samsi
NSF Duke NCSU UNC

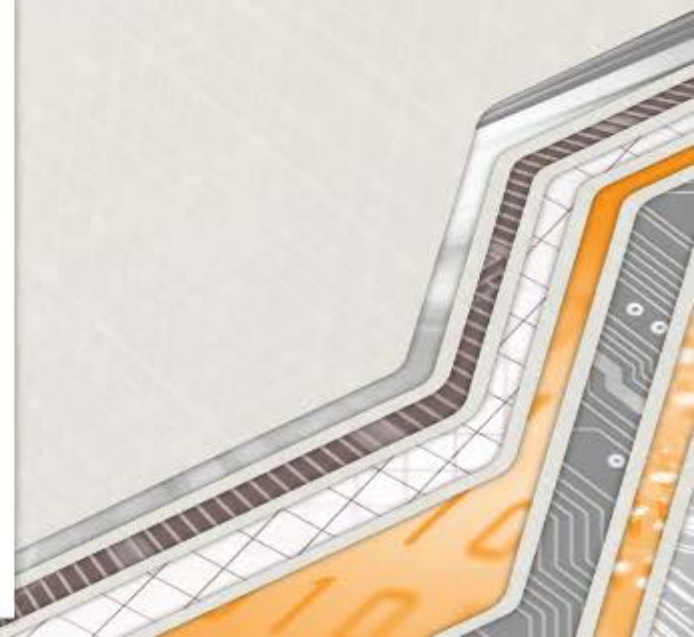
Workshop on the Interface of Statistics and Optimization (WISO)

The Road to Exascale and Legacy Software for Dense Linear Algebra

Jack Dongarra
University of Tennessee
Oak Ridge National Lab
University of Manchester

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE



samsi
NSF Duke NCSU UNC

Workshop on the Interface of Statistics and Optimization (WISO)

The Road to Exascale and Legacy Software for Dense Linear Algebra

(or What I've Been Doing for the Last 43 Years)

Jack Dongarra
University of Tennessee
Oak Ridge National Lab
University of Manchester

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Outline for the Talk

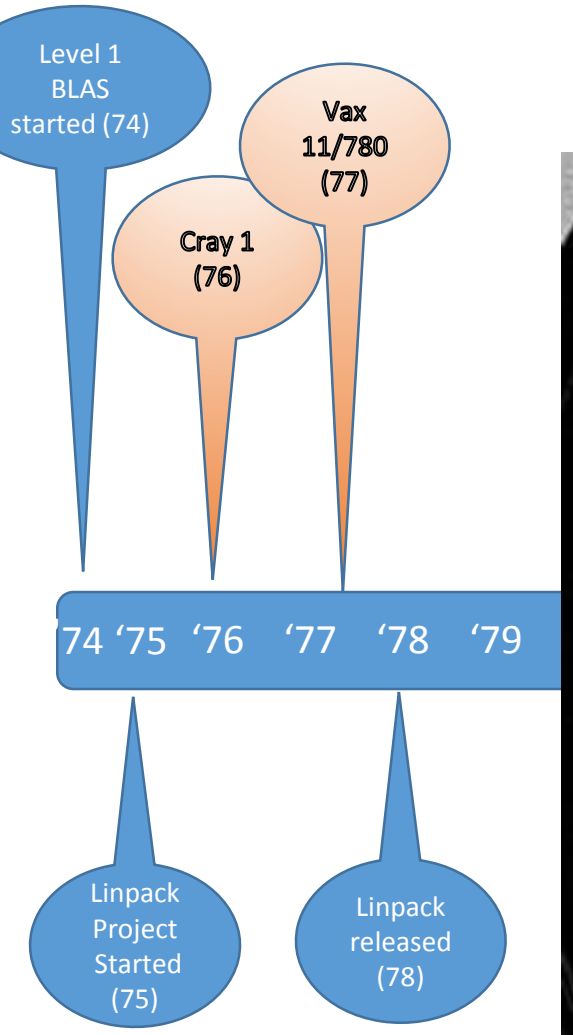
- What was going on before
- What’s the current situation
- What’s planned for exascale

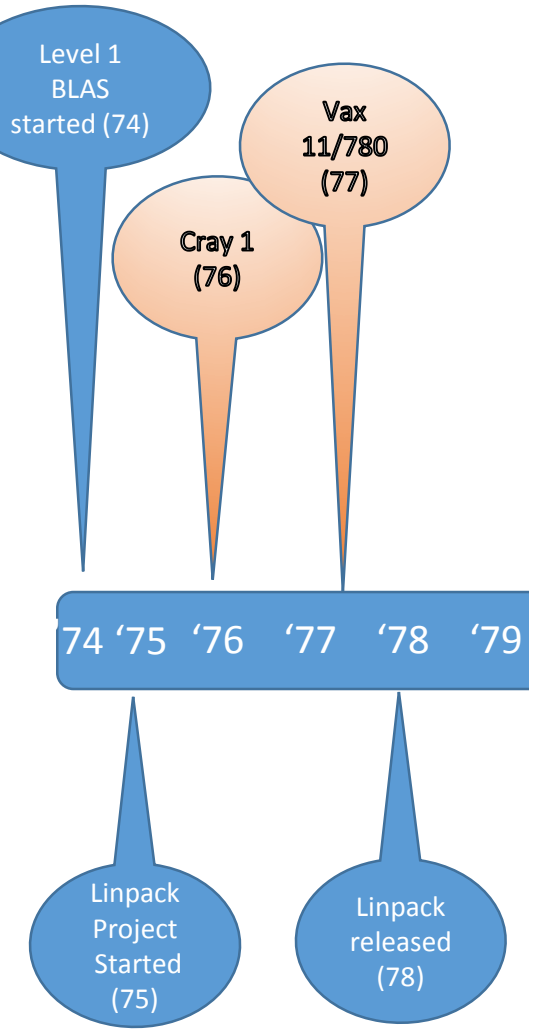
Software for Linear Algebra Targeting Exascale
Focused on Dense Linear Algebra Problems

- Linear systems of equations $Ax = b$
- Linear least squares $\min \| b - Ax \|_2$
- Singular value decomposition (SVD) $A = U\Sigma V^T$
- Eigenvalue value problems (EVP) $Ax = \lambda x$

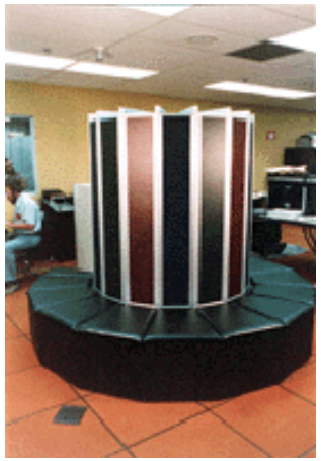
- Dense (square, rectangular)
- Band

But first, let’s go back in time.


Level 1 BLAS started (74)
Cray 1 (76)
Vax 11/780 (77)
'74 '75 '76 '77 '78 '79
Linpack Project Started (75)
Linpack released (78)



- 1974: Effort to standardize Basic Linear Algebra Subprograms
  - Basic LA vector operations
  - Referred to now as Level 1 BLAS
- 1975: LINPACK Project started
  - Effort to produce portable, efficient linear algebra software for dense matrix computations.
- 1976: Vector computers in use
- 1977: DEC VAX system in common use



ACM SIGNUM Newsletter
Volume 8 Issue 4, October 1973

Published in:
· Newsletter
ACM SIGNUM Newsletter archive
ACM New York, NY, USA
table of contents ISSN:0163-5778

IMPROVING THE EFFICIENCY OF PORTABLE SOFTWARE FOR LINEAR ALGEBRA

R. J. Hanson
(Washington State Univ.)
F. T. Krogh
(Jet Propulsion Lab)
C. L. Lawson
(Jet Propulsion Lab)

In algorithms for linear algebraic computations there are a fairly small number of basic operations which are generally responsible for a significant

As a Result of LINPACK, I Became an Accidental Benchmarker

- Appendix B of the Linpack Users' Guide
  - Designed to help users extrapolate execution Linpack software package

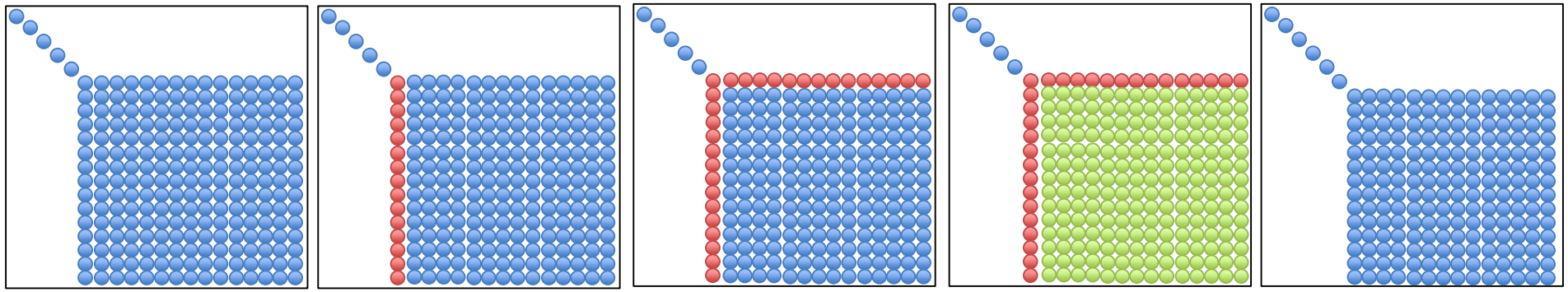
MATLAB Trivia question: What does the function dongarra(i) do?

/(1/3 100**3 + 100**2)

Facility	TIME N=100 secs.	UNIT micro-secs.	Computer	Type	Compiler
NCAR	.049	0.14	CRAY-1	S	CFT, Assembly BLAS
LASL	.148	0.43	CDC 7600	S	FTN, Assembly BLAS
NCAR	.192	0.56	CRAY-1	S	CFT
LASL	.210	0.61	CDC 7600	S	FTN
Argonne	.297	0.86	IBM 370/195	D	H
NCAR	.359	1.05	CDC 7600	S	Local
Argonne	.388	1.33	IBM 3033	D	H
NASA Langley	.489	1.42	CDC Cyber 175	S	FTN
U. Ill. Urbana	.506	1.47	CDC Cyber 175	S	Ext. 4.6
LLL	.554	1.61	CDC 7600	S	CHAT, No optimize
SLAC	.579	1.69	IBM 370/168	D	H Ext., Fast mult.
Michigan	.631	1.84	Amdahl 470/V6	D	H
Toronto	.890	2.59	IBM 370/165	D	H Ext., Fast mult.
Northwestern	1.44	4.20	CDC 6600	S	FTN
Texas	1.93*	5.63	CDC 6600	S	RUN
China Lake	1.95*	5.69	Univac 1110	S	V
Yale	2.59	7.53	DEC KL-20	S	F20
Bell Labs	3.46	10.1	Honeywell 6080	S	Y
Wisconsin	3.49	10.1	Univac 1110	S	V
Iowa State	3.54	10.2	Itel AS/5 mod3	D	H
U. Ill. Chicago	4.10	11.9	IBM 370/158	D	G1
Purdue	5.69	16.6	CDC 6500	S	FUN
U. C. San Diego	13.1	38.2	Burroughs 6700	S	H
Yale	17.1*	49.9	DEC KA-10	S	F40

* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGESL(75)

The Standard Factorization LINPACK 1970's HPC of the Day: Vector Architecture



Factor column with Level 1 BLAS

Divide by Pivot row

Schur complement update (Rank 1 update)

