Users' Meeting 2010

Past, Present, and Future of High Performance Computing

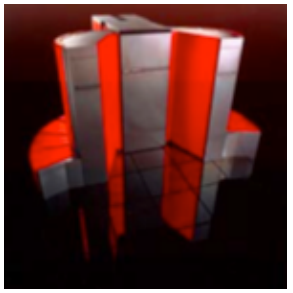
Jack Dongarra

University of Tennessee
Oak Ridge National Laboratory
University of Manchester

Architecture-aware Algorithms and Software for

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.

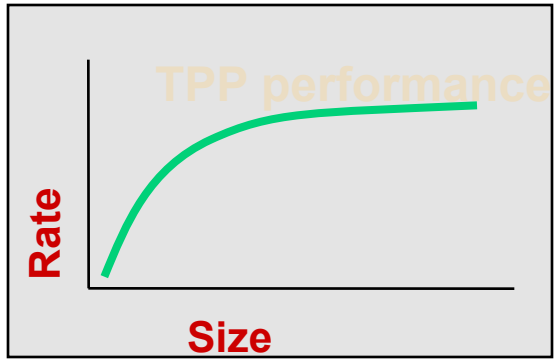
- 1 PFlop/s; 2008; Cray XT5; 1.5×10^5 Processors
  - Superconductive materials

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

H. Meuer, H. Simon, E. Strohmaier, & JD

- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

$Ax=b$, *dense problem*



- Updated twice a year
  SC‘xy in the States in November
  Meeting in Germany in June

- All data available from **www.top500.org**

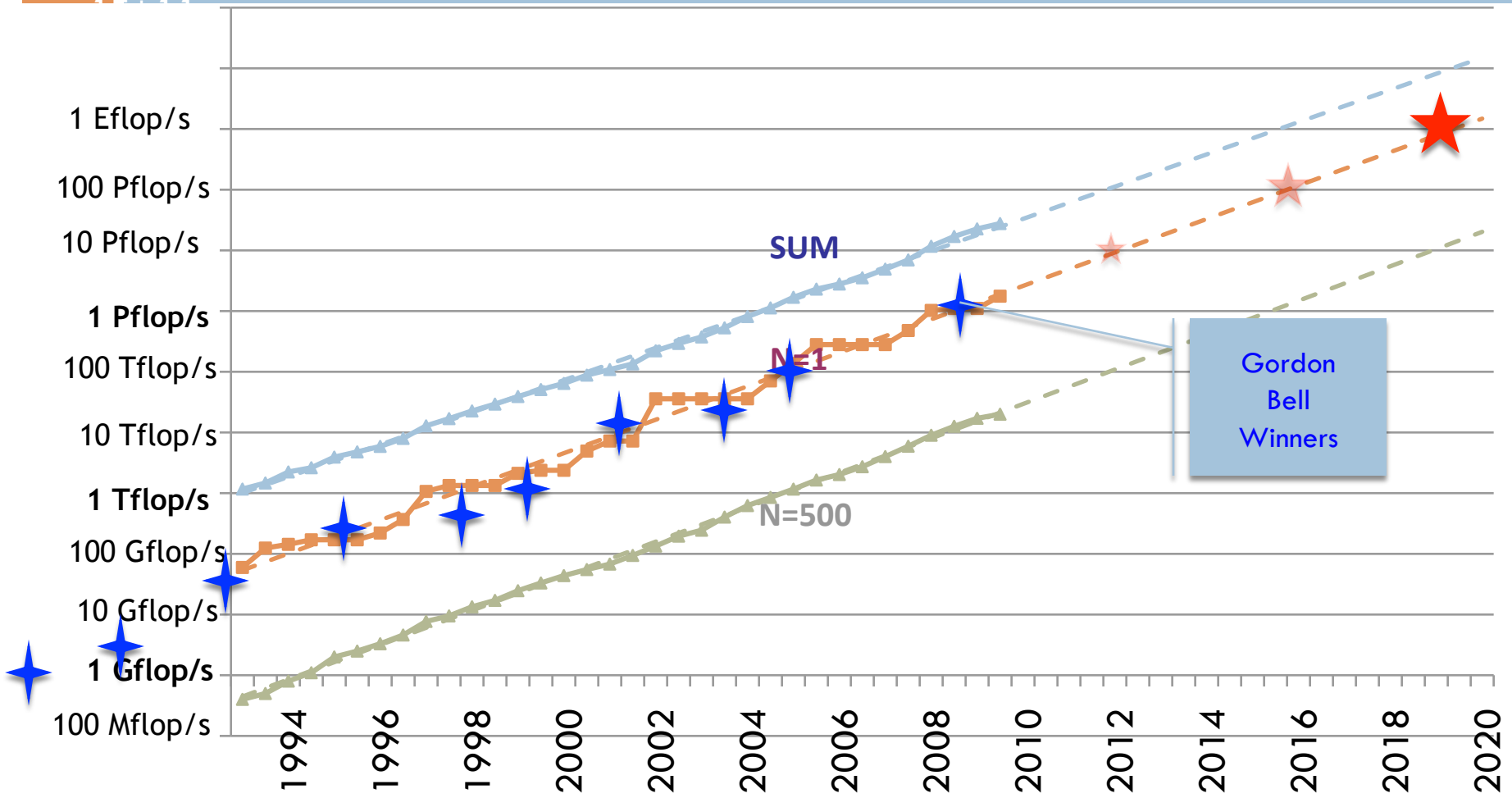
36rd List: The TOP10

Rank	Site	Computer	Country	Cores	Rmax [Tflops]	% of Peak
1	Nat. SuperComputer Center in Tianjin	NUDT YH Cluster, X5670 2.93Ghz 6C, NVIDIA GPU	China	186,368	2.57	55
2	DOE / OS Oak Ridge Nat Lab	Jaguar / Cray Cray XT5 sixCore 2.6 GHz	USA	224,162	1.76	75
3	Nat. Supercomputer Center in Shenzhen	Nebulea / Dawning / TC3600 Blade, Intel X5650, Nvidia C2050 GPU	China	120,640	1.27	43
4	GSIC Center, Tokyo Institute of Technology	Tusbame 2.0 HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU	Japan	73,278	1.19	52
5	DOE/SC/LBNL/NERSC	Hopper, Cray XE6 12-core 2.1 GHz	USA	153,408	1.054	82
6	Commissariat a l'Energie Atomique (CEA)	Tera-100 Bull bullx super-node S6010/S6030	France	138,368	1.050	84
7	DOE / NNSA Los Alamos Nat Lab	Roadrunner / IBM BladeCenter QS22/LS21	USA	122,400	1.04	76
8	NSF / NICS / U of Tennessee	Jaguar / Cray Cray XT5 sixCore 2.6 GHz	USA	98,928	.831	81
9	Forschungszentrum Juelich (FZJ)	Jugene / IBM Blue Gene/P Solution	Germany	294,912	.825	82
10	DOE/ NNSA / Los Alamos Nat Lab	Cray XE6 8-core 2.4 GHz	USA	107,152	.817	79

36rd List: The TOP10

Rank	Site	Computer	Country	Cores	Rmax [Tflops]	% of Peak	Power [MW]	Flops/ Watt
1	Nat. SuperComputer Center in Tianjin	NUDT YH Cluster, X5670 2.93Ghz 6C, NVIDIA GPU	China	186,368	2.57	55	4.04	636
2	DOE / OS Oak Ridge Nat Lab	Jaguar / Cray Cray XT5 sixCore 2.6 GHz	USA	224,162	1.76	75	7.0	251
3	Nat. Supercomputer Center in Shenzhen	Nebulea / Dawning / TC3600 Blade, Intel X5650, Nvidia C2050 GPU	China	120,640	1.27	43	2.58	493
4	GSIC Center, Tokyo Institute of Technology	Tusbame 2.0 HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU	Japan	73,278	1.19	52	1.40	850
5	DOE/SC/LBNL/NERSC	Hopper, Cray XE6 12-core 2.1 GHz	USA	153,408	1.054	82	2.91	362
6	Commissariat a l'Energie Atomique (CEA)	Tera-100 Bull bullx super-node S6010/S6030	France	138,368	1.050	84	4.59	229
7	DOE / NNSA Los Alamos Nat Lab	Roadrunner / IBM BladeCenter QS22/LS21	USA	122,400	1.04	76	2.35	446
8	NSF / NICS / U of Tennessee	Jaguar / Cray Cray XT5 sixCore 2.6 GHz	USA	98,928	.831	81	3.09	269
9	Forschungszentrum Juelich (FZJ)	Jugene / IBM Blue Gene/P Solution	Germany	294,912	.825	82	2.26	365
10	DOE/ NNSA / Los Alamos Nat Lab	Cray XE6 8-core 2.4 GHz	USA	107,152	.817	79	2.95	277

