The 2015 International Conference on High Performance Computing & Simulation
(HPCS 2015)

The Thirteen Annual Meeting

July 20 – 24, 2015
The Hilton Amsterdam Hotel
Amsterdam, the Netherlands

Architecture-aware Algorithms and Software for Peta- and Exascale Computing

Jack Dongarra

University of Tennessee
Oak Ridge National Laboratory
University of Manchester

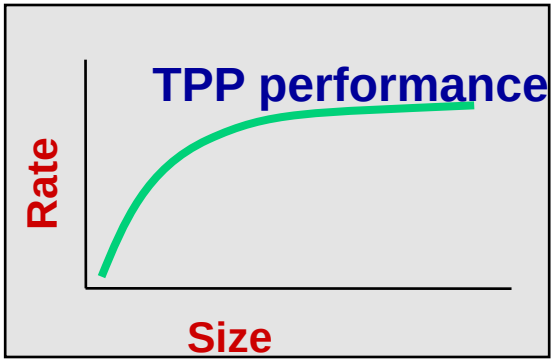
Outline

- Overview of High Performance Computing
- Look at an implementation for some linear algebra algorithms on today's High Performance Computers
  - As an examples of the kind of thing needed.

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*



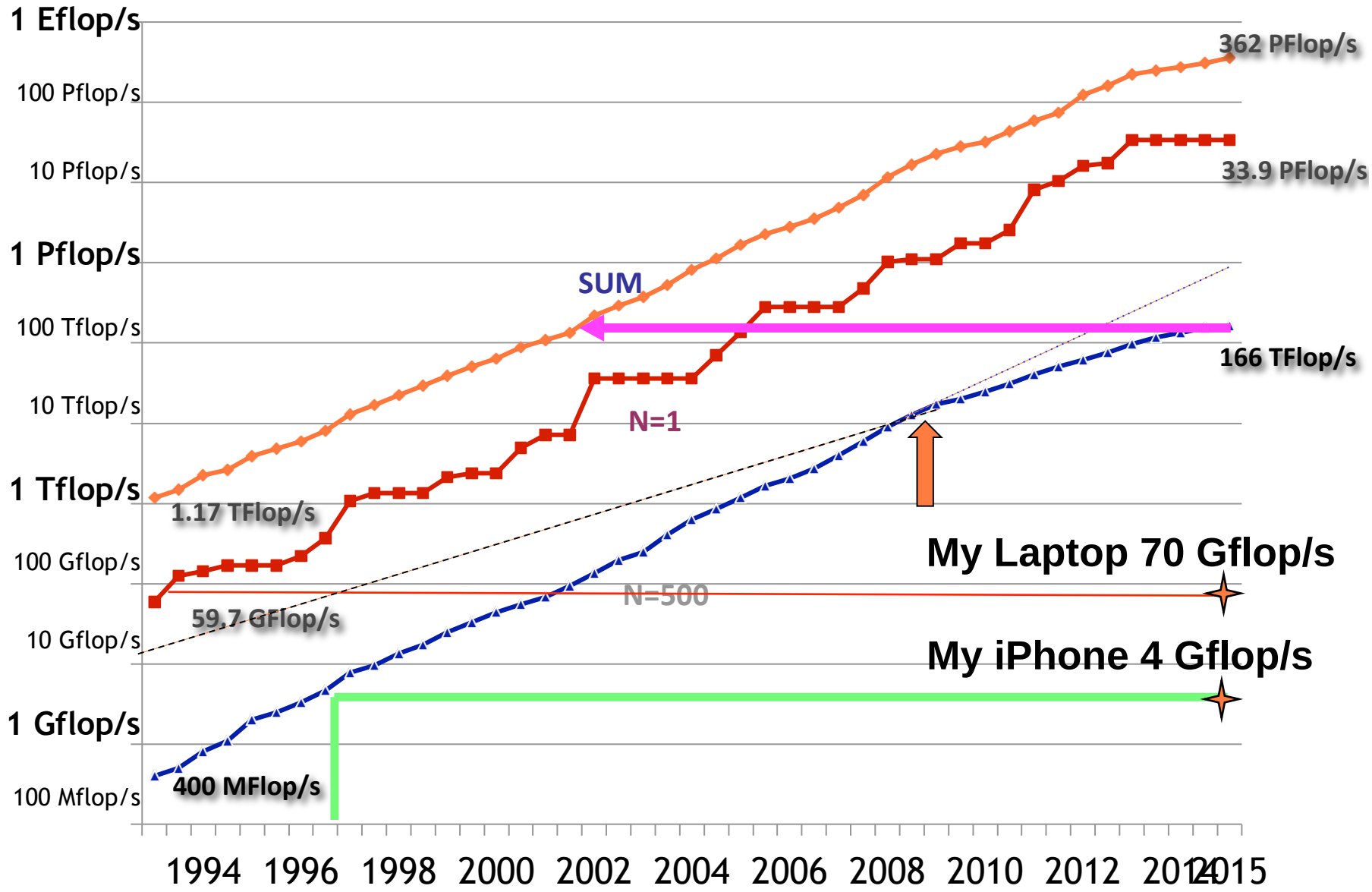
- Updated twice a year

  SC‘xy in the States in November

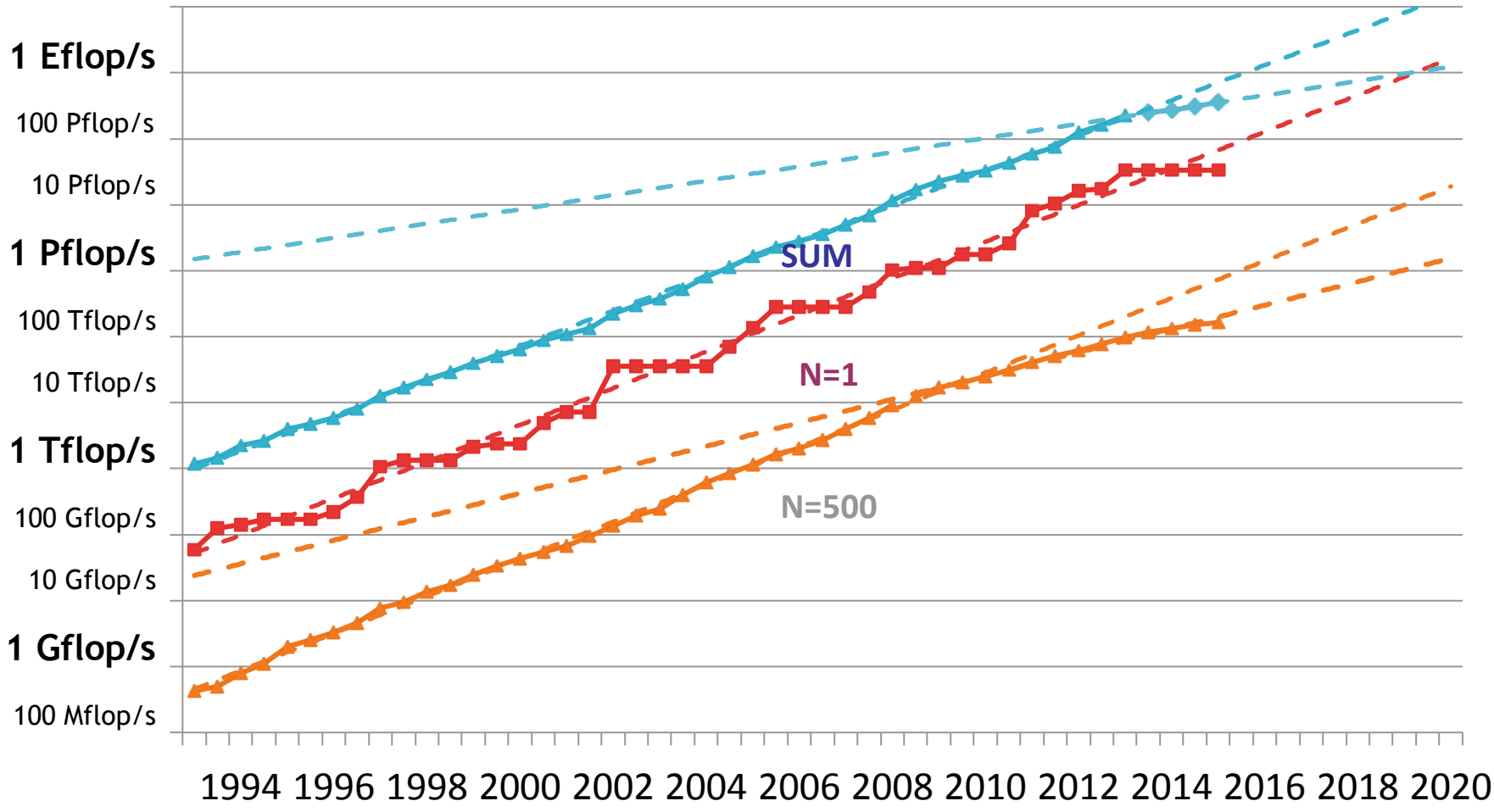
  Meeting in Germany in June

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

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



Projected Performance Development



State of Supercomputing in 2015

- Pflops 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, Knights Landing)
- Interest in supercomputing is now worldwide, and growing in many new markets (over 50% of Top500 computers are in industry).
- Exascale 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		