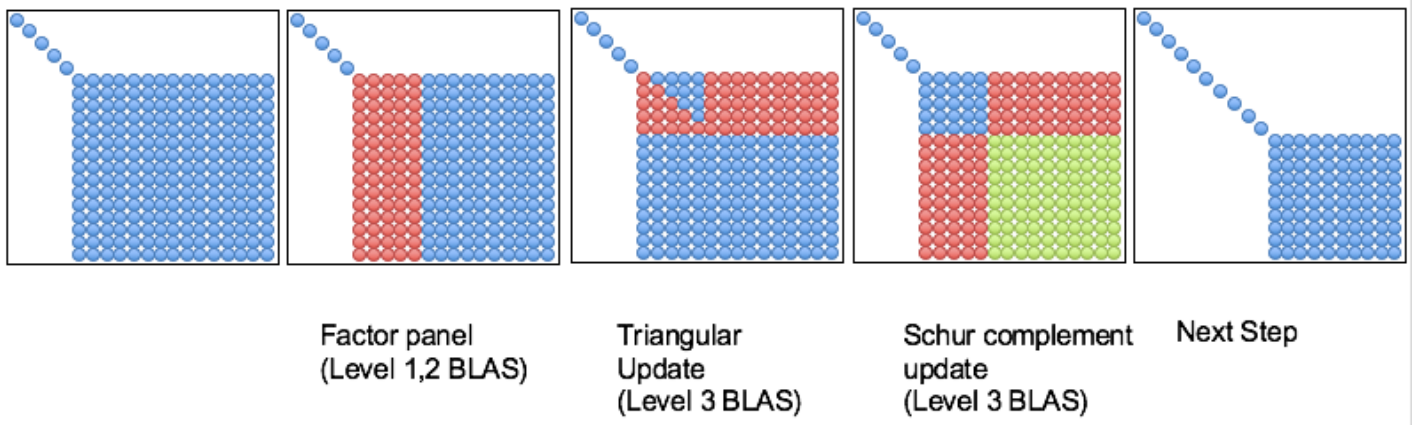
Next Step

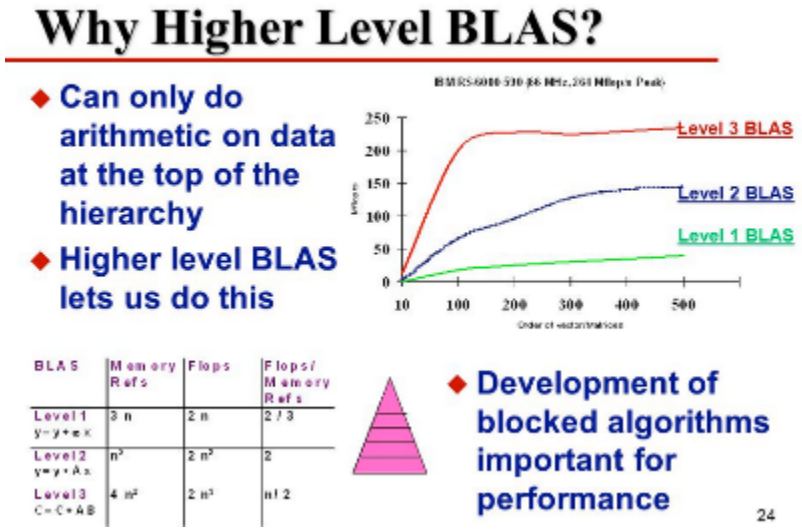
Main points

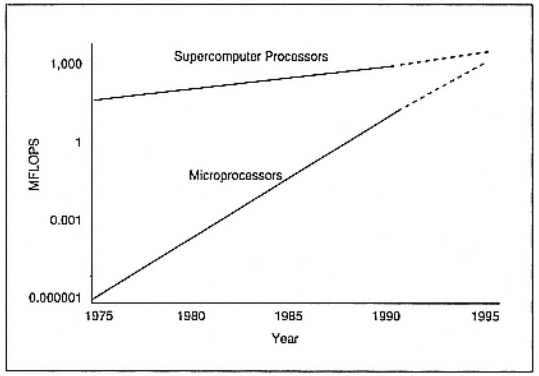
- Factorization column (zero) mostly sequential due to memory bottleneck
- Level 1 BLAS
- Divide pivot row has little parallelism
- OK on machines with excess memory bandwidth, but
- Too much data movement per step

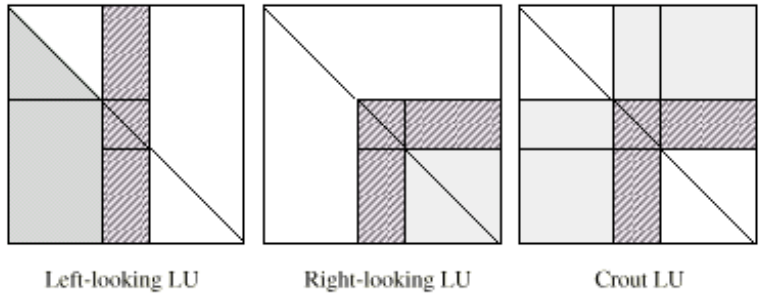
1984 - 1990

- Level 3 BLAS standardization started
- Level 2 BLAS standard published
- “Attack of the Killer Micros”, Brooks @ SC90
- Cache based & SMP machines
- Blocked partitioned algorithms was the way to go
  - Reduce data movement; today’s buzzword “Communication avoiding”









Level 1 BLAS started (74)

Cray 1 (76)

Vax 11/780 (77)

Level 1 BLAS Published (79)

Netlib MathWorks Started (84)

IEEE 754 standard (85)

Blocked Partitioned Algorithms (89)

MPI started (91)

'74 '75 '76 '77 '78 '79 '80 '81 '82 '83 '84 '85 '86 '87 '88 '89 '90 '91 '92 '93 '94 '95 '96 '97 '98 '99

Linpack Project Started (75)

Linpack released (78)

Unrolling Loops Paper (79)

Unrolling Loops Outer-level (83)

IJK Paper Level 2 BLAS started EISPACK3 (84)

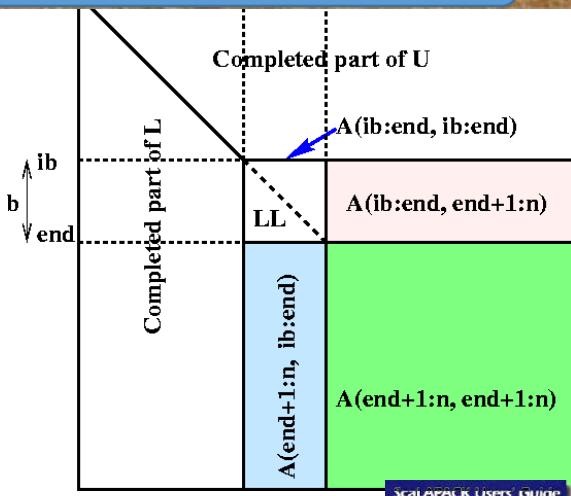
Level 3 BLAS started LAPACK started (87)

Level 2 BLAS Publish (88)

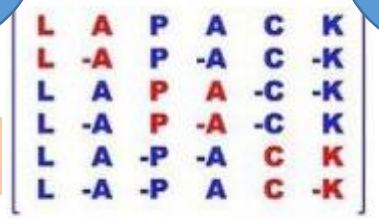
Level 3 BLAS Publish (90)

LAPACK Publish (92)

ScaLAPACK started (93)



What the inverse of the LAPACK matrix?



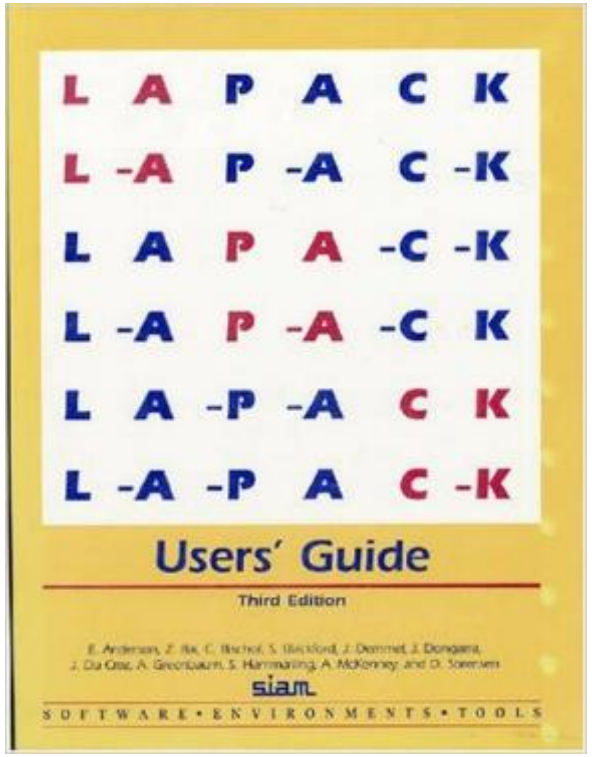
- LAPACK Published
- ScaLAPACK started

LAPACK Functionality

Type of Problem	Acronyms
Linear system of equations	SV
Linear least squares problems	LLS
Linear equality-constrained least squares problem	LSE
General linear model problem	GLM
Symmetric eigenproblems	SEP
Nonsymmetric eigenproblems	NEP
Singular value decomposition	SVD
Generalized symmetric definite eigenproblems	GSEP
Generalized nonsymmetric eigenproblems	GNEP
Generalized (or quotient) singular value decomposition	GSVD (QSVD)

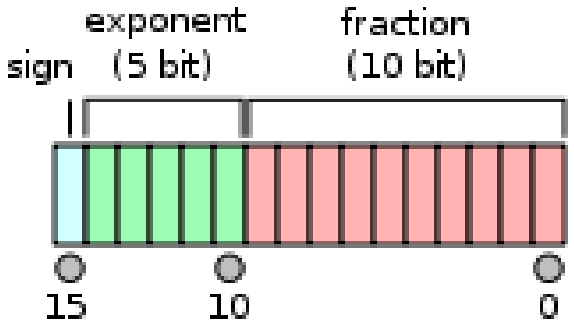
LAPACK Software

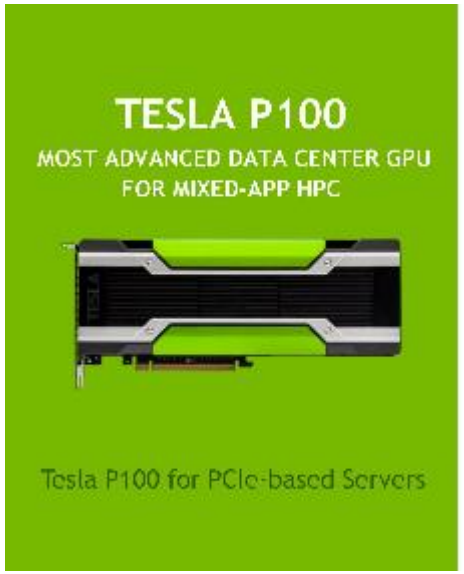
Jointly with UTK and UCB and Many Other Contributors



- First release in February 1992 (Silver Anniversary)
- Current: LAPACK Version 3.7.0 (Dec, 2017) ~2M LoC
- Public GITHub repository
- **4 Precisions:** single, double, complex, double complex
  - *Considering 16-bit flpt version*
- **Multi-OS** *nix, Mac OS/X, Windows
- **Multi-build** support (Make and Cmake)
- **Used by MATLAB, R, Intel, Cray, Fujitsu, NEC,…**
- **LAPACKE: Standard C language APIs for LAPACK**
- Prebuilt Libraries for **Windows**
- Extensive test suite
- **LICENSE**: Mod-BSD, freely-available software package - Thus, it can be included in commercial software packages (and has been). We only ask that proper credit be given to the authors.
- **Forum and User support**: http://icl.cs.utk.edu/lapack-forum/
- Goal: bug free library – Since 2009, 165 bugs reported, only 11 pending correction

IEEE 754 Half Precision Floating Point Standard





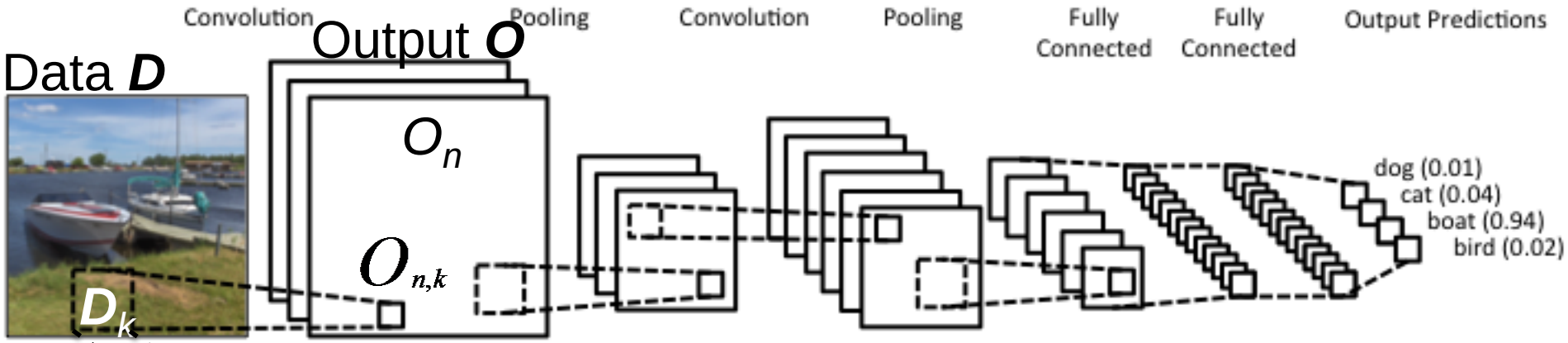
AMD

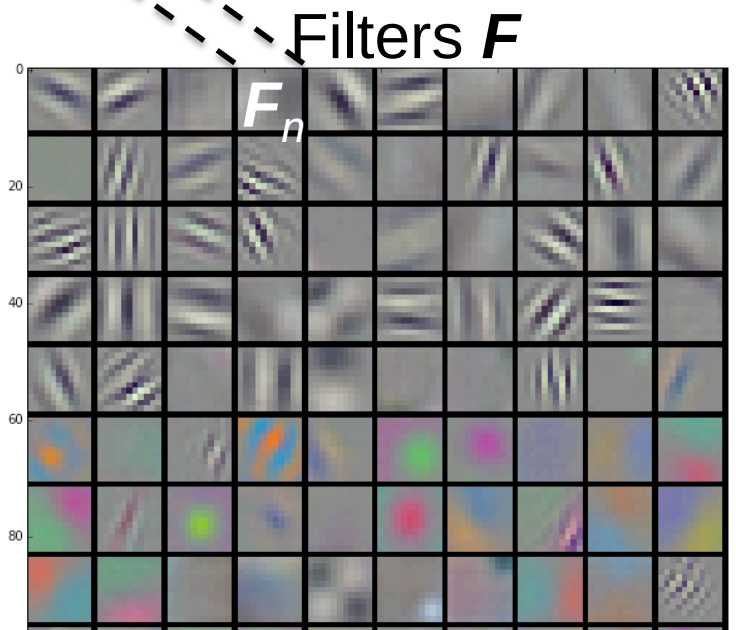
AMD Radeon Instinct			
	Instinct MI6	Instinct MI8	Instinct MI25
Memory Type	16GB GDDR5	4GB HBM	"High Bandwidth Cache and Controller"
Memory Bandwidth	224GB/sec	512GB/sec	?
Single Precision (FP32)	5.7 TFLOPS	8.2 TFLOPS	12.5 TFLOPS
Half Precision (FP16)	5.7 TFLOPS	8.2 TFLOPS	25 TFLOPS
TDP	<150W	<175W	<300W
Cooling	Passive	Passive (SFF)	Passive
GPU	Polaris 10	Fiji	Vega
Manufacturing Process	GloFo 14nm	TSMC 28nm	?

