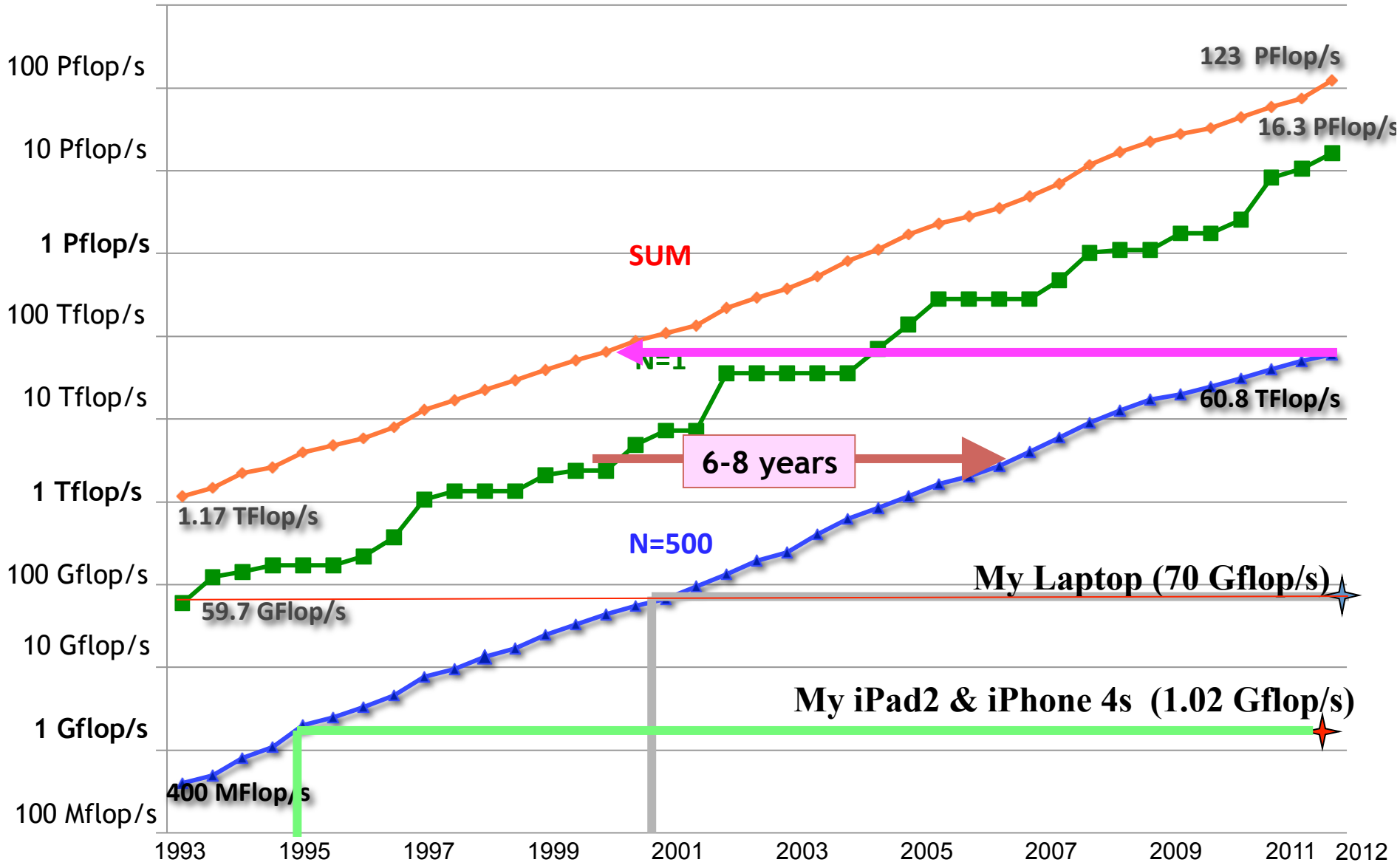
The SPEEDUP Society
The SWISS forum for Grid and High Performance Computing

ON THE FUTURE OF HIGH PERFORMANCE COMPUTING: HOW TO THINK FOR PETA AND EXASCALE COMPUTING

JACK DONGARRA
UNIVERSITY OF TENNESSEE
OAK RIDGE NATIONAL LAB

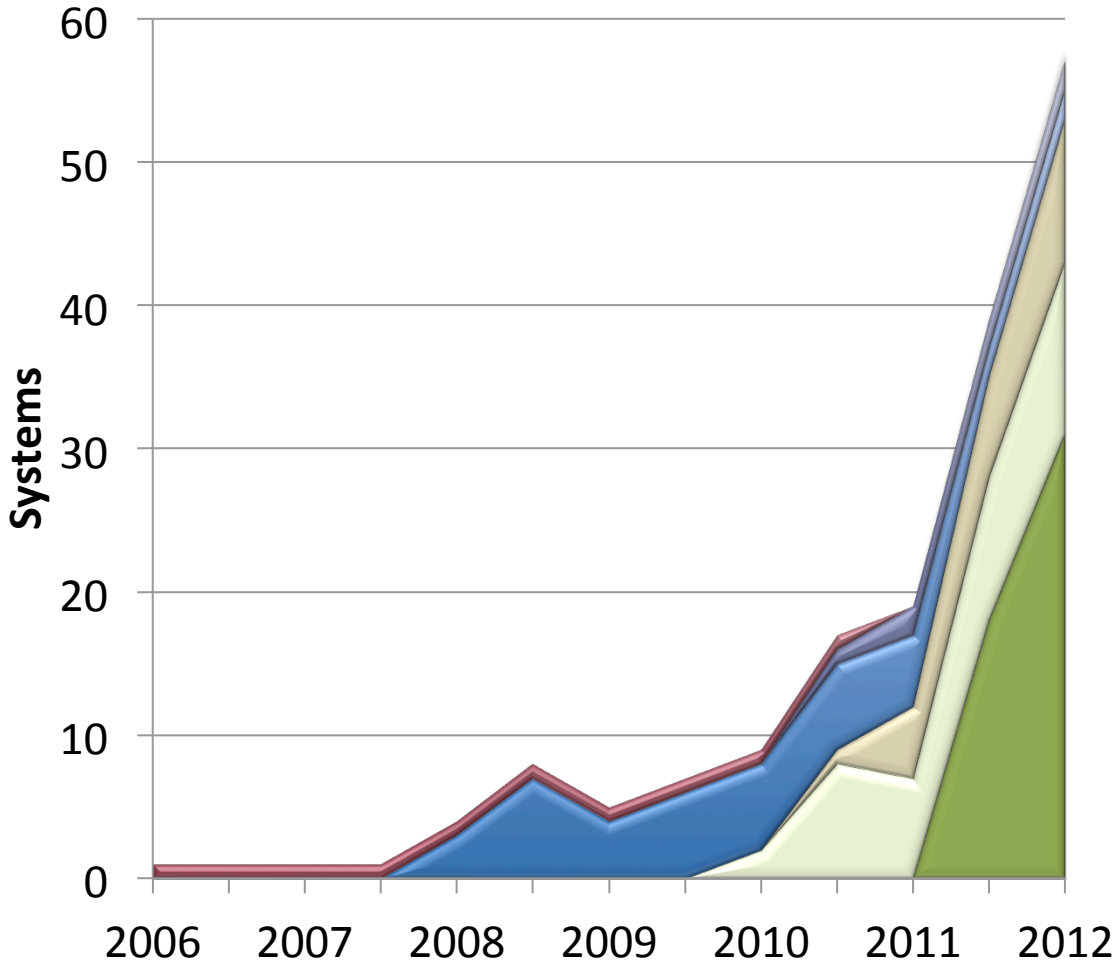
Over Last 20 Years - Performance Development



June 2012: The TOP10

Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	MFlops /Watt
1	DOE / NNSA L Livermore Nat Lab	Sequoia, BlueGene/Q (16c) + custom	USA	1,572,864	16.3	81	8.6	1895
2	RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx (8c) + custom	Japan	705,024	10.5	93	12.7	830
3	DOE / OS Argonne Nat Lab	Mira, BlueGene/Q (16c) + custom	USA	786,432	8.16	81	3.95	2069
4	Leibniz Rechenzentrum	SuperMUC, Intel (8c) + IB	Germany	147,456	2.90	90*	3.52	823
5	Nat. SuperComputer Center in Tianjin	Tianhe-1A, NUDT Intel (6c) + Nvidia GPU (14c) + custom	China	186,368	2.57	55	4.04	636
6	DOE / OS Oak Ridge Nat Lab	Jaguar, Cray AMD (16c) + custom	USA	298,592	1.94	74	5.14	377
7	CINECA	Fermi, BlueGene/Q (16c) + custom	Italy	163,840	1.73	82	.821	2099
8	Forschungszentrum Juelich (FZJ)	JuQUEEN, BlueGene/Q (16c) + custom	Germany	131,072	1.38	82	.657	2099
9	Commissariat a l'Energie Atomique (CEA)	Curie, Bull Intel (8c) + IB	France	77,184	1.36	82	2.25	604
10	Nat. Supercomputer Center in Shenzhen	Nebulea, Dawning Intel (6) + Nvidia GPU (14c) + IB	China	120,640	1.27	43	2.58	493
500	Energy Comp	IBM Cluster, Intel + IB	Italy	4096	.061	93*		

