A Look at High Performance Computing

Jack Dongarra

University of Tennessee
Oak Ridge National Laboratory
University of Manchester

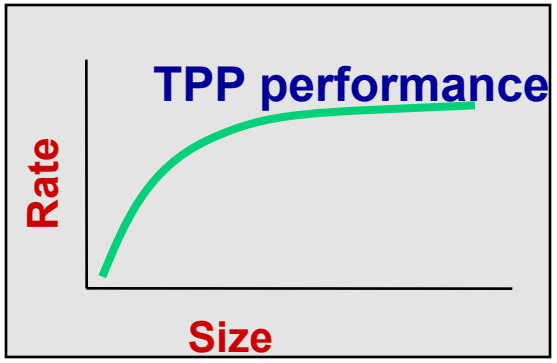
Outline

- Overview of High Performance Computing
- With Extreme Computing the “rules” for computing have changed

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

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

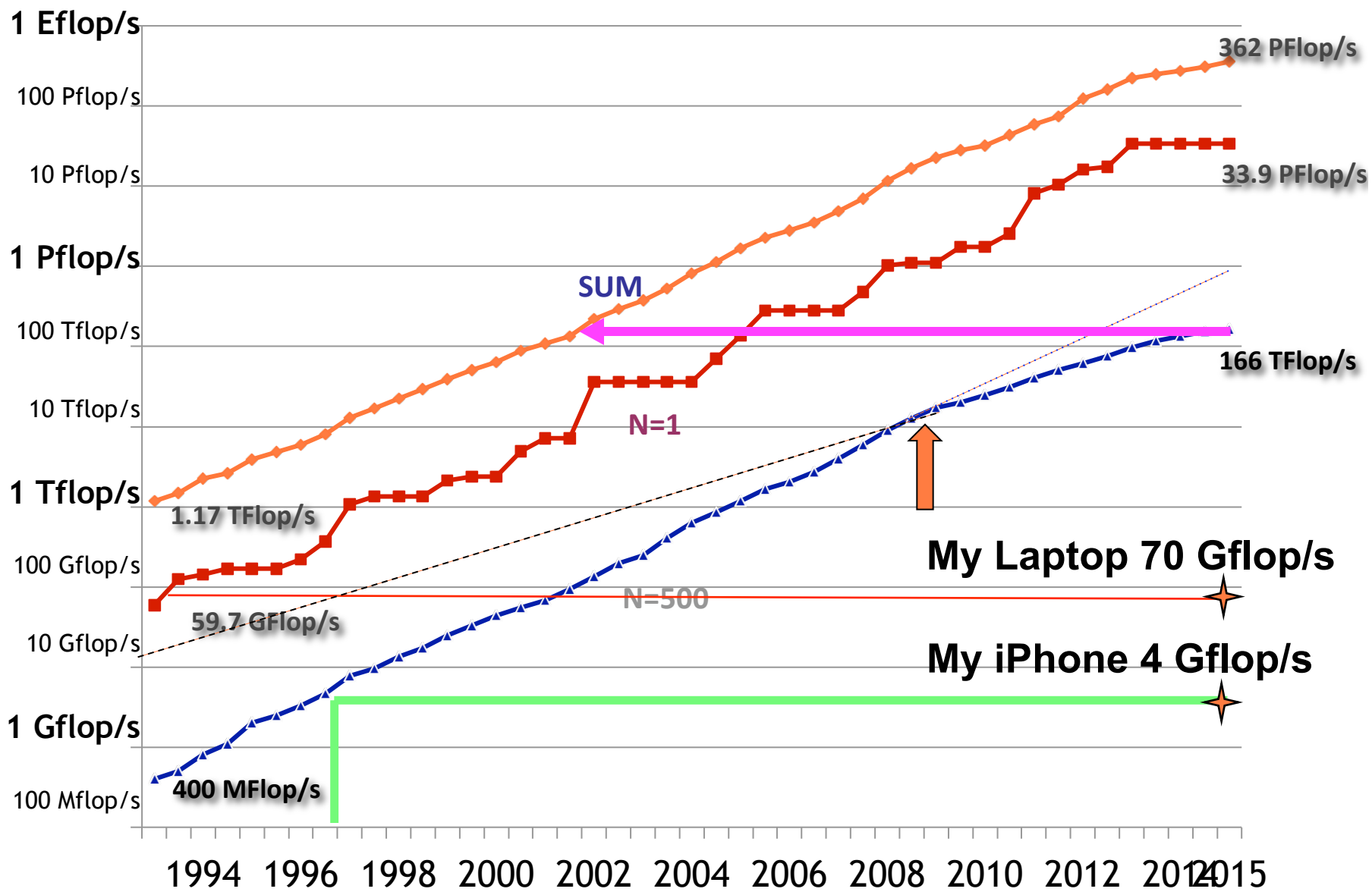
  $Ax=b$, *dense problem*



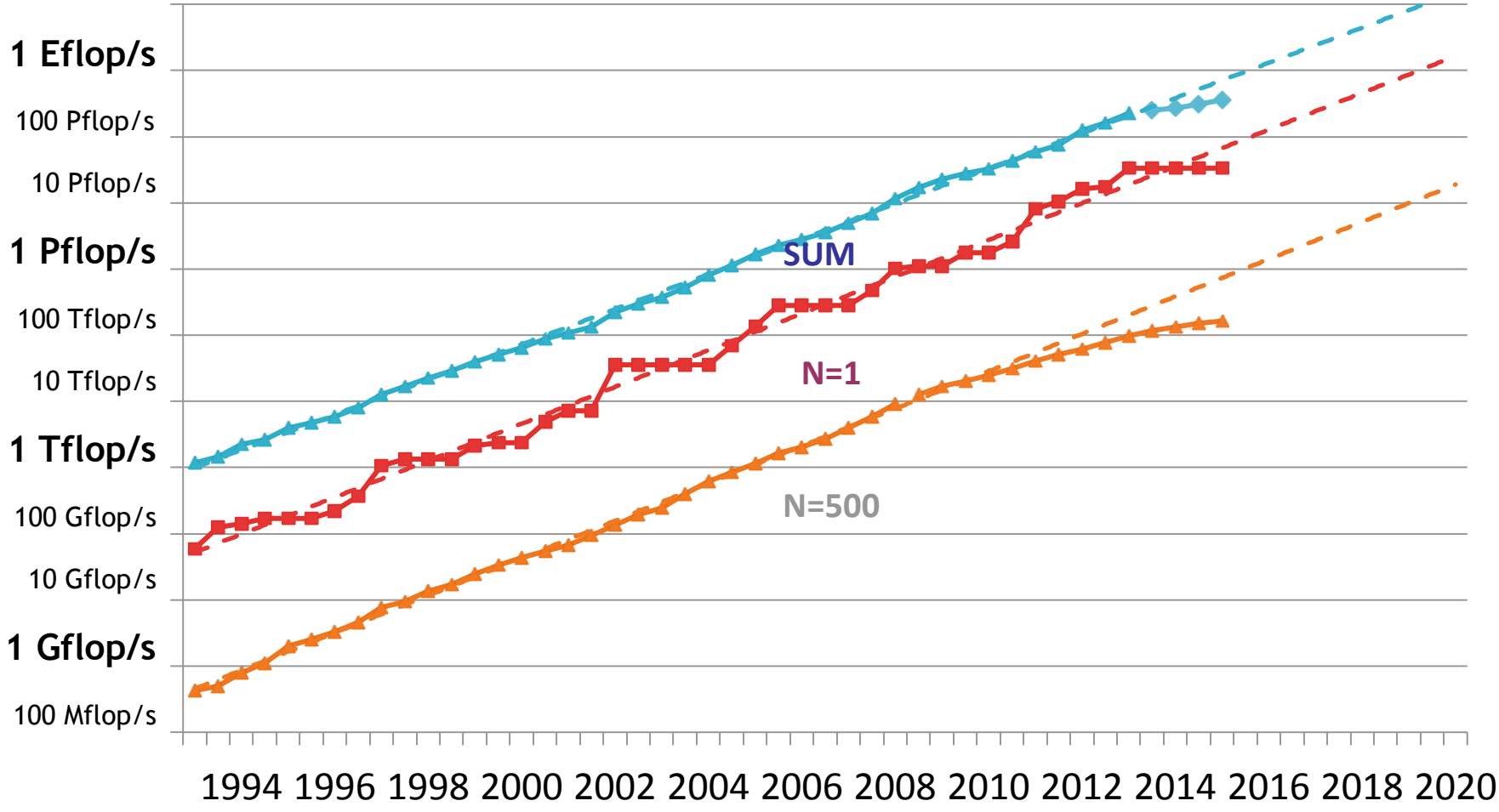
- Updated twice a year
  SC‘xy in the States in November
  Meeting in Germany in June