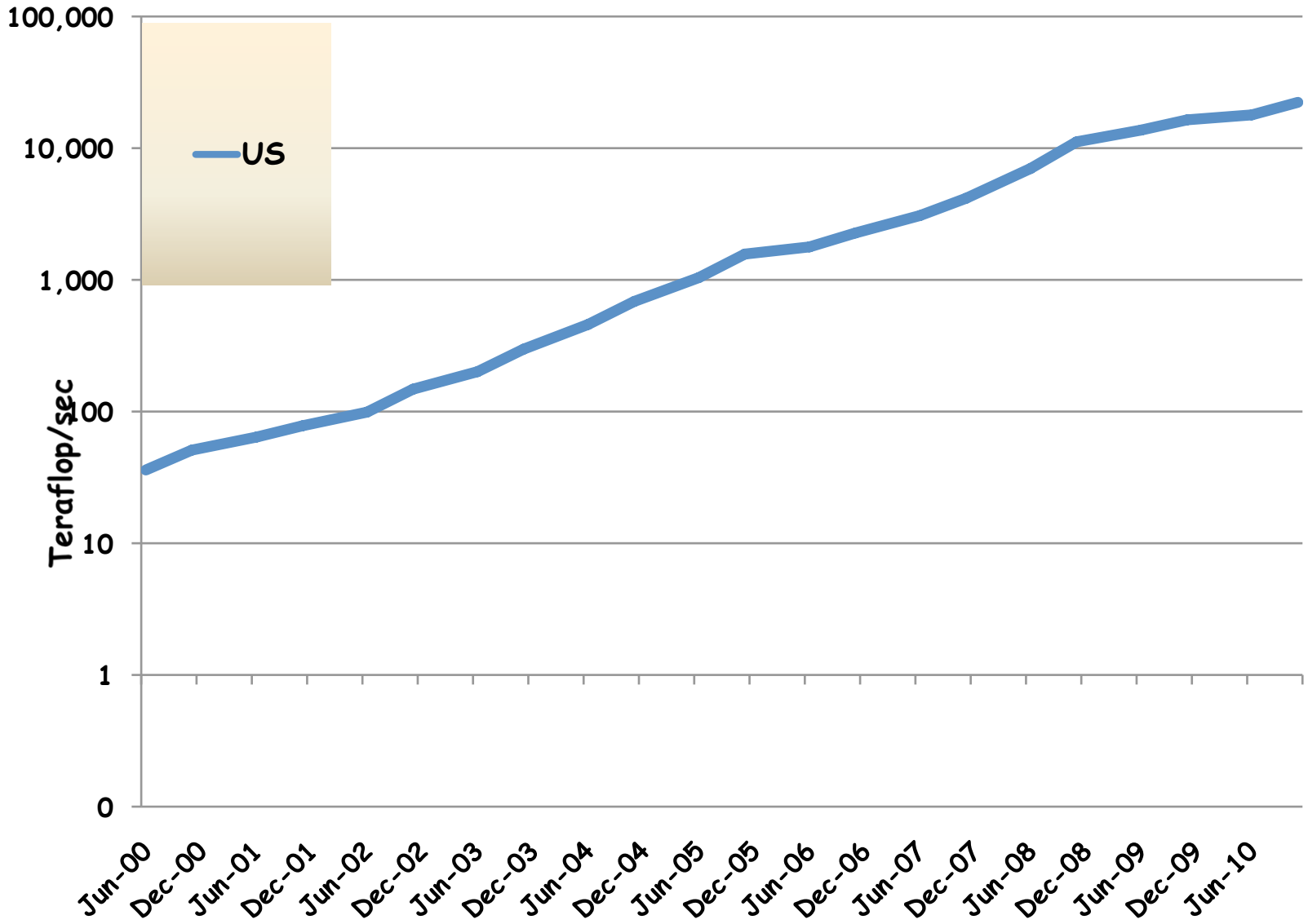
Performance Development in Top500



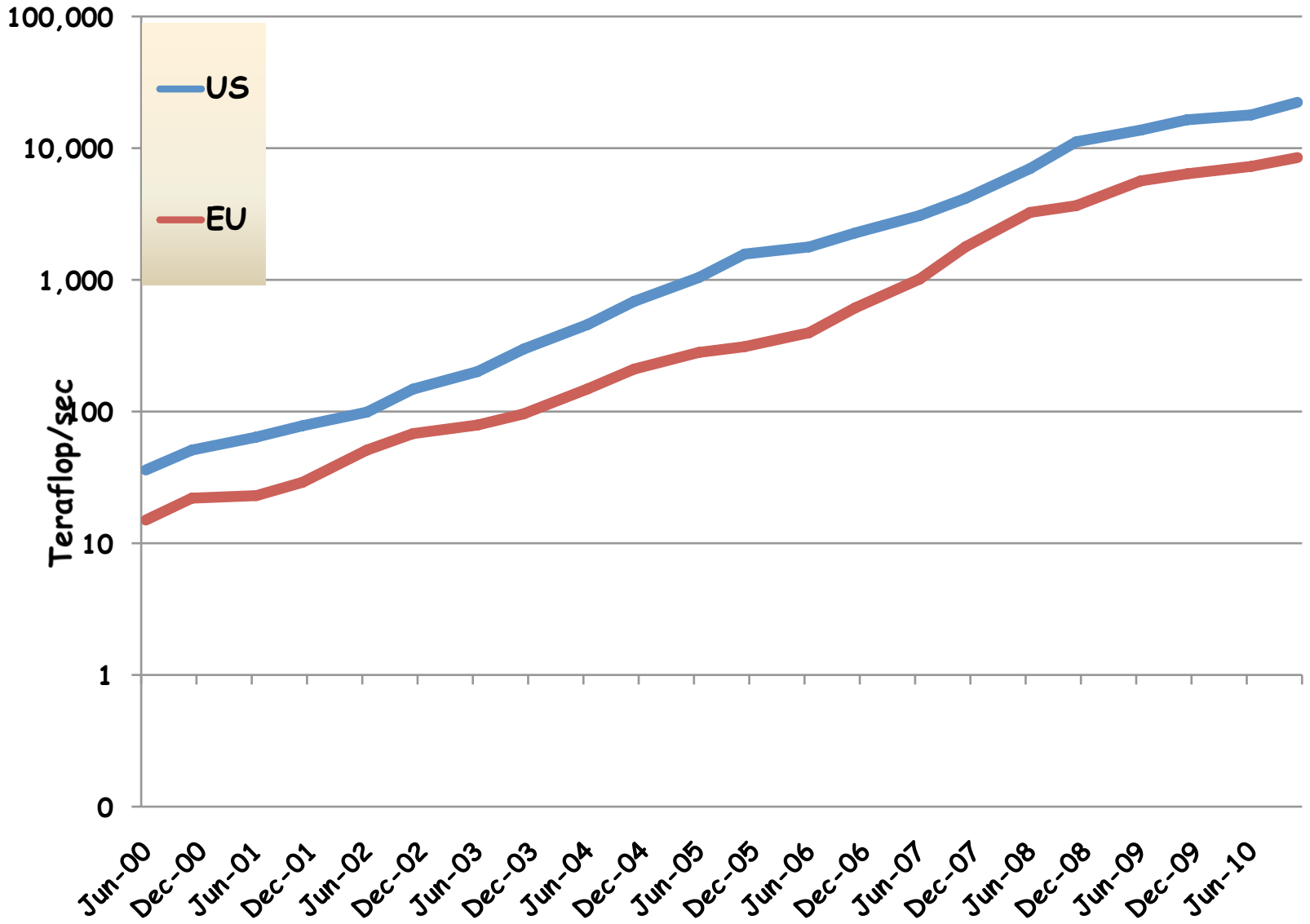
Pflop/s Club (11 systems; Peak)

Name	Peak Pflop/s	"Linpack" Pflop/s	Country	
Tianhe-1A	4.70	2.57	China	NUDT: Hybrid Intel/Nvidia/ Self
Nebula	2.98	1.27	China	Dawning: Hybrid Intel/ Nvidia/IB
Jaguar	2.33	1.76	US	Cray: AMD/Self
Tsubame 2.0	2.29	1.19	Japan	HP: Hybrid Intel/Nvidia/IB
RoadRunner	1.38	1.04	US	IBM: Hybrid AMD/Cell/IB
Hopper	1.29	1.054	US	Cray: AMD/Self
Tera-100	1.25	1.050	France	Bull: Intel/IB
Mole-8.5	1.14	.207	China	CAS: Hybrid Intel/Nvidia/IB
Kraken	1.02	.831	US	Cray: AMD/Self
Cielo	1.02	.817	US	Cray: AMD/Self
JuGene	1.00	.825	Germany	IBM: BG-P/Self