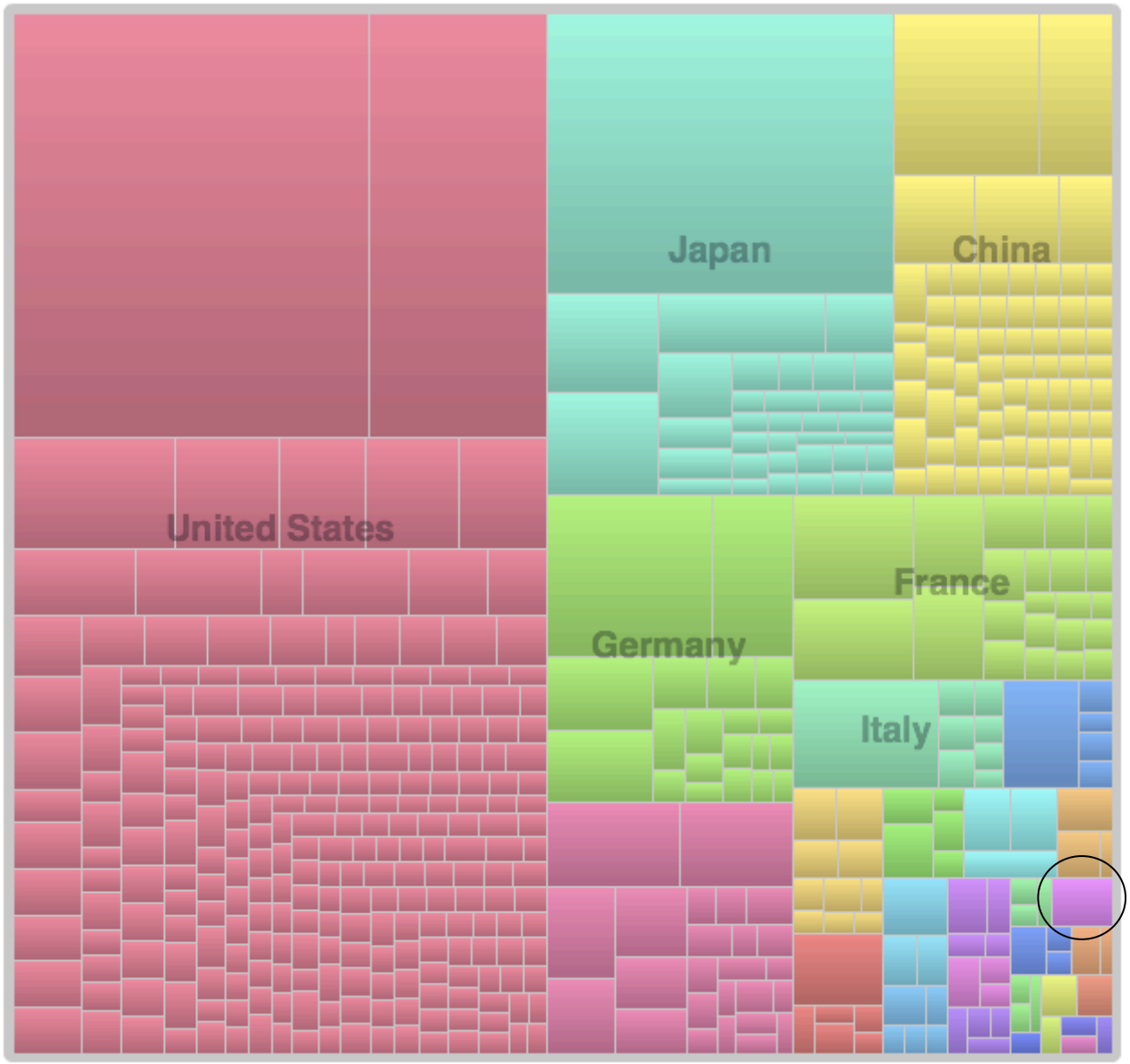
Accelerators (58 systems)



- Intel MIC (1)
- Clearspeed CSX600 (0)
- ATI GPU (2)
- IBM PowerXCell 8i (2)
- NVIDIA 2070 (10)
- NVIDIA 2050(12)
- NVIDIA 2090 (31)

27 US	2 Poland
7 China	1 Australia
4 Japan	1 Brazil
3 Russia	1 Canada
2 France	1 Singapore
2 Germany	1 Spain
2 India	1 Taiwan
2 Italy	1 UK

Countries Share

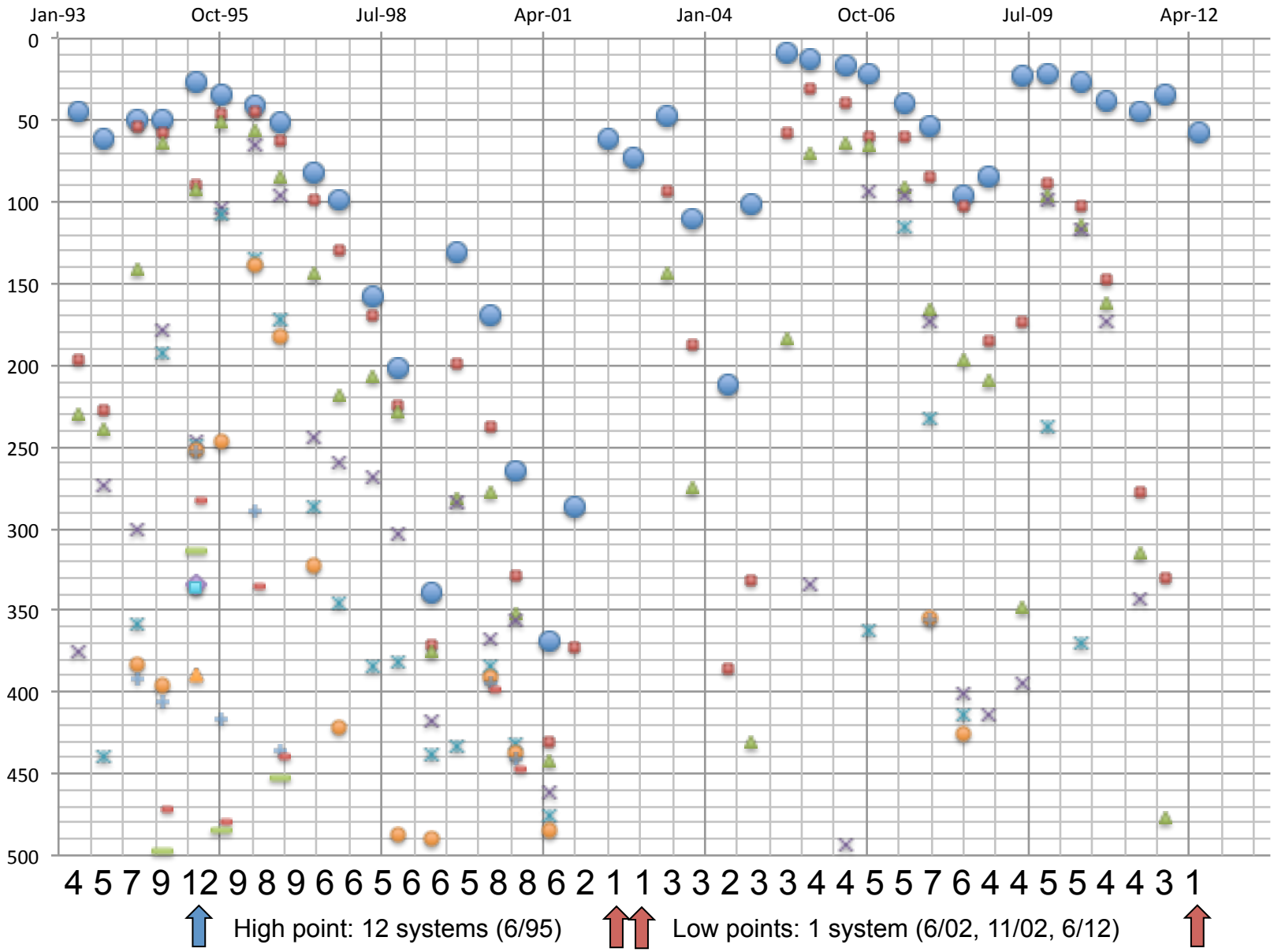


Absolute Counts

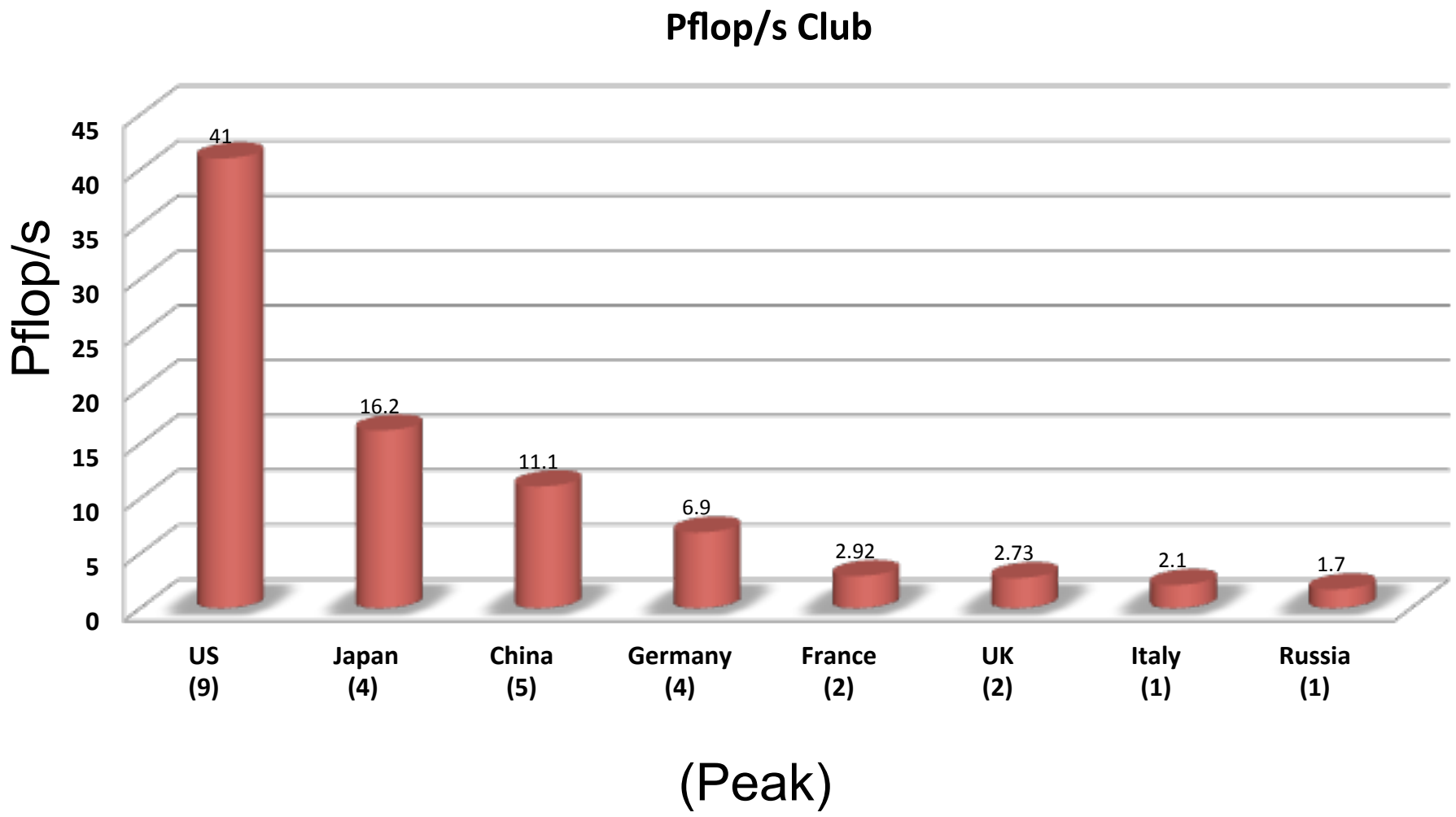
US:	252
China:	68
Japan:	35
UK:	25
France:	22
Germany:	20