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

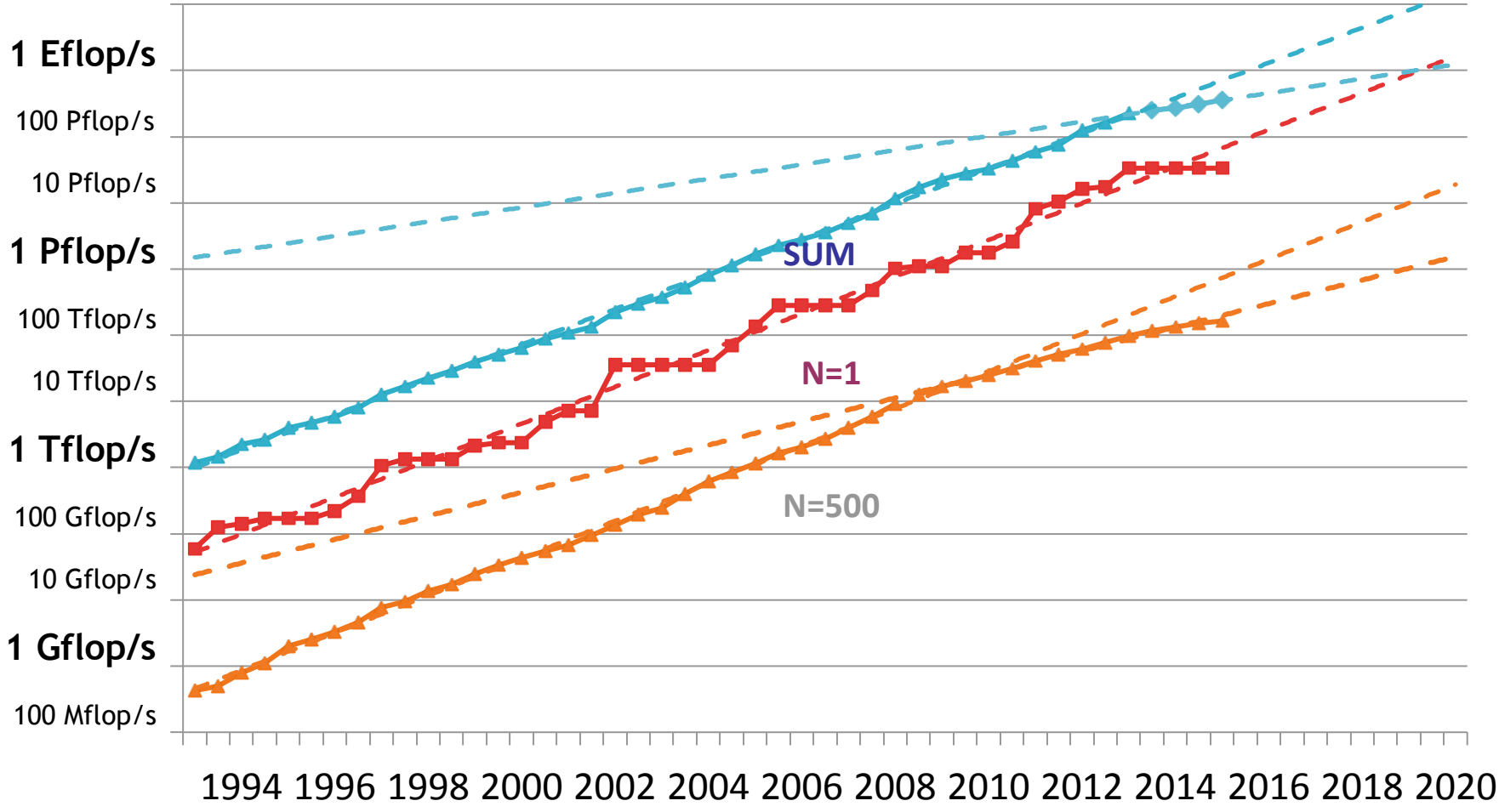
Performance Development of HPC over the Last 23 Years from the Top500