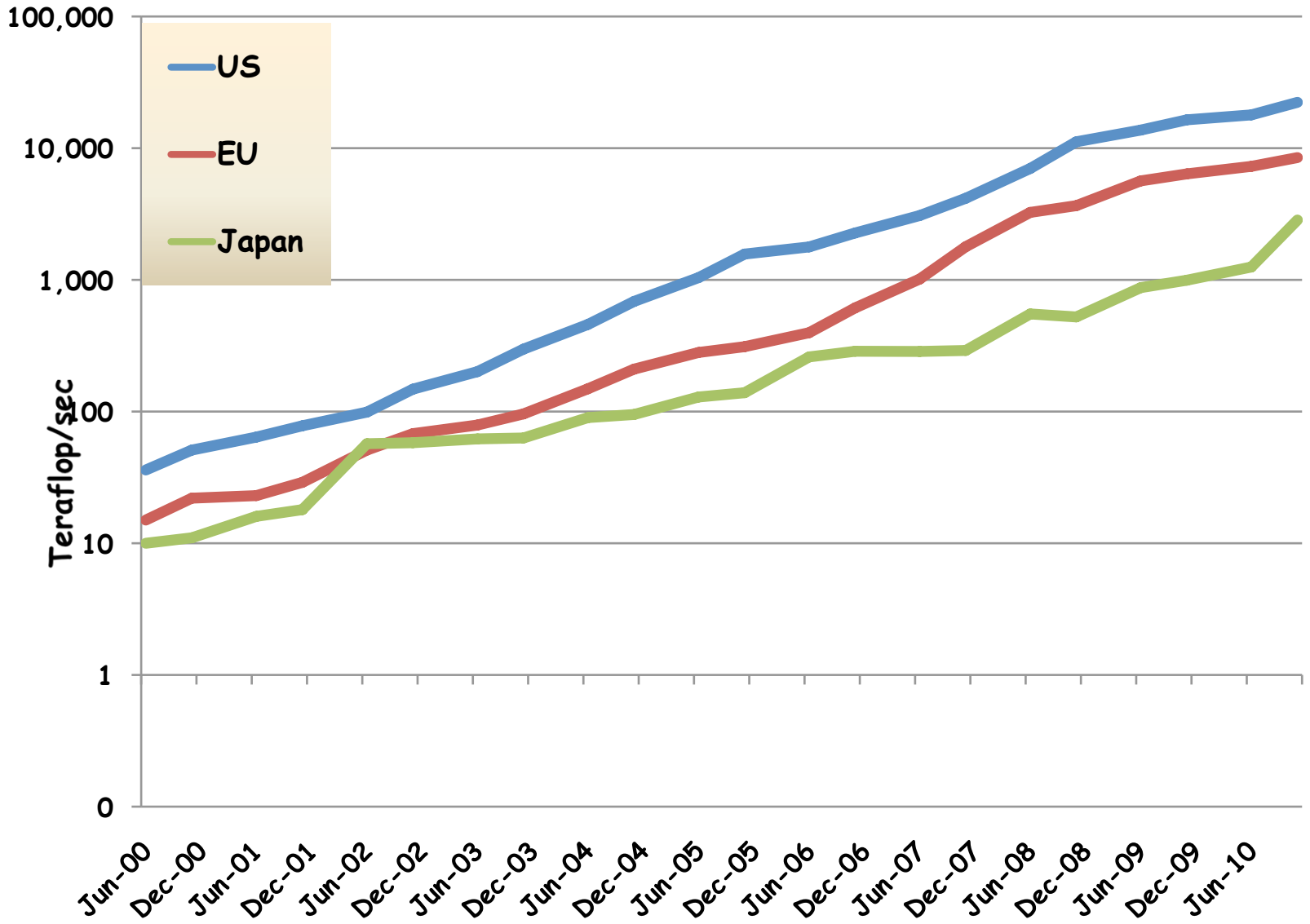
Performance of Countries



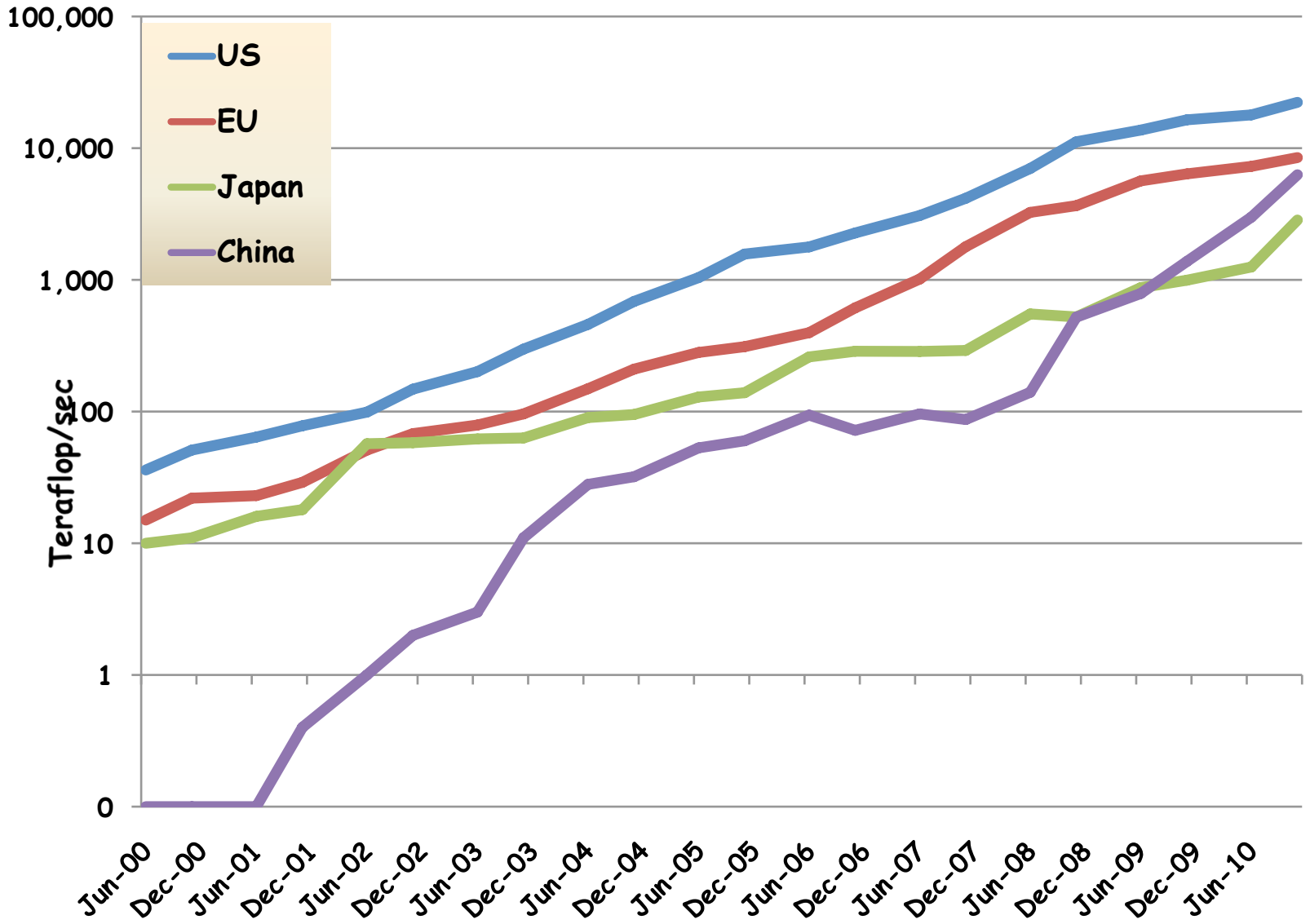
Performance of Countries



Performance of Countries

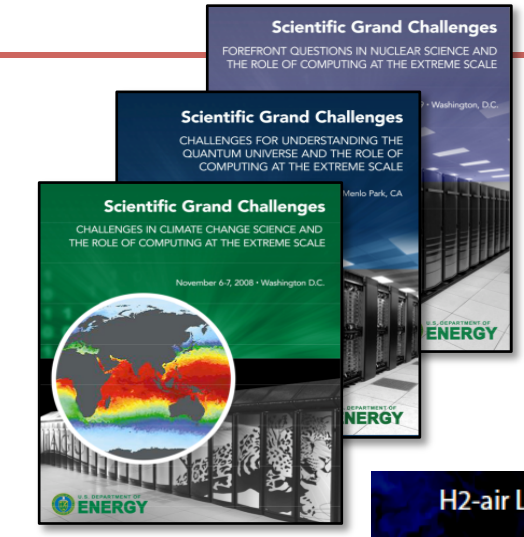


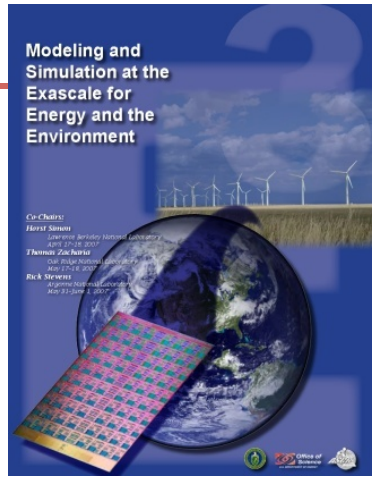
Performance of Countries

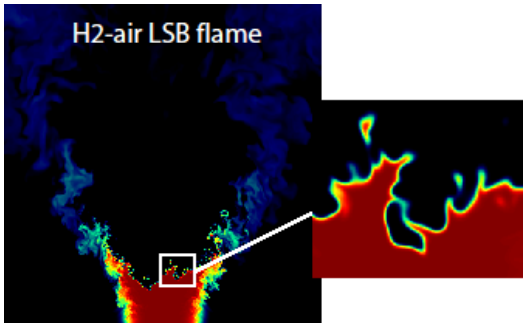


Exascale Applications and Technology

- Town Hall Meetings April-June 2007
- Scientific Grand Challenges Workshops November 2008 – October 2009
  - 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) (with NNSA)
- Cross-cutting workshops
  - Architecture and Technology (12/09)
  - Architecture, Applied Math and CS (2/10)
- Meetings with industry (8/09, 11/09)
- External Panels
  - ASCAC Exascale Charge (FACA)
  - Trivelpiece Panel







"The key finding of the Panel is that there are compelling needs for exascale computing capability to support the DOE's missions in energy, national security, fundamental sciences, and the environment. The DOE has the necessary assets to initiate a program that would accelerate the development of such capability to meet its own needs and by so doing benefit other national interests. Failure to initiate an exascale program could lead to a loss of U. S. competitiveness in several critical technologies."

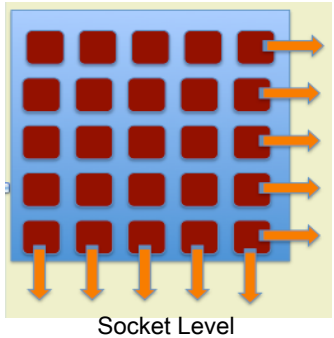
Trivelpiece Panel Report, January, 2010

Potential System Architectures

Systems	2010
System peak	2 Pflop/s
System memory	0.3 PB
Node performance	125 Gflop/s
Node memory BW	25 GB/s
Node concurrency	12
Interconnect BW	1.5 GB/s
System size (nodes)	18,700
Total concurrency	225,000
Storage	15 PB
IO	0.2 TB/s
MTTI	days
Power	7 MW

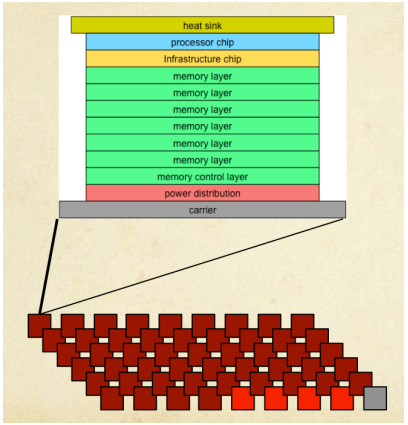
Exascale (10^{18} Flop/s) Systems: Two possible paths

- Light weight processors (think BG/P)
  - ~1 GHz processor (10^9)
  - ~1 Kilo cores/socket (10^3)
  - ~1 Mega sockets/system (10^6)



Socket Level
Cores scale-out for planar geometry

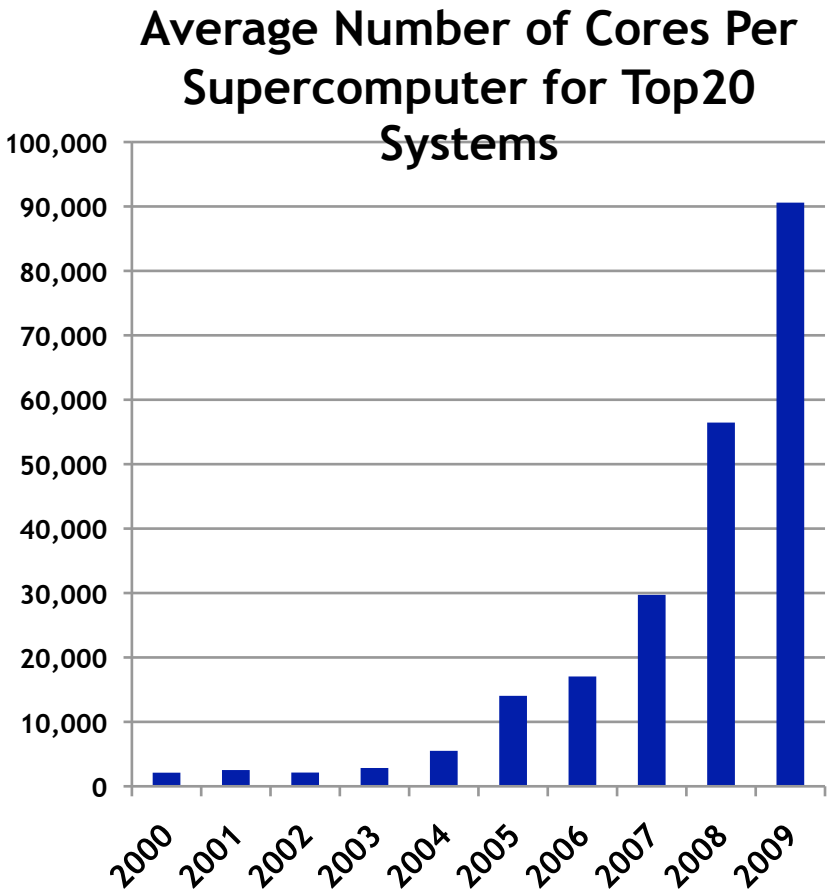
- Hybrid system (think GPU based)
  - ~1 GHz processor (10^9)
  - ~10 Kilo FPUs/socket (10^4)
  - ~100 Kilo sockets/system (10^5)