Switzerland

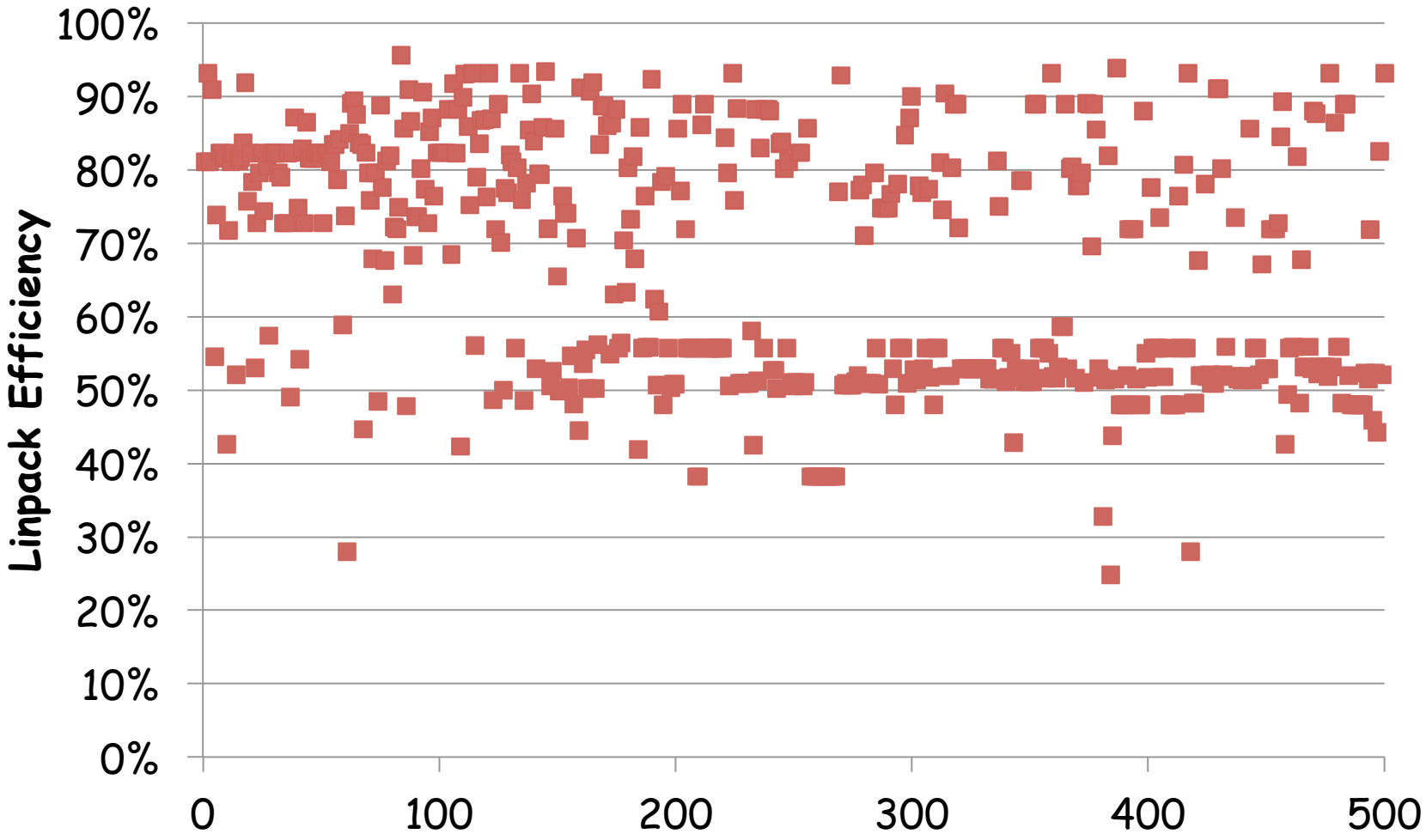
Swiss Machines in Top500 (max:12 min:1)



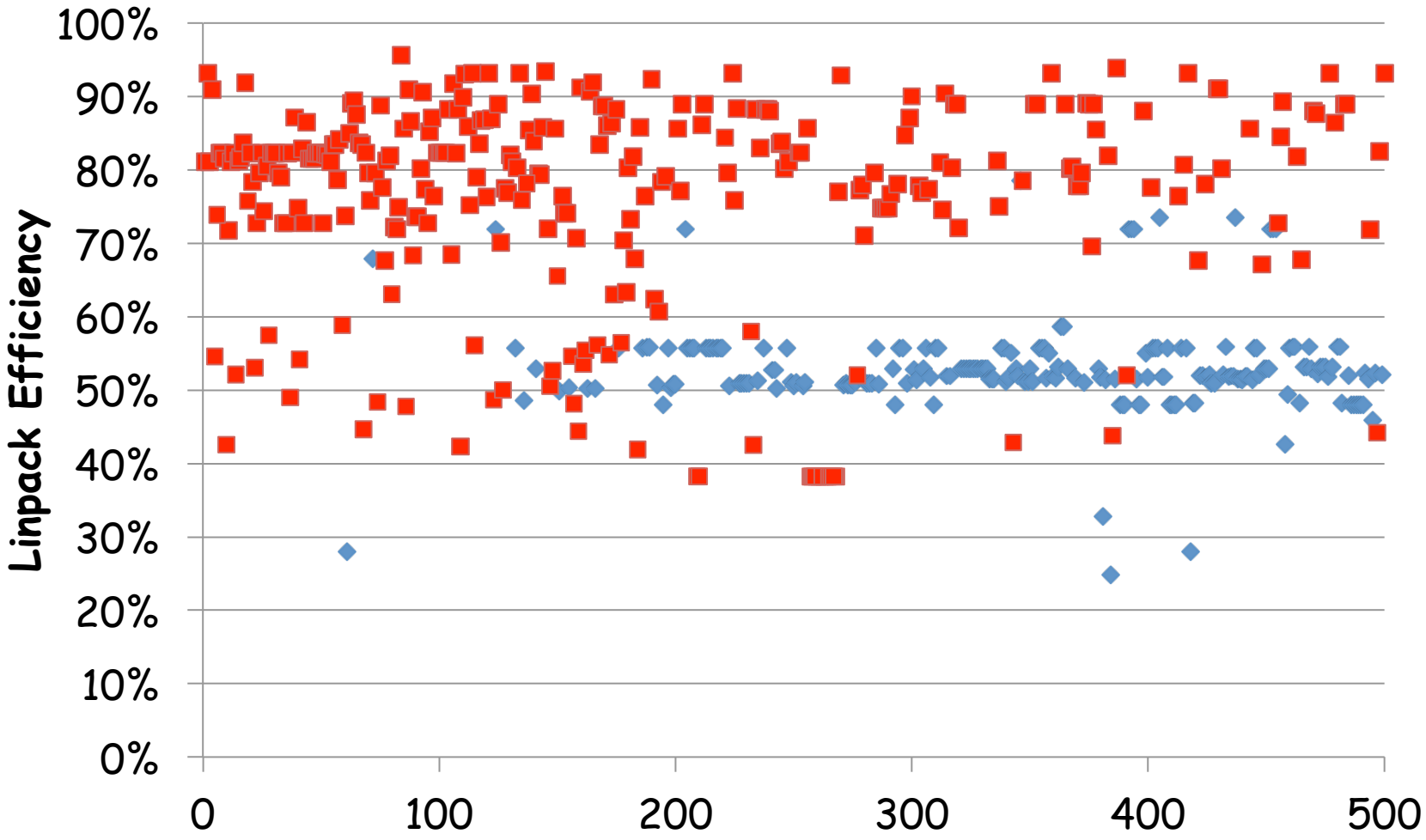
28 Systems at > Pflop/s (Peak)