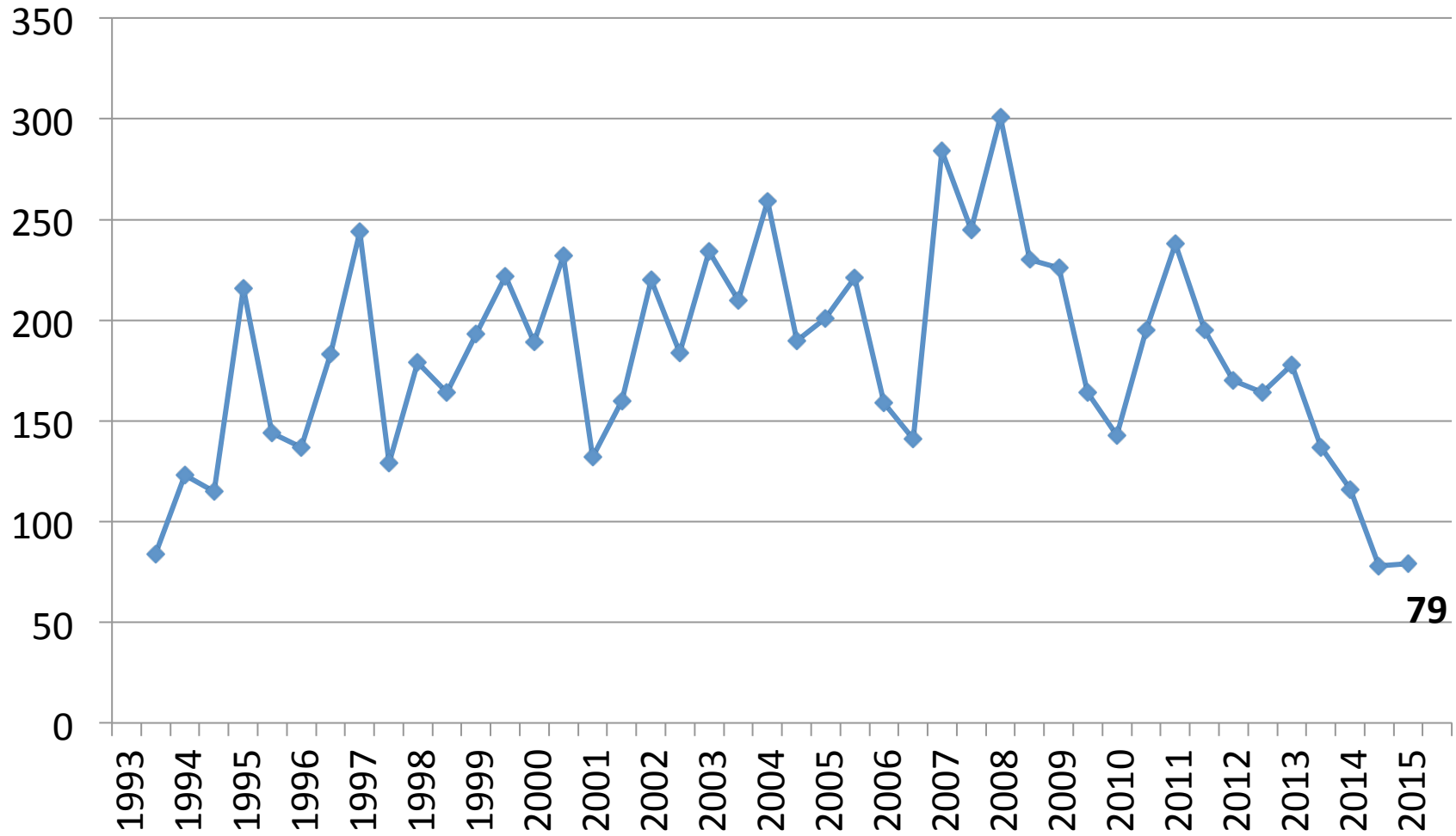
Projected Performance Development



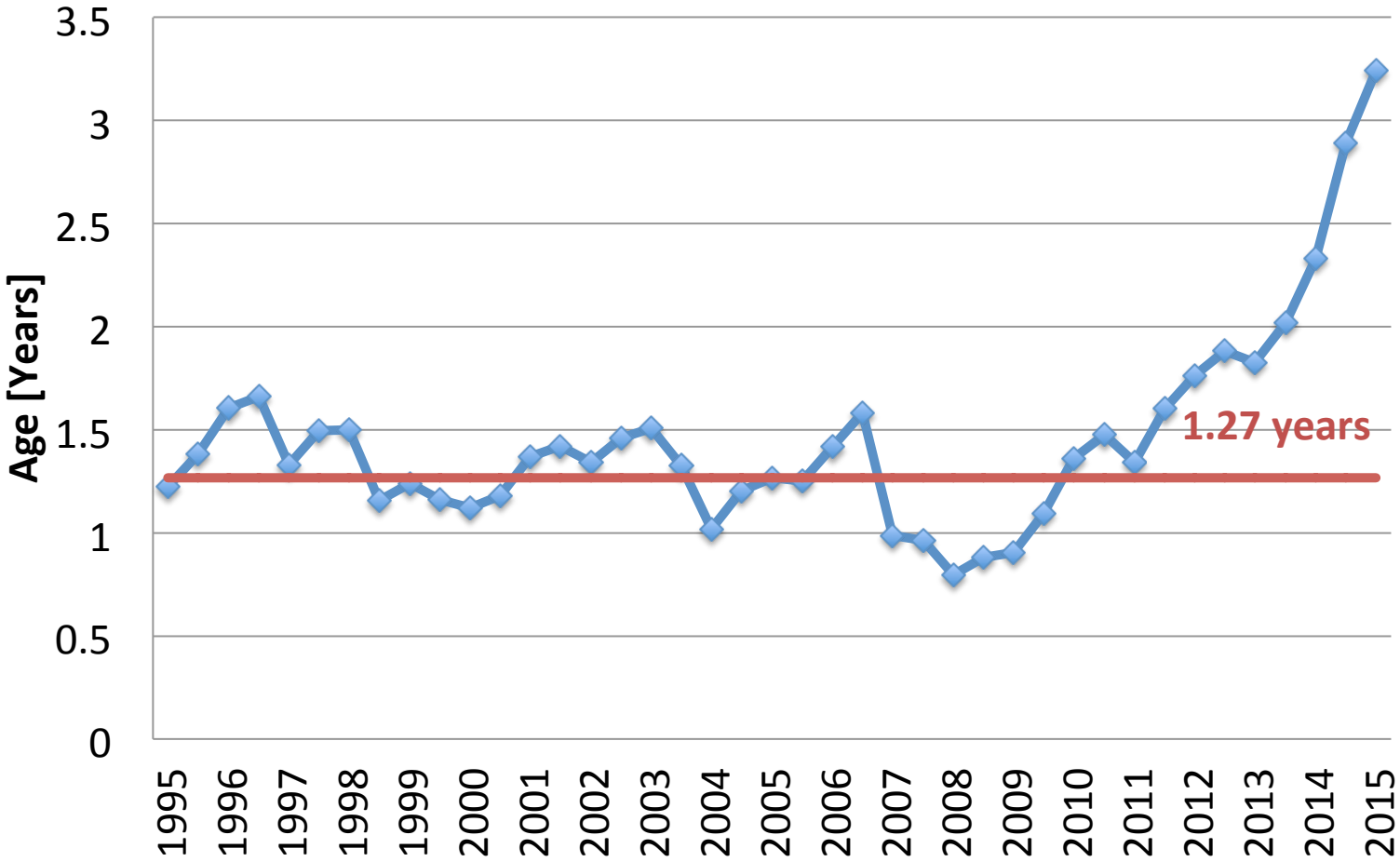
Projected Performance Development



Replacement Rate in List



Average System Age



State of Supercomputing in 2015

- Pflops (> 10^{15} Flop/s) computing fully established with 67 systems.
- Three technology architecture possibilities or "swim lanes" are thriving.
  - Commodity (e.g. Intel)
  - Commodity + accelerator (e.g. GPUs) (88 systems)
  - Special purpose lightweight cores (e.g. IBM BG, ARM, Knights Landing)
- Interest in supercomputing is now worldwide, and growing in many new markets (over 50% of Top500 computers are in industry).
- Exascale (10^{18} Flop/s) projects exist in many countries and regions.
- Intel processors largest share, 86% followed by AMD, 4%.

July 2015: The TOP 10 Systems

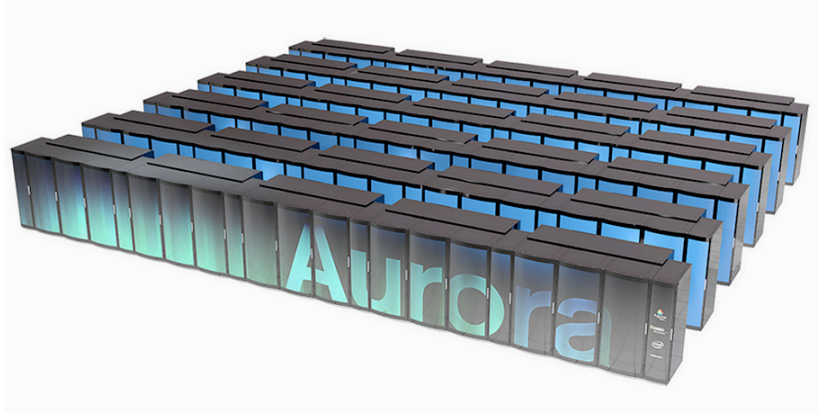
Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	MFlops /Watt
1	National Super Computer Center in Guangzhou	Tianhe-2 NUDT, Xeon 12C 2.2GHz + IntelXeon Phi (57c) + Custom	China	3,120,000	33.9	62	17.8	1905
2	DOE / OS Oak Ridge Nat Lab	Titan, Cray XK7, AMD (16C) + Nvidia Kepler GPU (14c) + Custom	USA	560,640	17.6	65	8.3	2120
3	DOE / NNSA L Livermore Nat Lab	Sequoia, BlueGene/Q (16c) + custom	USA	1,572,864	17.2	85	7.9	2063
4	RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx (8c) + Custom	Japan	705,024	10.5	93	12.7	827
5	DOE / OS Argonne Nat Lab	Mira, BlueGene/Q (16c) + Custom	USA	786,432	8.16	85	3.95	2066
6	Swiss CSCS	Piz Daint, Cray XC30, Xeon 8C + Nvidia Kepler (14c) + Custom	Swiss	115,984	6.27	81	2.3	2726
7	KAUST	Shaheen II, Cray XC30, Xeon 16C + Custom	Saudi Arabia	196,608	5.54	77	4.5	1146
8	Texas Advanced Computing Center	Stampede, Dell Intel (8c) + Intel Xeon Phi (61c) + IB	USA	204,900	5.17	61	4.5	1489
9	Forschungszentrum Juelich (FZJ)	JuQUEEN, BlueGene/Q, Power BQC 16C 1.6GHz+Custom	Germany	458,752	5.01	85	2.30	2178
10	DOE / NNSA L Livermore Nat Lab	Vulcan, BlueGene/Q, Power BQC 16C 1.6GHz+Custom	USA	393,216	4.29	85	1.97	2177
500 (422)	Software Comp	HP Cluster	USA	18,896	.309	48		

Eight Russian Top500 Systems

Rank	Name	Site	Manufacturer	Cores	Rmax (Pflop/s)	Rpeak (Pflop/s)	Acc
30	Lomonosov-2	MSU	T-Platforms	37,120	1.84	2.57	Nvidia
77	Lomonosov	MSU	T-Platforms	78,660	.901	1.70	Nvidia
106	Polytechnic RSC Tornado	St. Petersburg Polytechnic U	RSC Group	19,936	.658	.829	none
176	MVS-10P	Joint Supercomputer Center	RSC Group	28,704	.375	.523	Phi
242	Lobachevsky	Lobachevsky State U of Nizhni Novgorod	Niagara Computers, Supermicro	5310	.289	.348	Nvidia
245	RSC Tornado SUSU	South Ural State University	RSC Group	28,032	.288	.473	Phi
416		IT Services Provider	Hewlett-Packard	13,180	.189	.295	none
469	RSC PetaStream	St. Petersburg Polytechnic U	RSC Group	22,528	.170	.572	Phi

Recent Developments

- US DOE planning to deploy O(100) Pflop/s systems for 2017-2018 - $525M hardware
- Oak Ridge Lab and Lawrence Livermore Lab to receive IBM and Nvidia based systems
- Argonne Lab to receive Intel based system
  - After this Exaflops



Peak Performance - Per Core

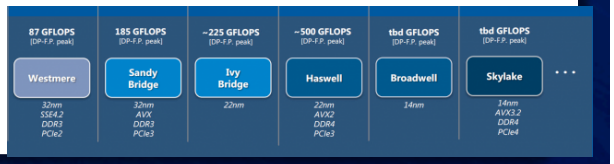
$$\text{FLOPS} = \text{cores} \times \text{clock} \times \frac{\text{FLOPs}}{\text{cycle}}$$