Commodity plus Accelerator
Today 88 of the Top500 Systems

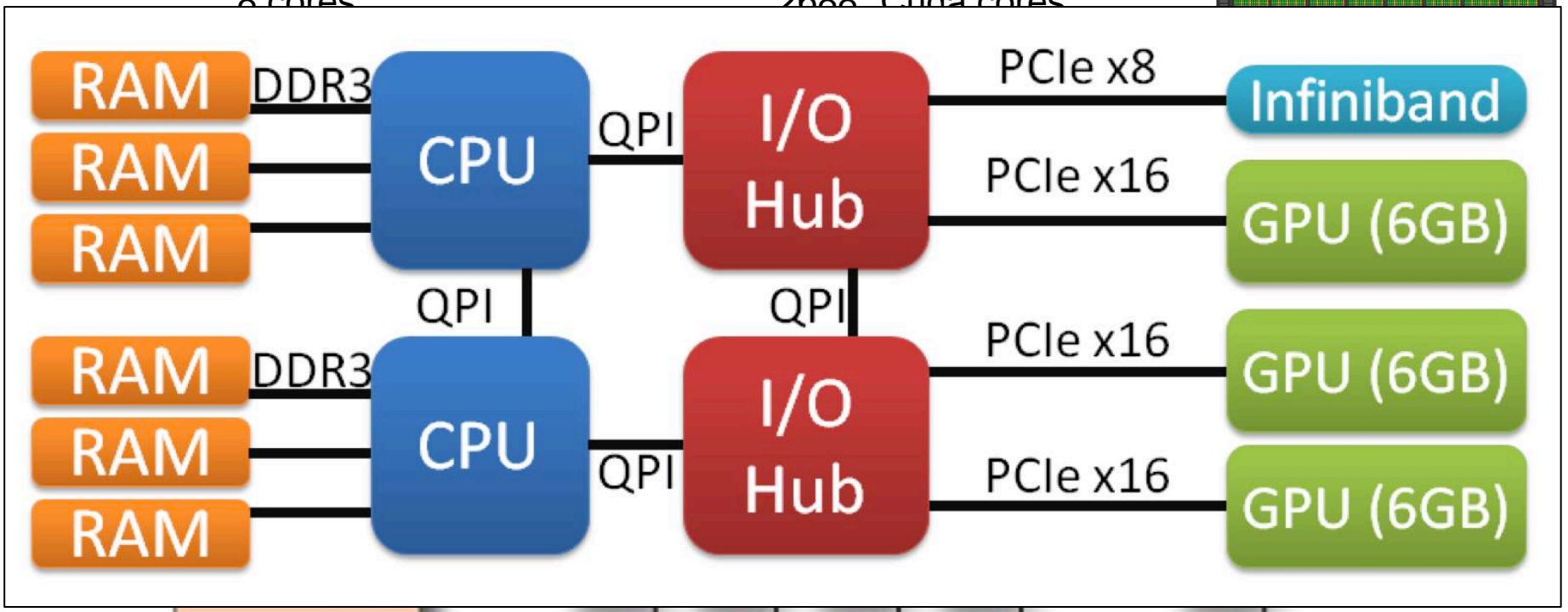
Commodity

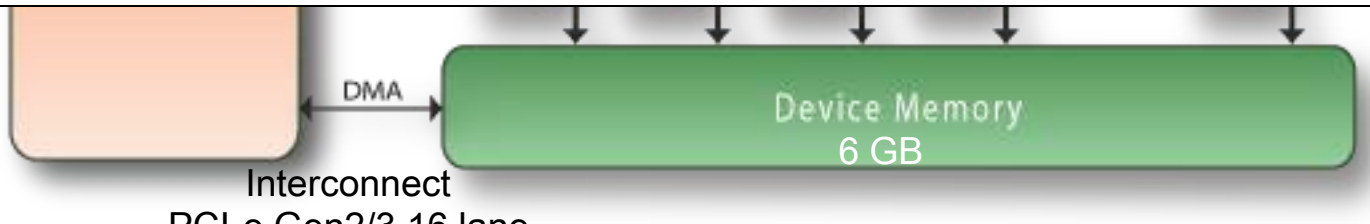
Intel Xeon
8 cores

Accelerator (GPU)

Nvidia K20X "Kepler"
2688 "Cuda cores"

192 Cuda cores/SMX
2688 "Cuda cores"





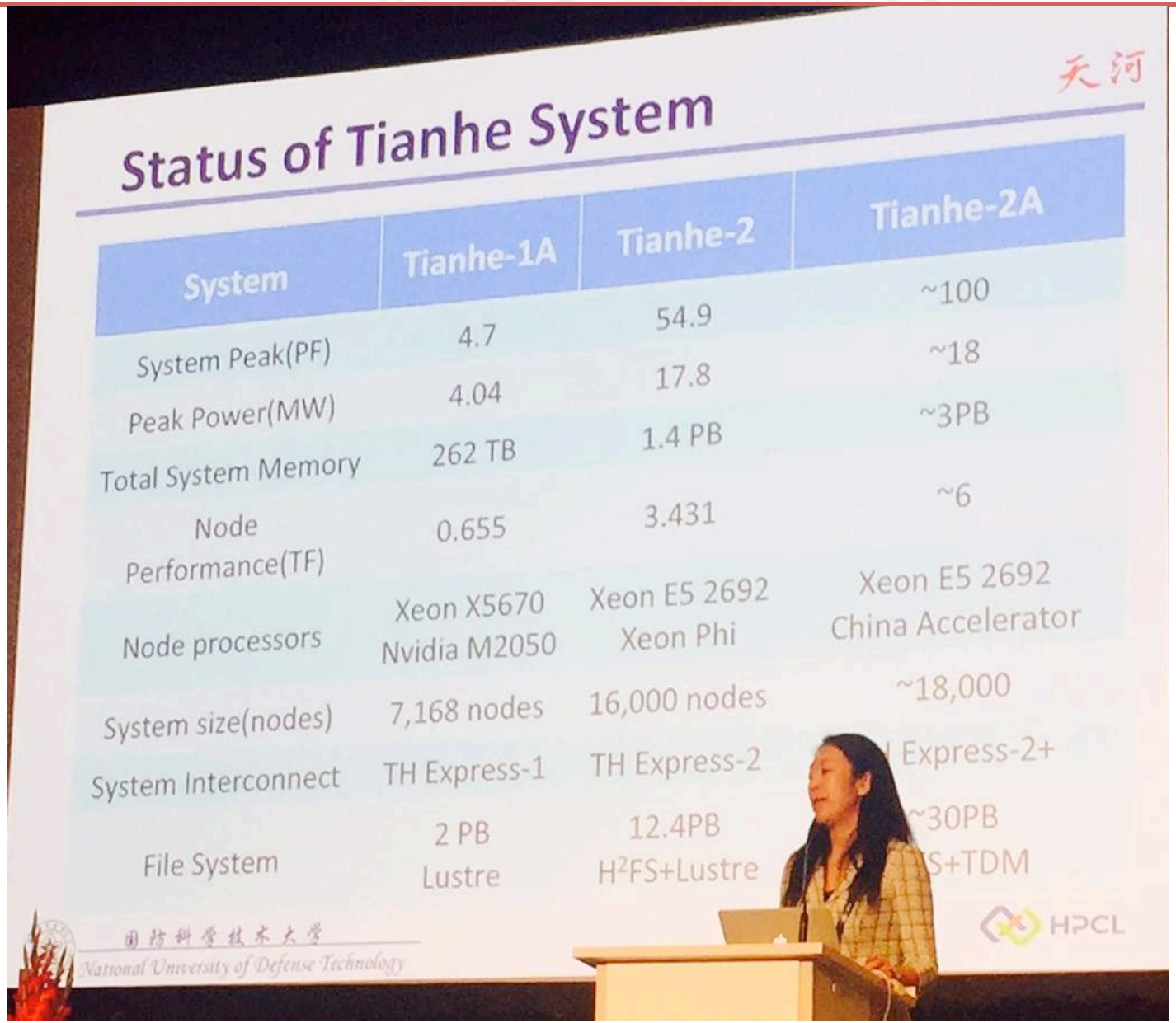
Interconnect
PCI-e Gen2/3 16 lane
64 Gb/s (8 GB/s)
1 GW/s

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
- US Dept of Commerce is [illegible] a groups from receiving In[illegible]ith the [illegible]5.