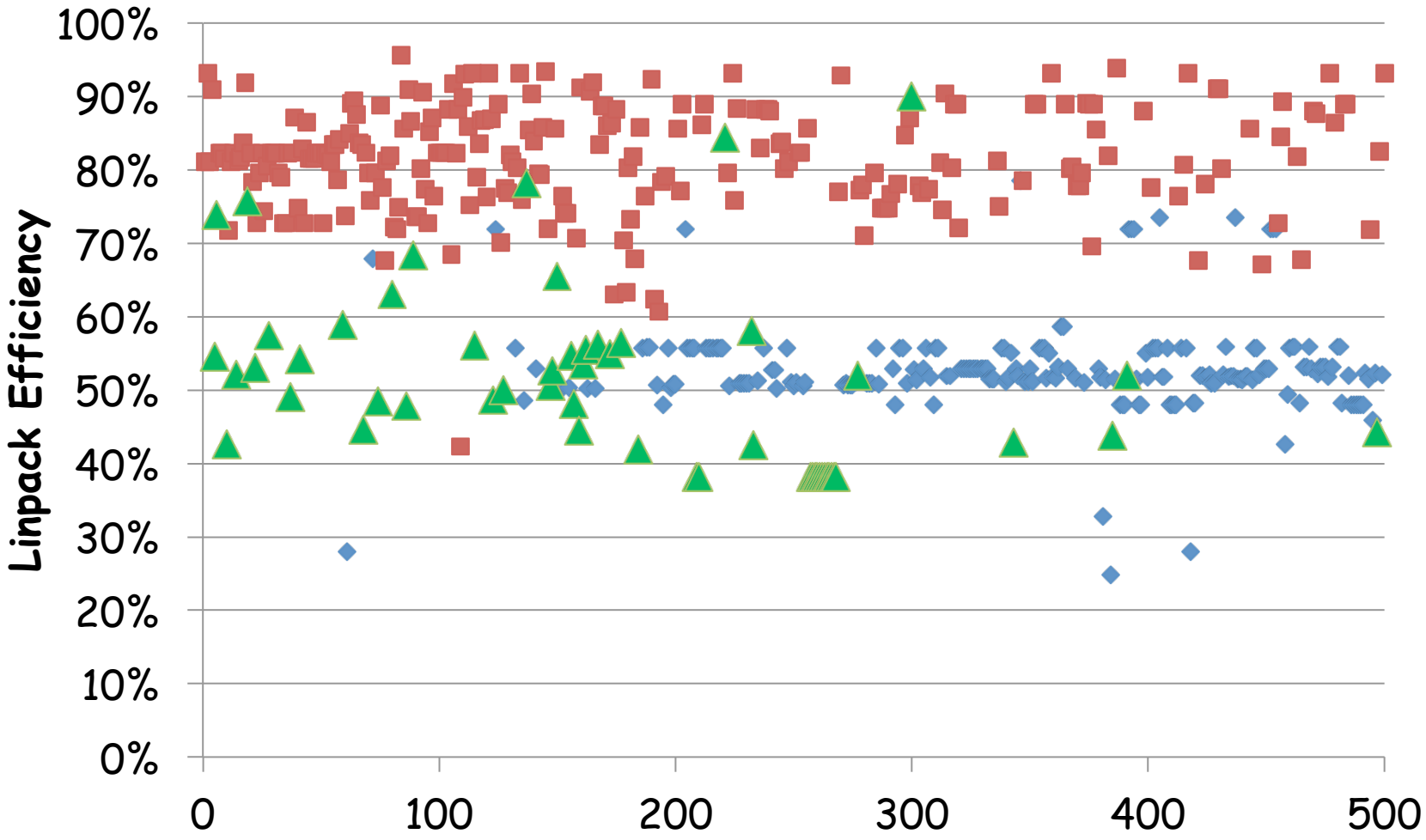
Linpack Efficiency



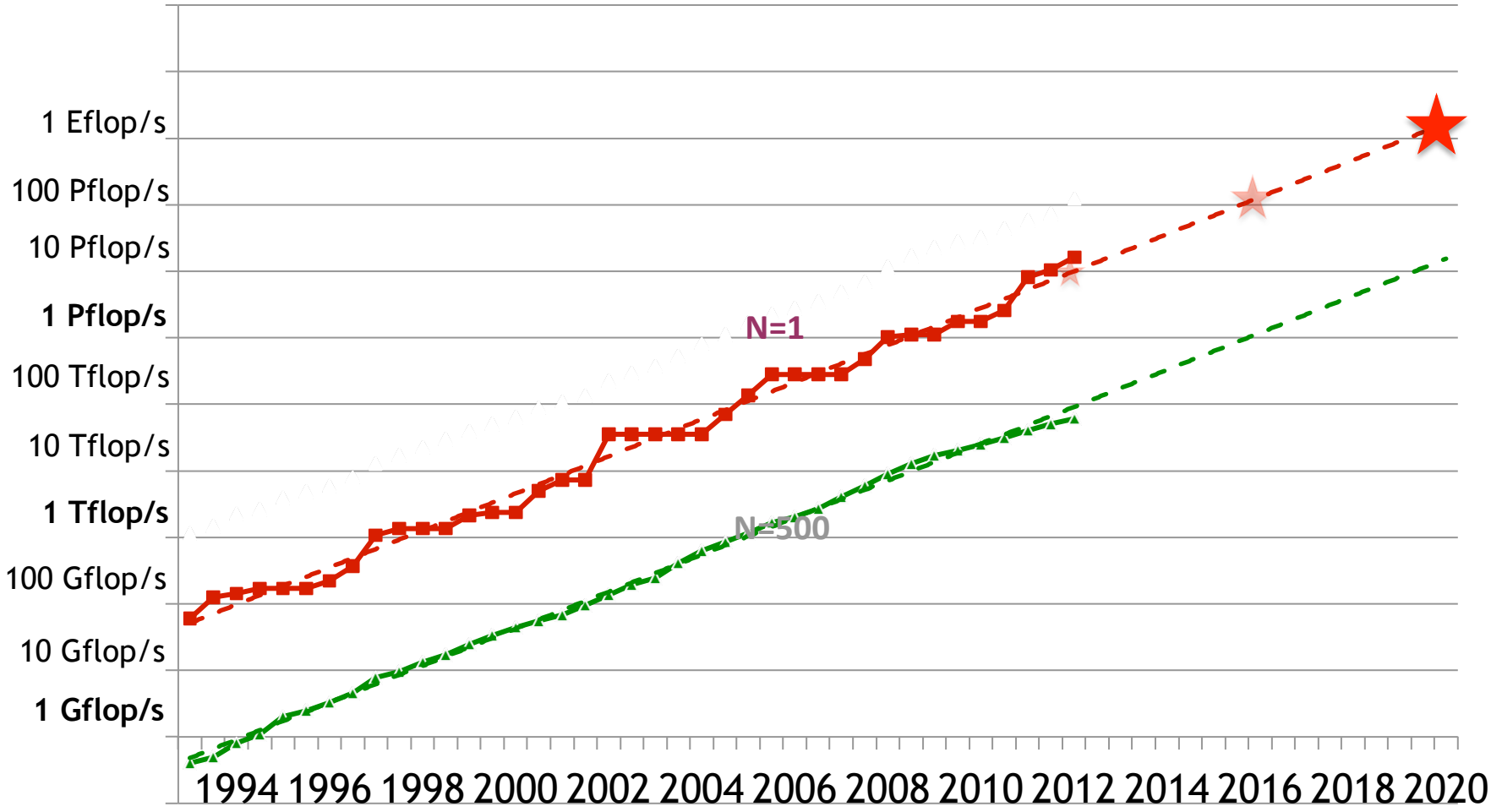
Linpack Efficiency



Linpack Efficiency



Performance Development in Top500



The High Cost of Data Movement

•Flop/s or percentage of peak flop/s become much less relevant

Approximate power costs (in picoJoules)

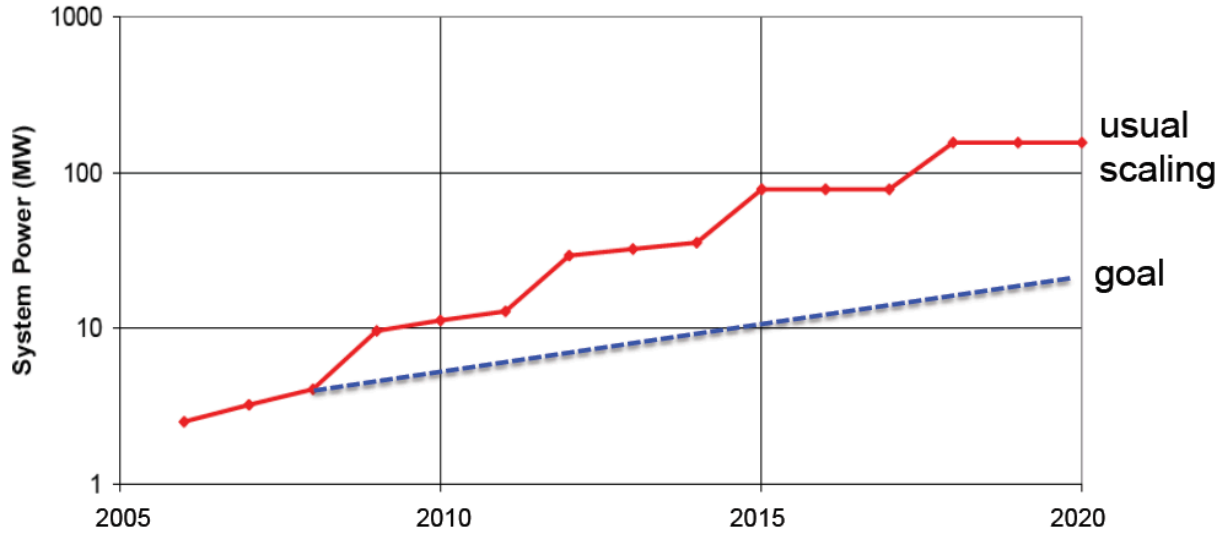
	2011
DP FMADD flop	100 pJ
DP DRAM read	4800 pJ
Local Interconnect	7500 pJ
Cross System	9000 pJ

Source: John Shalf, LBNL

•Algorithms & Software: minimize data movement; perform more work per unit data movement.

Energy Cost Challenge

- At ~$1M per MW energy costs are substantial
  - 10 Pflop/s in 2011 uses ~10 MWs
  - 1 Eflop/s in 2018 > 100 MWs



  - DOE Target: 1 Eflop/s in 2018 at 20 MWs

Potential System Architecture with a cap of $200M and 20MW

Systems	2012 BG/Q Computer
System peak	**20 Pflop/s**
Power	**8.6 MW**
System memory	1.6 PB (16*96*1024)
Node performance	205 GF/s (16*1.6GHz*8)
Node memory BW	42.6 GB/s
Node concurrency	64 Threads
Total Node Interconnect BW	20 GB/s
System size (nodes)	98,304 (96*1024)
Total concurrency	5.97 M
MTTI	4 days

Potential System Architecture with a cap of $200M and 20MW

Systems	2012 BG/Q Computer	2022	Difference Today & 2022
System peak	**20 Pflop/s**	**1 Eflop/s**	**O(100)**
Power	**8.6 MW**	**~20 MW**	
System memory	1.6 PB (16*96*1024)	32 - 64 PB	O(10)
Node performance	205 GF/s (16*1.6GHz*8)	1.2 or 15TF/s	O(10) – O(100)
Node memory BW	42.6 GB/s	2 - 4TB/s	O(1000)
Node concurrency	64 Threads	O(1k) or 10k	O(100) – O(1000)
Total Node Interconnect BW	20 GB/s	200-400GB/s	O(10)
System size (nodes)	98,304 (96*1024)	O(100,000) or O(1M)	O(100) - O(1000)
Total concurrency	5.97 M	O(billion)	O(1,000)
MTTI	4 days	O(<1 day)	- O(10)

Critical Issues at Peta & Exascale for Algorithm and Software Design

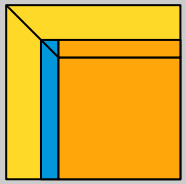
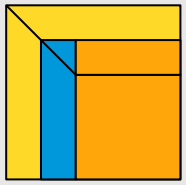
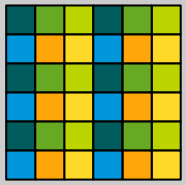
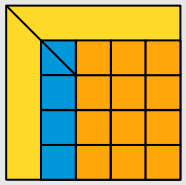
- Synchronization-reducing algorithms
  - Break Fork-Join model
- Communication-reducing algorithms
  - Use methods which have lower bound on communication
- Mixed precision methods
  - 2x speed of ops and 2x speed for data movement
- Autotuning
  - Today's machines are too complicated, build "smarts" into software to adapt to the hardware
- Fault resilient algorithms
  - Implement algorithms that can recover from failures/bit flips
- Reproducibility of results
  - Today we can't guarantee this. We understand the issues, but some of our "colleagues" have a hard time with this.

