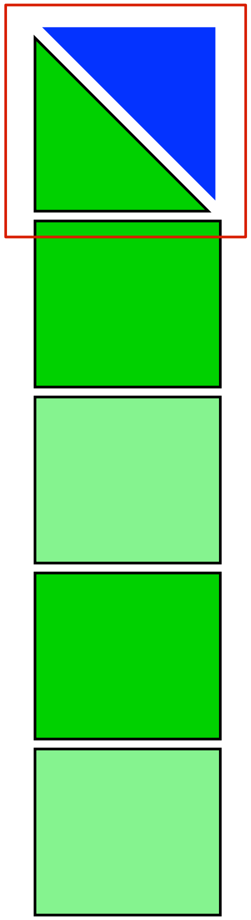
Parallel Tasks in LU



- Break into smaller tasks and remove dependencies

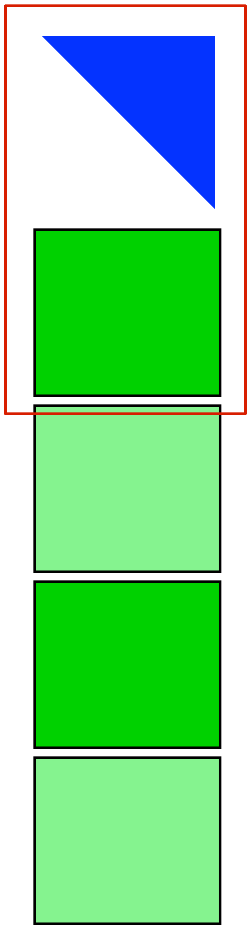


Parallel Tasks in LU



Step 1: LU of block 1,1 (w/partial pivoting)

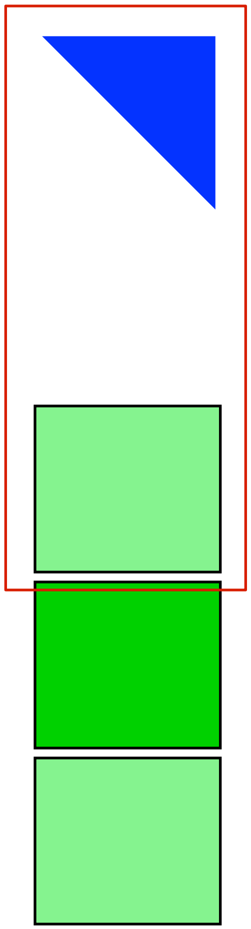
Parallel Tasks in LU



Step 1: LU of block 1,1 (w/partial pivoting)

Step 2: Use $U_{1,1}$ to zero $A_{1,2}$ (w/partial pivoting)

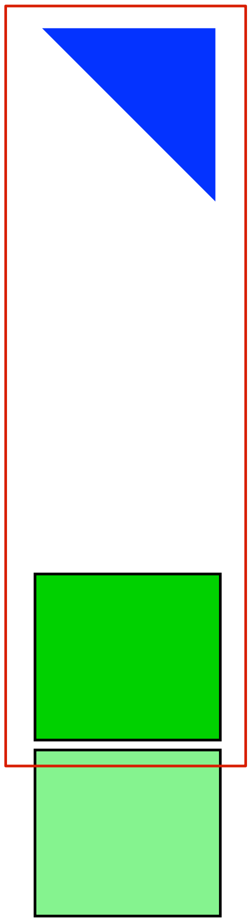
Parallel Tasks in LU



Step 1: LU of block 1,1 (w/partial pivoting)

Step 2: Use $U_{1,1}$ to zero $A_{1,2}$ (w/partial pivoting)

Parallel Tasks in LU



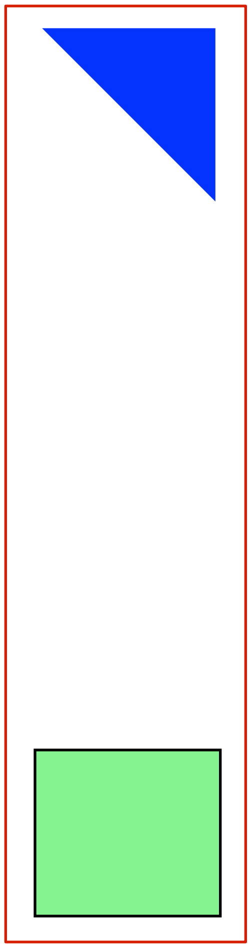
Step 1: LU of block 1,1 (w/partial pivoting)

Step 2: Use $U_{1,1}$ to zero $A_{1,2}$ (w/partial pivoting)

Step3: Use $U_{1,1}$ to zero $A_{1,3}$ (w/partial pivoting)

.
.
.