Machine Learning

Need of Batched and/or Tensor contraction routines in machine learning

e.g., Convolutional Neural Networks (CNNs) used in computer vision
Key computation is convolution of Filter Fi (feature detector) and input image D (data):





Convolution operation:

- For every filter F_n and every channel, the computation for every pixel value $O_{n,k}$ is a tensor contraction:

$$O_{n,k} = \sum D_{k,i} F_{n,i}$$

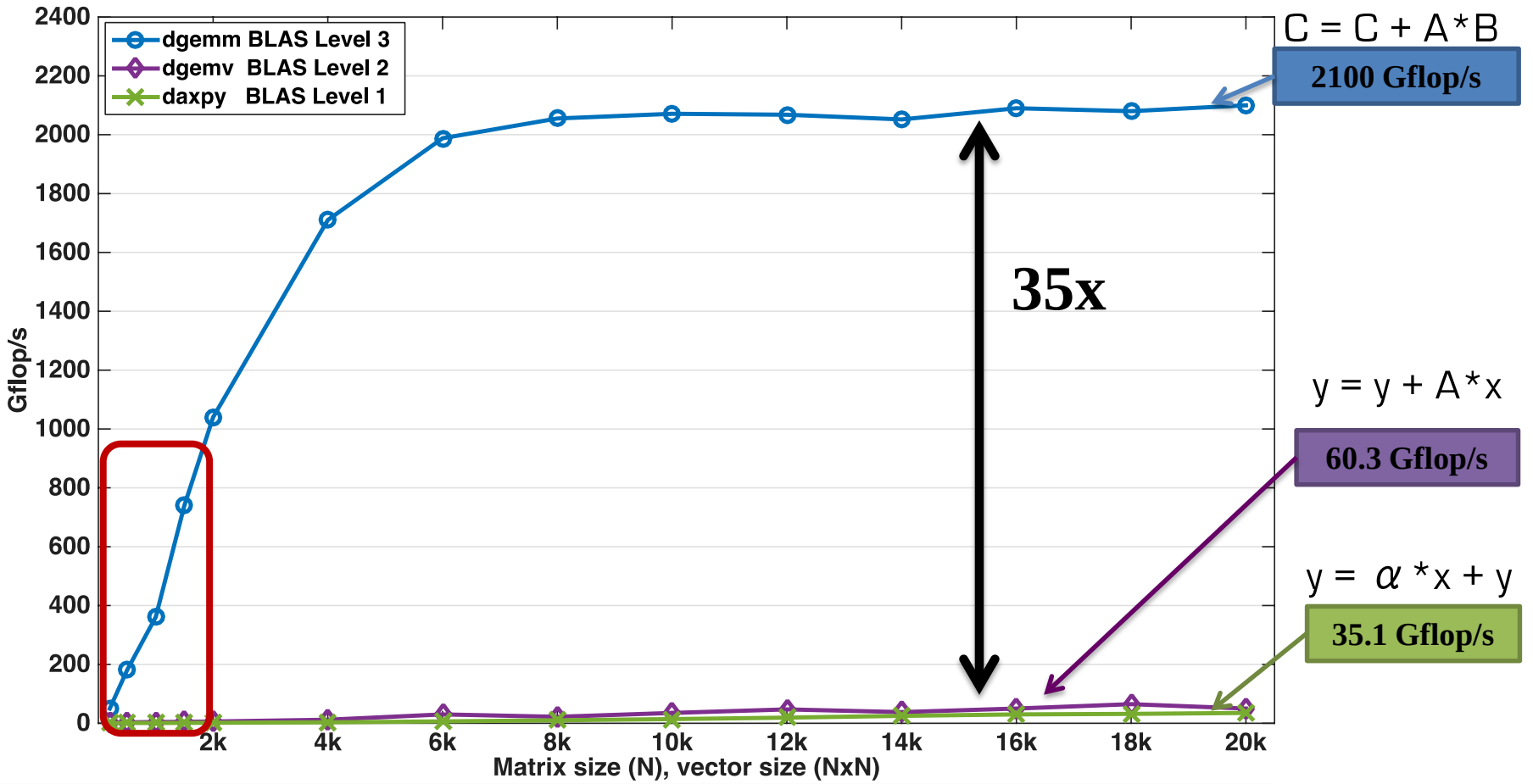
This problem may be able to get away with 16 bit floating point.
=> Some vendors are now implementing this in hardware.

API for Batching BLAS Operations

- We are proposing, as a community standard, an API for Batched Basic Linear Algebra Operations
- The focus is on multiple independent BLAS operations
  - Think "small" matrices (n<500) that are operated on in a single routine.
- Goal to be more efficient and portable for multi/manycore & accelerator systems.
- We can show 2x speedup and 3x better energy efficiency.

Level 1, 2 and 3 BLAS

68 cores **Intel Xeon Phi KNL**, 1.3 GHz, Peak DP = 2662 Gflop/s



68 cores Intel Xeon Phi KNL, 1.3 GHz
The theoretical peak double precision is 2662 Gflop/s
Compiled with icc and using Intel MKL 2017b1 20160506

Examples

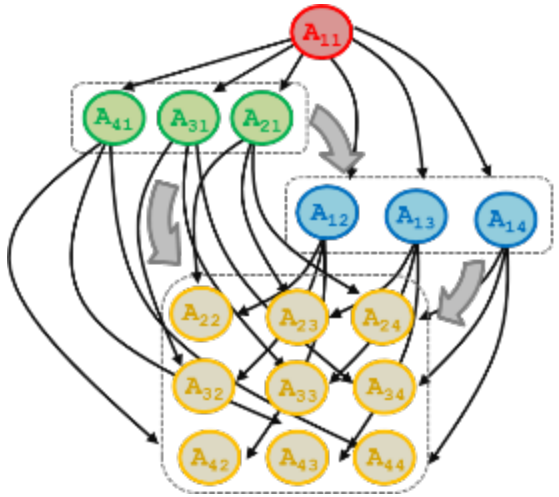
Need of Batched routines for Numerical LA

[e.g., sparse direct multifrontal methods, preconditioners for sparse iterative methods, tiled algorithms in dense linear algebra, etc.;]
[collaboration with Tim Davis at al., Texas A&M University]

Sparse / Dense Matrix System

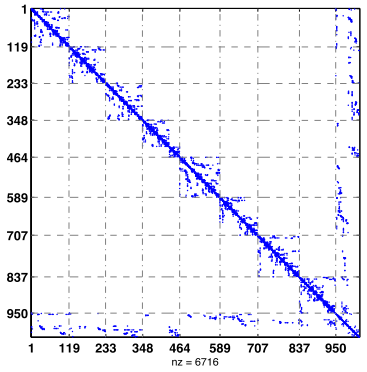
$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}$$

DAG-based factorization

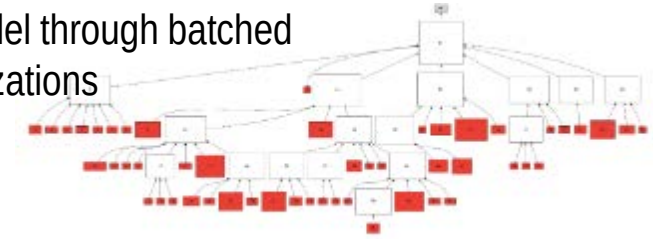


To capture main LA patterns needed in a numerical library for Batched LA

- **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



- Example matrix from Quantum chromodynamics
- Reordered and ready for sparse direct multifrontal solver
- Diagonal blocks can be handled in parallel through batched LU, QR, or Cholesky factorizations



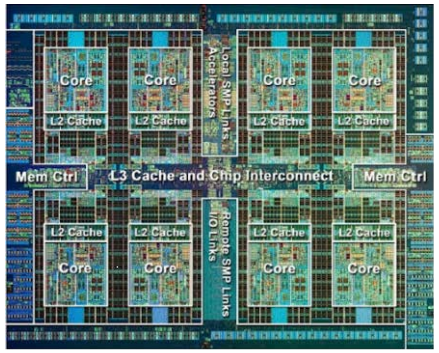
Batched Computations CPU

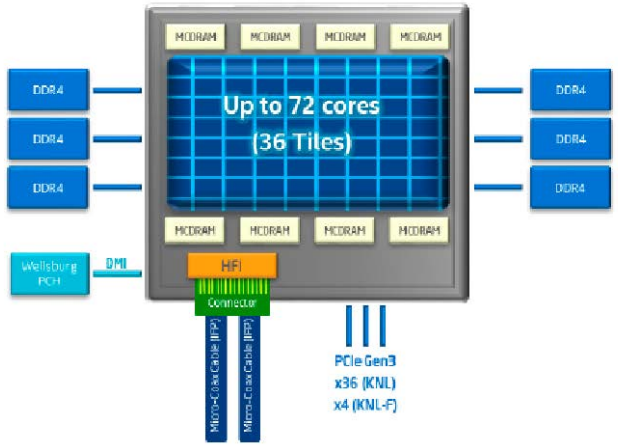
1. Non-batched computation

loop over the matrices one by one and compute either:

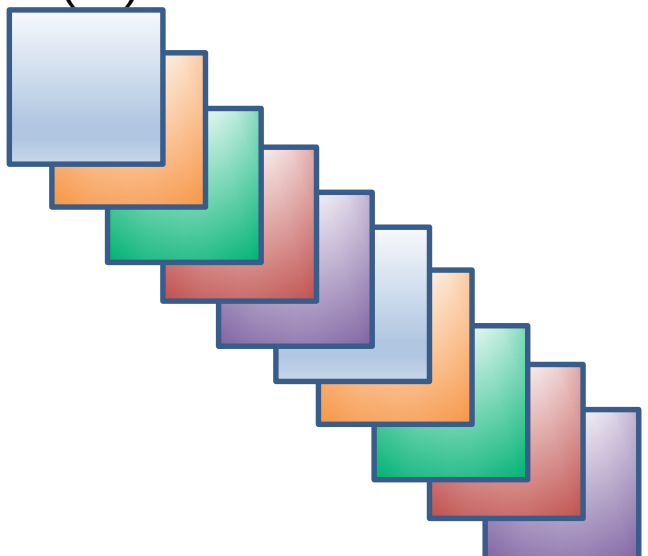
- One call for each matrix.
- Sequentially wasting all the other cores, and attaining very poor performance

- Or using multithread (note that for small matrices there is not enough work for all cores so expect low efficiency as well as threads contention can affect the performance)






```
for (i=0; i<batchount; i++)
    dgemm(…)
```

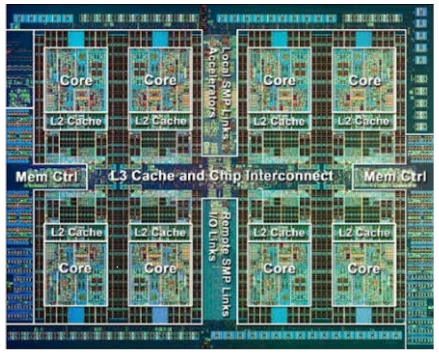


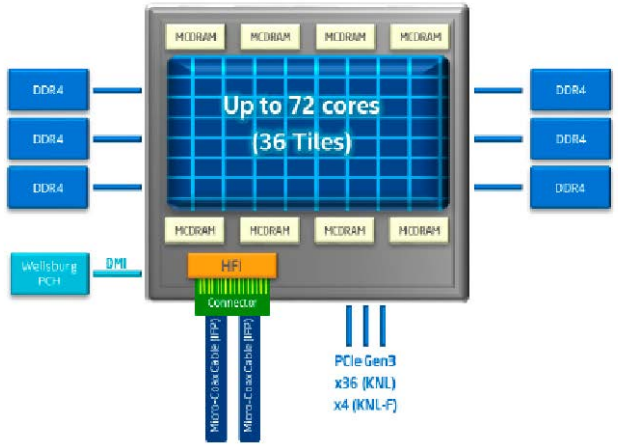
Batched Computations CPU

2. Batched computation

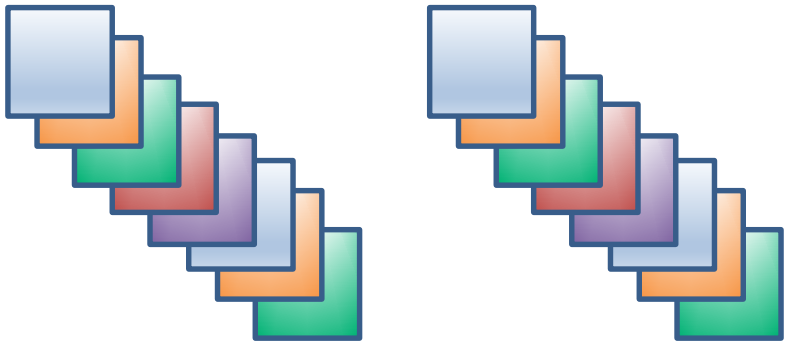
loop over the matrices and assign a matrix to each core working on it sequentially and independently