Major Changes to Algorithms/Software

- **Must rethink the design of our algorithms and software**
  - **Manycore and Hybrid architectures are disruptive technology**
    - **Similar to what happened with cluster computing and message passing**
  - **Rethink and rewrite the applications, algorithms, and software**
  - **Data movement is expensive**
  - **Flops are cheap**

Dense Linear Algebra

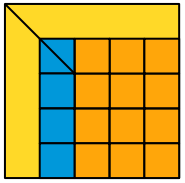
Software Evolution

	**LINPACK** (70's) vector operations	• Level 1 BLAS
	**LAPACK** (80's) block operations	• Level 3 BLAS
	**ScaLAPACK** (90's) block cyclic data distribution	• PBLAS • BLACS (message passing)
	**PLASMA** (00's) tile operations	• tile layout • dataflow scheduling

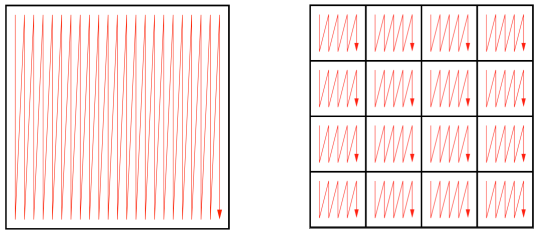
PLASMA

Principles

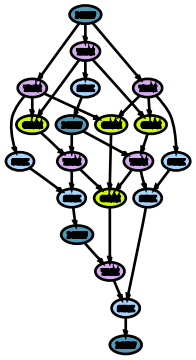
- **Tile Algorithms**
  - minimize capacity misses

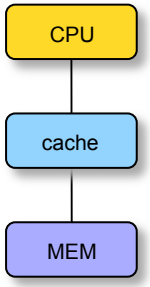


- **Tile Matrix Layout**
  - minimize conflict misses



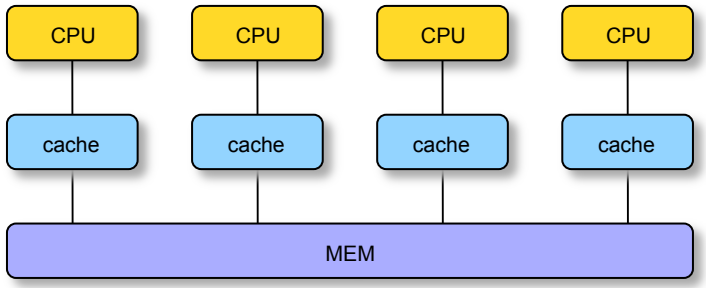
- **Dynamic DAG Scheduling**
  - minimizes idle time
  - More overlap
  - Asynchronous ops





LAPACK

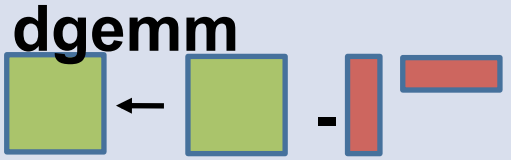
PLASMA

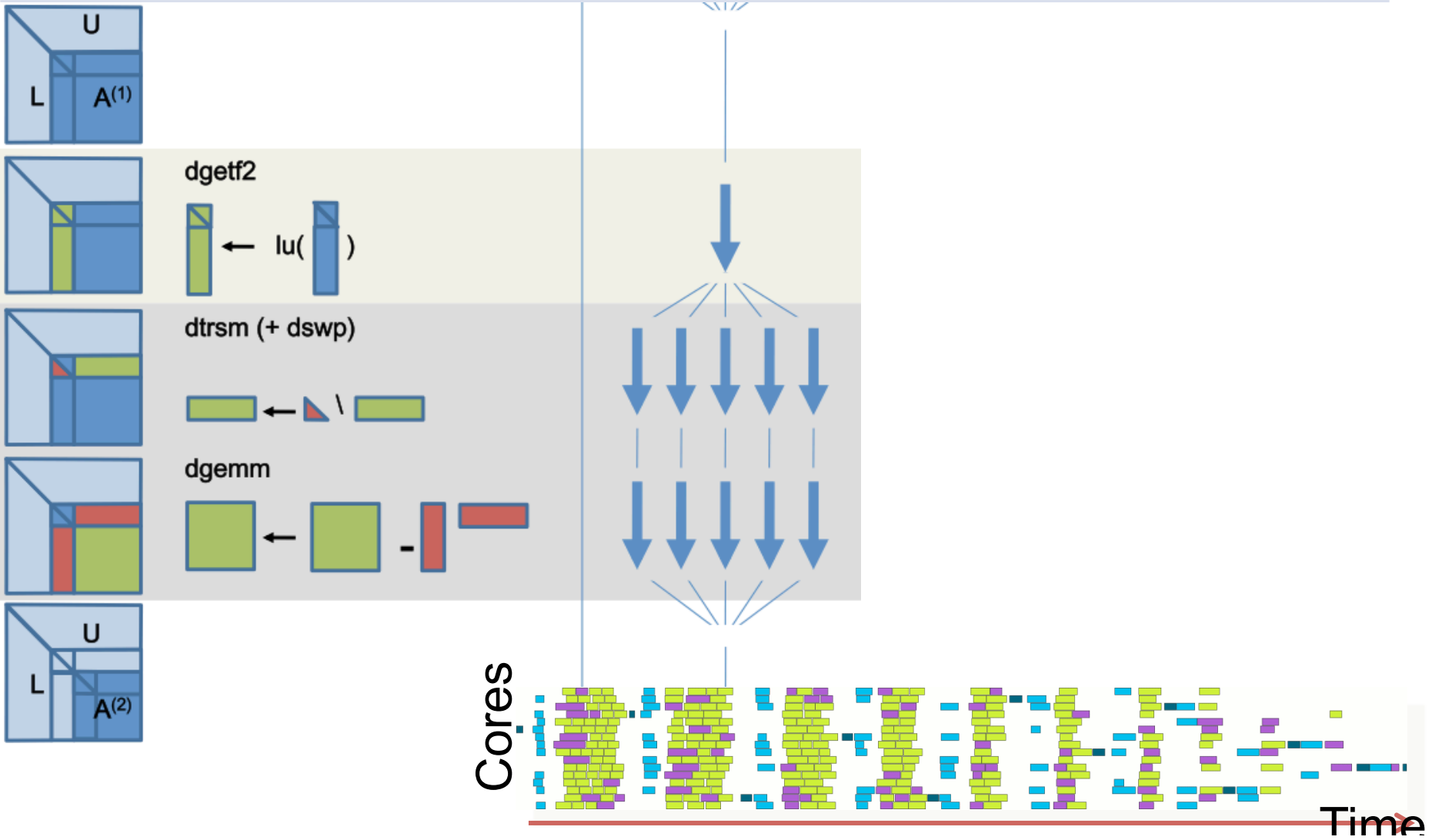


Fork-Join Parallelization of LU and QR.

Parallelize the update:

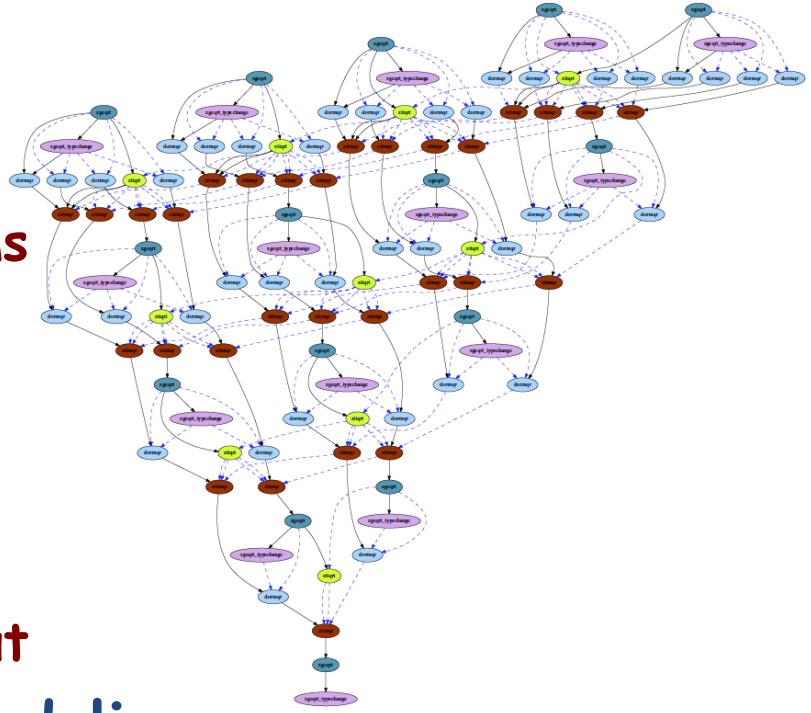
- Easy and done in any reasonable software.
- This is the $2/3n^3$ term in the FLOPs count.
- Can be done efficiently with LAPACK+multithreaded BLAS

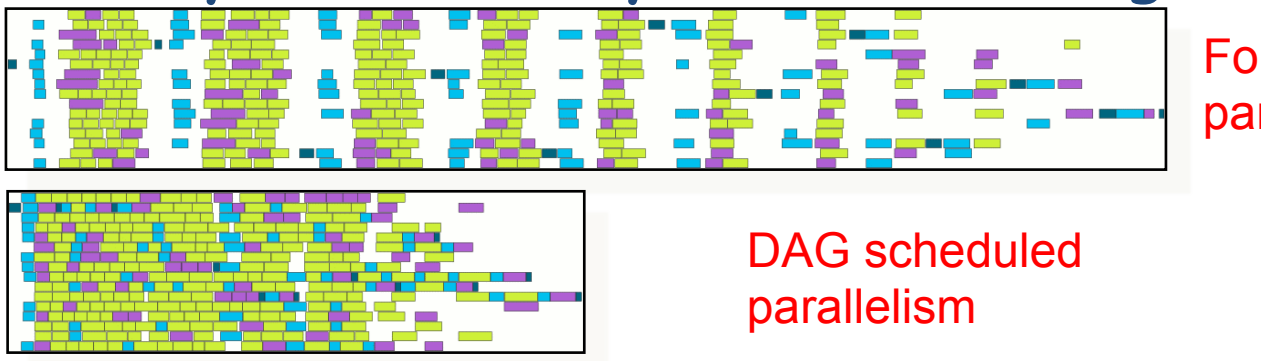




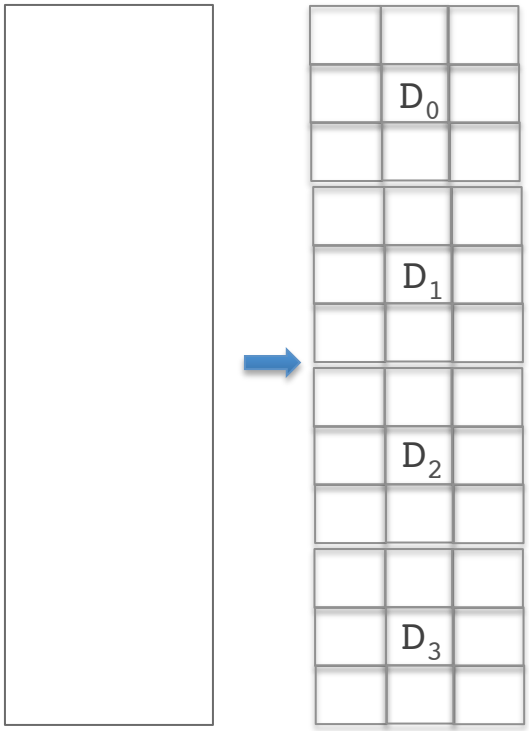
PLASMA/MAGMA: Parallel Linear Algebra s/w for Multicore/Hybrid Architectures