  - National SC Center Chang[illegible]
- For the first time, < 50% of [illegible]
  - 231 of the systems are U.S.-bas[illegible]

Yutong Lu from NUDT at ISC Last Week

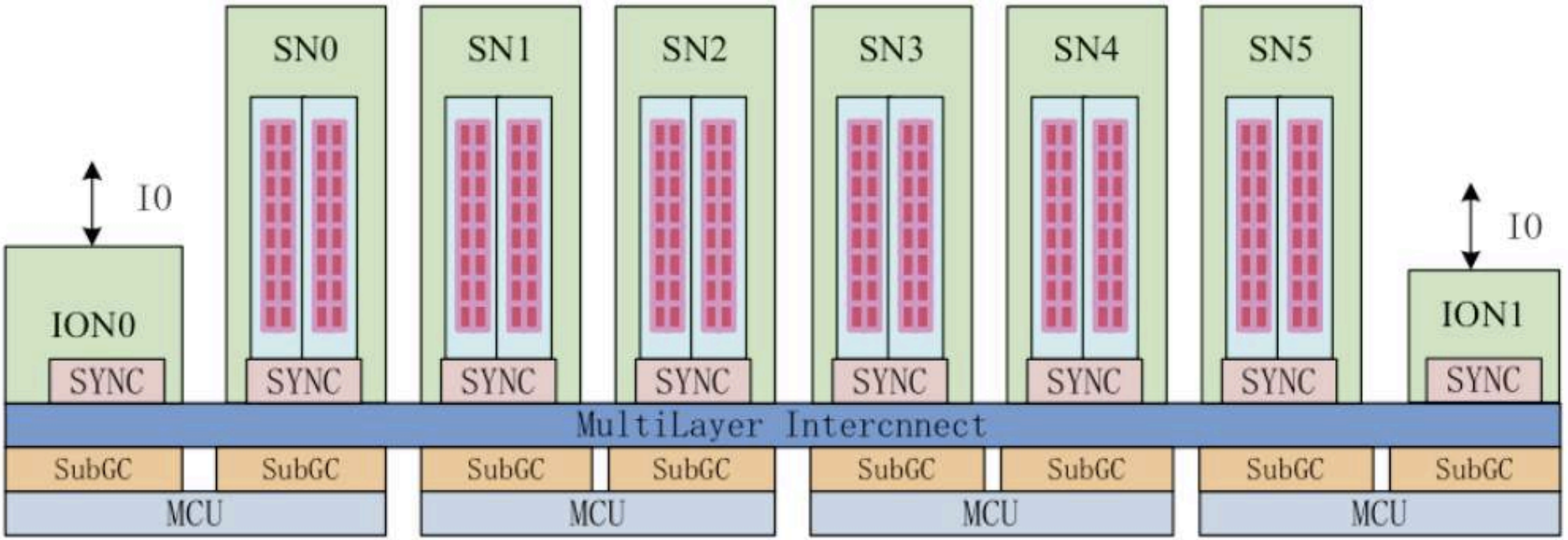


China Accelerator

天河

Matrix2000 GPDSP

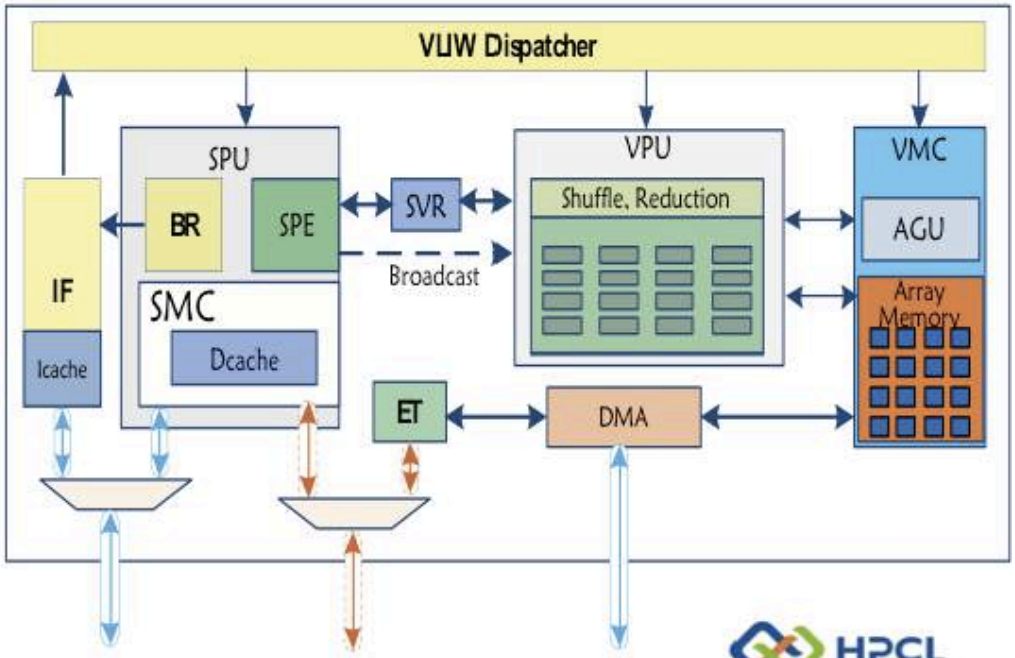
- High Performance
  - 64bit Supported
  - ~2.4/4.8TFlops(DP/SP)
  - 1GHz, ~200W
- High Throughput
  - High-bandwidth Memory
  - 32~64GB
  - PCIE 3.0, 16x



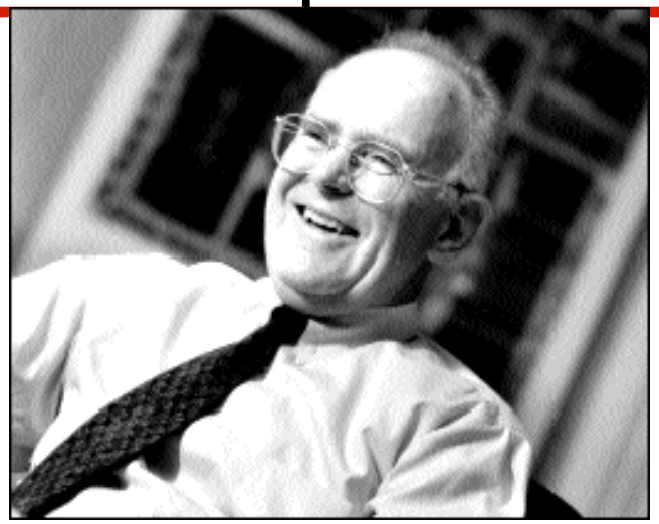
China Accelerator

天河

- Highly optimized GPDSP-Cores
  - Scalar Unit + Vector Unit
  - VLIW
  - Vector Shuffle and Reduction
  - Dedicated Vector Memory
- Non-blocking High-Speed NoC
  - 4Tb/s
- Global shared Cache
  - Cache validation on address scopes
  - Cache policy configurable
- Inter-core synchronization support
- Complexes DMA engine
  - Broadcast support
  - For global cache and memories



Technology Trends: Microprocessor Ca

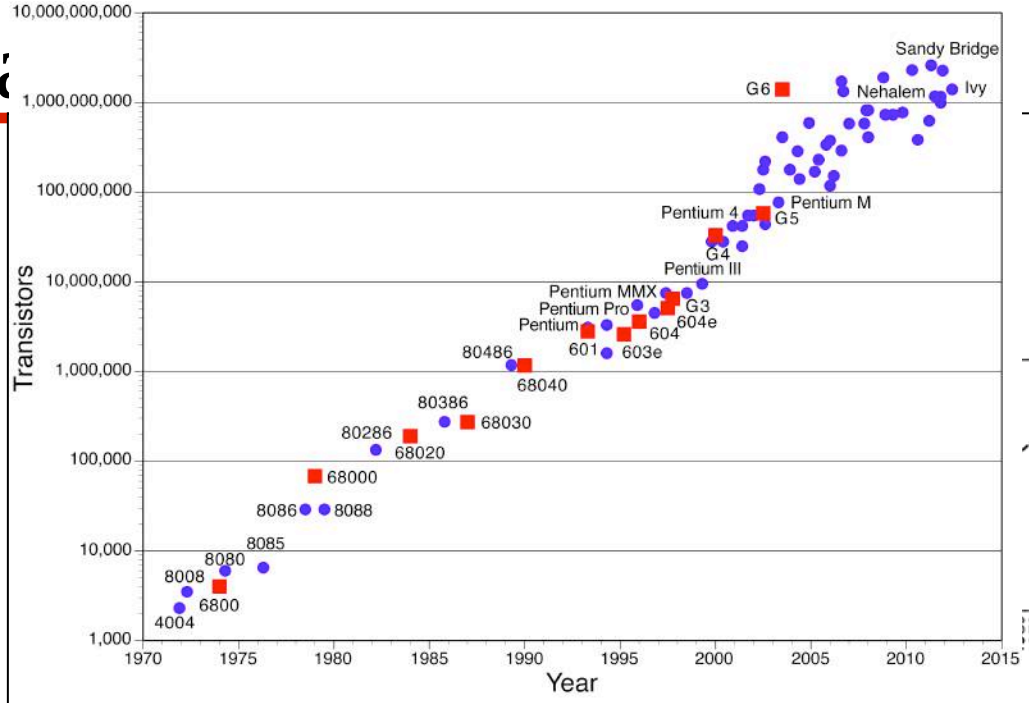


Gordon Moore (co-founder of Intel) Electronics Magazine, 1965

Number of devices/chip doubles every 18 months

2X transistors/Chip Every 1.5 years

Called "Moore's Law"



The future of integrated electronics is the future of electronics itself. The advantages of integration will bring about a proliferation of electronics, pushing this science into many new areas.