- Since matrices are very small, all the n_cores matrices will fit into L2 cache thus we do not increase L2 cache misses while performing in parallel n_cores computations reaching the best of each core



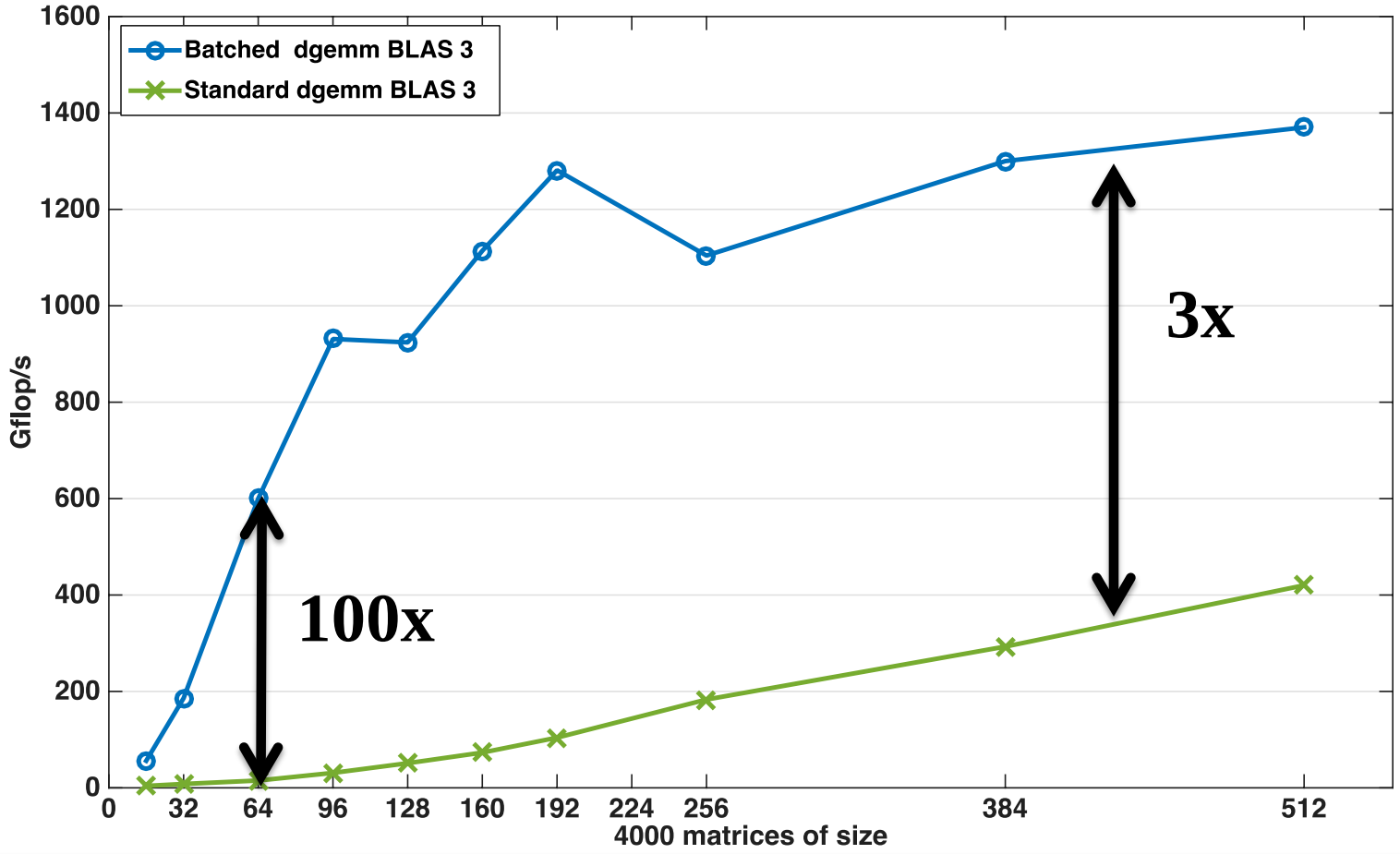



```
for (i=cpu_id; i<batchcount; i+=n_cpu)
    batched_dgemm(…)
```



Level 1, 2 and 3 BLAS

68 cores **Intel Xeon Phi KNL**, 1.3 GHz, Peak DP = 2662 Gflop/s



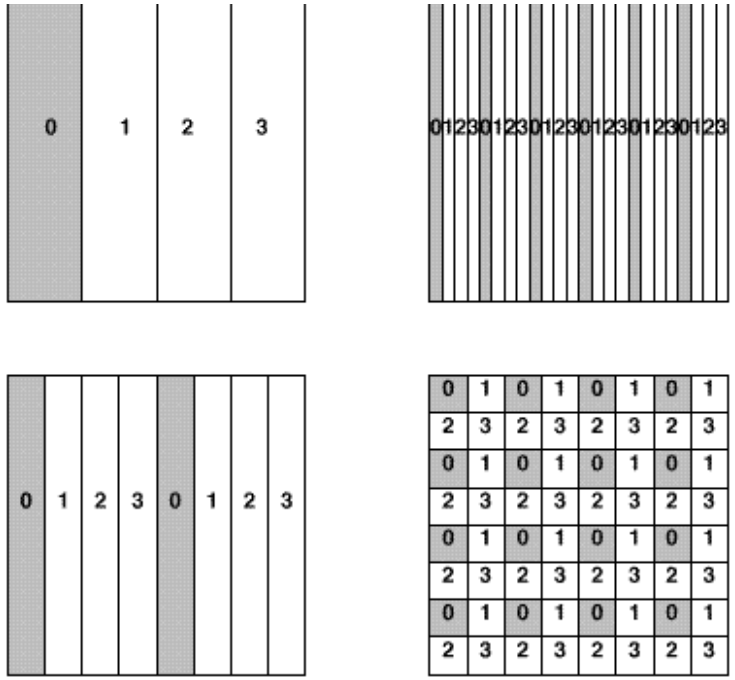
C = C + A*B

68 cores Intel Xeon Phi KNL, 1.3 GHz
The theoretical peak double precision is 2662 Gflop/s
Compiled with icc and using Intel MKL 2017b1 20160506

Next Evolution For Distributed Memory Computers

- **LAPACK ⇒ ScaLAPACK**
- Explicit message passing required
- Library of software dealing with dense & banded routines
- MPI used for message passing
- Data layout critical for performance

- Relies on LAPACK / BLAS and BLACS / MPI
- Includes PBLAS (Parallel BLAS)



Performance Issues with ScaLAPACK

- The major problem with ScaLAPACK is the lack of overlap of computation and communication .
- Each phase done separately, bulk synchronous.
  - Computation phase then a communication phase.
  - All (most) processes compute then a communication phase (broadcast)
  - This is how the PBLAS operate.
- No overlap, resulting in performance issues

- Need an “new” interface which allows computation and communication to take place simultaneously, in an asynchronous fashion.

Since LAPACK and ScaLAPACK

- A lot has changed since then…
  - Manycore and accelerators
  - Data movement very expensive
  - Use a different set of ideas to provide efficient use of underlying hardware
    - PLASMA
    - MAGMA

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
- 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 (server) AVX 512
  - 32 flops/cycle DP & 64 flops/cycle SP
  - Knight's Landing

We are here

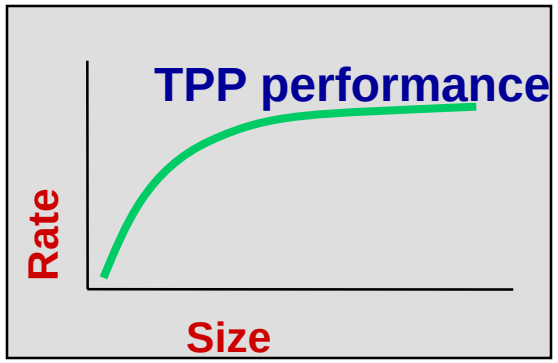
State of Supercomputing in 2017

- **Pflops (> 10^{15} Flop/s) computing fully established with 117 systems.**
- **Three technology architecture possibilities or "swim lanes" are thriving.**
  - **Commodity (e.g. Intel)**
  - **Commodity + accelerator (e.g. GPUs) (88 systems)**
  - **Lightweight cores (e.g. IBM BG, ARM, Knights Landing)**
- **Interest in supercomputing is now worldwide, and growing in many new markets (~50% of Top500 computers are in industry).**
- **Exascale (10^{18} Flop/s) projects exist in many countries and regions.**
- **Intel processors largest share, 92% followed by AMD, 1%.**