Node Level
3D packaging

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

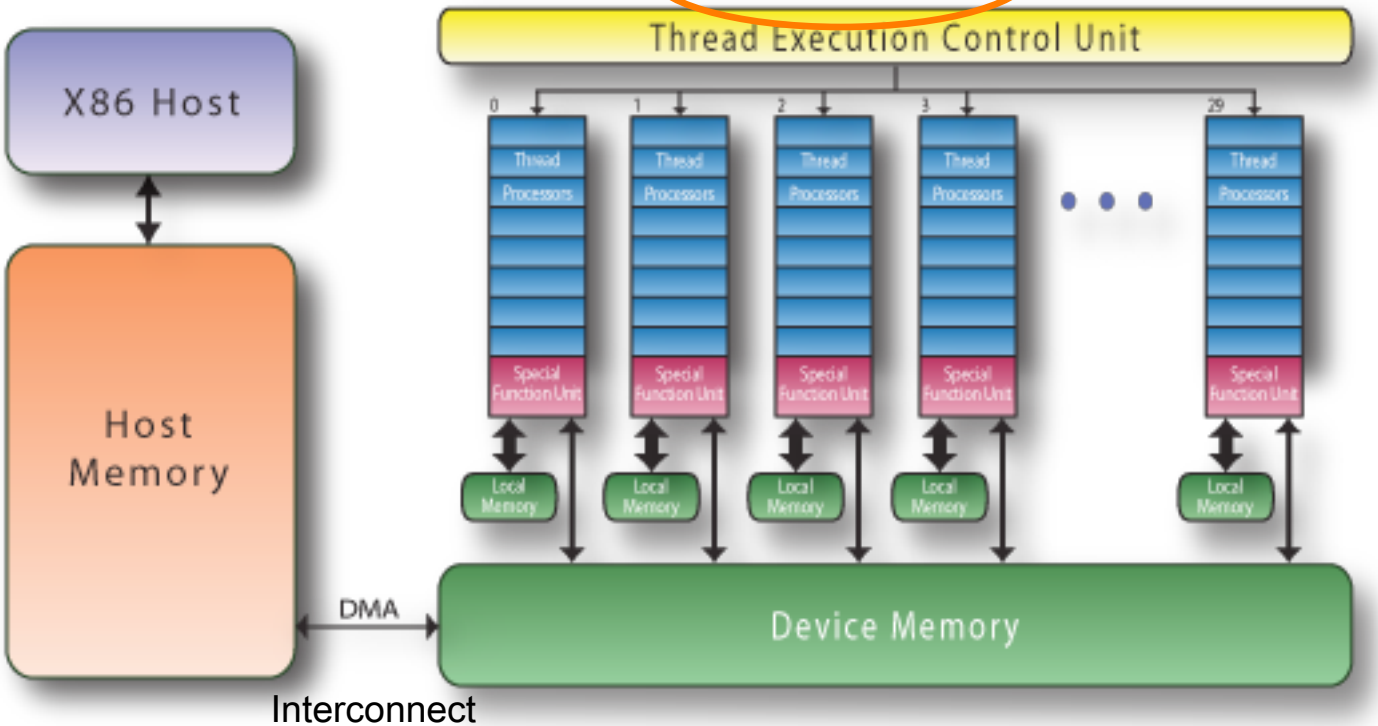
Commodity plus Accelerators

Commodity

Intel Xeon
8 cores
3 GHz
8*4 ops/cycle
96 Gflop/s (DP)

Accelerator (GPU)

Nvidia C2050 “Fermi”
448 “Cuda cores”
1.15 GHz
~~448 ops/cycle~~
515 Gflop/s (DP)



Interconnect
PCI Express
512 MB/s to 32GB/s
8 MW – 512 MW

07

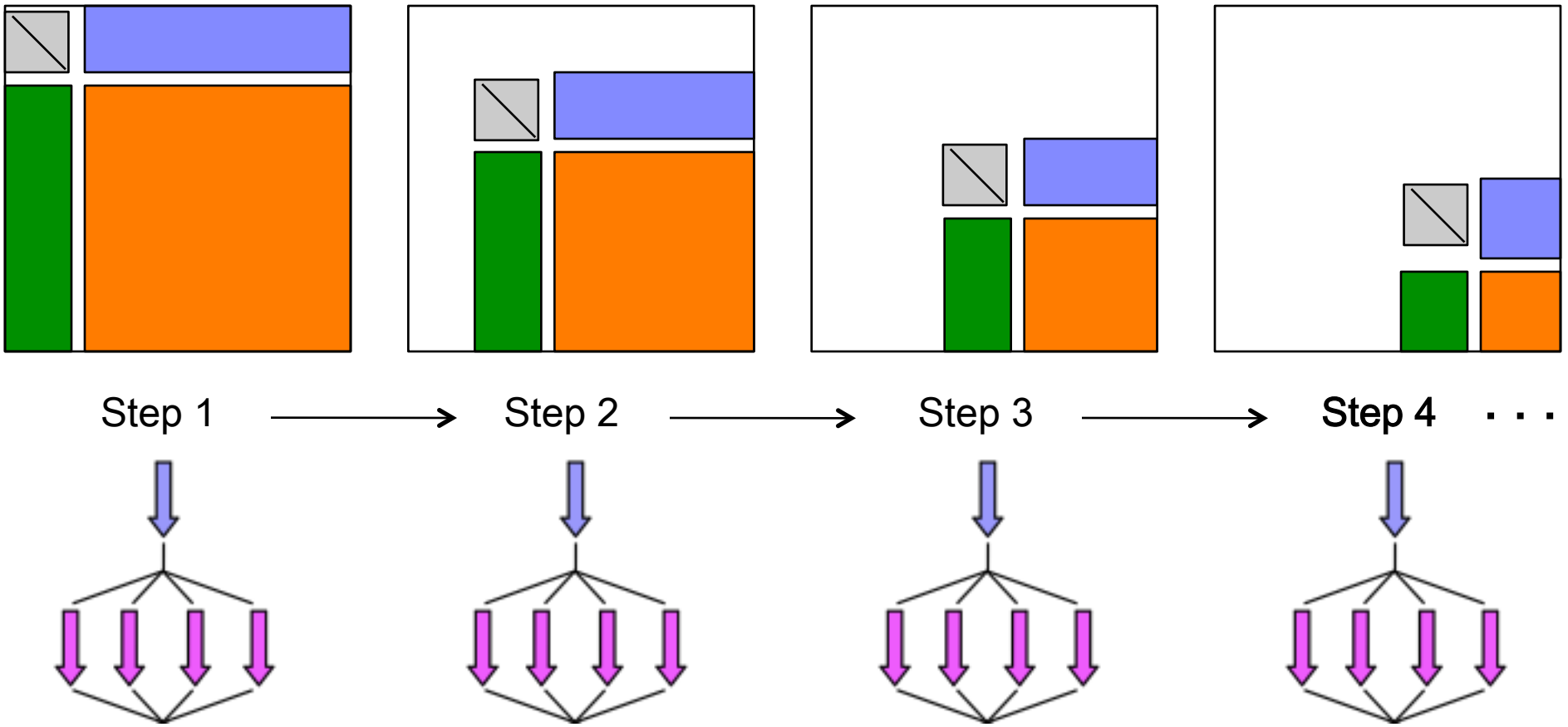
Major Changes to Software

- 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 Software Features to Consider When Computing at Scale

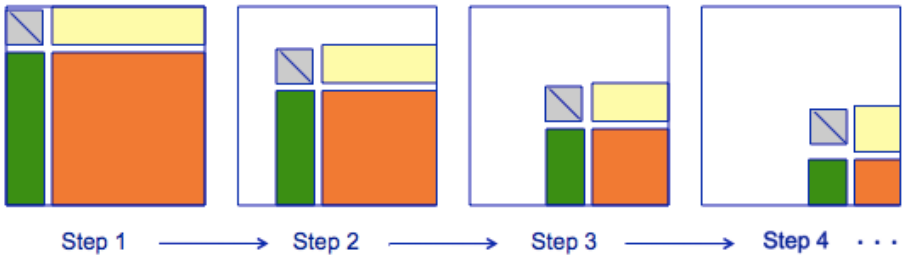
1. Effective Use of Many-Core and Hybrid architectures
   - Break fork-join parallelism
   - Dynamic Data Driven Execution
   - Block Data Layout
2. Exploiting Mixed Precision in the Algorithms
   - Single Precision is 2X faster than Double Precision
   - With GP-GPUs 10x
   - Power saving issues
3. Self Adapting / Auto Tuning of Software
   - Too hard to do by hand
4. Fault Tolerant Algorithms
   - With 1,000,000's of cores things will fail
5. 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, … A^kx)