Parallel Tasks in LU



Step 1: LU of block 1,1 (w/partial pivoting)

Step 2: Use $U_{1,1}$ to zero $A_{1,2}$ (w/partial pivoting)

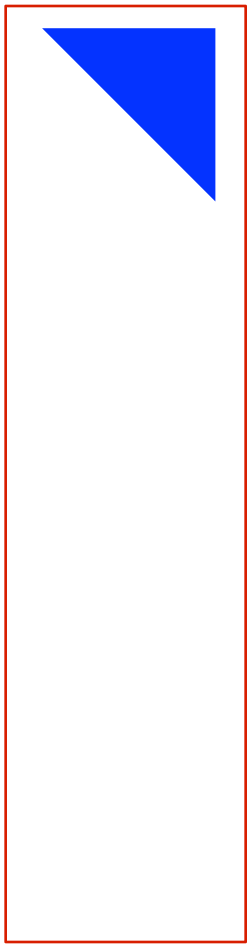
Step3: Use $U_{1,1}$ to zero $A_{1,3}$ (w/partial pivoting)

.

.

.

Parallel Tasks in LU



Step 1: LU of block 1,1 (w/partial pivoting)

Step 2: Use $U_{1,1}$ to zero $A_{1,2}$ (w/partial pivoting)

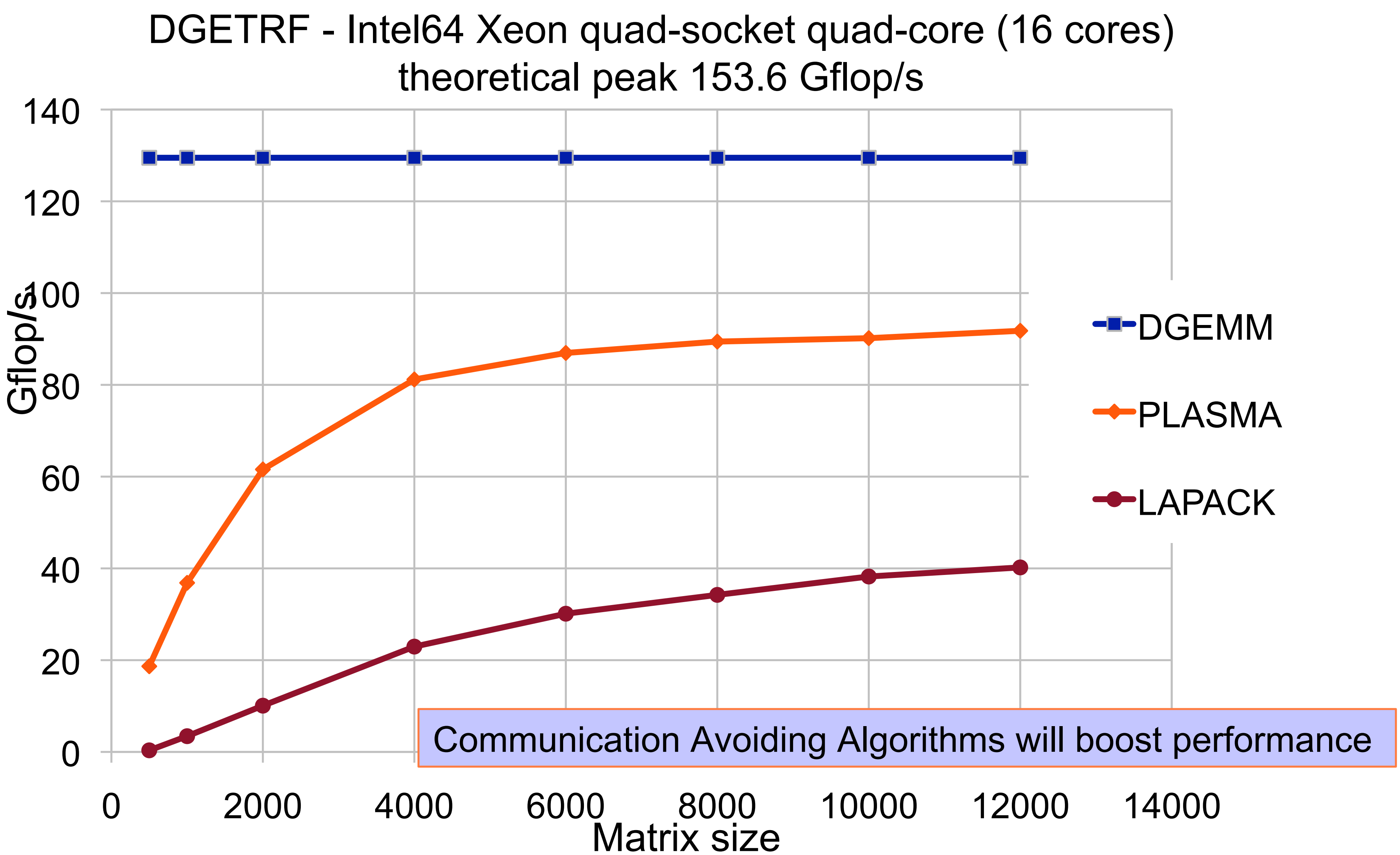
Step3: Use $U_{1,1}$ to zero $A_{1,3}$ (w/partial pivoting)