Floating point operations per cycle per core

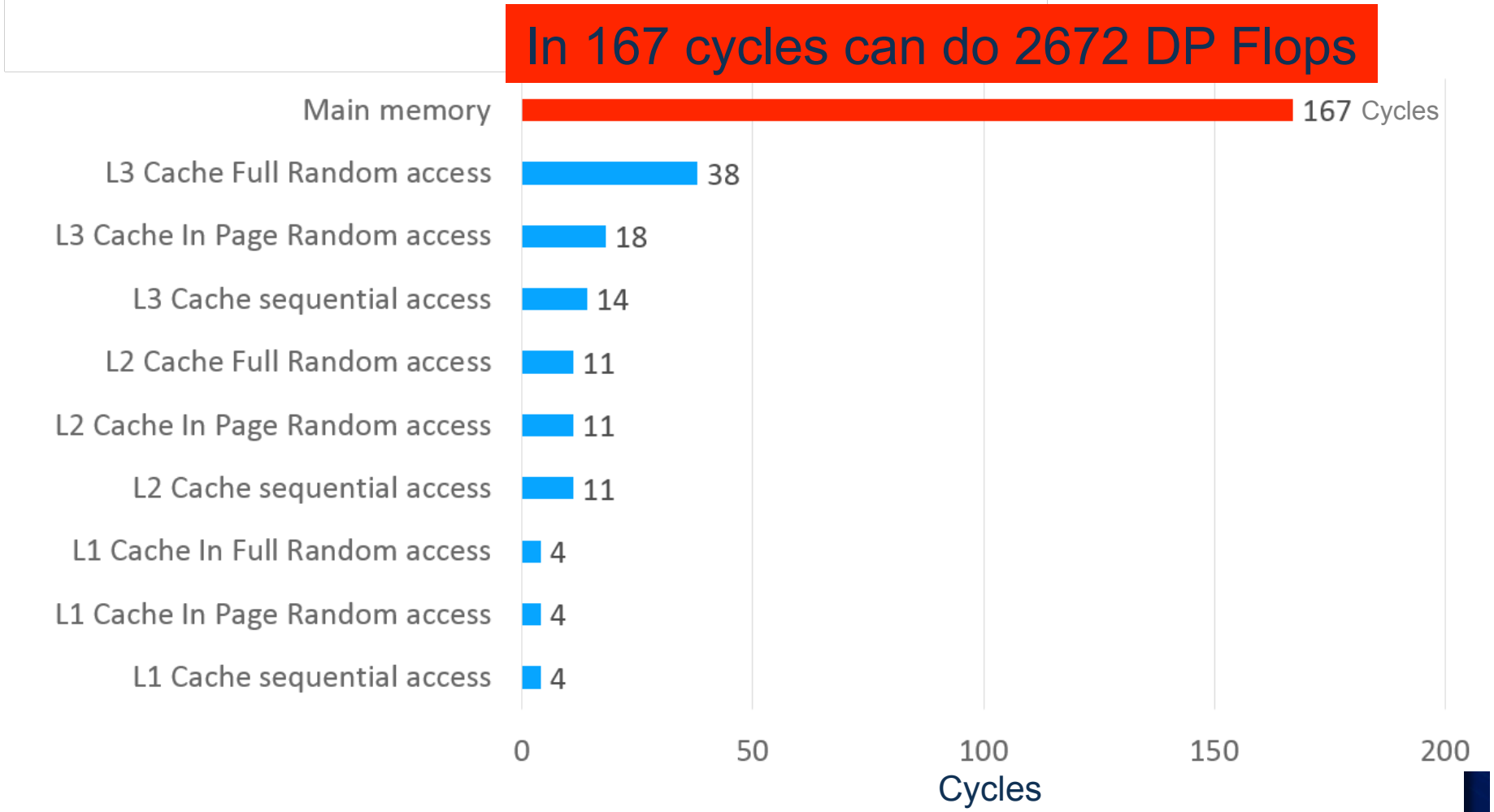
- Most of the recent computers have FMA (Fused multiple add): (i.e. x ←x + y*z in one cycle)
- Intel Xeon earlier models and AMD Opteron have SSE2
  - 2 flops/cycle DP & 4 flops/cycle SP
- Intel Xeon Nehalem ('09) & Westmere ('10) have SSE4
  - 4 flops/cycle DP & 8 flops/cycle SP
- Intel Xeon Sandy Bridge('11) & Ivy Bridge ('12) have AVX
  - 8 flops/cycle DP & 16 flops/cycle SP

We are here

- Intel Xeon Haswell ('13) & (Broadwell ('14)) AVX2
  - 16 flops/cycle DP & 32 flops/cycle SP
  - Xeon Phi (per core) is at 16 flops/cycle DP & 32 flops/cycle SP
- Intel Xeon Skylake ('15)
  - 32 flops/cycle DL & 64 flops/cycle SP



CPU Access Latencies in Clock Cycles



Classical Analysis of Algorithms May Not be Valid

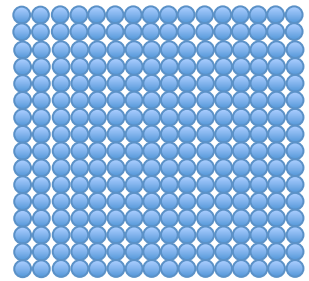
- Processors over provisioned for floating point arithmetic
- Data movement extremely expensive
- Operation count is not a good indicator of the time to solve a problem.
- Algorithms that do more ops may actually take less time.