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

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

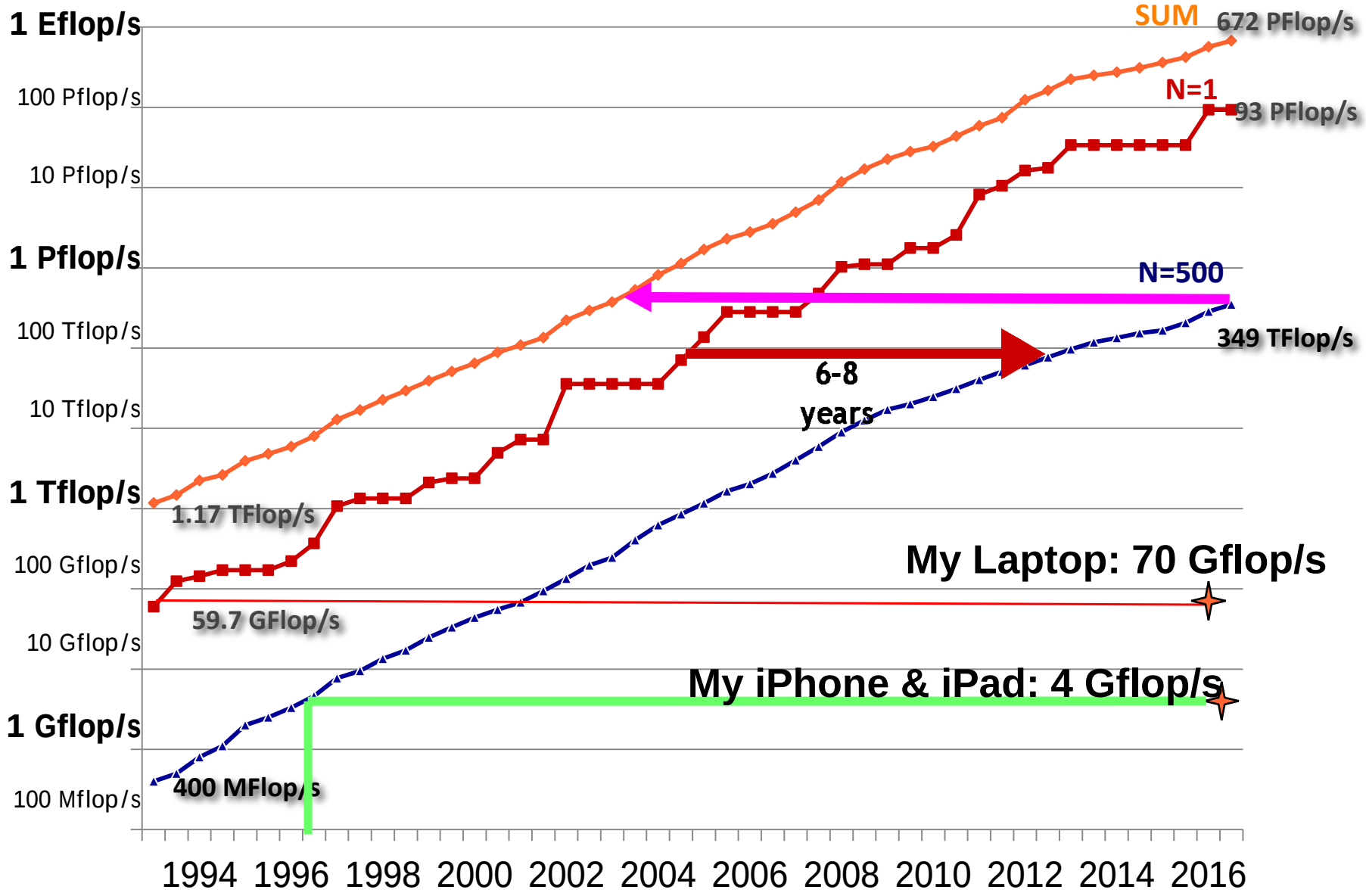
$Ax=b$, *dense problem*



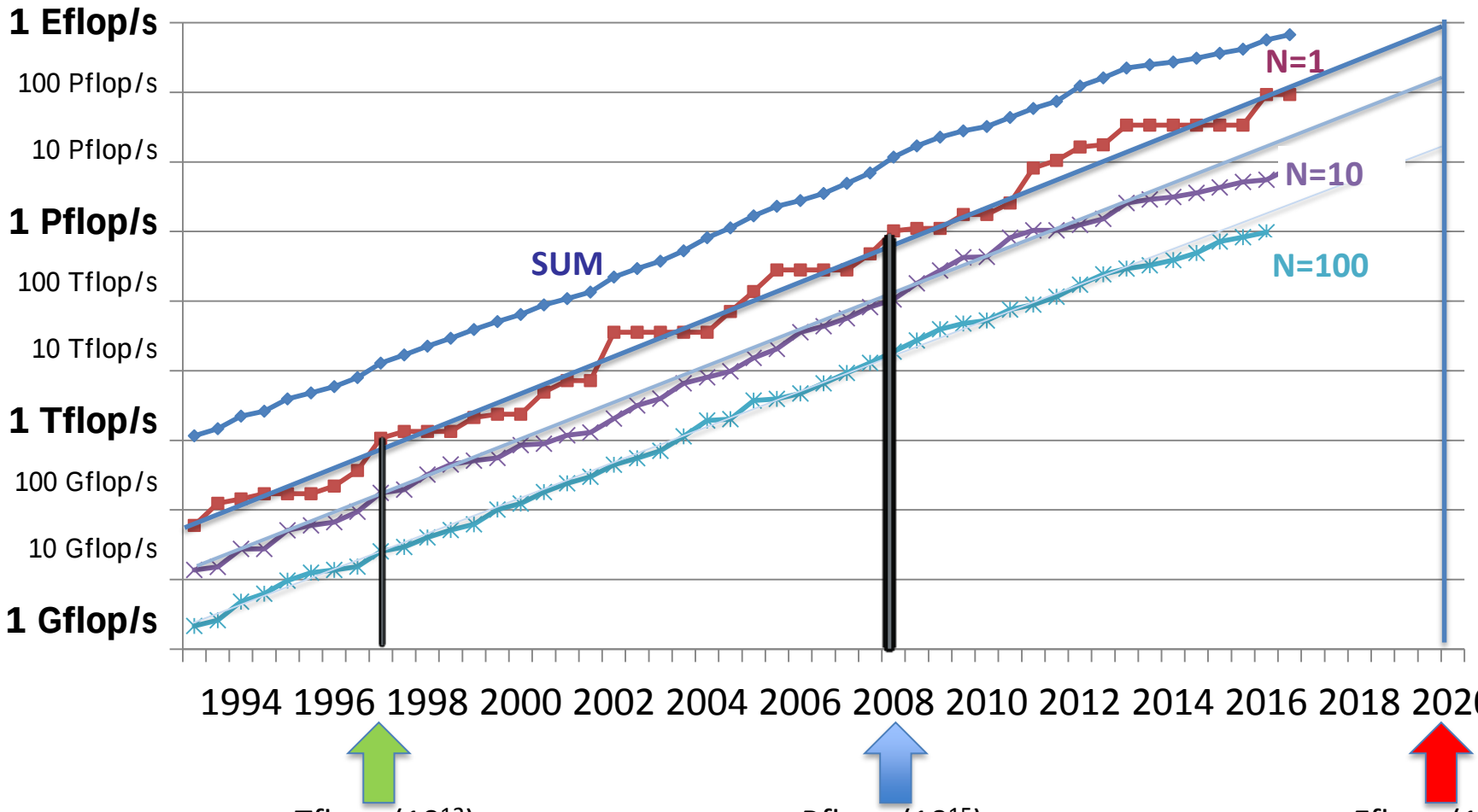
- Updated twice a year
  SC‘xy in the States in November
  Meeting in Germany in June

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

Performance Development of HPC over the Last 24 Years from the Top500



PERFORMANCE DEVELOPMENT



Tflops (10^{12}) Achieved ASCI Red Sandia NL

Pflops (10^{15}) Achieved RoadRunner Los Alamos NL

Eflops (10^{18}) Achieved?

China says 2020
U.S. says 2021

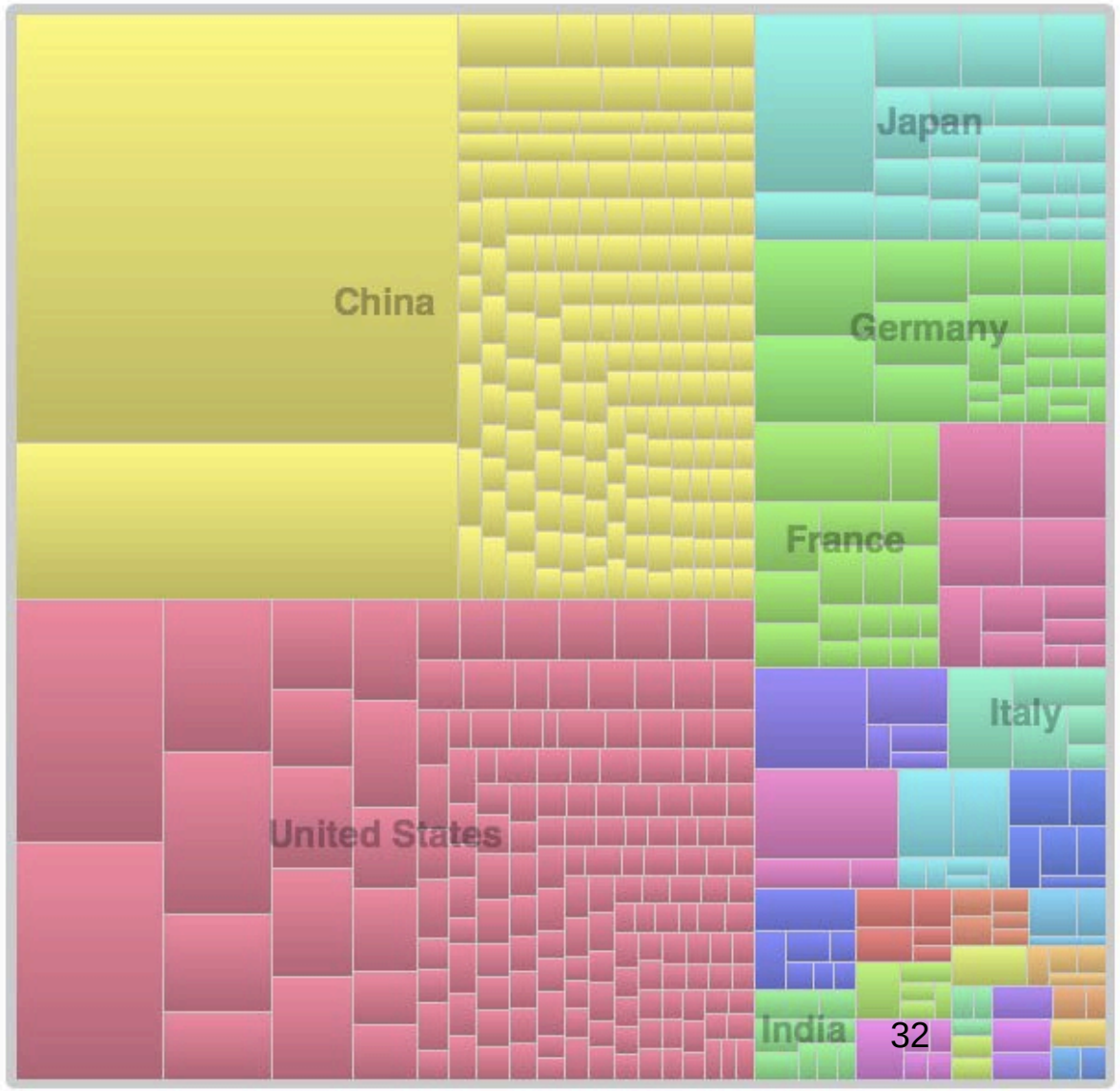
November 2016: The TOP 10 Systems

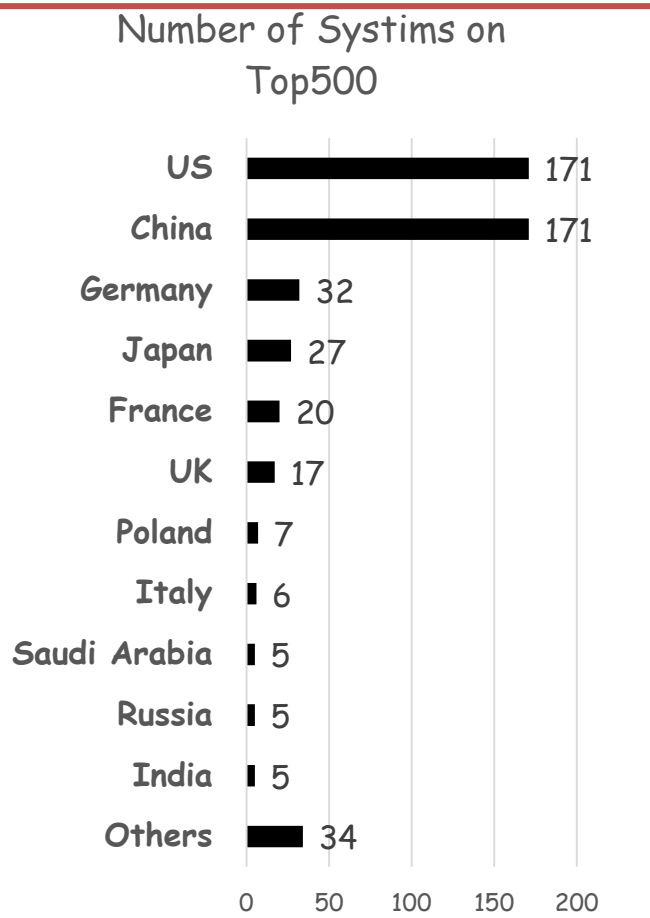
Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	GFlops/ Watt
1	National Super Computer Center in Wuxi	Sunway TaihuLight, SW26010 (260C) + Custom	China	10,649,000	93.0	74	15.4	6.04
2	National Super Computer Center in Guangzhou	Tianhe-2 NUDT, Xeon (12C) + IntelXeon Phi (57C) + Custom	China	3,120,000	33.9	62	17.8	1.91
3	DOE / OS Oak Ridge Nat Lab	Titan, Cray XK7, AMD (16C) + Nvidia Kepler GPU (14C) + Custom	USA	560,640	17.6	65	8.21	2.14
6	Advanced HPC	Primergy CX1640, Xeon Phi (68C) + Omni-Path	Japan	558,144	13.6	54	2.72	4.98
7	RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx (8C) + Custom	Japan	705,024	10.5	93	12.7	.827
8	Swiss CSCS	Piz Daint, Cray XC50, Xeon (12C) + Nvidia P100(56C) + Custom	Swiss	206,720	9.78	61	1.31	7.45
9	DOE / OS Argonne Nat Lab	Mira, BlueGene/Q (16C) + Custom	USA	786,432	8.59	85	3.95	2.07
10	DOE / NNSA / Los Alamos & Sandia	Trinity, Cray XC40,Xeon (16C) + Custom	USA	301,056	8.10	80	4.23	1.92
500	Internet company	Inspur Intel (8C) + Nnvidia	China	5440	.286	71		

TaihuLight is 5.2 X Performance of Titan

TaihuLight is 1.1 X Sum of All DOE Systems

Countries Share





China has 1/3 of the systems, while the number of systems in the US has fallen to the lowest point since the TOP500 list was created.

Each rectangle represents one of the Top500 computers, area of rectangle reflects its performance.

Toward Exascale

- China plans for Exascale 2020
  - Three separate developments in HPC; "Anything but from the US"
  - Wuxi
    - ShenWei O(100) Pflops all Chinese, June 2016
  - National University for Defense Technonlogy
    - Tianhe-2A O(100) Pflops will be Chinese ARM processor + accelerator, 2017
  - Sugon - CAS ICT
    - X86 based; collaboration with AMD
- US DOE - Exascale Computing Program – 7 Year Program
  - Initial exascale system based on *advanced architecture* and delivered in 2021
  - Enable *capable exascale* systems, based on ECP R&D, delivered in 2022 and deployed in 2023

Many Other Benchmarks

- TOP500
- Green 500
- Graph 500
- Sustained Petascale Performance
- HPC Challenge
- Perfect
- ParkBench
- SPEC-hpc
- Big Data Top100
- Livermore Loops
- EuroBen
- NAS Parallel Benchmarks
- Genesis
- RAPS
- SHOC
- LAMMPS
- Dhrystone
- Whetstone
- I/O Benchmarks
- WRF
- Yellowstone
- Roofline
- Neptune

High Performance Linpack (HPL)

- Is a **widely recognized** and discussed metric for ranking high performance computing systems
- When HPL gained prominence as a performance metric in the early 1990s there **was a strong correlation between its predictions of system rankings and the ranking that full-scale applications would realize**.
- **Computer system vendors pursued designs that would increase their HPL performance**, which would in turn improve overall application performance.
- Today HPL remains **valuable as a measure of historical trends**, and as a stress test, especially for leadership class systems that are pushing the boundaries of current technology.