Integrated circuits will lead to such wonders as home computers—or at least terminals connected to a central computer—automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today.

But the biggest potential lies in the production of large systems. In telephone communications, integrated circuits in digital filters will separate channels on multiplex equipment. Integrated circuits will also switch telephone circuits and perform data processing.

Computers will be more powerful, and will be organized in completely different ways. For example, memories built of integrated electronics may be distributed throughout the machine instead of being concentrated in a central unit. In addition, the improved reliability made possible by integrated circuits will allow the construction of larger processing units. Machines similar to those in existence today will be built at lower costs and with faster turn-around.

Present and future

By integrated electronics, I mean all the various technologies which are referred to as microelectronics today as well as any additional ones that result in electronics functions supplied to the user as irreducible units. These technologies were first investigated in the late 1950's. The object was to miniaturize electronics equipment to include increasingly complex electronic functions in limited space with minimum weight. Several approaches evolved, including microassembly techniques for individual components, thin-film structures and semiconductor integrated circuits.

Each approach evolved rapidly and converged so that each borrowed techniques from another. Many researchers believe the way of the future to be a combination of the various approaches.

The advocates of semiconductor integrated circuitry are already using the improved characteristics of thin-film resistors by applying such films directly to an active semiconductor substrate. Those advocating a technology based upon

The author

Dr. Gordon E. Moore is one of the new breed of electronic engineers, schooled in the physical sciences rather than in electronics. He earned a B.S. degree in chemistry from the

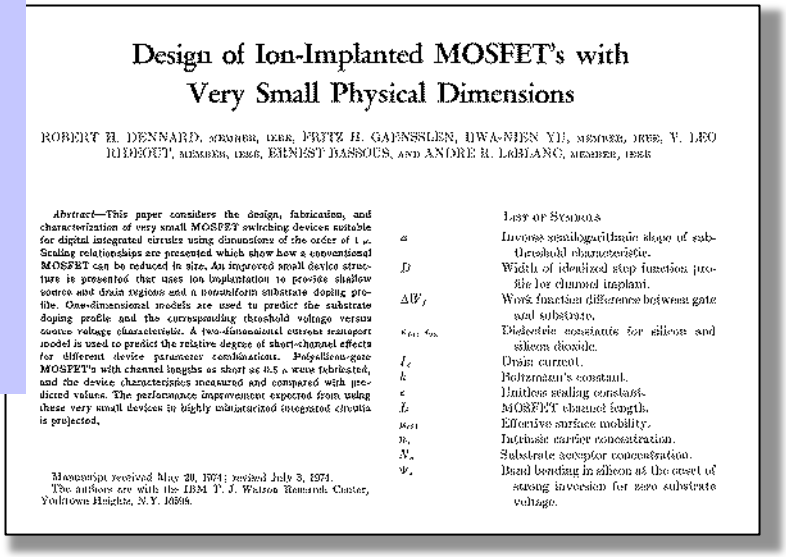
Moore's *Secret Sauce*: Dennard Scaling

Moore's Law put lots more transistors on a chip…but it's Dennard's Law that made them useful

Dennard Scaling :

- Decrease feature size by a factor of λ and decrease voltage by a factor of λ ; then
- # transistors increase by λ^2
- Clock speed increases by λ
- **Energy consumption does not change**

2x transistor count
40% faster
50% more efficient



Design of Ion-Implanted MOSFET's with Very Small Physical Dimensions

ROBERT H. DENNARD, MEMBER, IEEE, FRITZ H. GAENSSLEN, HWA-NIEN YU, MEMBER, IEEE, V. LEO RIDEOUT, MEMBER, IEEE, ERNEST BASSOUS, AND ANDRE R. LEBLANC, MEMBER, IEEE

Abstract—This paper considers the design, fabrication, and characterization of very small MOSFET switching devices suitable for digital integrated circuits using dimensions of the order of 1 μ. Scaling relationships are presented which show how a conventional MOSFET can be reduced in size. An improved small device structure is presented that uses ion implantation to provide shallow source and drain regions and a nonuniform substrate doping profile. One-dimensional models are used to predict the substrate doping profile and the corresponding threshold voltage versus source voltage characteristic. A two-dimensional current transport model is used to predict the relative degree of short-channel effects for different device parameter combinations. Polysilicon-gate MOSFET's with channel lengths as short as 0.5 μ were fabricated, and the device characteristics measured and compared with predicted values. The performance improvement expected from using these very small devices in highly miniaturized integrated circuits is projected.

Manuscript received May 20, 1974; revised July 3, 1974.
The authors are with the IBM T. J. Watson Research Center, Yorktown Heights, N.Y. 10598.

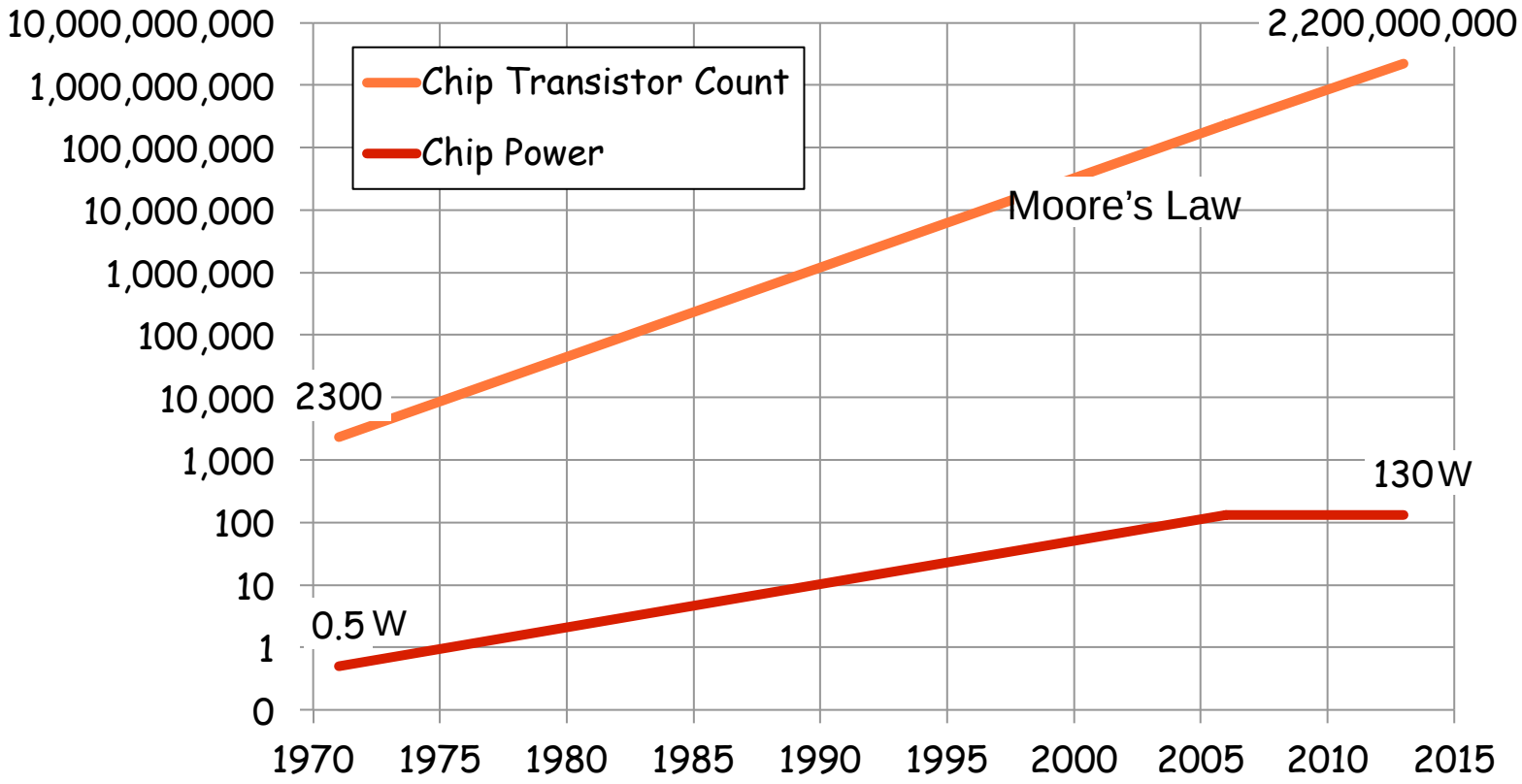
LIST OF SYMBOLS

a	Inverse semilogarithmic slope of subthreshold characteristic.
D	Width of idealized step function profile for channel implant.
ΔW_f	Work function difference between gate and substrate.
κ_{ox}, κ_s	Dielectric constants for silicon and silicon dioxide.
I_d	Drain current.
k	Boltzmann's constant.
ϵ	Unitless scaling constant.
L	MOSFET channel length.
μ_{eff}	Effective surface mobility.
n_i	Intrinsic carrier concentration.
N_a	Substrate acceptor concentration.
Ψ_s	Band bending in silicon at the onset of strong inversion for zero substrate voltage.

[Dennard, Gaensslen, Yu, Rideout, Bassous, Leblanc, **IEEE JSSC**, 1974]

Unfortunately Dennard Scaling is Over: What is the Catch?