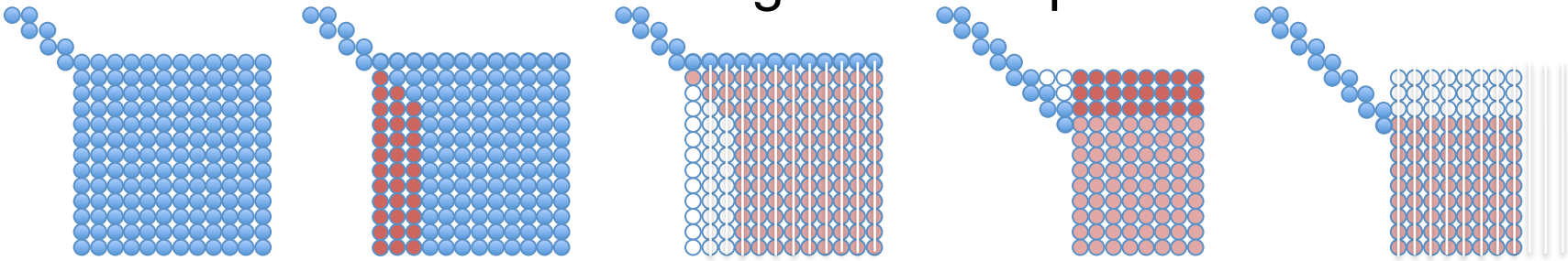
Singular Value Decomposition

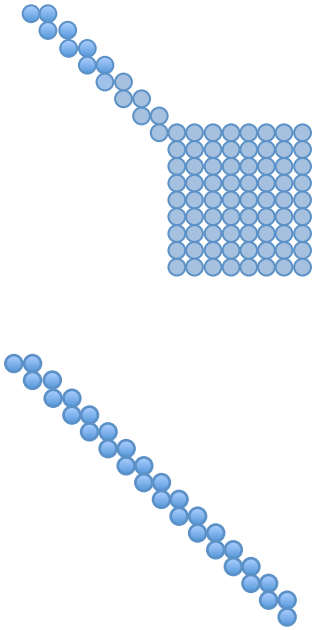
LAPACK Version 1991

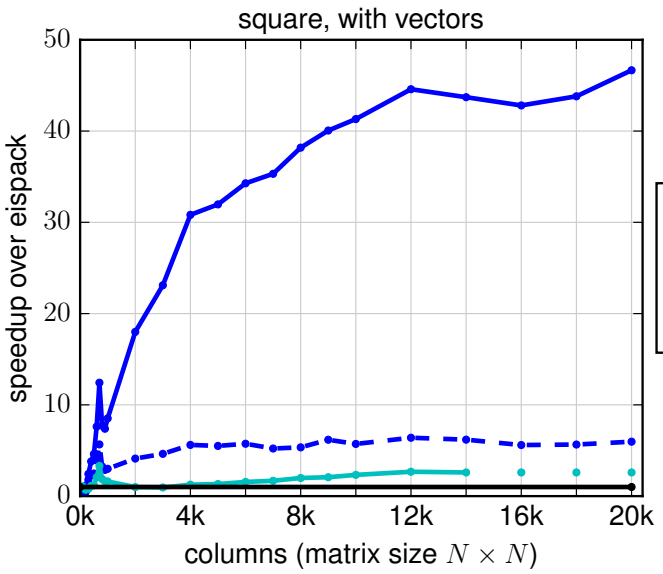
Level 1, 2, & 3 BLAS

First Stage $8/3\ n^3$ Ops

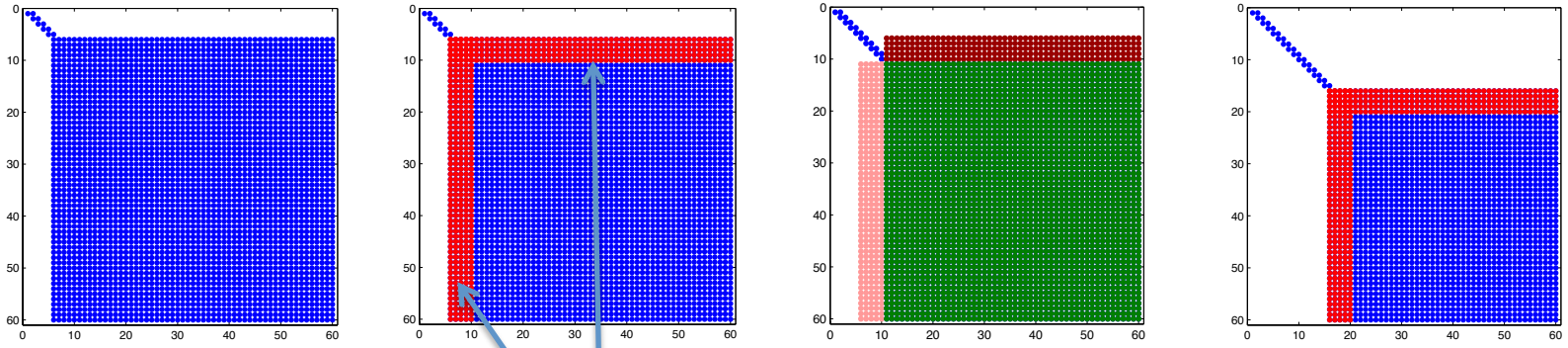








Bottleneck in the Bidiagonalization
The Standard Bidiagonal Reduction: xGEBRD
Two Steps: Factor Panel & Update Tailing Matrix



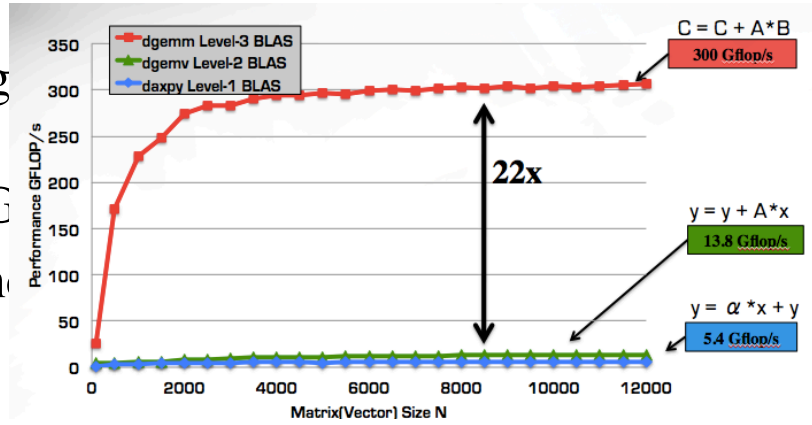
factor panel k then update ➔ **factor panel k+1**

Requires 2 GEMVs

$Q^*A^*P^H$

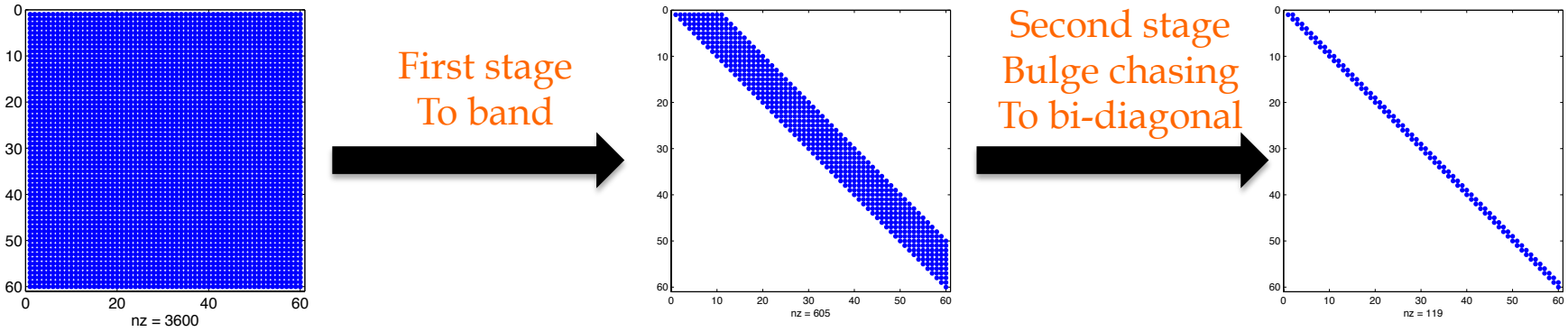
✸ Characteristics

- Total cost $8n^3/3$, (reduction to bi-diag
- Too many Level 2 BLAS operations
- $4/3\ n^3$ from GEMV and $4/3\ n^3$ from G
- Performance limited to 2* performan
- ➔**Memory bound algorithm.**



16 cores Intel Sandy Bridge, 2.6 GHz, 20 MB shared L3 cache.
The theoretical peak per core double precision is 20.4 Gflop/s per core.
Compiled with icc and using MKL 2015.3.187

Recent Work on 2-Stage Algorithm

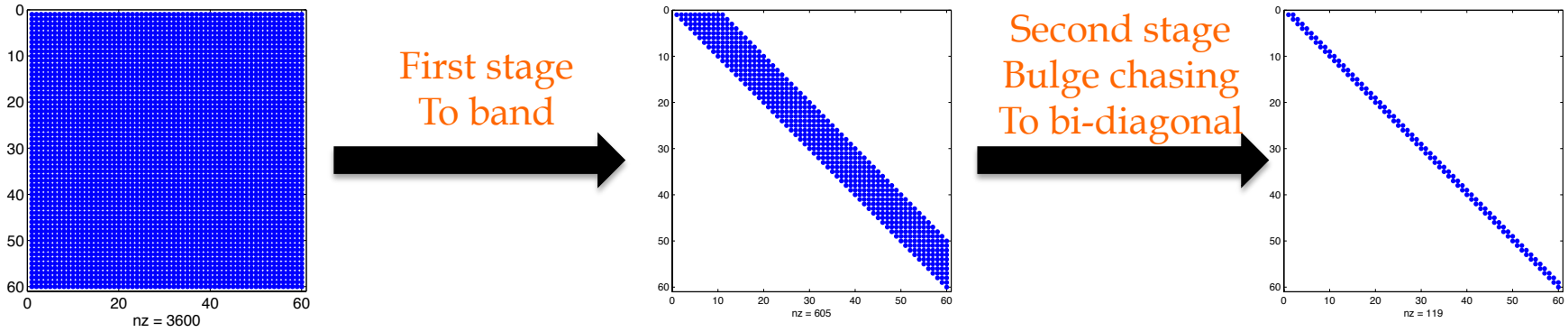


✸ Characteristics

- **Stage 1:**
  - Fully Level 3 BLAS
  - Dataflow Asynchronous execution

- **Stage 2:**
  - Level "BLAS-1.5"
  - Asynchronous execution
  - Cache friendly kernel (reduced communication)

Recent work on developing new 2-stage algorithm



$$\text{flops} \approx \sum_{s=1}^{\frac{n-n_b}{n_b}} 2n_b^3 + (nt-s)3n_b^3 + (nt-s)\frac{10}{3}n_b^3 + (nt-s)\times(nt-s)5n_b^3$$