The Problem

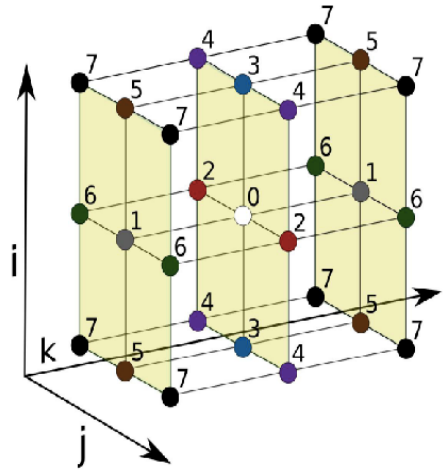
- HPL performance of computer systems are **no longer so strongly correlated to real application performance**, especially for the broad set of HPC applications governed by partial differential equations.

- **Designing a system for good HPL performance can actually lead to design choices that are wrong** for the real application mix, or add unnecessary components or complexity to the system.

hpcg-benchmark.org

HPCG

- High Performance Conjugate Gradients (HPCG).
- Solves *Ax=b*, *A* large, sparse, *b* known, *x* computed.
- An optimized implementation of PCG contains essential computational and communication patterns that are prevalent in a variety of methods for discretization and numerical solution of PDEs
- Synthetic discretized 3D PDE (FEM, FVM, FDM).
- Sparse matrix:
  - 27 nonzeros/row interior.
  - 8 – 18 on boundary.
  - Symmetric positive definite.



27-point stencil operator

- Patterns:
  - Dense and sparse computations.
  - Dense and sparse collectives.
  - Multi-scale execution of kernels via MG (truncated) V cycle.
  - Data-driven parallelism (unstructured sparse triangular solves).
- Strong verification (via spectral properties of PCG).

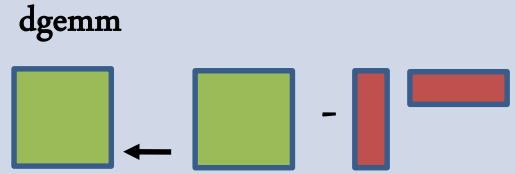
HPCG Results, Nov 2016, 1-10

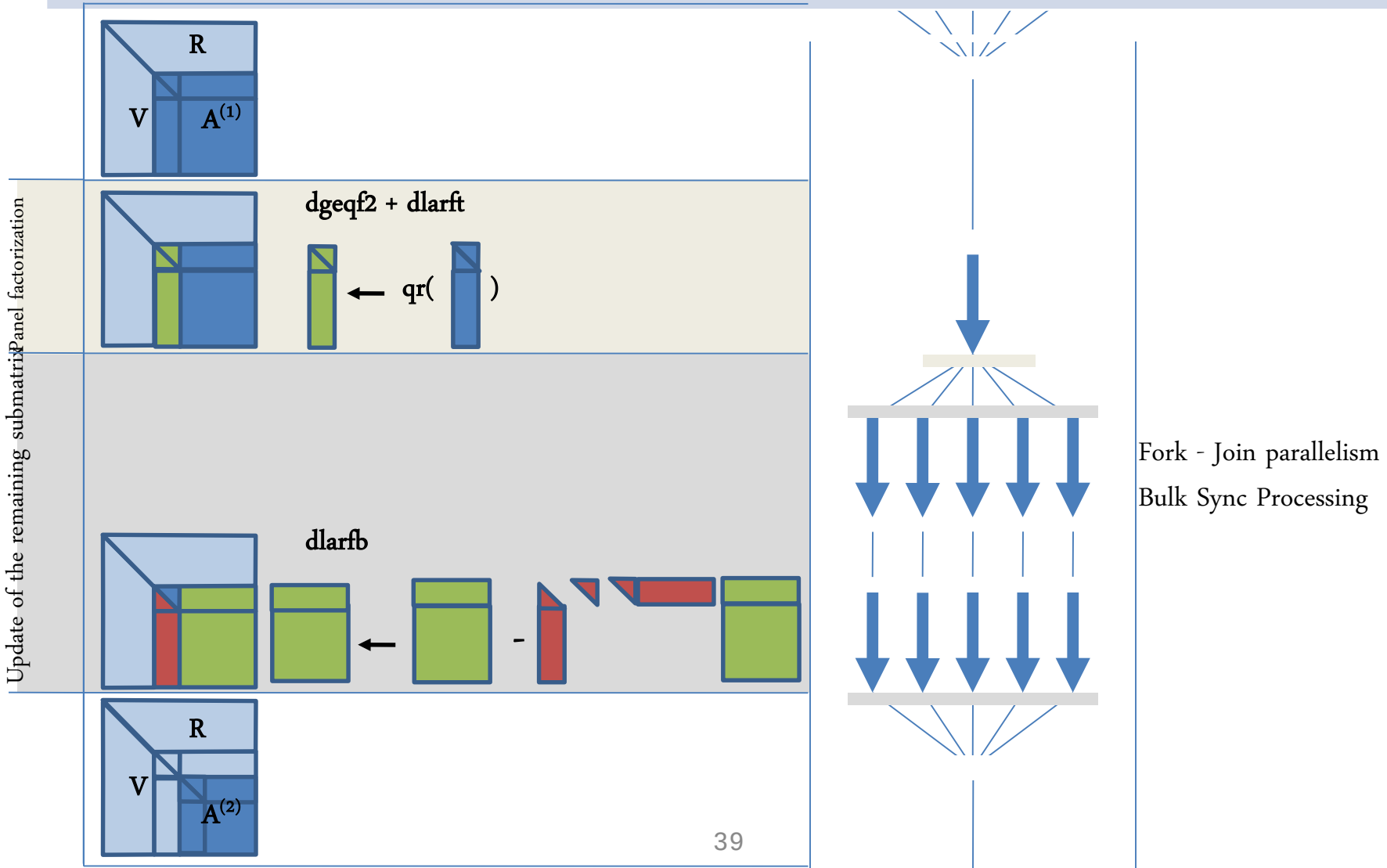
#	Site	Computer	Cores	HPL Pflops	HPCG Pflops	% of Peak
1	RIKEN Advanced Institute for Computational Science	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect	705,024	10.5	0.603	5.3%
2	NSCC / Guangzhou	Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom	3,120,000	33.8	0.580	1.1%
3	Joint Center for Advanced HPC, Japan	Oakforest-PACS – PRIMERGY CX600 M1, Intel Xeon Phi	557,056	24.9	0.385	2.8%
4	National Supercomputing Center in Wuxi, China	Sunway TaihuLight – Sunway MPP, SW26010	10,649,600	93.0	0.3712	0.3%
5	DOE/SC/LBNL/NERSC USA	Cori – XC40, Intel Xeon Phi Cray	632,400	13.8	0.355	1.3%
6	DOE/NNSA/LLNL USA	Sequoia – IBM BlueGene/Q, IBM	1,572,864	17.1	0.330	1.6%
7	DOE/SC/Oak Ridge Nat Lab	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x	560,640	17.5	0.322	1.2%
8	DOE/NNSA/LANL/SNL	Trinity - Cray XC40, Intel E5-2698v3, Aries custom	301,056	8.10	0.182	1.6%
9	NASA / Mountain View	Pleiades - SGI ICE X, Intel E5-2680, E5-2680V2, E5-2680V3, Infiniband FDR	243,008	5.90	0.175	2.5%
10	DOE/SC/Argonne National Laboratory	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom	786,432	8.58	0.167	1.7%

Parallelization of Factorization

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

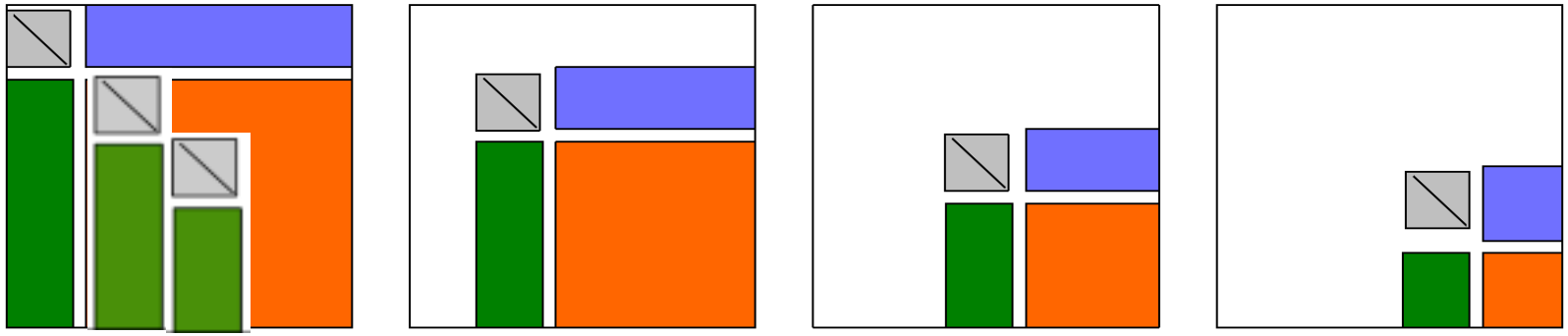


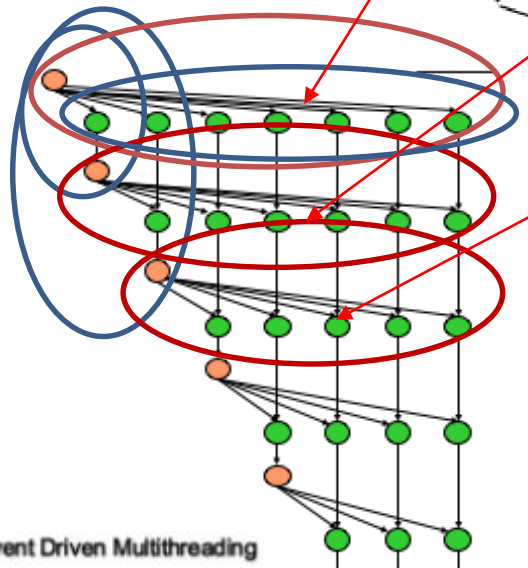


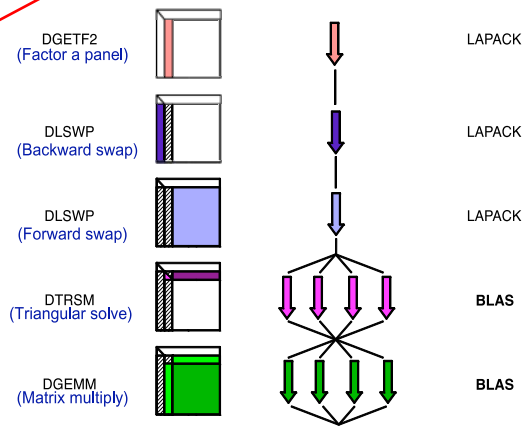
Fork - Join parallelism

Bulk Sync Processing

Synchronization (in LAPACK)