LAPACK LU/LL^T/QR

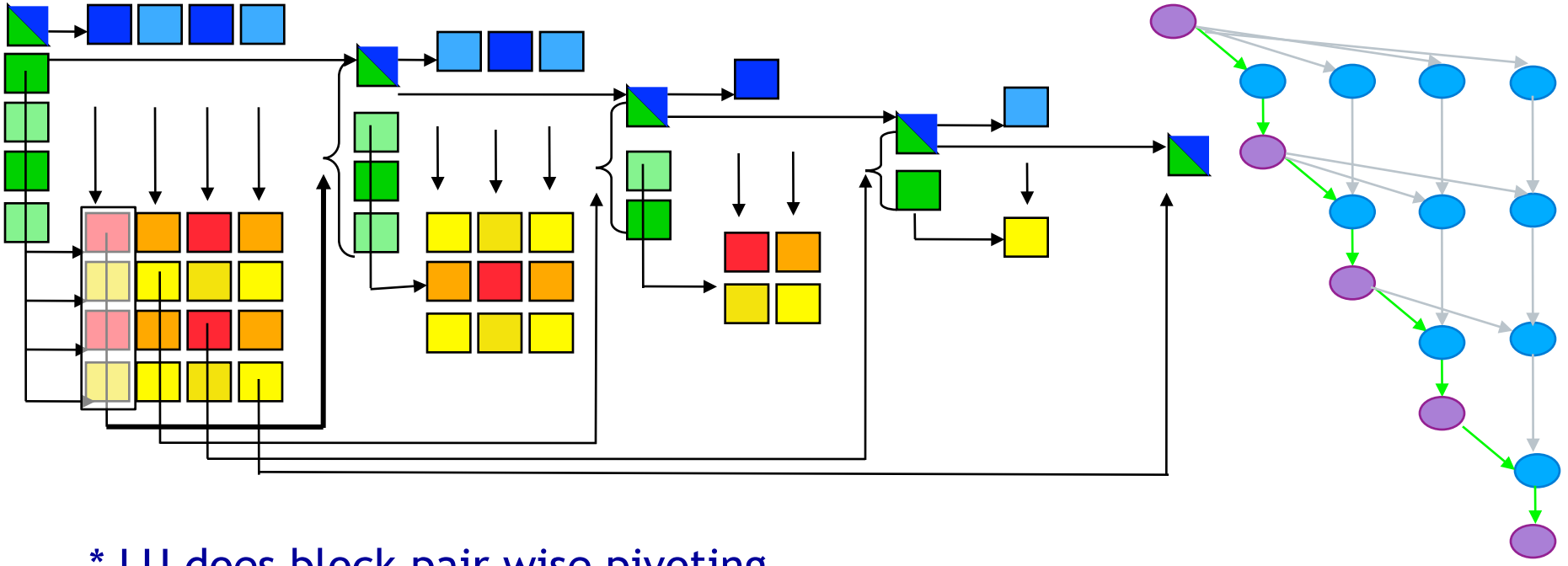


- **Fork-join, bulk synchronous processing**

Parallel Tasks in LU/LL^T/QR



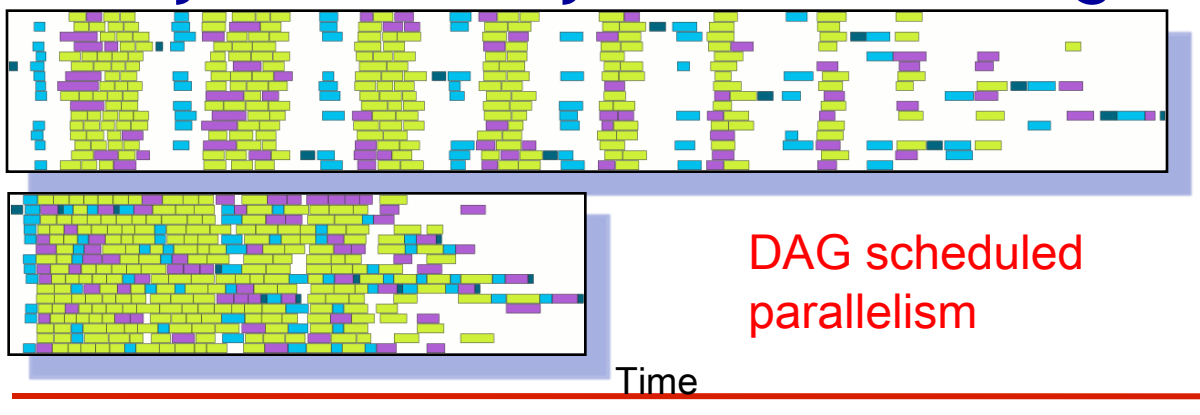
- Break into smaller tasks and remove dependencies

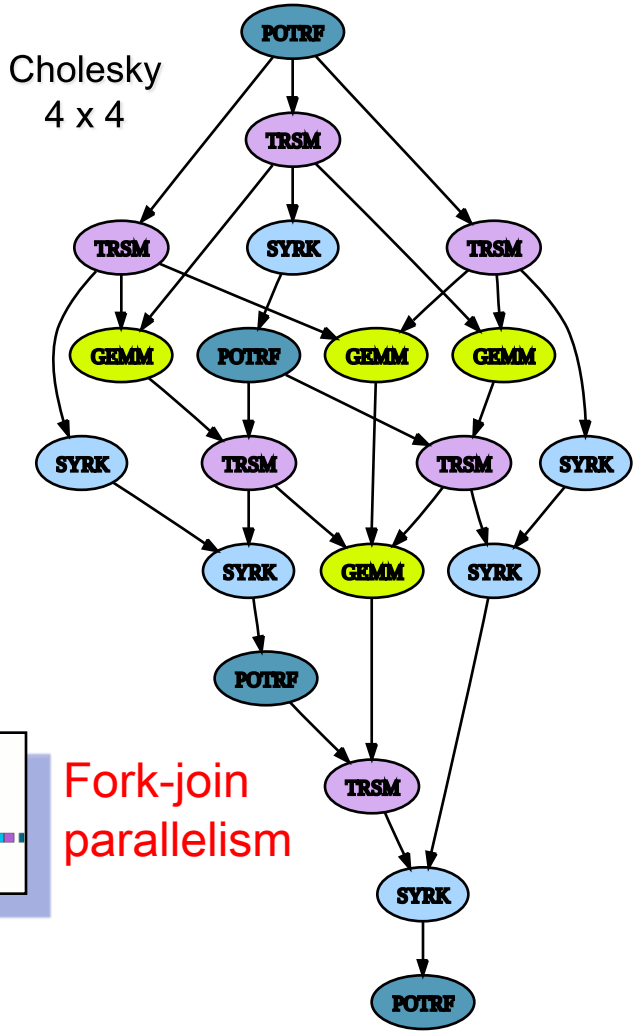


* LU does block pair wise pivoting

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**
  - Dynamic DAG scheduling
  - Explicit parallelism
  - Implicit communication
  - Fine granularity / block data layout
- **Arbitrary DAG with dynamic scheduling**