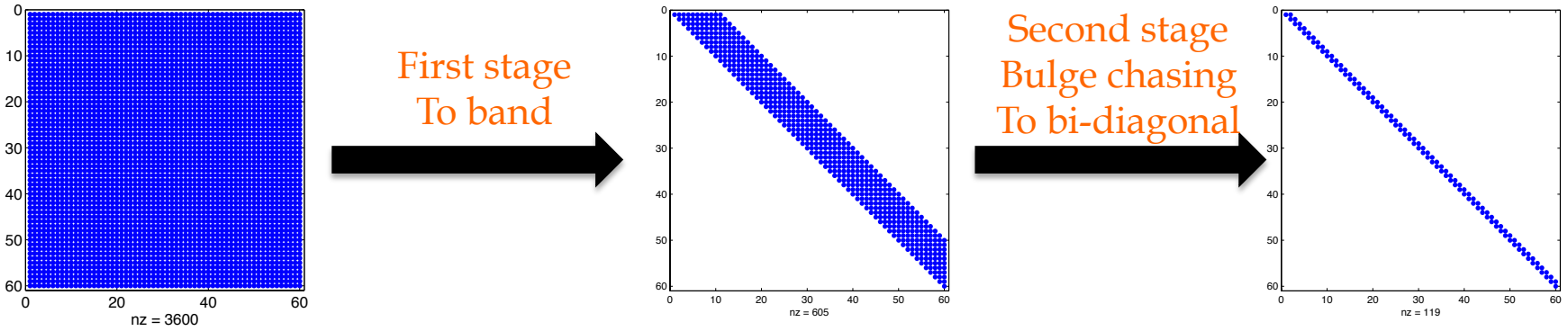
$$+ \sum_{s=1}^{\frac{n-n_b}{n_b}} 2n_b^3 + (nt-s-1)3n_b^3 + (nt-s-1)\frac{10}{3}n_b^3 + (nt-s)\times(nt-s-1)5n_b^3$$

$$\approx \frac{10}{3}n^3 + \frac{10n_b}{3}n^2 + \frac{2n_b}{3}n^3$$

$$\approx \frac{10}{3}n^3(\text{gemm})_{\text{first stage}} \qquad \text{flops} = 6\times n_b \times n^2(\text{gemv})_{\text{second stage}}$$

More Flops, original did 8/3 n ^{3}

Recent work on developing new 2-stage algorithm

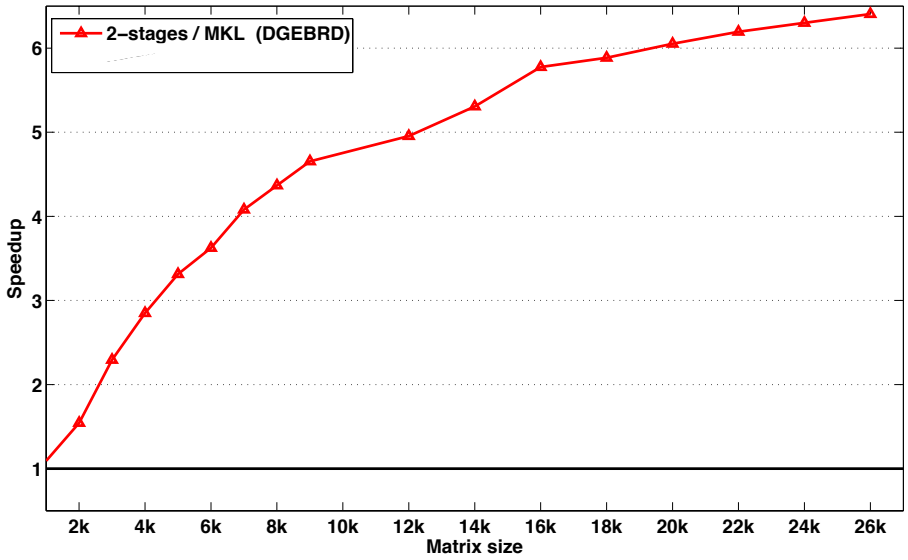


$$\text{speedup} = \frac{\text{time of one-stage}}{\text{time of two-stage}}$$

$$= \frac{4n^3/3P_{gemv} + 4n^3/3P_{gemm}}{10n^3/3P_{gemm} + 6n_b n^2/P_{gemv}}$$

$$\Longrightarrow \frac{84}{70} \leq \text{Speedup} \leq \frac{84}{15}$$

$$\Longrightarrow 1.8 \leq \text{Speedup} \leq 7$$

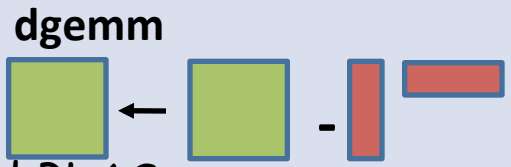


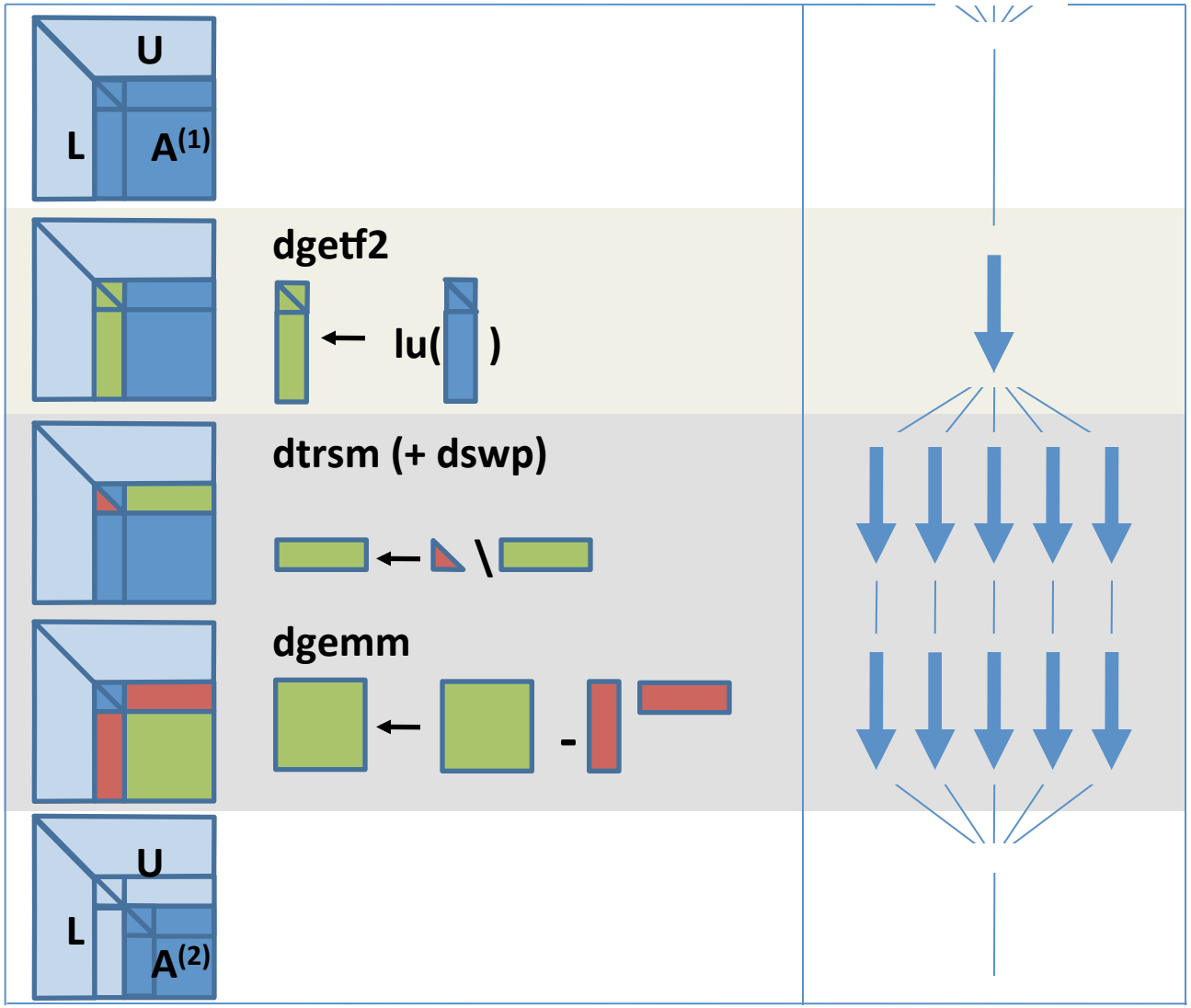
if P_{gemm} is about 22x P_{gemv} and $120 \leq n_b \leq 240$.

Parallelization of LU and QR.