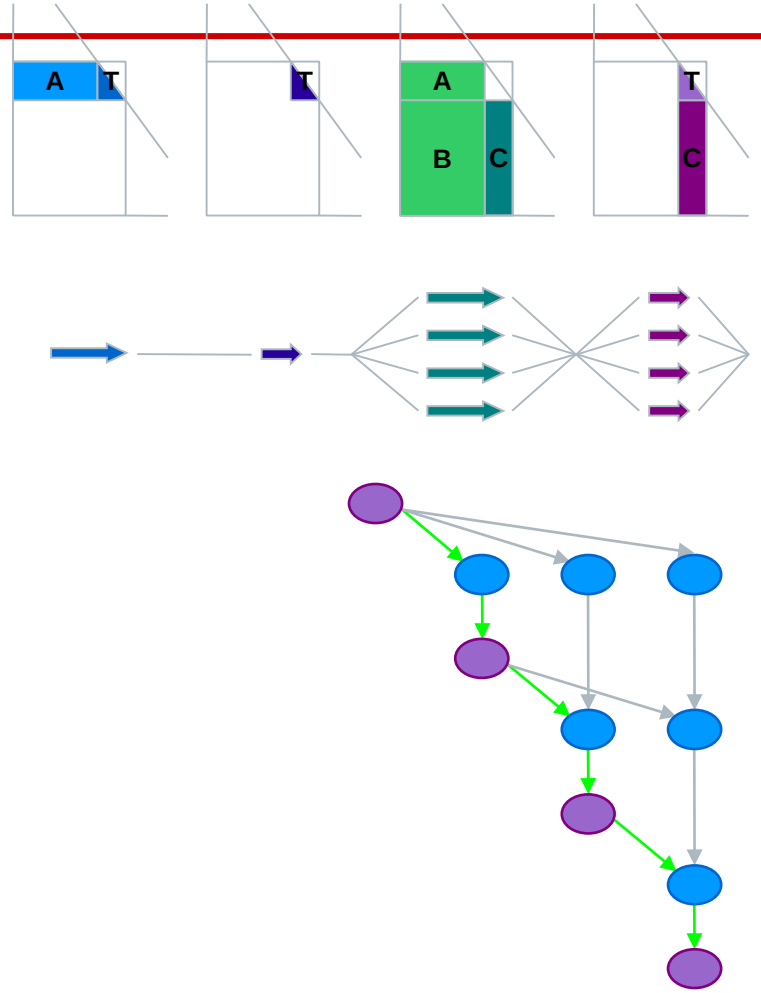




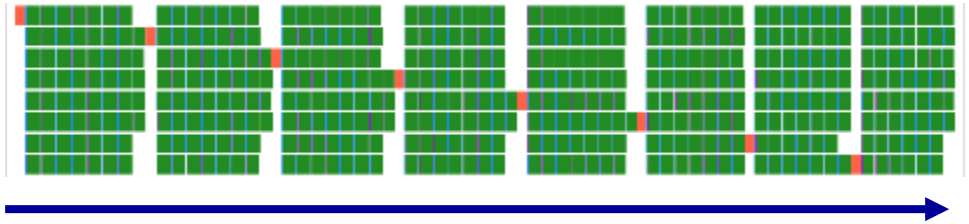


- fork join
- bulk synchronous processing

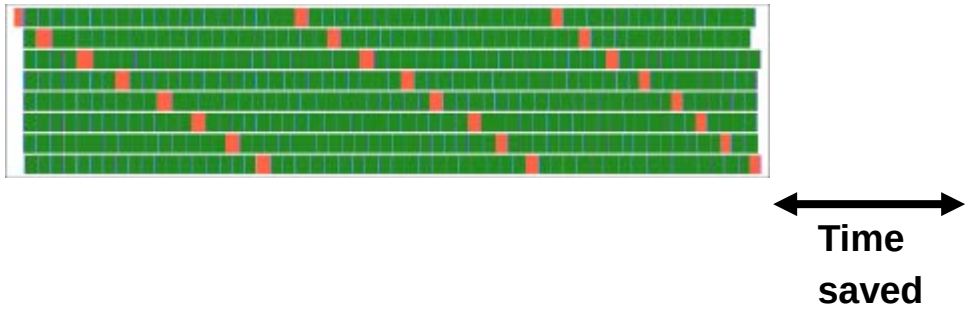
Fork-Join vs. Dynamic Execution



Fork-Join – parallel BLAS



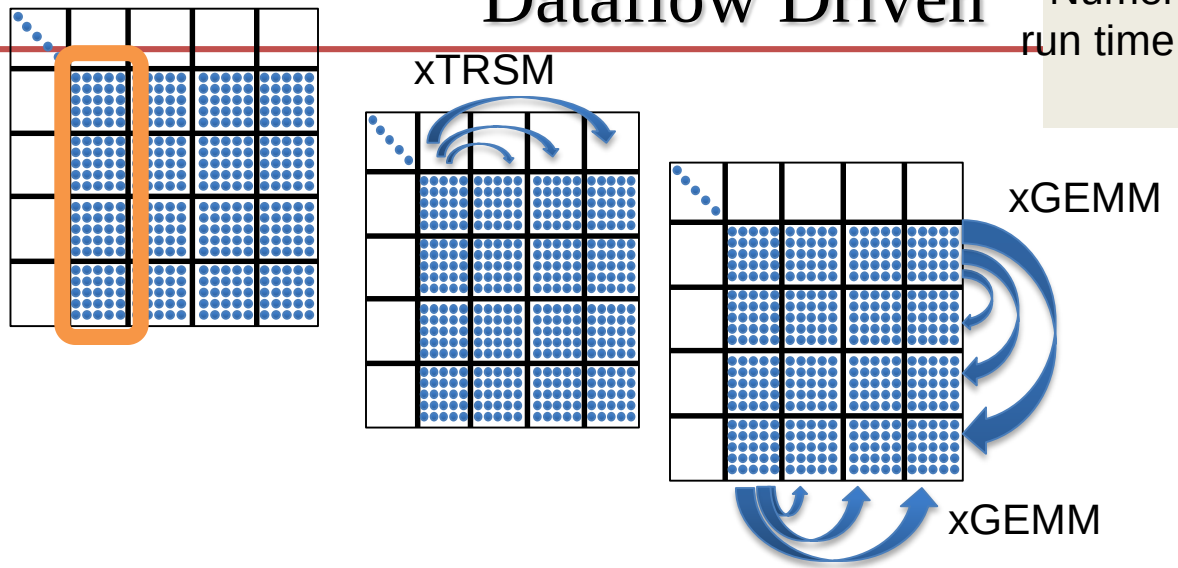
DAG-based – dynamic scheduling



PLASMA LU Factorization

Dataflow Driven

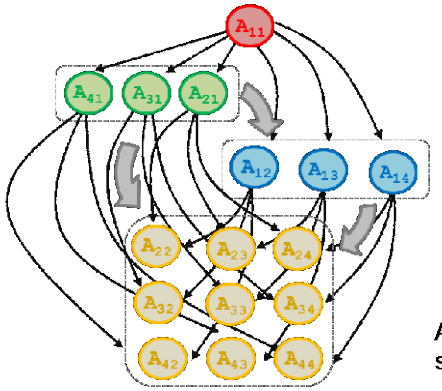
Numerical program generates tasks and run time system executes tasks respecting data dependences.



Sparse / Dense Matrix System

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

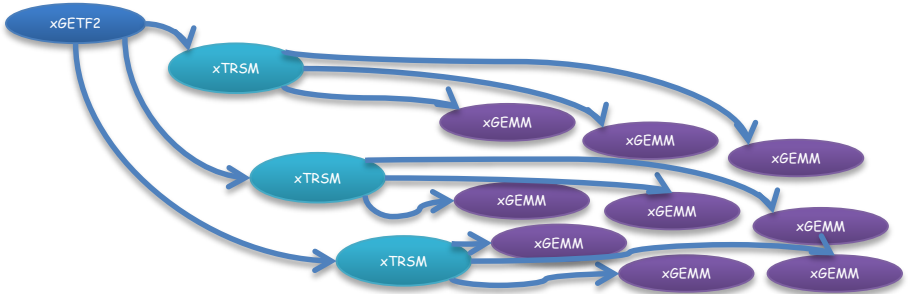
DAG-based factorization



Batched LA

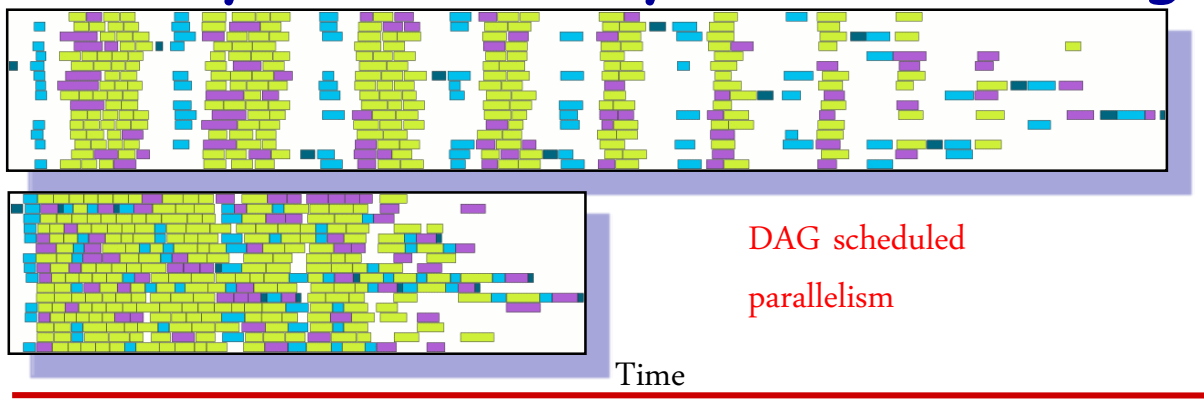
- **LU, QR, or Cholesky** on small diagonal matrices
- **TRSMs, QRs, or LUs**
- **TRSMs, TRMMs**
- **Updates (Schur complement) GEMMs, SYRKs, TRMMs**

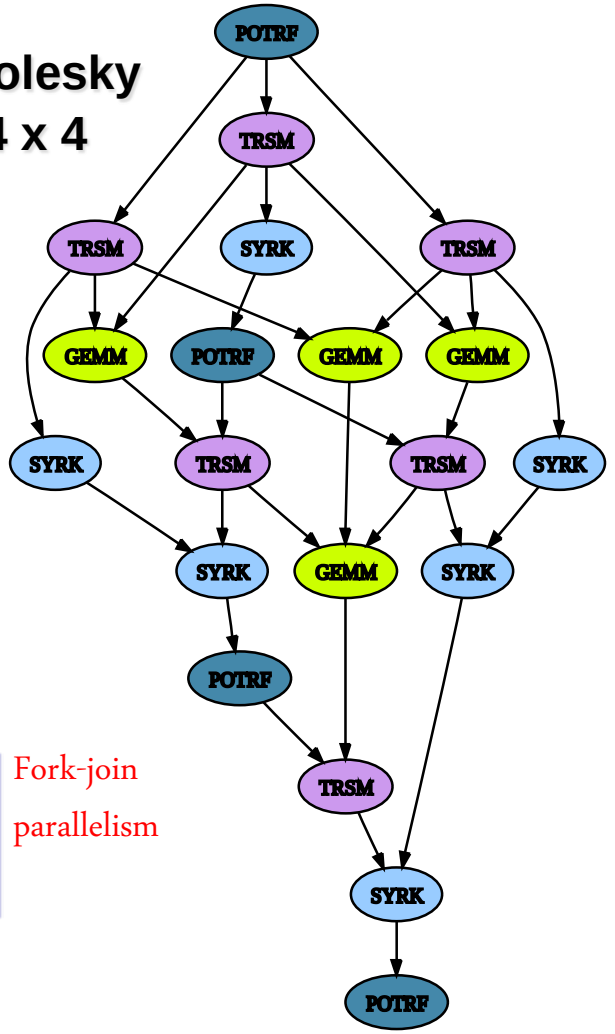
And many other BLAS/LAPACK, e.g., for application specific solvers, preconditioners, and matrices



PLASMA: Parallel Linear Algebra s/w for Multicore Architectures

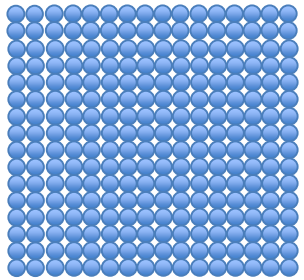
- **Objectives**
  - High utilization of each core
  - Scaling to large number of cores
  - Shared or distributed memory
- **Methodology**
  - Dynamic DAG scheduling
  - Split phases task generation and execution
  - Explicit parallelism/Implicit communication
  - Fine granularity / block data layout
- **Arbitrary DAG with dynamic scheduling**

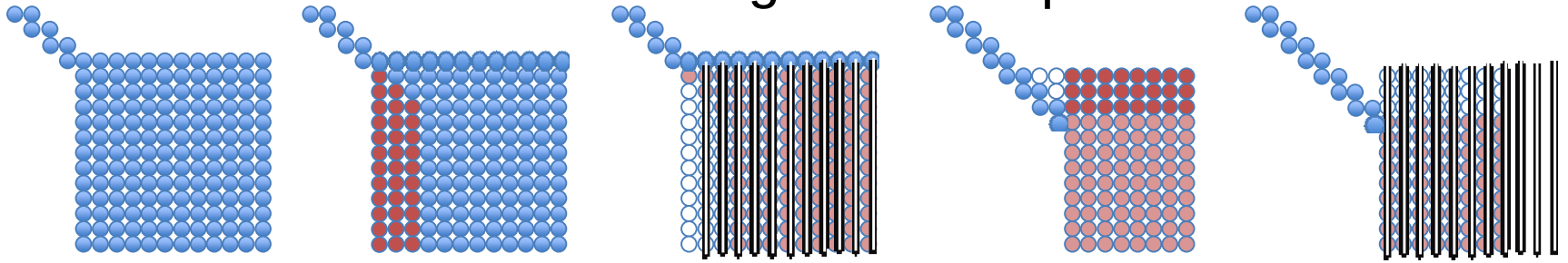


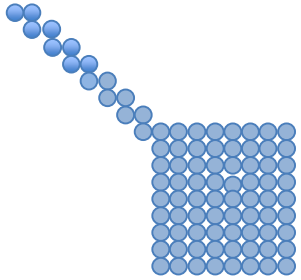


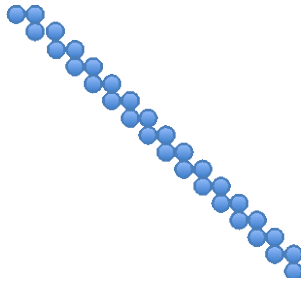
Singular Value Decomposition
EISPACK, LINPACK, & LAPACK