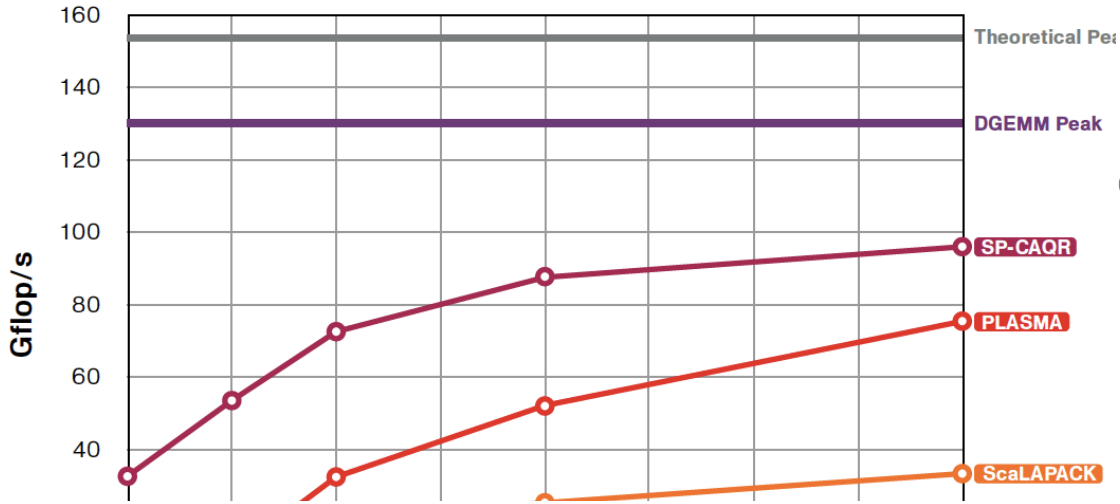


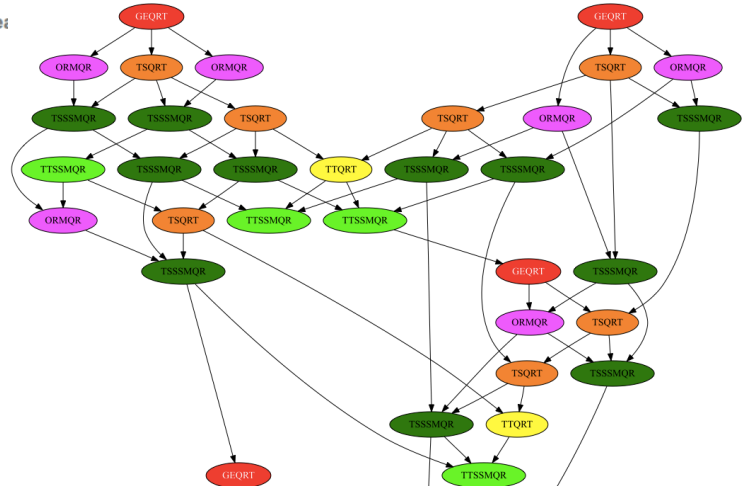
Communication Avoiding Algorithms

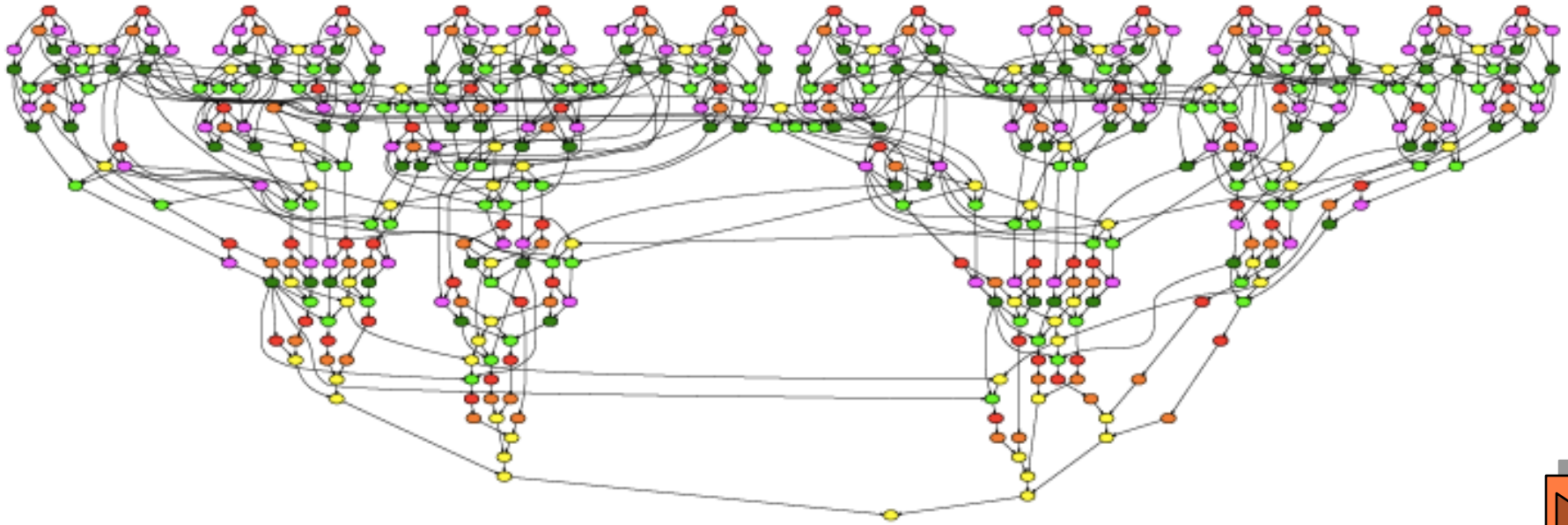
- Goal: Algorithms that communicate as little as possible
- Jim Demmel and company have been working on algorithms that obtain a provable minimum communication.
- Direct methods (BLAS, LU, QR, SVD, other decompositions)
  - Communication lower bounds for *all* these problems
  - Algorithms that attain them (*all* dense linear algebra, some sparse)
    - Mostly not in LAPACK or ScaLAPACK (yet)
- Iterative methods – Krylov subspace methods for Ax=b, Ax=λx
  - Communication lower bounds, and algorithms that attain them (depending on sparsity structure)
    - Not in any libraries (yet)
- For QR Factorization they can show:

	Lower bound
# flops	$\Theta(mn^2)$
# words	$\Theta(\frac{mn^2}{\sqrt{W}})$
# messages	$\Theta(\frac{mn^2}{W^{3/2}})$

Communication Reducing QR Factorization



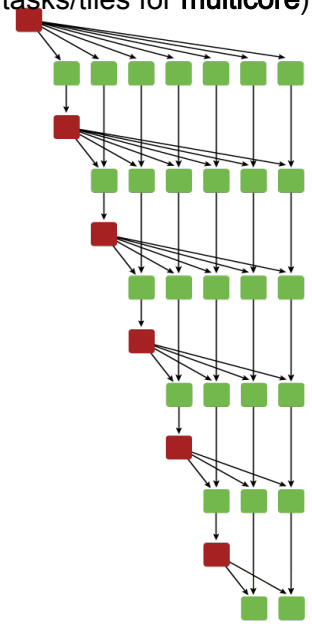




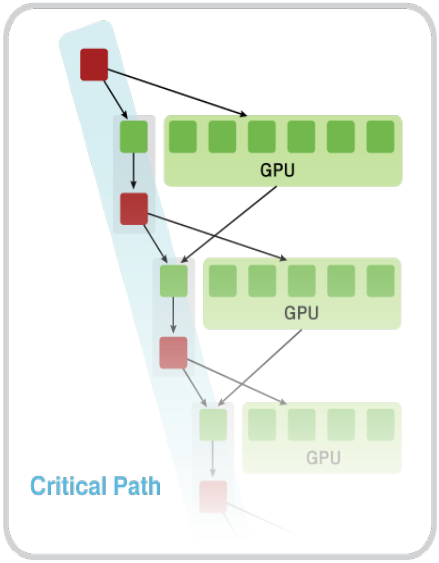
Hybrid Computing

- Match algorithmic requirements to architectural strengths of the hybrid components
  Multicore : small tasks/tiles
  Accelerator: large data parallel tasks

Algorithms as DAGs
(small tasks/tiles for **multicore**)

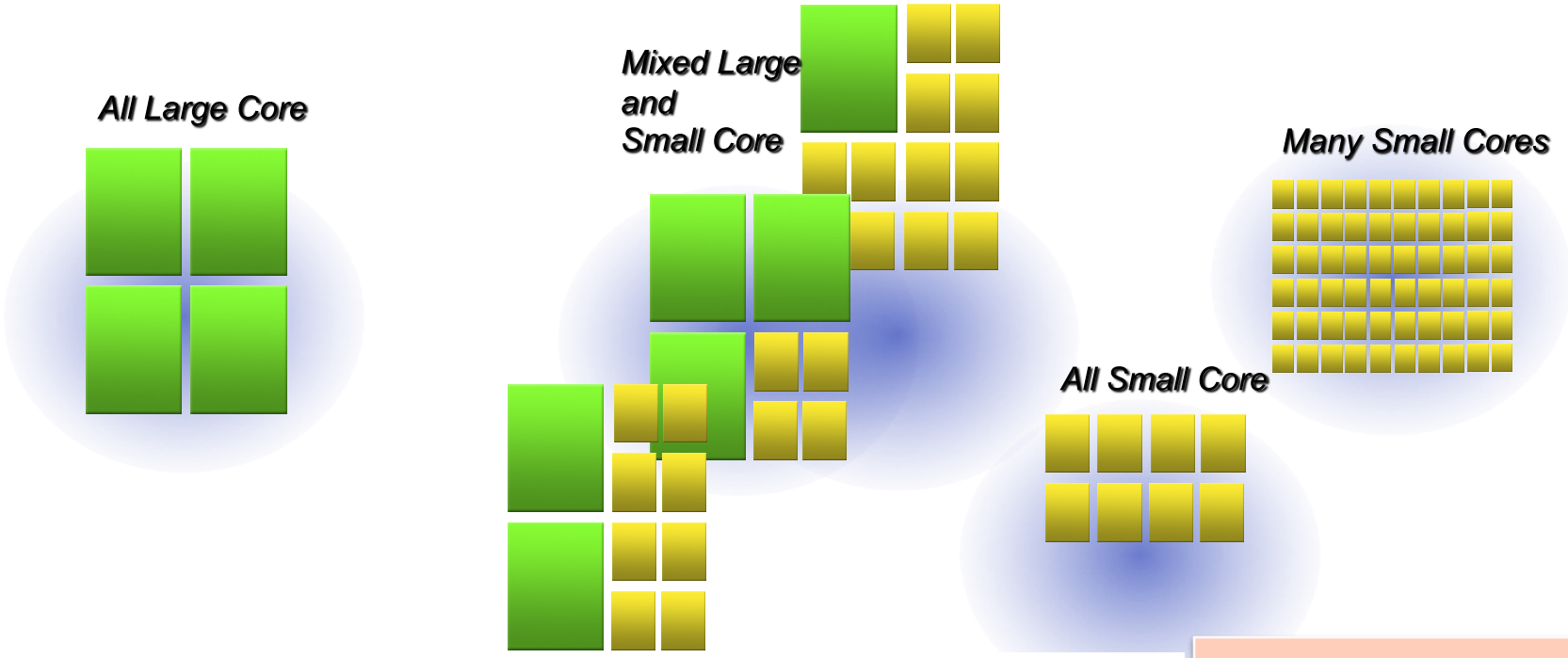


Current hybrid CPU+GPU algorithms
(small tasks for multicores and large tasks for GPUs)

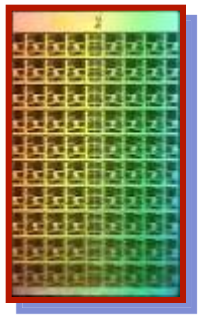


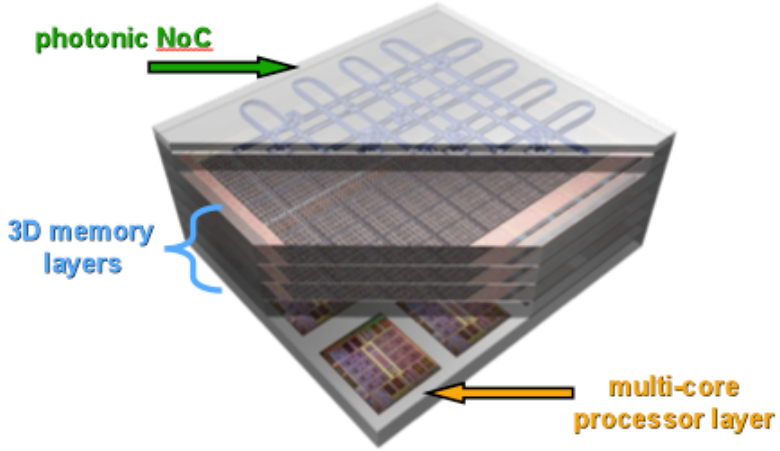
- e.g. split the computation into tasks; define critical path that "clears" the way for other large data parallel tasks; proper schedule the tasks execution
- Design algorithms with well defined "*search space*" to facilitate auto-tuning

What's Next?