.

.

.

LU - Intel64 - 16 cores

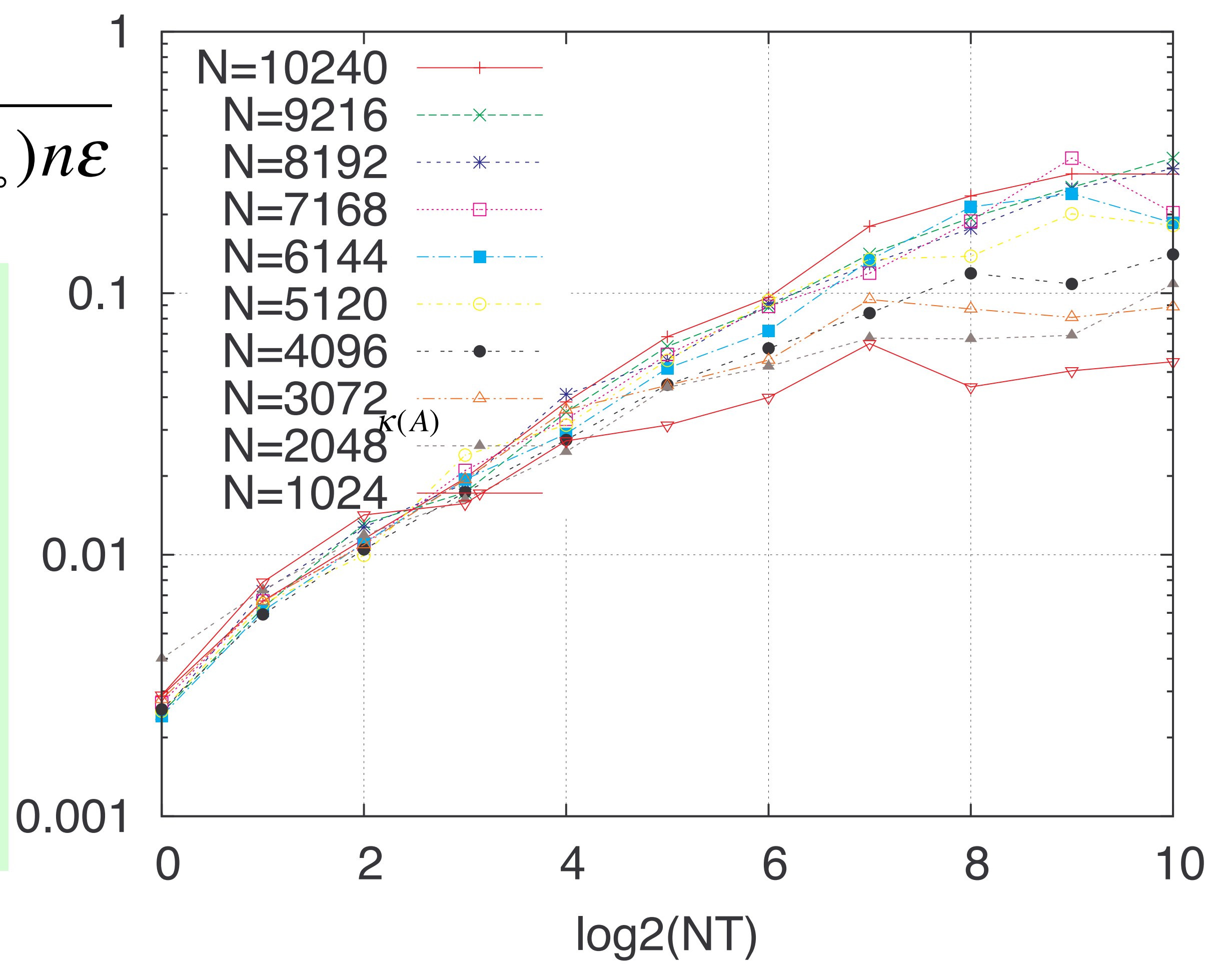


Residual from PLASMA's Tiled LU

$$\frac{\| Ax - b \|_\infty}{(\| A \|_\infty \| x \|_\infty + \| b \|_\infty) n\varepsilon}$$

Validation and verification

- Compute the answer fast with a possibly (but rarely) unreliable/ unstable algorithm
- Quickly check that the answer is ok (exception flags, small residual..)