- Objectives
  - High utilization of each core
  - Scaling to large number of cores
  - Synchronization reducing algorithms
- Methodology
  - Dynamic DAG scheduling (QUARK)
  - Explicit parallelism
  - Implicit communication
  - Fine granularity / block data layout
- Arbitrary DAG with dynamic scheduling



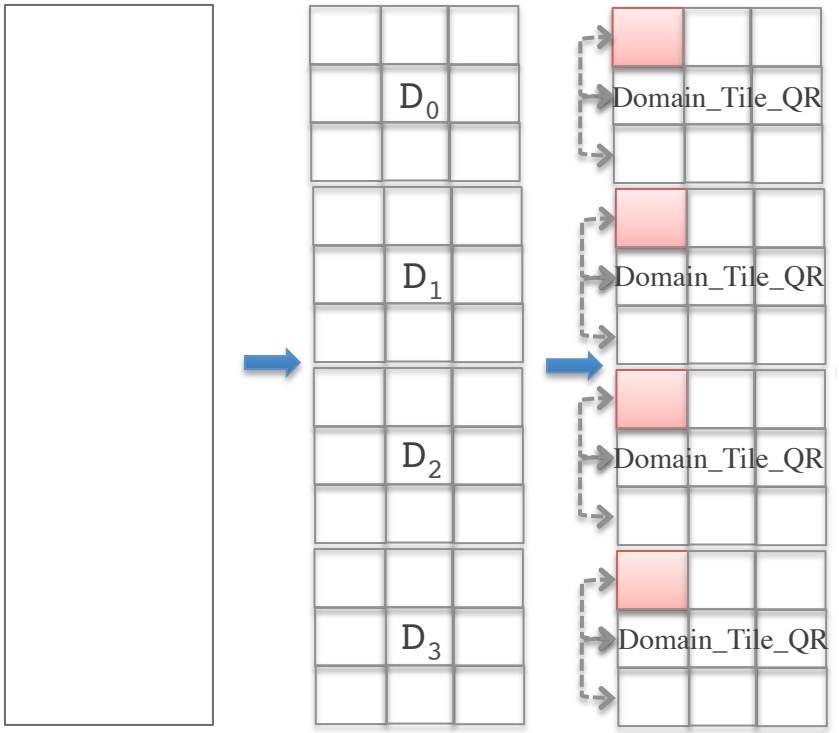


Communication Avoiding QR Example



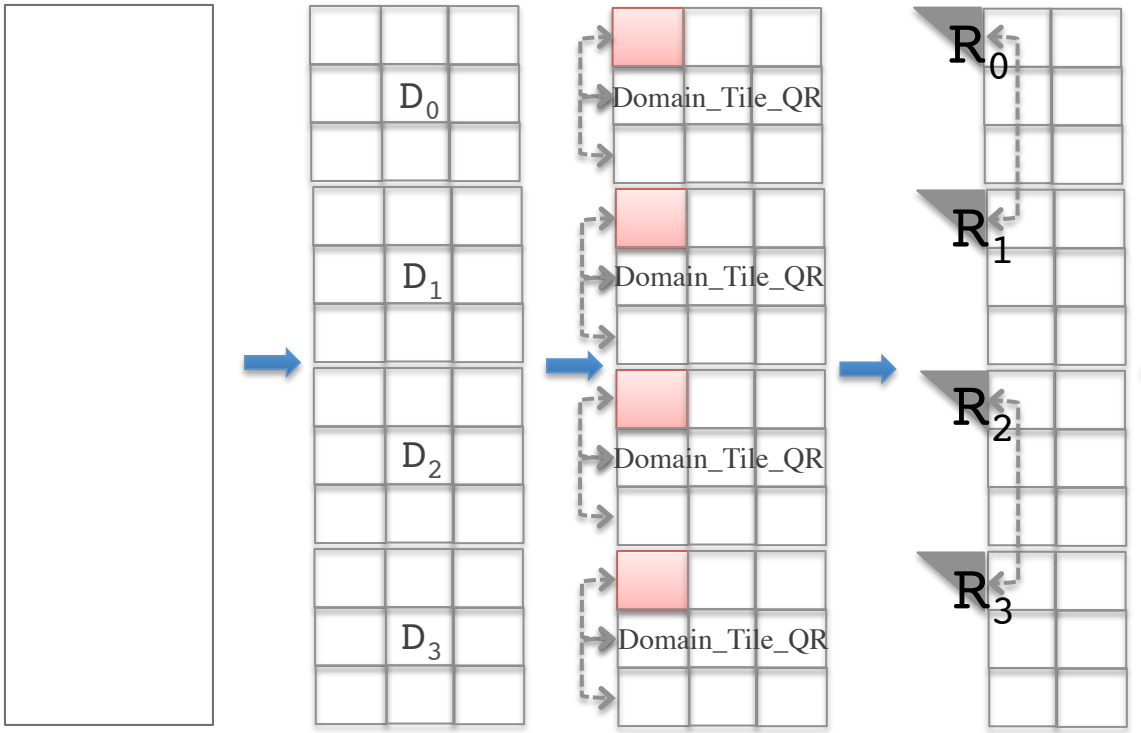
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Avoiding QR Example



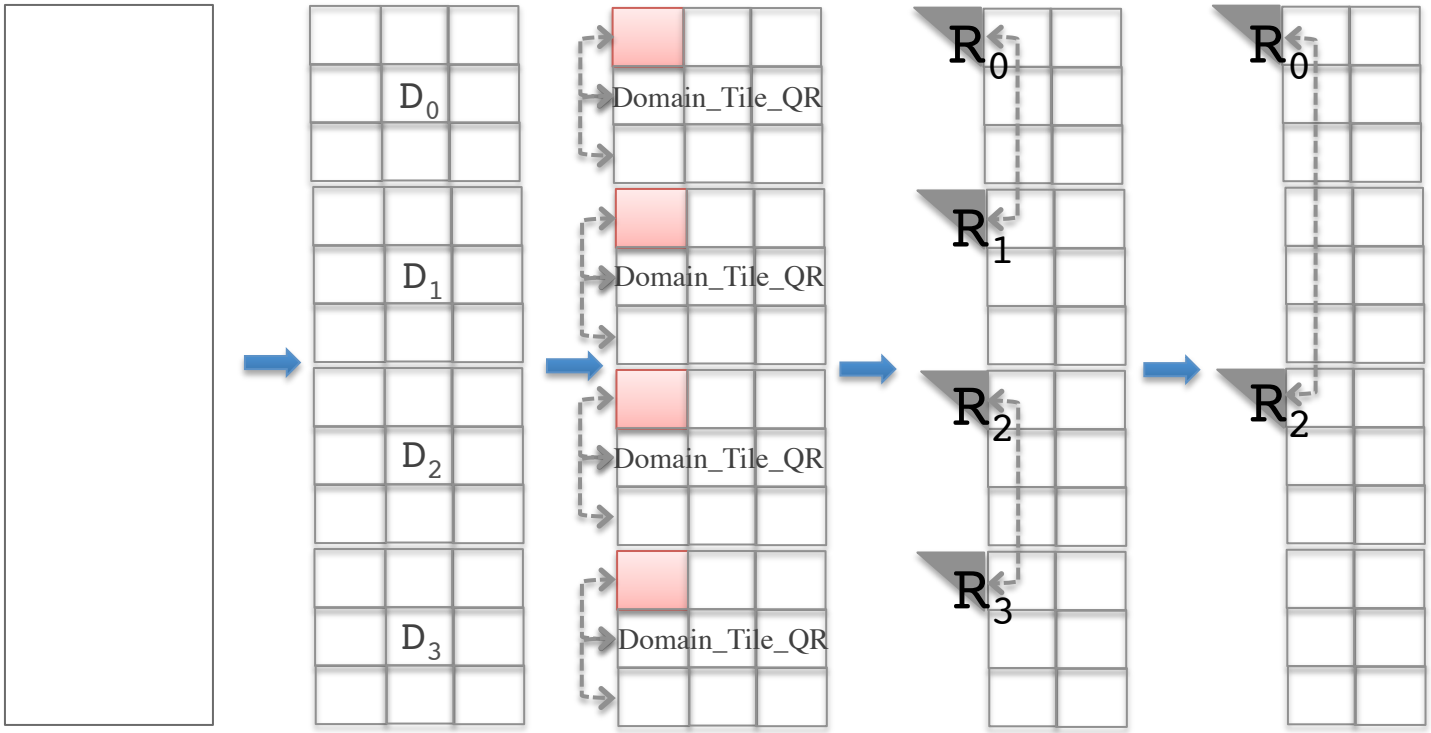
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Avoiding QR Example