Many Floating-Point Cores

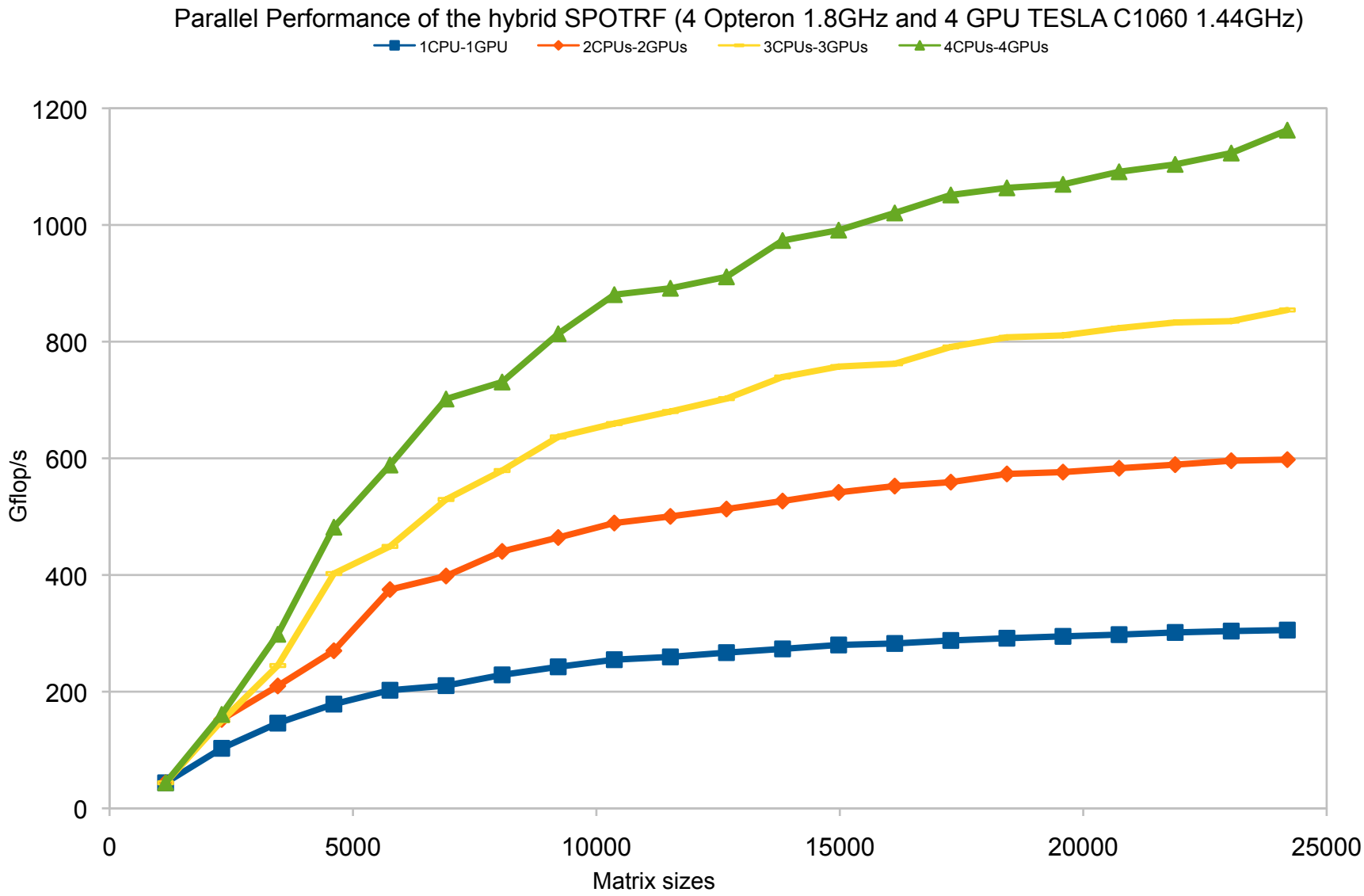




+ 3D Stacked Memory

Different Classes of Chips
- Home
- Games / Graphics
- Business
- Scientific

SP Cholesky on Multicore + Multi GPUs



Exploiting Mixed Precision Computations

- Single precision is faster than DP because:
  - **Higher parallelism within floating point units**
    - 4 ops/cycle (usually) instead of 2 ops /cycle
  - **Reduced data motion**
    - 32 bit data instead of 64 bit data
  - **Higher locality in cache**
    - More data items in cache

Idea Goes Something Like This…

- Exploit 32 bit floating point as much as possible.
  - Especially for the bulk of the computation
- Correct or update the solution with selective use of 64 bit floating point to provide a refined results
- Intuitively:
  - Compute a 32 bit result,
  - Calculate a correction to 32 bit result using selected higher precision and,
  - Perform the update of the 32 bit results with the correction using high precision.

Mixed-Precision Iterative Refinement

- Iterative refinement for dense systems, $Ax = b$, can work this way.

L U = lu(A)	$O(n^3)$
x = L\(U\b)	$O(n^2)$
r = b – Ax	$O(n^2)$
WHILE || r || not small enough	
z = L\(U\r)	$O(n^2)$
x = x + z	$O(n^1)$
r = b – Ax	$O(n^2)$
END	

  - Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.

Mixed-Precision Iterative Refinement

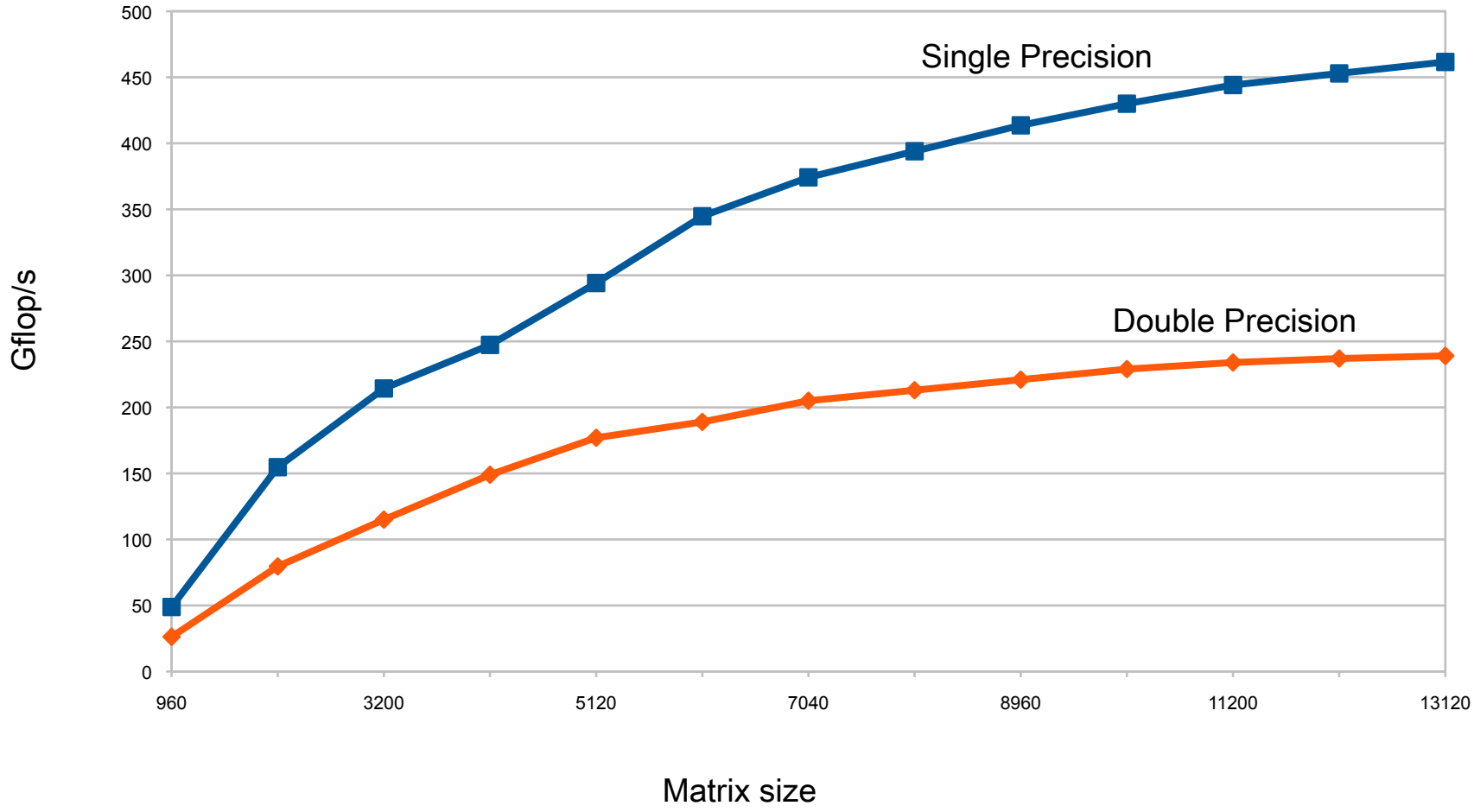
- Iterative refinement for dense systems, $Ax = b$, can work this way.

L U = lu(A)	SINGLE	$O(n^3)$
x = L\(U\b)	SINGLE	$O(n^2)$
r = b – Ax	DOUBLE	$O(n^2)$
WHILE || r || not small enough		
z = L\(U\r)	SINGLE	$O(n^2)$
x = x + z	DOUBLE	$O(n^1)$
r = b – Ax	DOUBLE	$O(n^2)$
END		

- Wilkinson, Moler, Stewart, & Higham provide error bound for SP fl pt results when using DP fl pt.
- It can be shown that using this approach we can compute the solution to 64-bit floating point precision.