- In the rare event of a problem, recompute carefully and slowly using classical approach



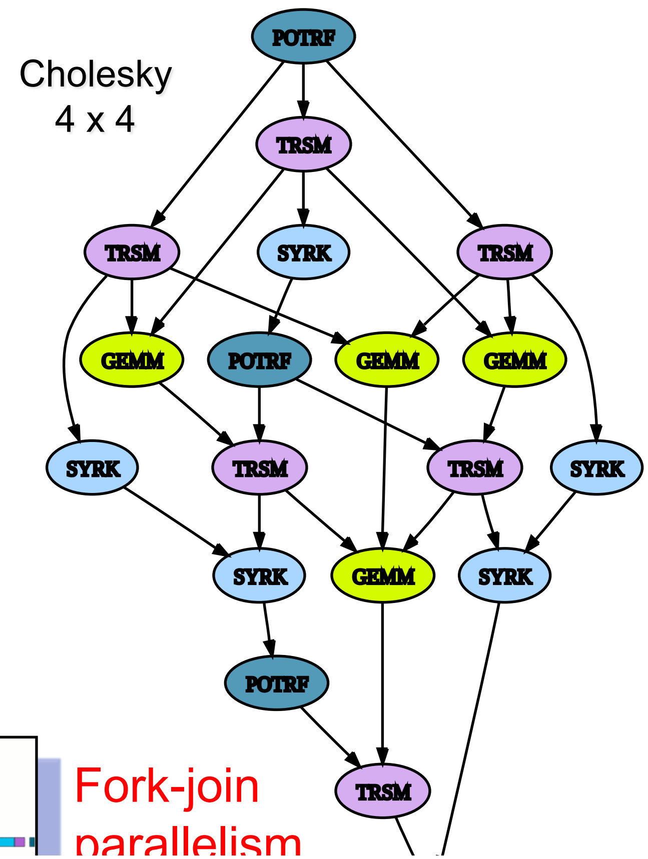
Random Matrices

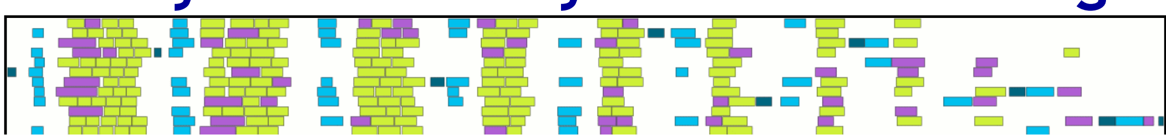
NT (Number of Tiles)

$\kappa(A) : 10^5 - 10^8$

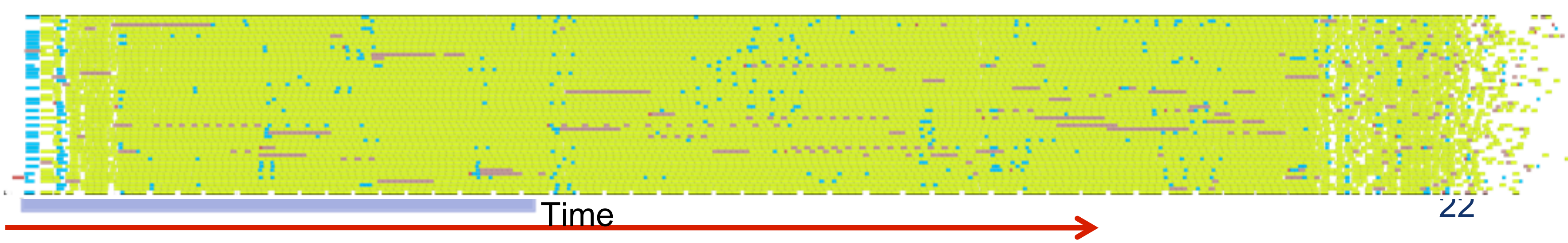
PLASMA: Parallel Linear Algebra s/w for Multicore Architectures