Breakdown is the result of small feature sizes, current leakage poses greater challenges, and also causes the chip to heat up

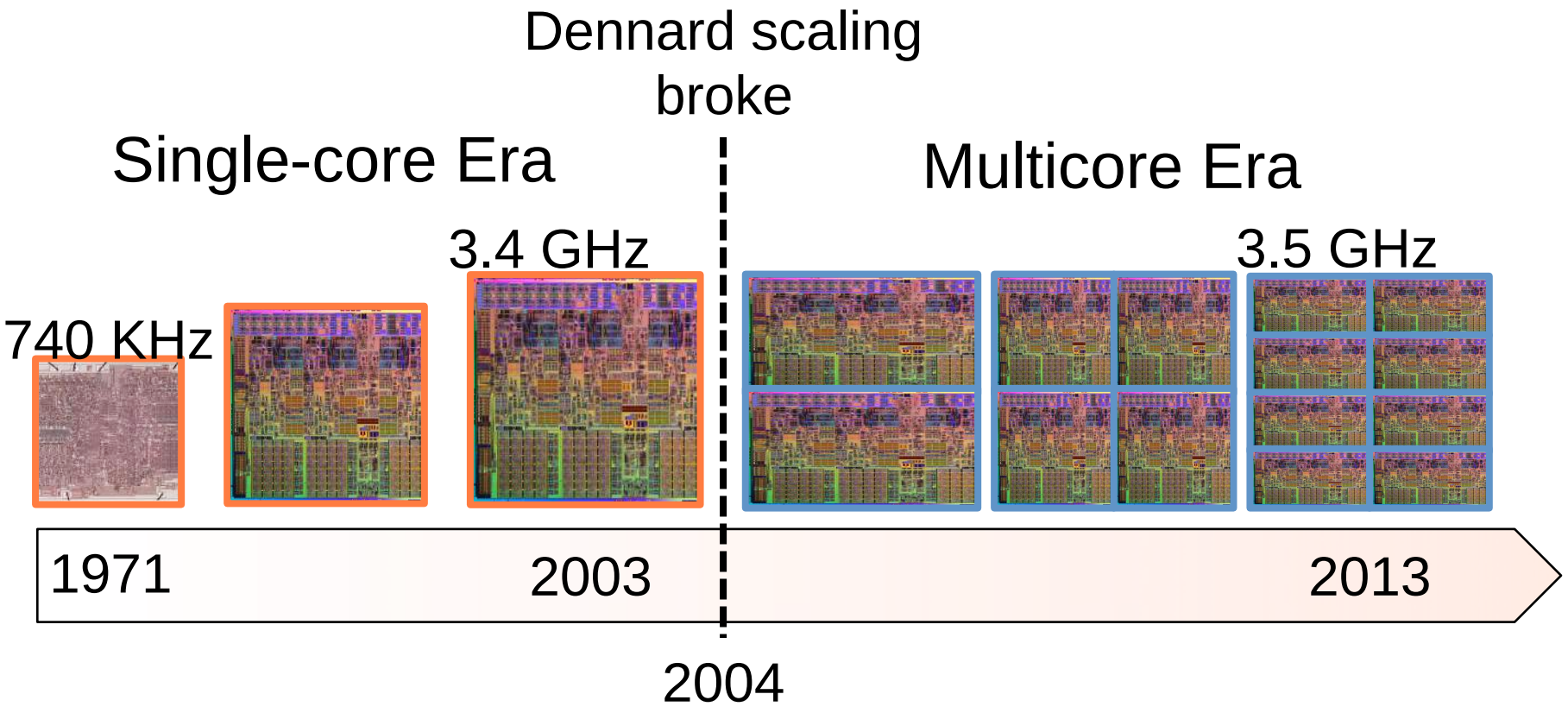


Powering the transistors without melting the chip

Dennard Scaling Over

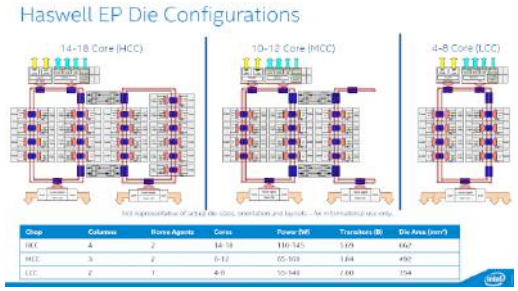
Evolution of processors

The primary reason cited for the breakdown is that at small sizes, current leakage poses greater challenges, and also causes the chip to heat up, which creates a threat of thermal runaway and therefore further increases energy costs.

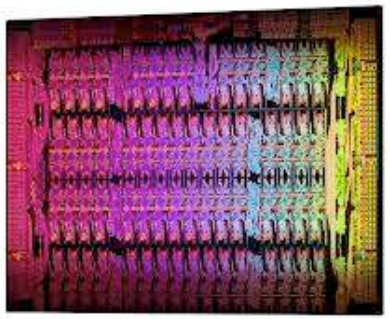


Today's Multicores

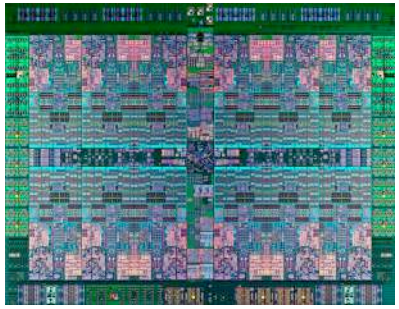
All of Top500 Systems Are Based on Multicore



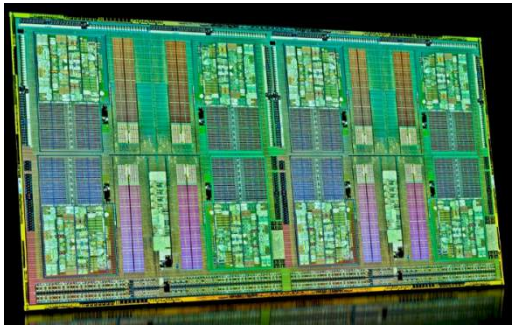
Intel Haswell (18 cores)



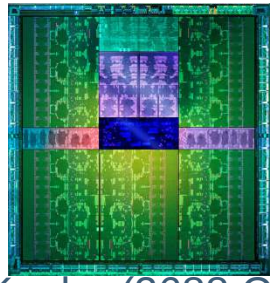
Intel Xeon Phi
(60 cores)



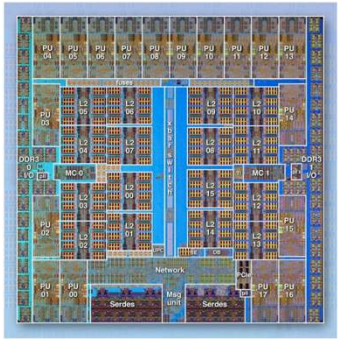
IBM Power 8 (12 cores)



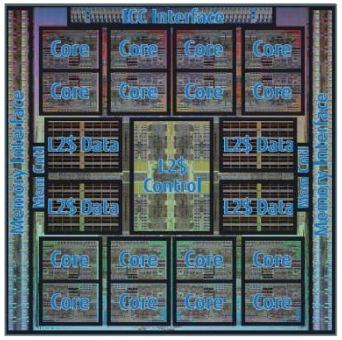
AMD Interlagos (16 cores)



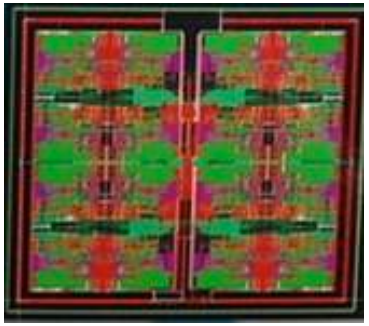
Nvidia Kepler (2688 Cuda cores)



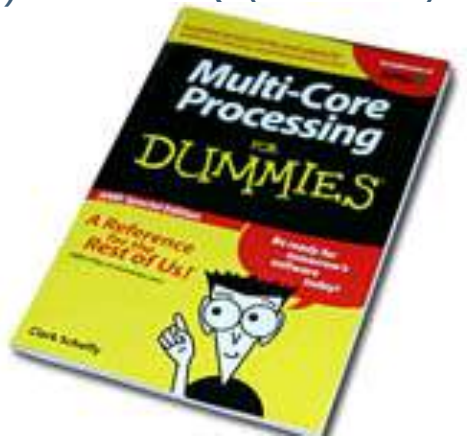
IBM BG/Q (18 cores)