Parallelize the update:

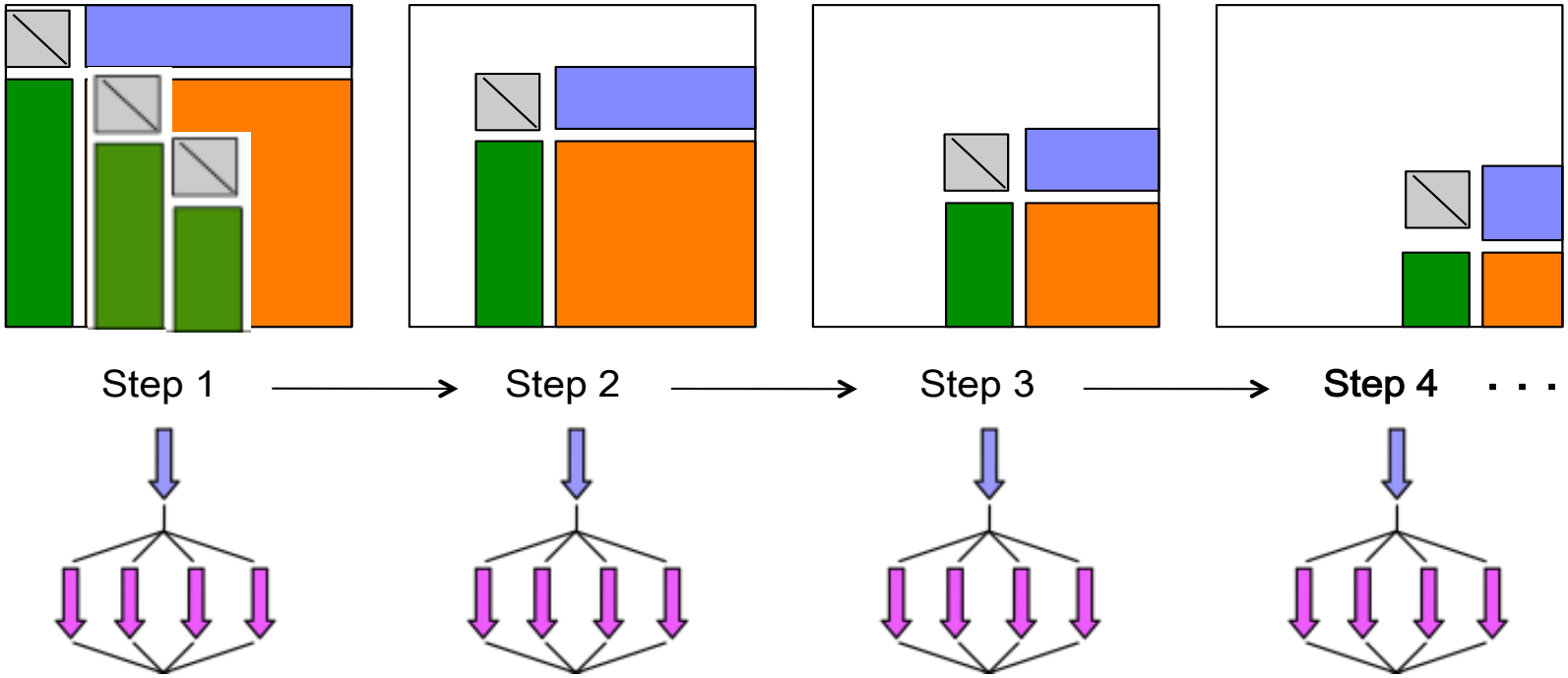
- Easy and done in any reasonable software.
- This is the $2/3n^3$ term in the FLOPs count.
- Can be done efficiently with LAPACK+multithreaded BLAS

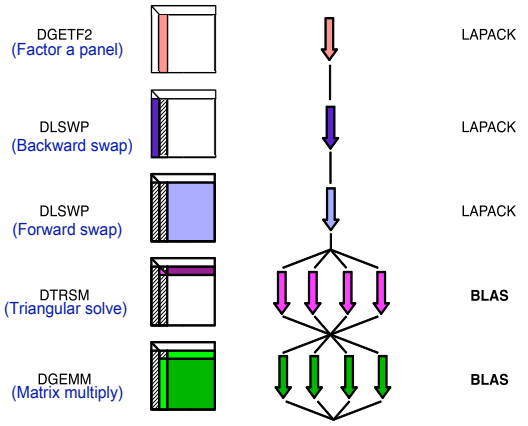




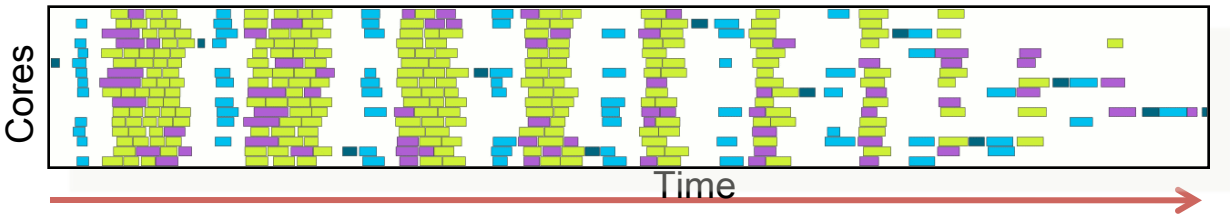
Fork - Join parallelism
Bulk Sync Processing

Synchronization (in LAPACK LU)





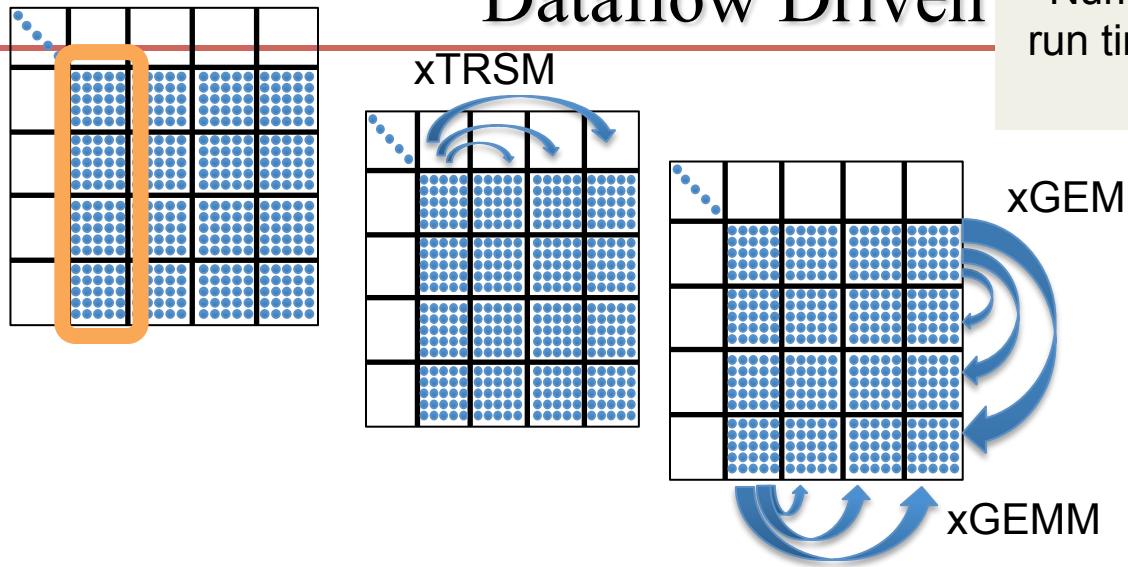
- fork join
- bulk synchronous processing



PLASMA LU Factorization

Dataflow Driven

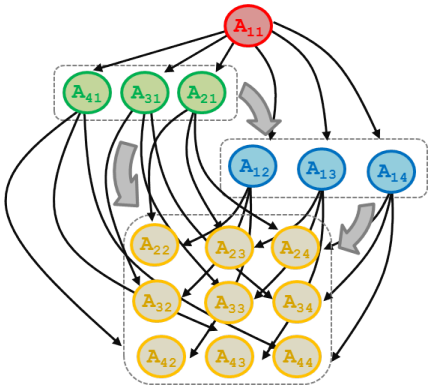
Numerical program generates tasks and run time system executes tasks respecting data dependences.