- **Objectives**
  - high utilization of each core
  - scaling to large number of cores
  - shared or distributed memory
- **Methodology**
  - DAG scheduling
  - explicit parallelism
  - implicit communication
  - Fine granularity / block data layout
- **Arbitrary DAG with dynamic scheduling**





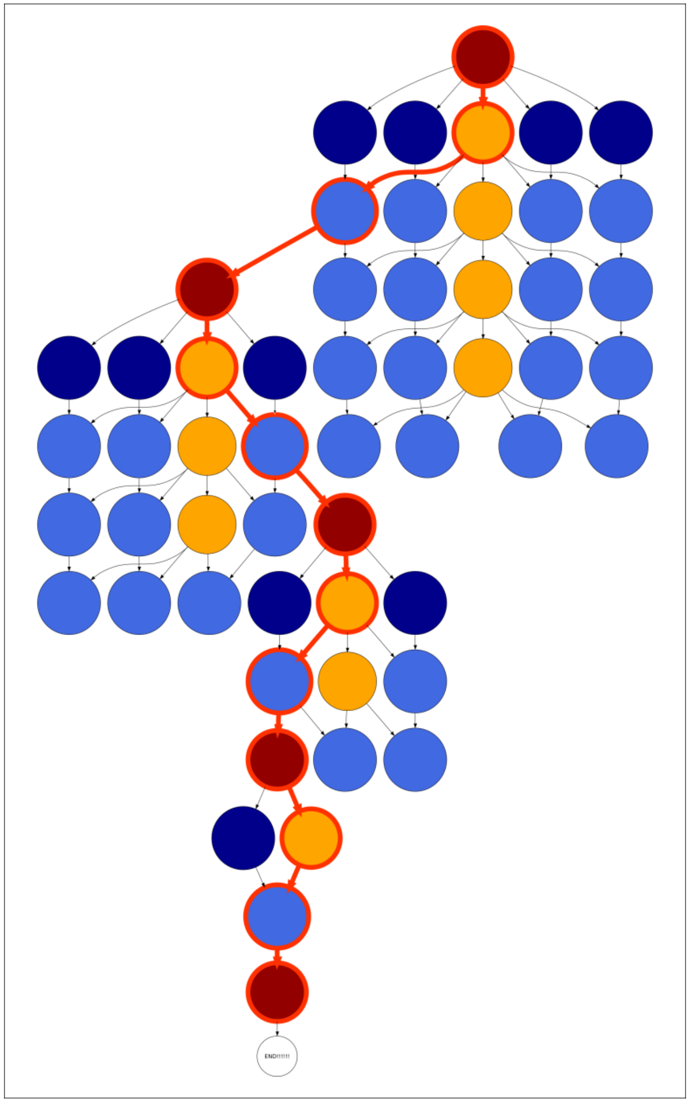
Fork-join parallelism



Time

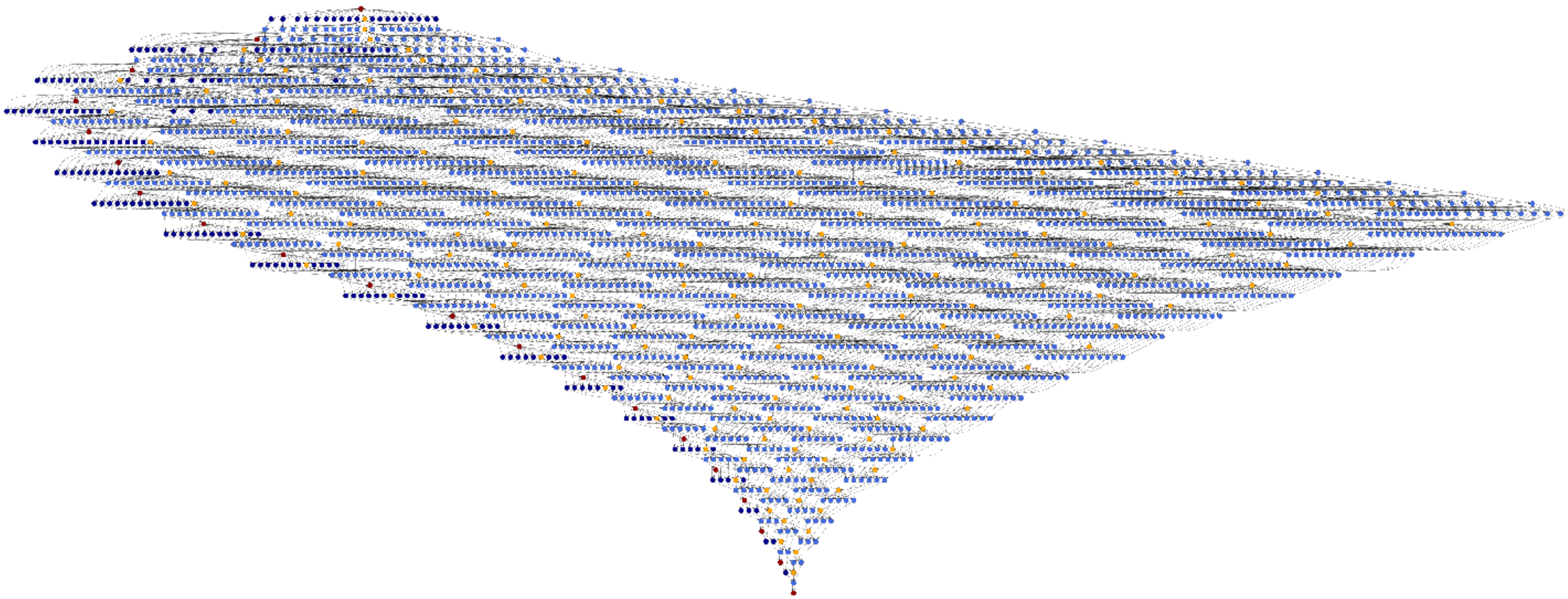
If We Had A Small Matrix Problem

- We would generate the DAG, find the critical path and execute it.
- DAG too large to generate ahead of time
  - Not explicitly generate
  - Dynamically generate the DAG as we go
- Machines will have large number of cores in a distributed fashion
  - Will have to engage in message passing
  - Distributed management
  - Locally have a run time system



The DAGs are Large

- Here is the DAG for a factorization on a 20 x 20 matrix



- For a large matrix say $O(10^6)$ the DAG is huge
- Many challenges for the software

- **www.exascale.org**

INTERNATIONAL
EXASCALE
SOFTWARE PROJECT

ROADMAP

Jack Dongarra
Pete Beckman
Terry Moore
Jean-Claude Andre
Jean-Yves Berthou
Taisuke Boku