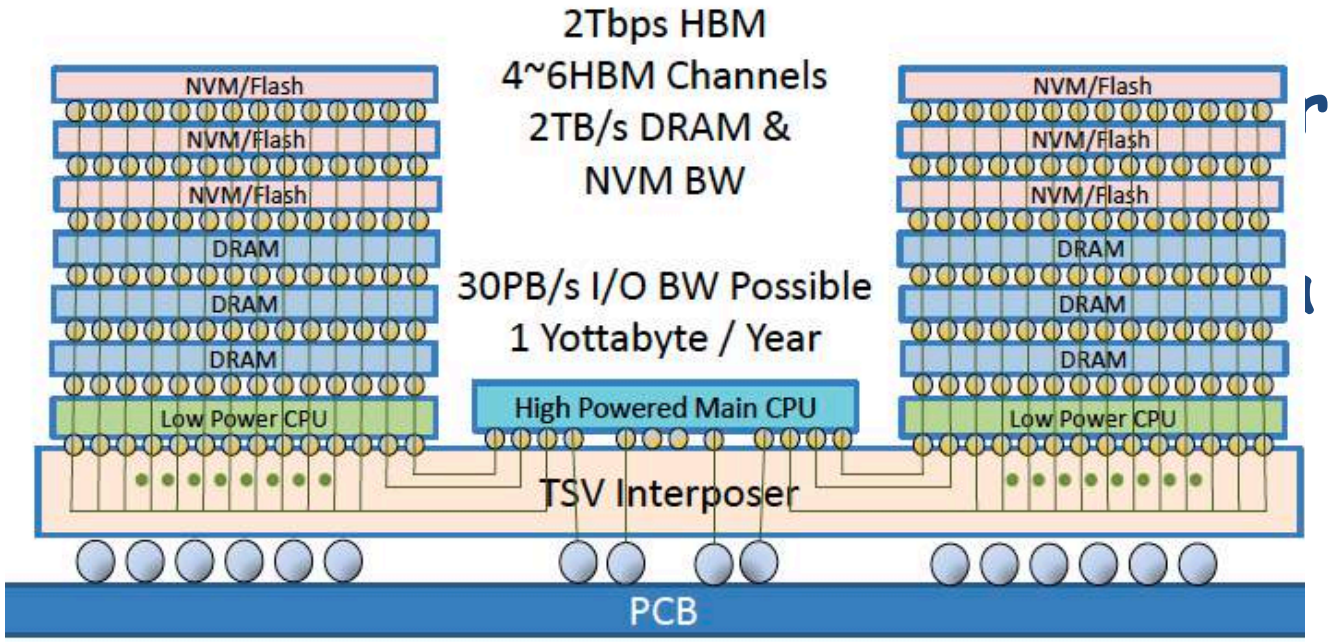
Fujitsu Venus (16 cores)



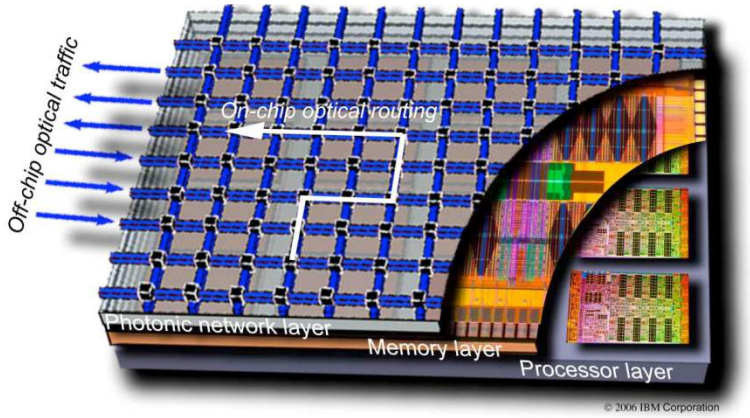
ShenWei (16 core)



Problem with Processors

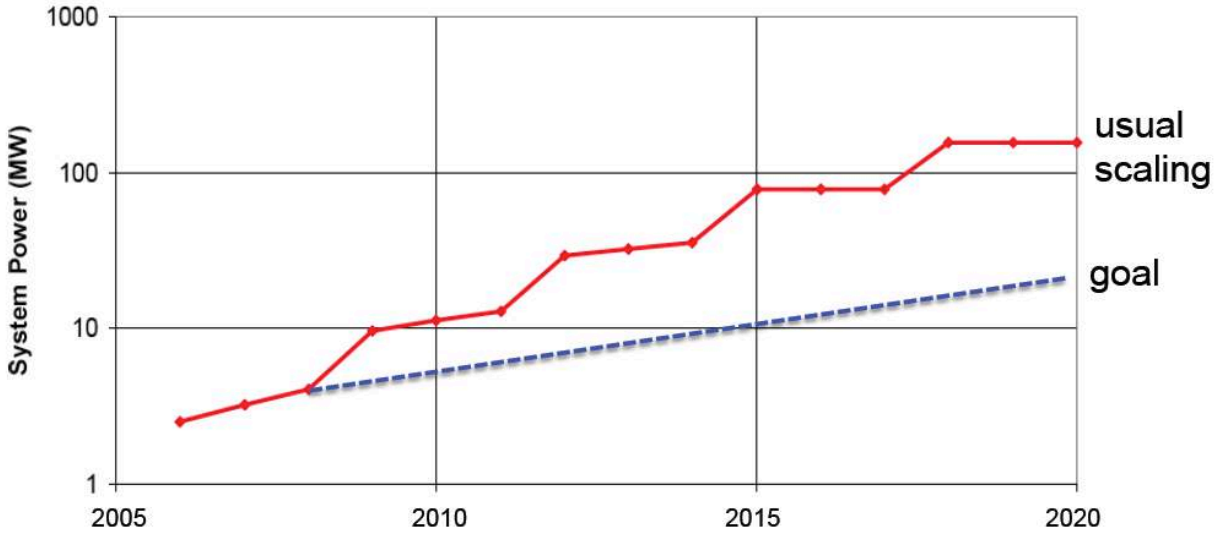


- Next generation will be more integrated, 3D design with a photonic network



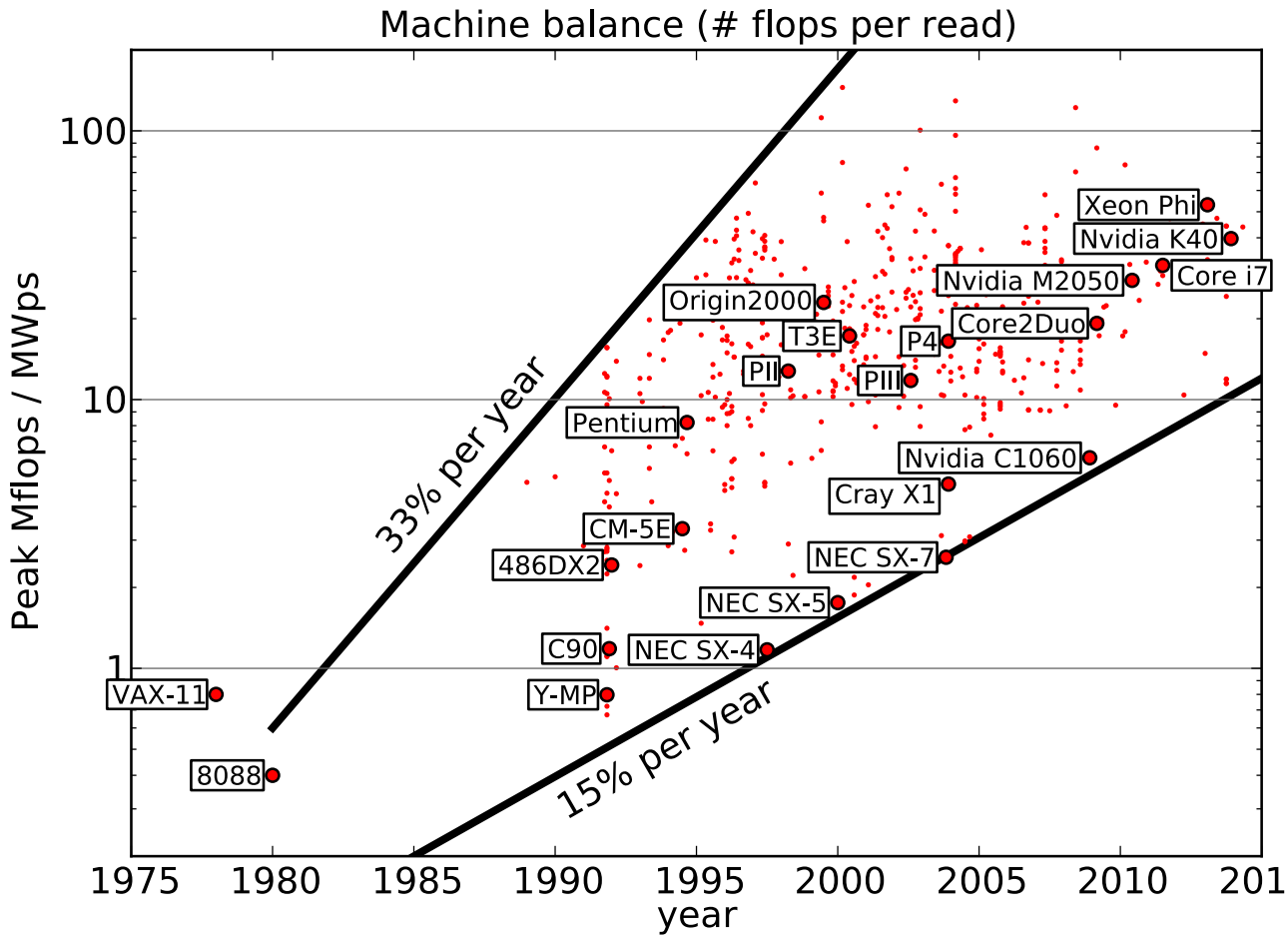
Energy Cost Challenge

- At ~$1M per MW energy costs are substantial
  - 10 Pflop/s in 2011 uses ~10 MWs
  - 1 Eflop/s in 2020 > 100 MWs