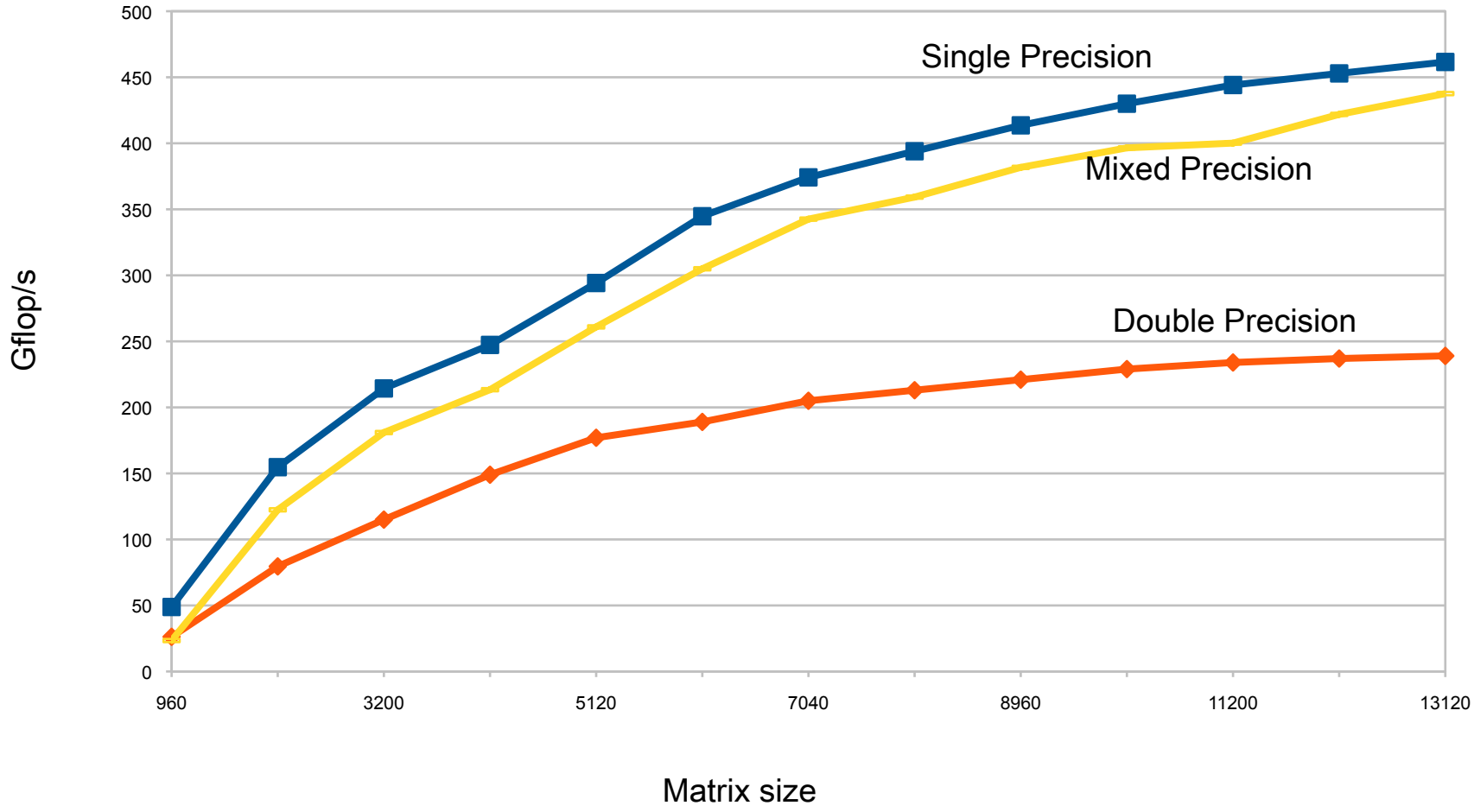
- Requires extra storage, total is 1.5 times normal;
- $O(n^3)$ work is done in lower precision
- $O(n^2)$ work is done in high precision
- Problems if the matrix is ill-conditioned in sp; $O(10^8)$

Ax = b



Tesla C2050, 448 CUDA cores (14 multiprocessors x 32) @ 1.15 GHz.,
3 GB memory, connected through PCIe to a quad-core Intel @2.5 GHz.

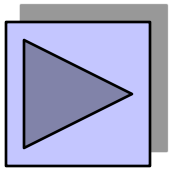
Ax = b



Tesla C2050, 448 CUDA cores (14 multiprocessors x 32) @ 1.15 GHz.,
3 GB memory, connected through PCIe to a quad-core Intel @2.5 GHz.

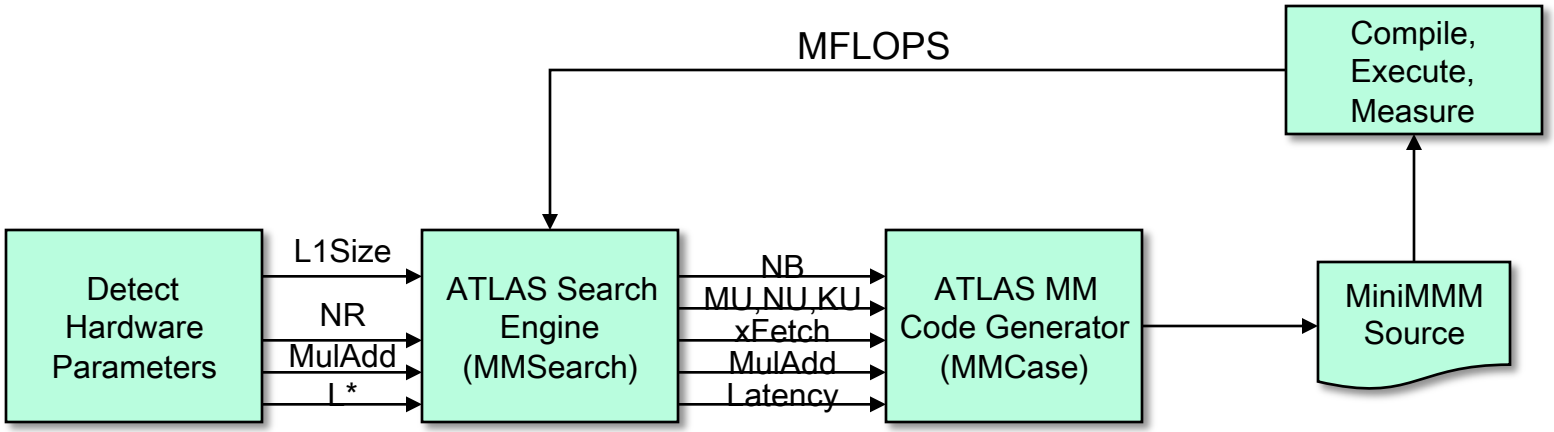
Automatic Performance Tuning

- Writing high performance software is hard
- Ideal: get high fraction of peak performance from one algorithm
- Reality: Best algorithm (and its implementation) can depend strongly on the problem, computer architecture, compiler,…
  - Best choice can depend on knowing a lot of applied mathematics and computer science
  - Changes with each new hardware, compiler release
- Automatic performance tuning
  - Use machine time in place of human time for tuning
  - Search over possible implementations
  - Use performance models to restrict search space
  - Past successes: ATLAS, FFTW, Spiral, Open-MPI



How to Deal with Complexity?

- Many parameters in the code needs to be optimized.
- Software adaptivity is the key for applications to effectively use available resources whose complexity is exponentially increasing



Auto-Tuning

- Best algorithm implementation can depend strongly on the problem, computer architecture, compiler,…
- There are 2 main approaches
  - Model-driven optimization
    [Analytical models for various parameters;
    Heavily used in the compilers community;
    May not give optimal results]
  - Empirical optimization
    [Generate large number of code versions and runs them on a given platform to determine the best performing one;
    Effectiveness depends on the chosen parameters to optimize and the search heuristics used]
- Natural approach is to combine them in a hybrid approach

[1st model-driven to limit the search space for a 2nd empirical part]
[Another aspect is adaptivity – to treat cases where tuning can not be restricted to optimizations at design, installation, or compile time]

PLASMA 2.3 for Multicore Systems

Functionality	Coverage
Linear systems and least squares	LU, Cholesky, QR & LQ
Mixed-precision linear systems	LU, Cholesky, QR
Tall and skinny factorization	QR
Generation of the Q matrix	QR, LQ, tall and skinny QR
Explicit matrix inversion	Cholesky
Level 3 BLAS	GEMM, HEMM, HER2K, HERK, SYMM, SYR2K, SYRK, TRMM, TRSM (complete set)
In-place layout translations	CM, RM, CCRB, CRRB, RCRB, RRRB (all combinations)

Features
Covering four precisions: Z, C, D, S (and mixed-precision: ZC, DS)
Static scheduling and dynamic scheduling with QUARK
Support for Linux, MS Windows, Mac OS and AIX

MAGMA 1.0 for Hybrid Systems

Functionality	Coverage
Linear systems and least squares	LU, Cholesky, QR & LQ
Mixed-precision linear systems	LU, Cholesky, QR
Eigenvalue and singular value problems	Reductions to upper Hessenberg, bidiagonal, and tridiagonal forms
Generation of the Q matrix	QR, LQ, Hessenberg, bidiagonalization, and tridiagonalization
MAGMA BLAS	Subset of BLAS, critical for MAGMA performance for Tesla and Fermi

Features
Covering four precisions: Z, C, D, S (and mixed-precision: ZC, DS)
Support for multicore and one NVIDIA GPU
CPU and GPU interfaces
Support for Linux and Mac OS

Summary

- Major Challenges are ahead for extreme computing
  - Parallelism
  - Hybrid
  - Fault Tolerance
  - Power
  - … and many others not discussed here
- We will need completely new approaches and technologies to reach the Exascale level
- This opens up many new opportunities for applied mathematicians and computer scientists

A Call to Action

- Hardware has changed dramatically while software ecosystem has remained stagnant
- Need to exploit new hardware trends (e.g., manycore, heterogeneity) that cannot be handled by existing software stack, memory per socket trends
- Emerging software technologies exist, but have not been fully integrated with system software, e.g., UPC, Cilk, CUDA, HPCS
- Community codes unprepared for sea change in architectures
- No global evaluation of key missing components

Exascale is a Global Challenge

- Formed in 2008
- Goal to engage international computer science community to address common software challenges for Exascale
- Focus on open source systems software that would enable multiple platforms
- Shared risk and investment
- Leverage international talent base

International Exascale Software Program

Improve the world's simulation and modeling capability by improving the coordination and development of the HPC software environment

Workshops:

Build an international plan for coordinating research for the next generation open source software for scientific high-performance computing

www.exascale.org

Roadmap Components

www.exascale.org

Where We Are Today:

- Ken Kennedy – Petascale Software Project (2006)
- SC08 (Austin TX) meeting to generate interest
- Funding from DOE's Office of Science & NSF Office of Cyberinfratructure and sponsorship by Europeans and Asians — Nov 2008
- US meeting (Santa Fe, NM) April 6-8, 2009 — Apr 2009
  - 65 people
- European meeting (Paris, France) June 28-29, 2009 — Jun 2009
  - Outline Report
- Asian meeting (Tsukuba Japan) October 18-20, 2009 — Oct 2009
  - Draft roadmap and refine report
- SC09 (Portland OR) BOF to inform others — Nov 2009
  - Public Comment; Draft Report presented
- European meeting (Oxford, UK) April 13-14, 2010 — Apr 2010
  - Refine and prioritize roadmap; look at management models
- Maui Meeting October 18-19, 2010 — Oct 2010
- SC10 (New Orleans) BOF to inform others (Wed 5:30, Room 389) — Nov 2010
- Kyoto Meeting – April 6-7, 2011 — Apr 2011

Conclusions

- For the last decade or more, the research investment strategy has been overwhelmingly biased in favor of hardware.
- This strategy needs to be rebalanced - barriers to progress are increasingly on the software side.
- Moreover, the return on investment is more favorable to software.
  - Hardware has a half-life measured in years, while software has a half-life measured in decades.
- High Performance Ecosystem out of balance
  - Hardware, OS, Compilers, Software, Algorithms, Applications
    - No Moore's Law for software, algorithms and applications

To be published in the January 2011 issue of The International Journal of High Performance Computing Applications

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

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

- *Alan Turing (1912—1954)*

- www.exascale.org