Franck Cappello
Barbara Chapman
Xuebin Chi
Alok Choudhary
Sudip Dosanjh
Al Geist
Bill Gropp
Robert Harrison
Mark Hereld
Michael Heroux
Adolfy Hoisie
Koh Hotta
Yutaka Ishikawa
Fred Johnson
Sanjay Kale
Richard Kenway
Bill Kramer
Jesus Labarta
Bob Lucas
Barney Maccabe
Satoshi Matsuoka
Paul Messina
Bernd Mohr
Matthias Mueller
Wolfgang Nagel
Hiroshi Nakashima
Michael E. Papka
Dan Reed
Mitsuhisa Sato
Ed Seidel
John Shalf
David Skinner
Thomas Sterling
Rick Stevens
William Tang
John Taylor
Rajeev Thakur
Anne Trefethen
Marc Snir
Aad van der Steen
Fred Streitz
Bob Sugar
Shinji Sumimoto
Jeffrey Vetter
Robert Wisniewski
Kathy Yelick

SPONSORS

Office of Science
U.S. Department of Energy

NSF

ANR cea CRAY EDF FUJITSU INRIA GENCI NVIDIA 東京大学 THE UNIVERSITY OF TOKYO 筑波大学

If you are wondering what's beyond ExaFlops

Mega, Giga, Tera, Peta, Exa, Zetta …

10^{3}	kilo
10^{6}	mega
10^{9}	giga
10^{12}	tera
10^{15}	peta
10^{18}	exa
10^{21}	zetta

10^{24}	yotta
10^{27}	xona
10^{30}	weka
10^{33}	vunda
10^{36}	uda
10^{39}	treda
10^{42}	sorta
10^{45}	rinta
10^{48}	quexa
10^{51}	pepta
10^{54}	ocha
10^{57}	nena
10^{60}	minga
10^{63}	luma