  - DOE Target: 1 Eflop/s around 2020-2022 at 20 MWs

Ratio of CPU speed to memory bandwidth increases 15-33% yearly



- Flops "free," memory expensive
- Good for dense, BLAS-3 operations (matrix multiply)

- Flops & memory access balanced
- Good for sparse & vector operations

Data from Stream benchmark (McCalpin) and vendor information sheets

Peak Performance - Per Core

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

Floating point operations per cycle per core

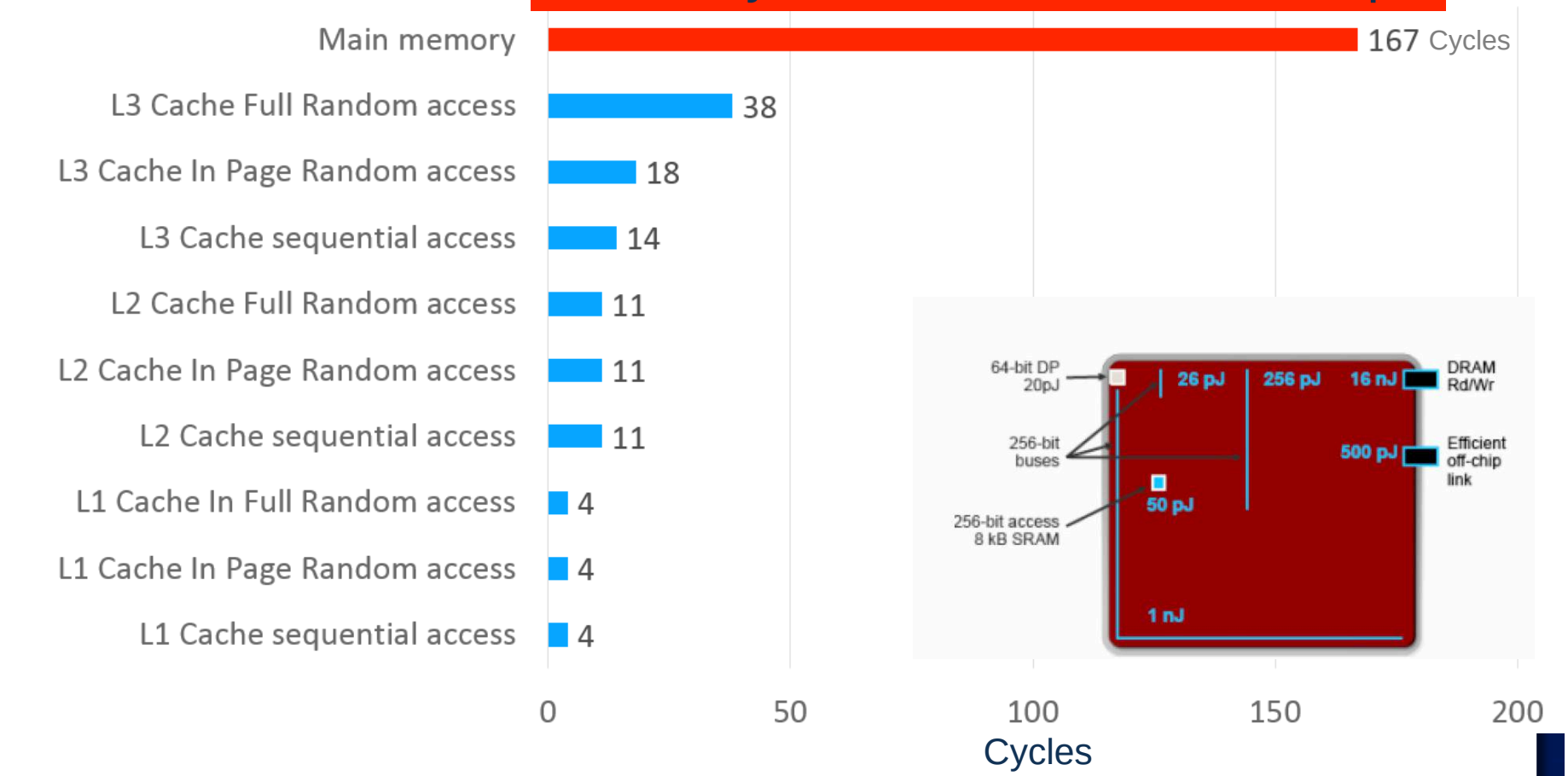
- Most of the recent computers have FMA (Fused multiple add): (i.e. x ←x + y*z in one cycle)
- Intel Xeon earlier models and AMD Opteron have SSE2
  - 2 flops/cycle DP & 4 flops/cycle SP
- Intel Xeon Nehalem ('09) & Westmere ('10) have AVX
  - 4 flops/cycle DP & 8 flops/cycle SP
- Intel Xeon Sandy Bridge('11) & Ivy Bridge ('12) have AVX & AVX2
  - 8 flops/cycle DP & 16 flops/cycle SP

We are here

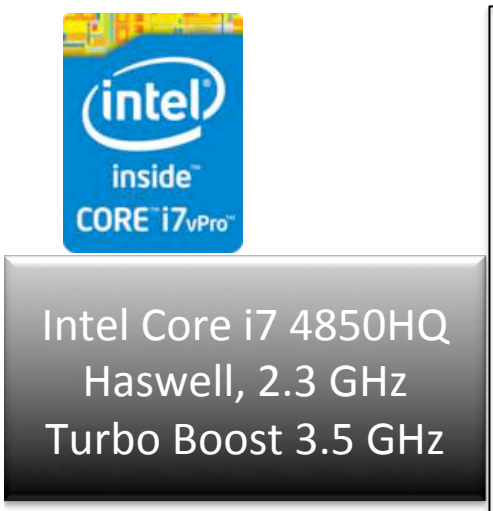
- Intel Xeon Haswell ('13) & (Broadwell ('14)) AVX2
  - 16 flops/cycle DP & 32 flops/cycle SP
  - Xeon Phi (per core) is at 16 flops/cycle DP & 32 flops/cycle SP
- Intel Xeon Skylake ('15)
  - 32 flops/cycle DL & 64 flops/cycle SP

CPU Access Latencies in Clock Cycles

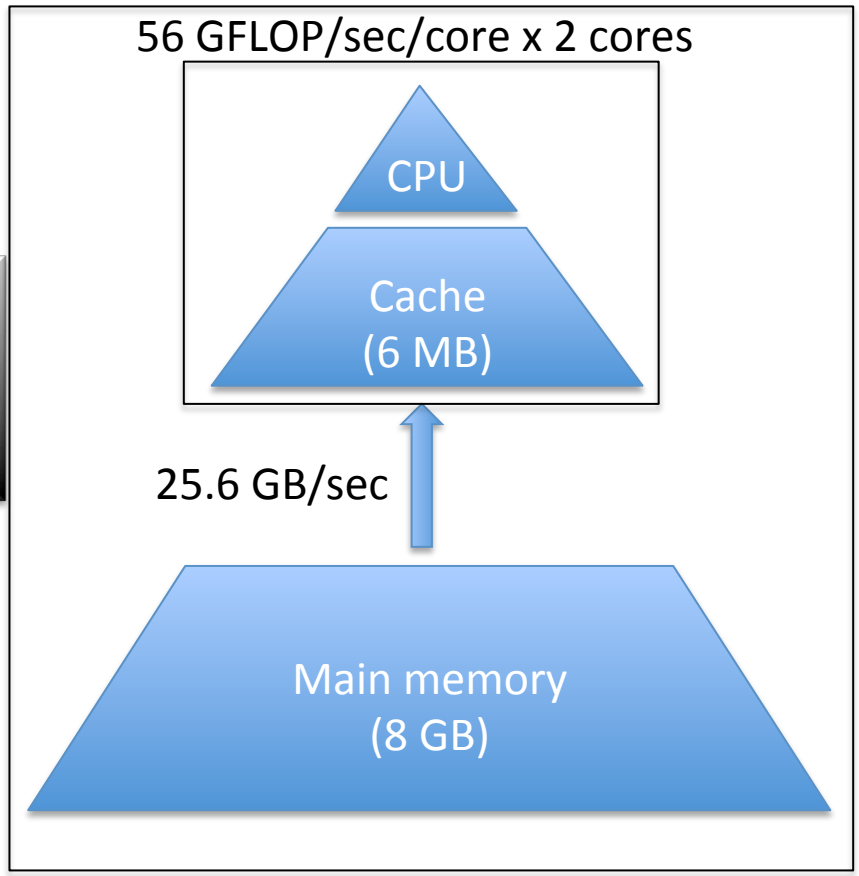
In 167 cycles can do 2672 DP Flops



Memory transfer
(Its All About Data Movement)
Example on my laptop: One level of memory



Intel Core i7 4850HQ
Haswell, 2.3 GHz
Turbo Boost 3.5 GHz

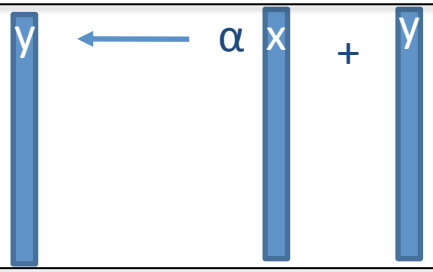


(Omitting latency here.)