Sparse / Dense Matrix System

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}$$

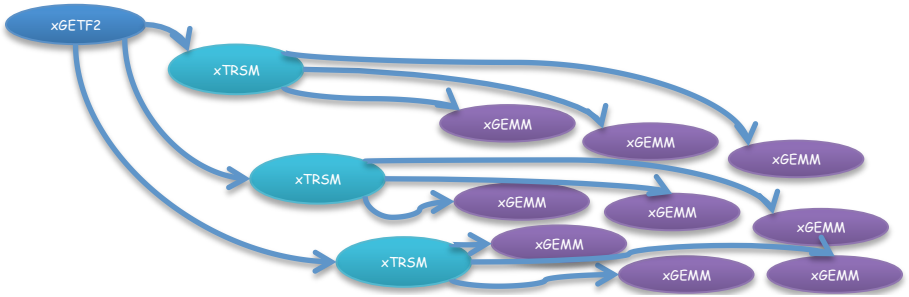
DAG-based factorization



Batched LA

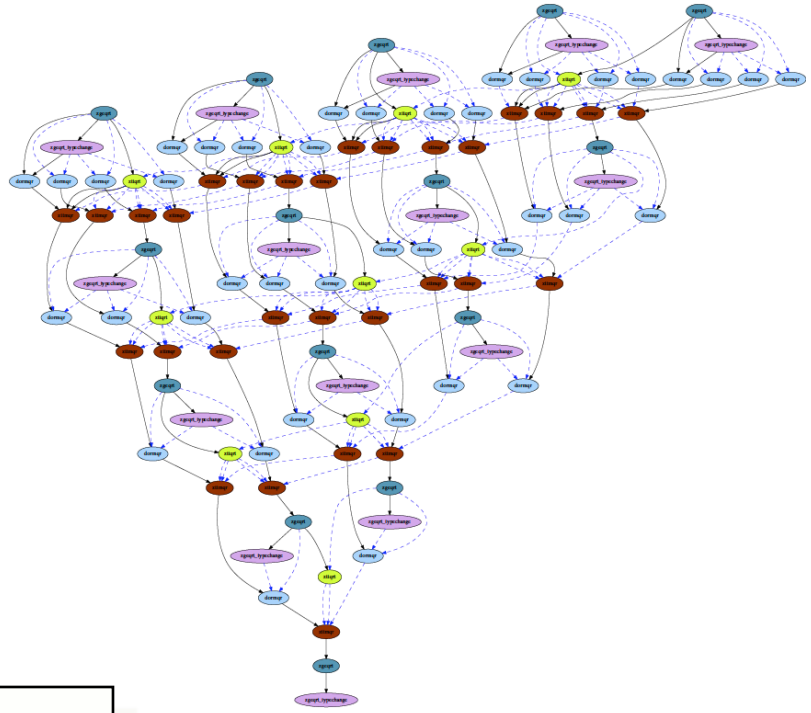
- **LU, QR,** or **Cholesky** on small diagonal matrices
- **TRSM**s, **QR**s, or **LU**s
- **TRSM**s, **TRMM**s
- **Updates (Schur complement) GEMM**s, **SYRK**s, **TRMM**s

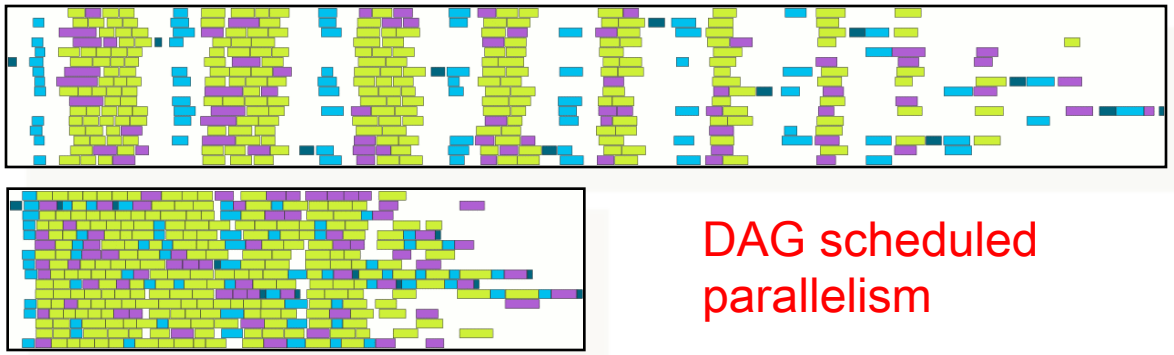
And many other BLAS/LAPACK, e.g., for application specific solvers, preconditioners, and matrices



Dataflow Based Design

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





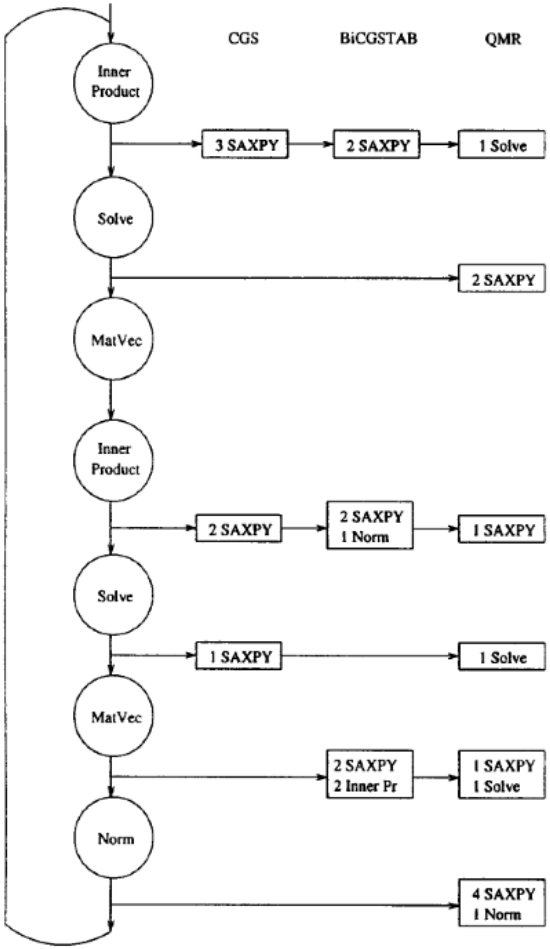
DAG scheduled parallelism

Fork-join parallelism
Notice the synchronization penalty in the presence of heterogeneity.

Algorithmic Bombardment

- Interesting in solving a large sparse non-symmetric system
- system on a parallel computer.
- Many algorithms to choose
  - CG Square
  - Bi CG Stabilized
  - QMR
  - GMRES
  - etc.
- Best choice not clear
- Some may not converge
- Develop a “poly-algorithm” combining multiple algorithms in one, leveraging data movement.

Poly-algorithm
Iterating in Lock Step



Amount of work per iteration

Method	$\alpha = x^T y$	$y = \alpha x + y$	$y = Ax$
CGS	2	6	2
Bi-CGSTAB	4	6	2
QMR	2	12	2

Number of Communications per iteration

Method	$\alpha = x^T y$	$y = \alpha x + y$	$y = Ax$	$x = M^{-1} y$	Storage
CGS	2	0	2	2	A + 6n
Bi-CGSTAB	3	0	2	2	A + 6n
QMR	2	0	2	2	A + 16n
Bombardment	3	0	2	3	A + 26n

Avoiding Synchronization

- "Responsibly Reckless" Algorithms
  - Try fast algorithm (unstable algorithm) that might fail (but rarely)
  - Check for instability
  - If needed, recompute with stable algorithm

Avoid Pivoting; Minimize Synchronization

- To solve Ax = b :
  - Compute $A_r = U^T A V$, with U and V random matrices
  - Factorize A_r without pivoting (GENP)
  - Solve $A_r\, y = U^T b$ and then Solve $x = Vy$
- U and V are Recursive Butterfly Matrices
  - Randomization is cheap (O(n^2) operations)
  - GENP is fast (can take advantage of the GPU)
  - Accuracy is in practice similar to GEPP (with iterative refinement), but…

Think of this as a preconditioner step.

Goal: Transform A into a matrix that would be sufficiently "random" so that, with a probability close to 1, pivoting is not needed.

A **butterfly matrix** is defined as any *n*-by-*n* matrix of the form:

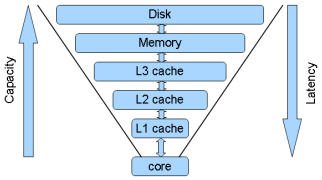
$$B = \frac{1}{\sqrt{2}} \begin{pmatrix} R & S \\ R & -S \end{pmatrix}$$

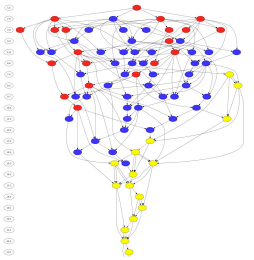
where *R* and *S* are random diagonal matrices.

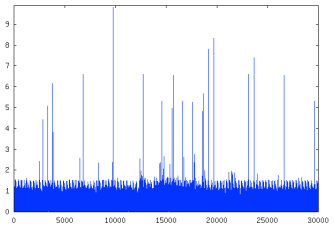
$$B = \begin{pmatrix} \diagdown & \diagup \\ \diagdown & \diagup \end{pmatrix}$$

Software and Algorithm Must Keep Pace with the Changes in Hardware

- Classical analysis of algorithms is not valid,
  - # of floating point ops ≠ computation time.
- Algorithms and software must take advantage by reducing data movement.
  - Need latency tolerance in our algorithms
- Communication and synchronization reducing algorithms and software are critical.
  - As parallelism grows
- Hardware presents a dynamically changing environment
  - Turbo Boost and OS jitter
- Many existing algorithms can't fully exploit the features of modern architecture







Summary

- **Major Challenges are ahead for extreme computing**
  - **Parallelism O(10^9)**
    - Programming issues
  - **Hybrid**
    - Peak and HPL may be very misleading
    - No where near close to peak for most apps
  - **Fault Tolerance**
    - Today Sequoia BG/Q node failure rate is 1.25 failures/day
  - **Power**
    - 50 Gflops/w (today at 2 Gflops/w)
- **We will need completely new approaches and technologies to reach the Exascale level**