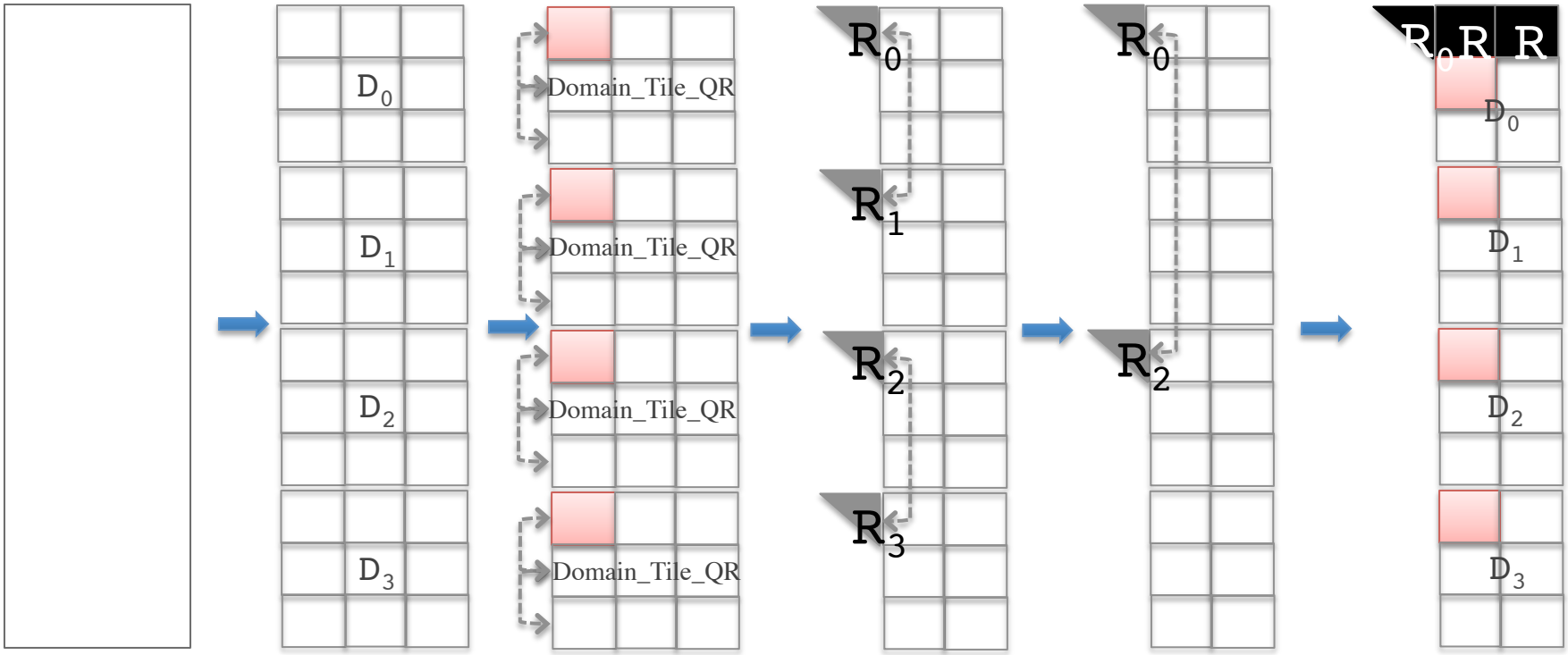
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Avoiding QR Example



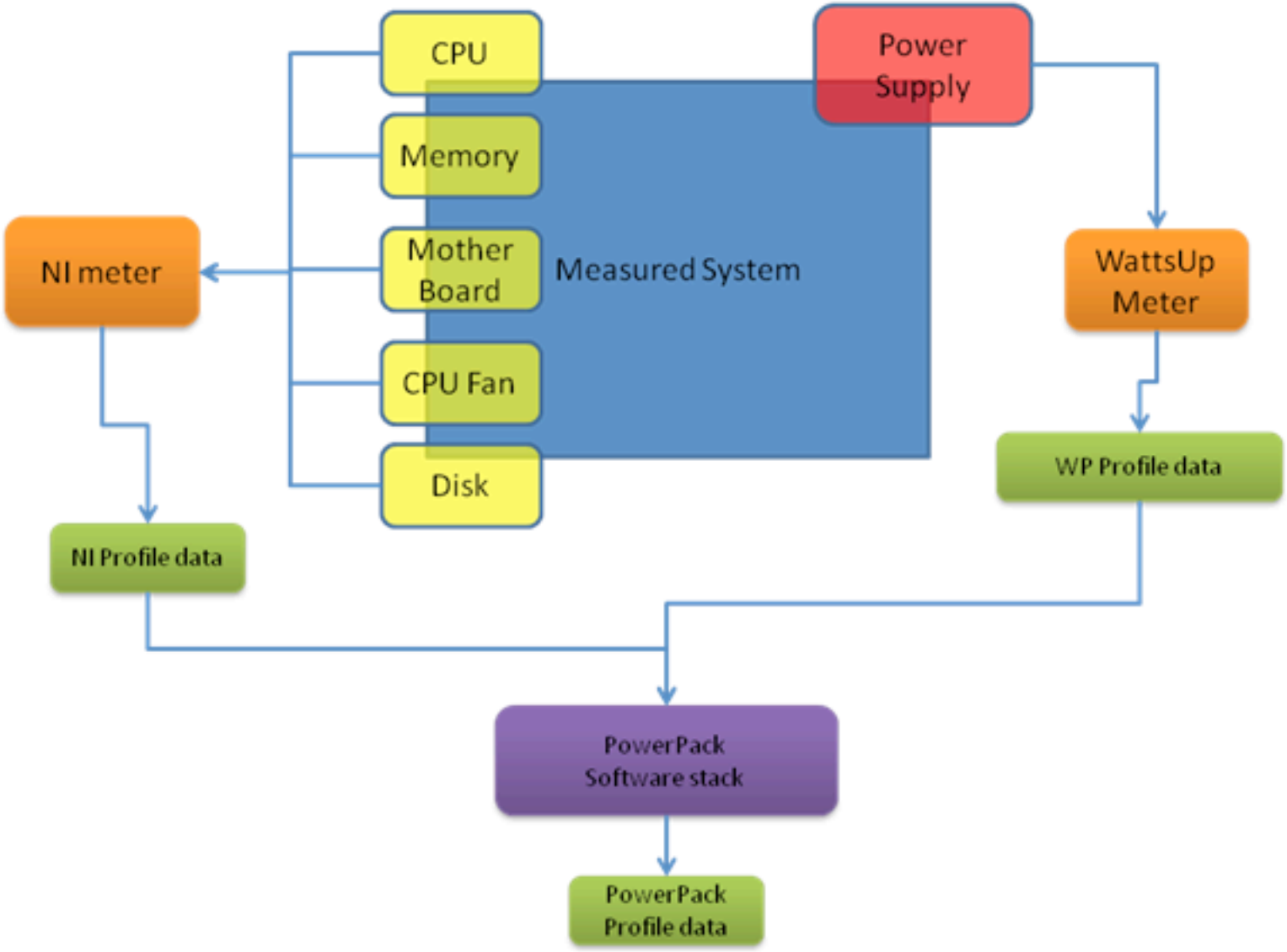
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Avoiding QR Example



A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

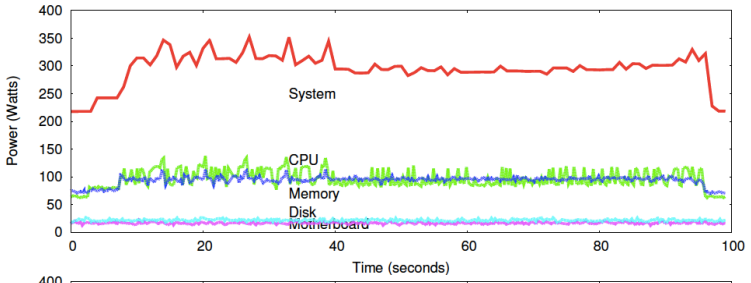
PowerPack 2.0



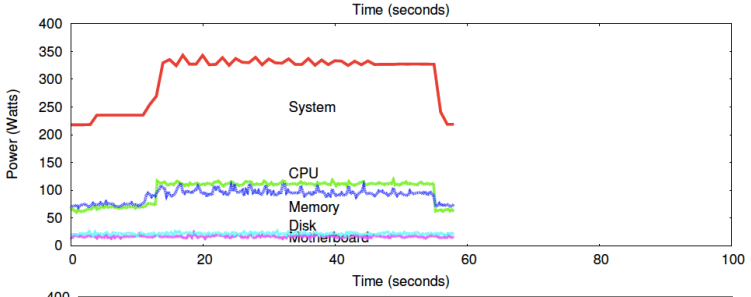
The PowerPack platform consists of software and hardwar

Kirk Cameron, Virginia Tech; http://scape.cs.vt.edu/softwar

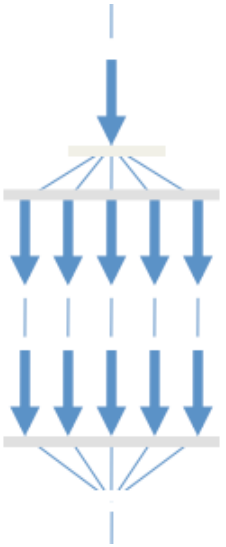
Power for QR Factorization

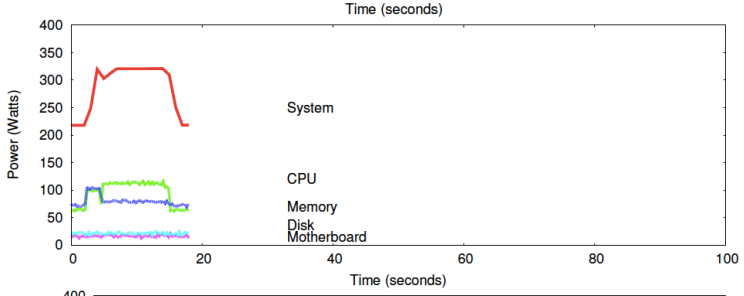


LAPACK's QR Factorization
Fork-join based

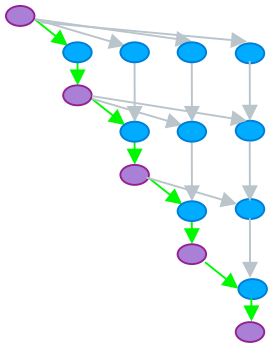


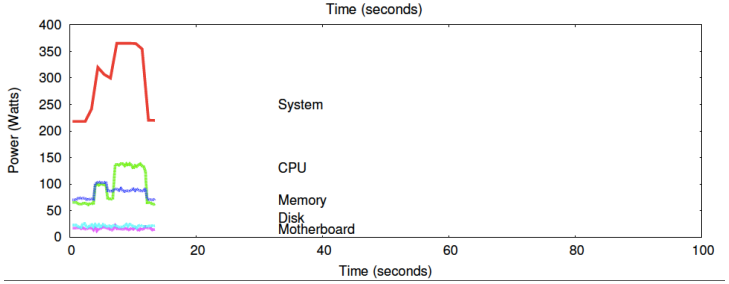
MKL's QR Factorization
Fork-join based





PLASMA's Conventional
QR Factorization
DAG based





PLASMA's Communication
Reducing QR Factorization
DAG based

dual-socket quad-core Intel Xeon E5462 (Harpertown) processor
@ 2.80GHz (8 cores total) w / MLK BLAS
matrix size is very tall and skinny (mxn is 1,152,000 by 288)

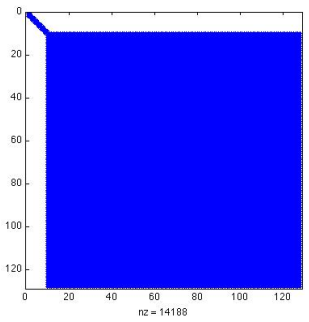
The standard Tridiagonal reduction xSYTRD