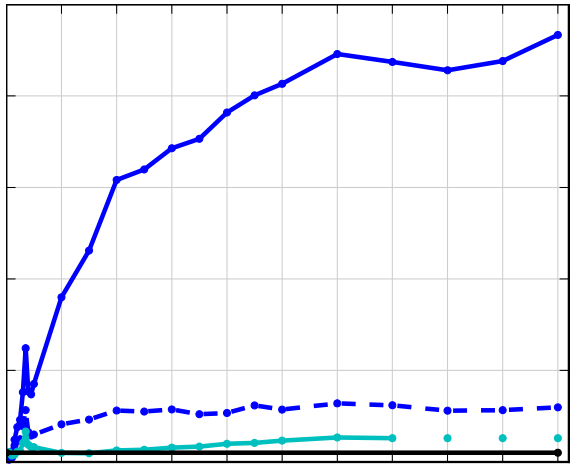
3 Generations of software compared
First Stage 8/3 n^3 Ops









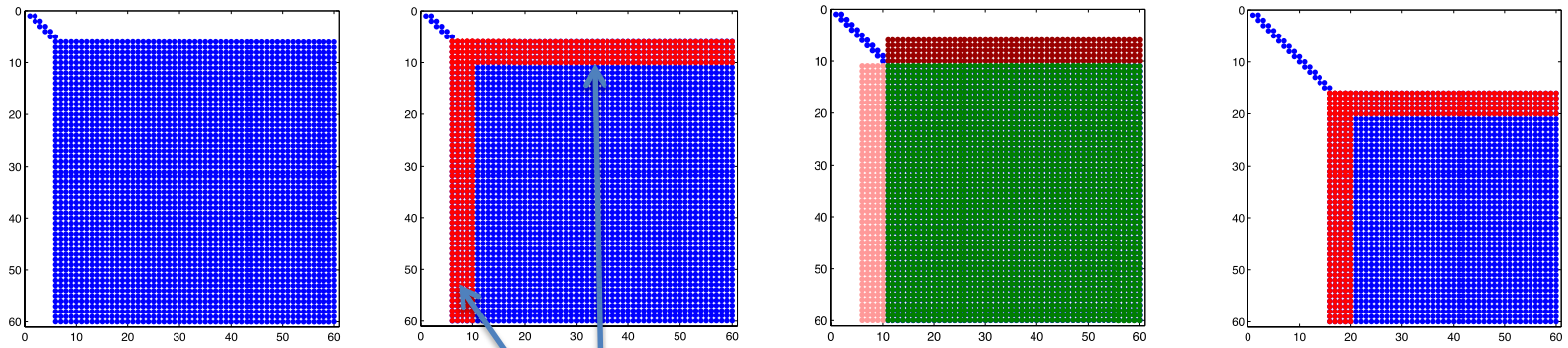


LAPACK QR (BLAS in ||, 16 cores)
LAPACK QR (using1 core)(1991)
LINPACK QR (1979)
EISPACK QR (1975)

QR refers to the QR algorithm
for computing the eigenvalues

Dual socket – 8 core
Intel Sandy Bridge 2.6 GHz
(8 Flops per core per cycle)

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



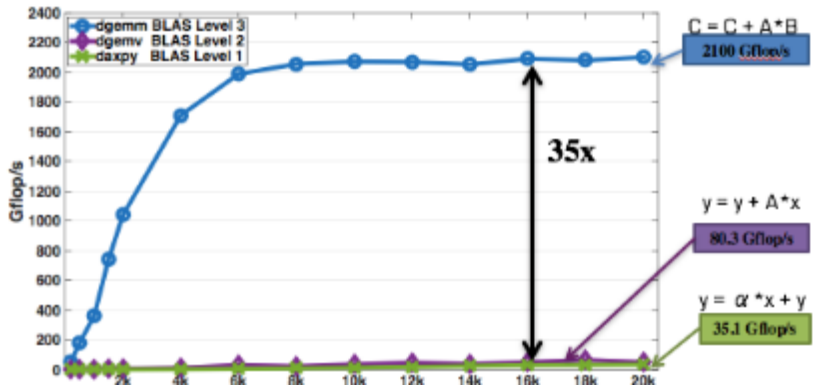
factor panel k then update ➔ **factor panel k+1**

Requires 2 GEMVs

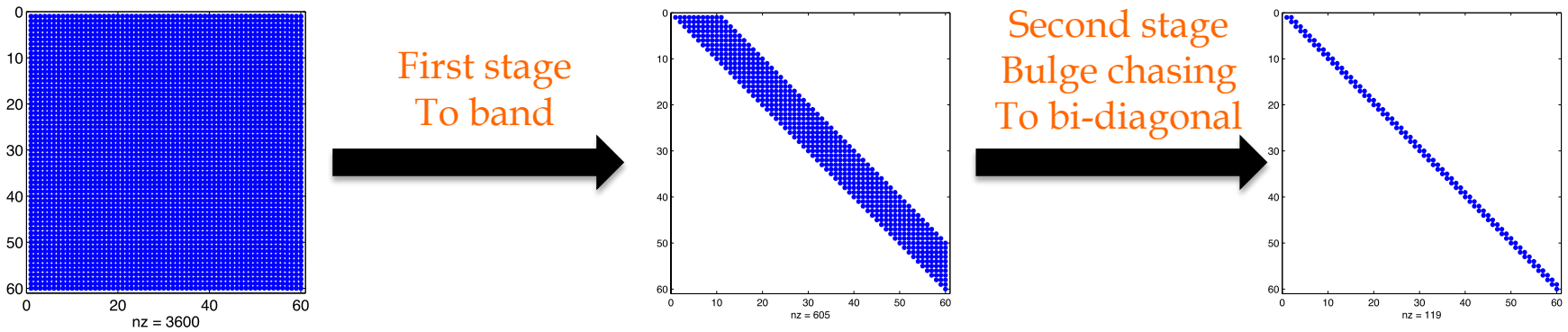
$Q^*A^*P^H$

✸ Characteristics

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



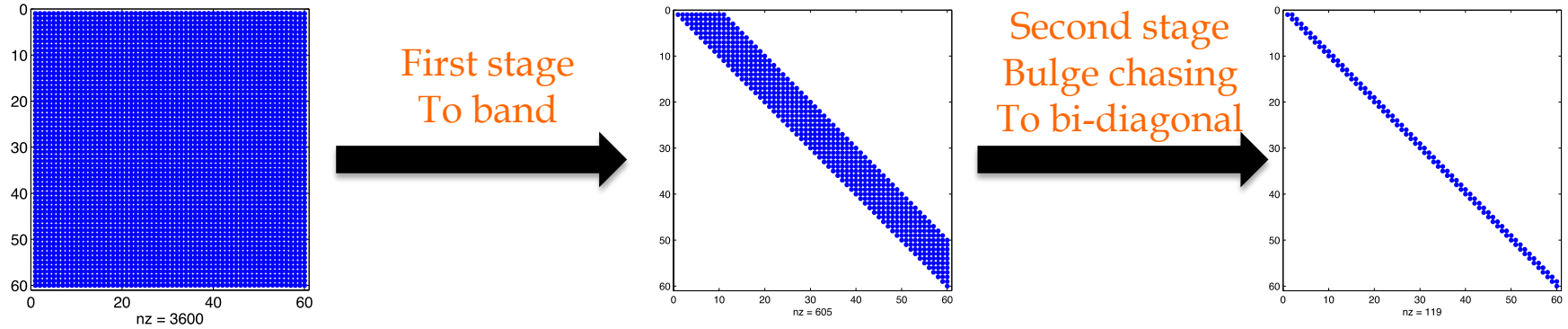
Recent Work on 2-Stage Algorithm



- **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



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

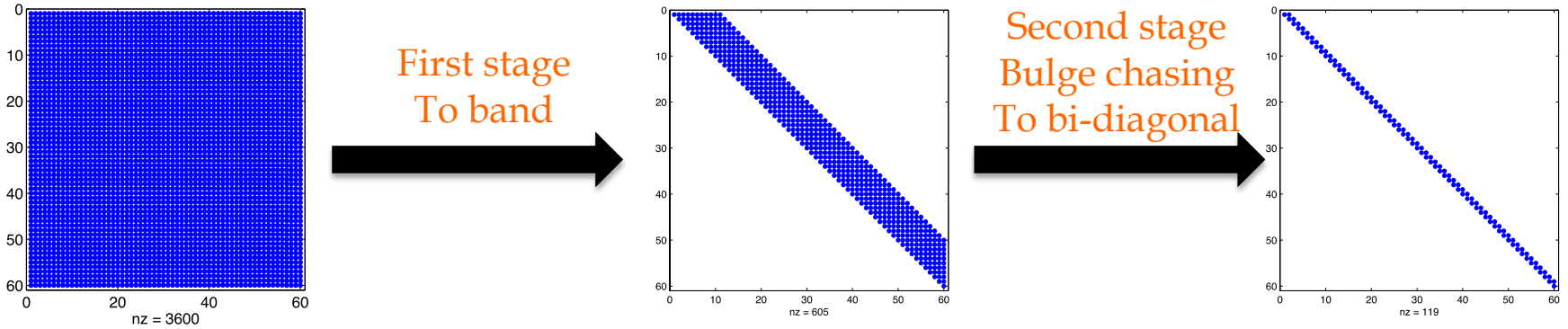
$$+ \sum_{s=1}^{\frac{n-n_b}{n_b}} 2n_b^3 + (nt-s-1)3n_b^3 + (nt-s-1)\tfrac{10}{3}n_b^3 + (nt-s)\times(nt-s-1)5n_b^3$$

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

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

More Flops, original did 8/3 n^3
25% More flops

Recent work on developing new 2-stage algorithm

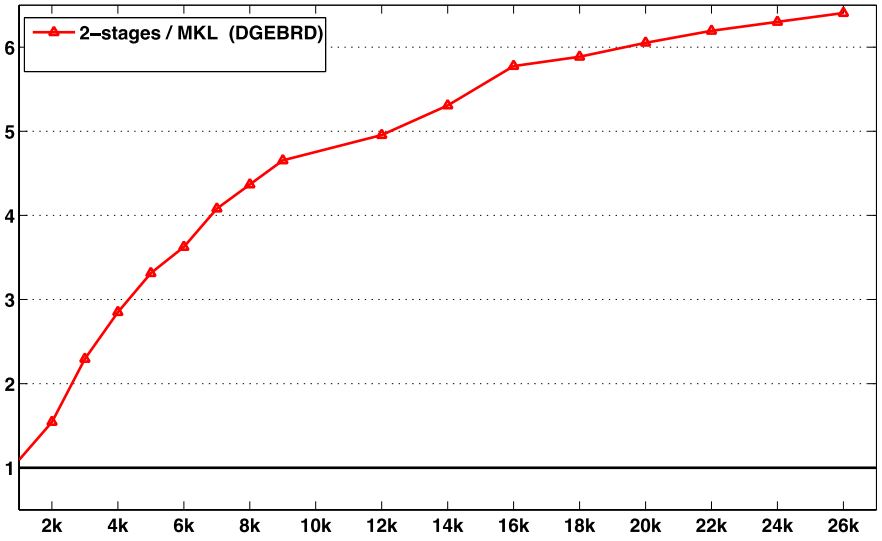


$$\textbf{speedup} = \frac{\textbf{time of one-stage}}{\textbf{time of two-stage}}$$

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

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

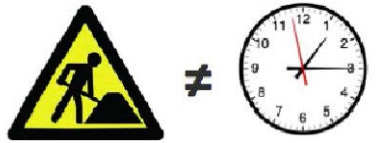
$$\Longrightarrow 1.8 \leq \textbf{Speedup} \leq 7$$



16 Sandy Bridge cores 2.6 GHz

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

25% More flops and 1.8 – 6 times faster



SLATE – Software for Linear Algebra Targeting Exascale

- Target Hardware DOE Exascale systems, as well as pre-Exascale
- Bring the best ideas of LAPACK, ScaLAPACK, PLASMA & MAGMA
- Goals
  - Efficiency – to run as fast as possible (close to theoretical peak);
  - Scalability – as the problem size and number of processors grow;
  - Reliability – including error bounds and rigorous LAPACK-derived testing suites;
  - Portability – across all important parallel machines (as described above);
  - Flexibility – so users can construct new routines from well-designed parts;
  - Ease of use – by making the interfaces look as similar as possible to LAPACK and ScaLAPACK.

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.

Collaborators and Support

MAGMA team

http://icl.cs.utk.edu/magma

PLASMA team

http://icl.cs.utk.edu/plasma

Collaborating partners

University of Tennessee, Knoxville
Lawrence Livermore National Laboratory, Livermore, CA
University of California, Berkeley
University of Colorado, Denver
INRIA, France (StarPU team)
KAUST, Saudi Arabia

Umeå University

INRIA

Rutherford Appleton Laboratory

University of Manchester