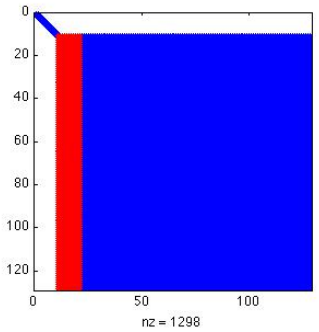
- **LAPACK xSYTRD:**

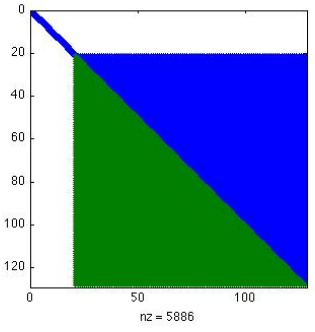
1. Apply left-right transformations Q A Q* to the panel $\begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix}$
2. Update the remaining submatrix A_{33}

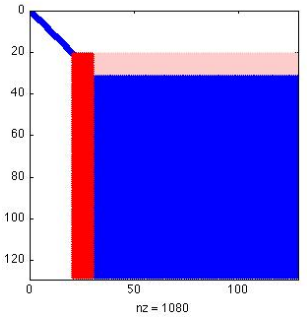
$$\begin{pmatrix} T_{11} & T_{21}^T & 0 \\ T_{21} & A_{22} & A_{32}^T \\ 0 & A_{32} & A_{33} \end{pmatrix} \equiv \begin{pmatrix} T_{11} & T_{21}^T & 0 \\ T_{21} & A_{22} & A_{32}^T \\ 0 & A_{32} & A_{33} \end{pmatrix} \Longrightarrow \begin{pmatrix} T_{11} & T_{21}^T & 0 \\ T_{21} & T_{22} & T_{23}^T \\ 0 & T_{23} & A_{33} \end{pmatrix}$$

where $A_{33} = A_{33} - YW^T - WY^T$









step *k*: Q A Q* then update ➔ step *k+1*

For the symmetric eigenvalue problem:
First stage takes:
- **90% of the time if only eigenvalues**
- **50% of the time if eigenvalues and eigenvectors**

The standard Tridiagonal reduction xSYTRD

Characteristics

1. Phase 1 requires :
   - 4 panel vector multiplications,
   - 1 symmetric matrix vector multiplication with A_{33},
   - Cost $2(n-k)^2b$ flops.
2. Phase 2 requires:
   - Symmetric update of A_{33} using SYRK,
   - Cost $2(n-k)^2b$ flops.

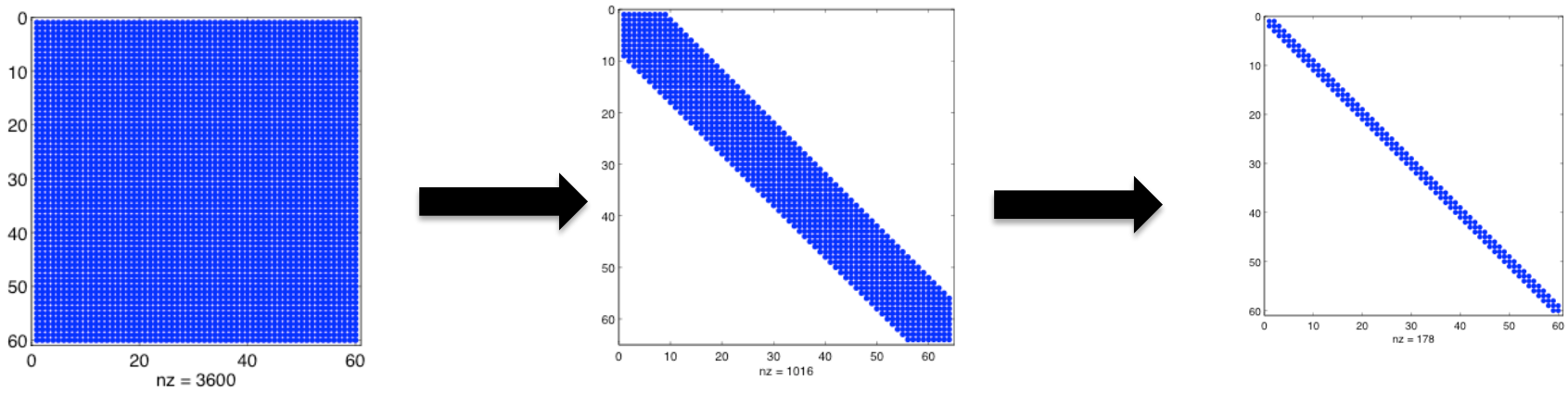
Observations

- Too many Level 2 BLAS ops,
- Relies on panel factorization,
- Total cost $4n^3/3$
- ➔Bulk sync phases,
- ➔Memory bound algorithm.

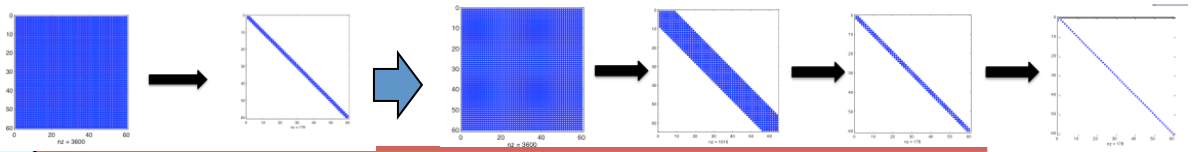
Symmetric Eigenvalue Problem

- Standard reduction algorithm are very slow on multicore.

- Step1: Reduce the dense matrix to band.
  - Matrix-matrix operations, high degree of parallelism
- Step2: Bulge Chasing on the band matrix
  - by group and cache aware



Symmetric

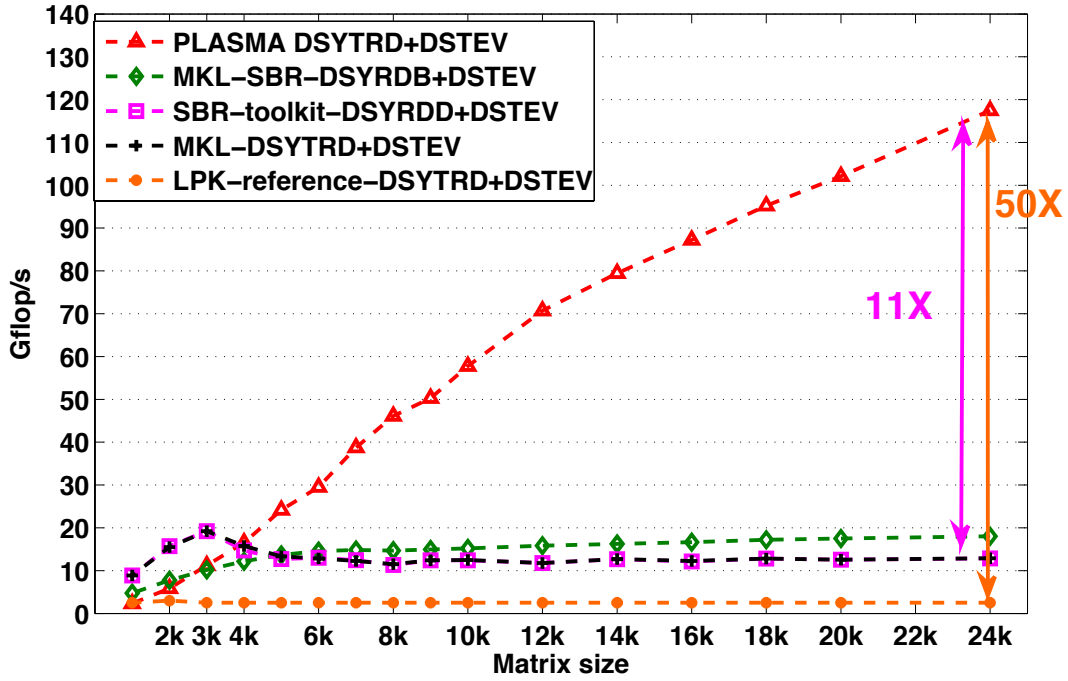


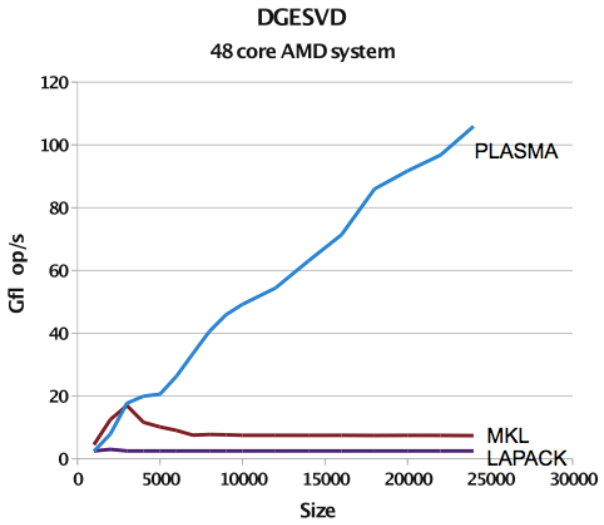
Eigenvalues

eigenvalues only

Singular Values

singular values only





Experiments on eight-socket six-core AMD Opteron 2.4 GHz processors with MKL V10.3.

- Block DAG based to banded form, then pipelined group chasing to tridiagaonal form.
- The reduction to condensed form accounts for the factor of 50 improvement over LAPACK
- Execution rates based on $4/3n^3$ ops

Summary

- These are old ideas (today SMPss, StarPU, Charm++, ParalleX, Swarm,…)
- Major Challenges are ahead for extreme computing
  - Power
  - Levels of Parallelism
  - Communication
  - Hybrid
  - Fault Tolerance
  - … and many others not discussed here
- Not just a programming assignment.
- This opens up many new opportunities for applied mathematicians and computer scientists

Collaborators / Software / Support

- **PLASMA**
  http://icl.cs.utk.edu/plasma/
- **MAGMA**
  http://icl.cs.utk.edu/magma/
- **Quark (RT for Shared Memory)**
  - **http://icl.cs.utk.edu/quark/**
- **PaRSEC**(Parallel Runtime Scheduling and Execution Control)
  - **http://icl.cs.utk.edu/parsec/**

- Collaborating partners
  University of Tennessee, Knoxville
  University of California, Berkeley
  University of Colorado, Denver

  INRIA, France
  KAUST, Saudi Arabia

These tools are being applied to a range of applications beyond dense LA:
Sparse direct, Sparse iterations methods and Fast Multipole Methods