The model IS simplified (see next slide) but it provides an upper bound on performance as well. I.e., we will never go faster than what the model predicts.
(And, of course, we can go slower …)

FMA: fused multiply-add

AXPY:

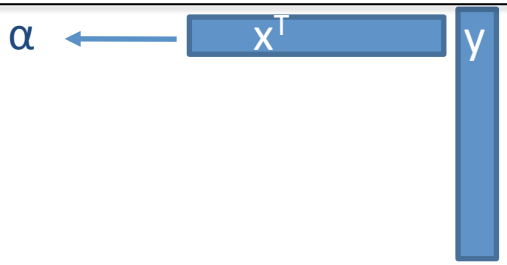



```
for ( j = 0; j < n; j++)
      y[i] += a * x[i];
```

(without increment)

n MUL
n ADD
2n FLOP
n FMA

DOT:




```
alpha = 0e+00;
for ( j = 0; j < n; j++)
      alpha += x[i] * y[i];
```

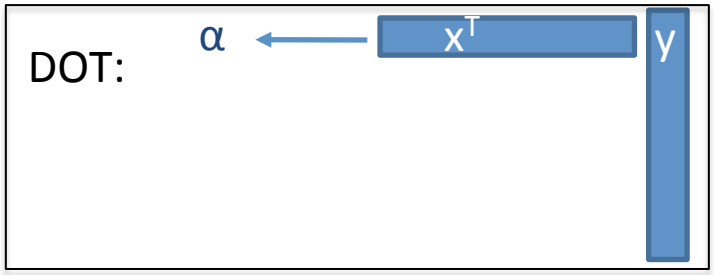
(without increment)

n MUL
n ADD
2n FLOP
n FMA

Note: It is reasonable to expect the one loop codes shown here to perform as well as their Level 1 BLAS counterpart (on multicore with an OpenMP pragma for example).

The true gain these days with using the BLAS is (1) Level 3 BLAS, and (2) portability.

- Take two double precision vectors x and y of size n=375,000.



- Data size:
  - (375,000 double) * (8 Bytes / double) = 3 MBytes per vector

  (Two vectors fit in cache (6 MBytes). OK.)

- Time to move the vectors from memory to cache:
  - (6 MBytes) / (25.6 GBytes/sec) = **0.23 ms**
- Time to perform computation of DOT:
  - (2n flop) / (56 Gflop/sec) = **0.01 ms**

Vector Operations

total_time ≥ max (time_comm , time_comp)

= max (0.23ms , 0.01ms) = 0.23ms

Performance = (2 x 375,000 flops)/.23ms = 3.2 Gflop/s

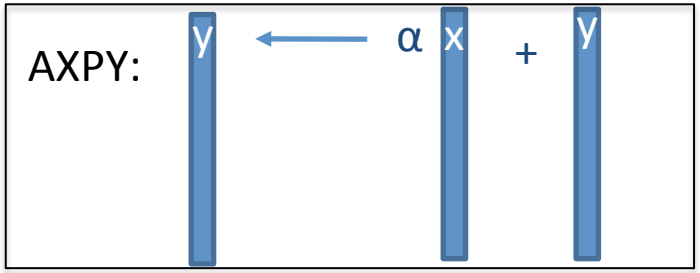
Performance for DOT ≤ 3.2 Gflop/s

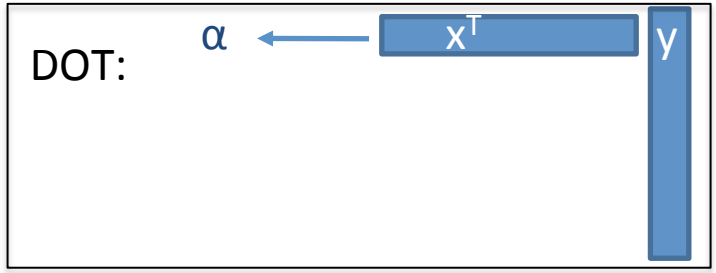
Peak is 56 Gflop/s

We say that the operation is communication bounded. No reuse of data.

Level 1, 2 and 3 BLAS

Level 1 BLAS Matrix-Vector operations

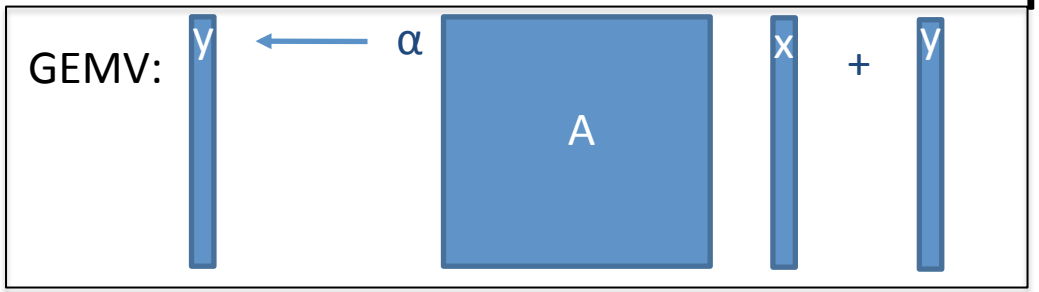




2n FLOP
2n memory reference
AXPY: 2n READ, n WRITE
DOT: 2n READ

RATIO: 1

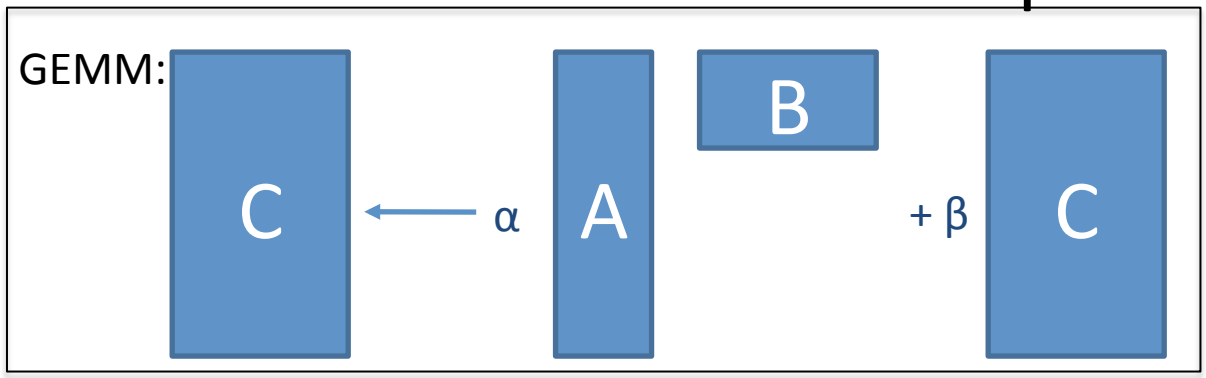
Level 2 BLAS Matrix-Vector operations



$2n^2$ FLOP
n^2 memory references

RATIO: 2

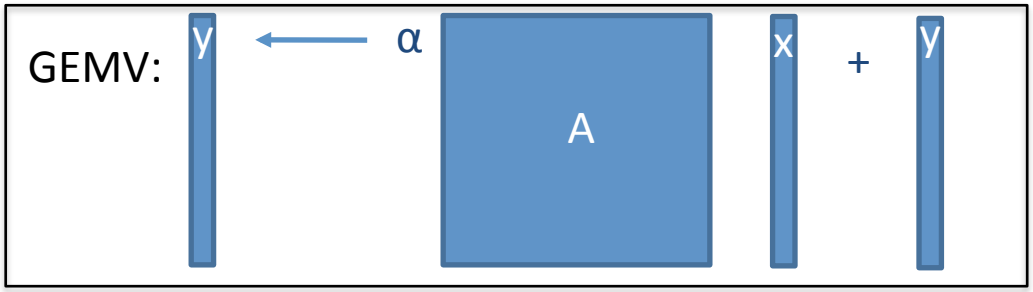
Level 3 BLAS Matrix-Matrix operations



$2n^3$ FLOP
$3n^2$ memory references
$3n^2$ READ, n^2 WRITE

RATIO: 2/3 n

- Double precision matrix A and vectors x and y of size n=860.



- Data size:
  - (860^2 + 2*860 double) * (8 Bytes / double) ~ 6 MBytes

  Matrix and two vectors fit in cache (6 MBytes).

- Time to move the data from memory to cache:
  - (6 MBytes) / (25.6 GBytes/sec) = **0.23 ms**

- Time to perform computation of DOT:
  - ($2n^2$ flop) / (56 Gflop/sec) = **0.26 ms**

Matrix - Vector Operations

total_time ≥ max (time_comm , time_comp)

= max (0.23ms , 0.26ms) = 0.26ms

Performance = (2 x 860^2 flops)/.26ms = 5.7 Gflop/s

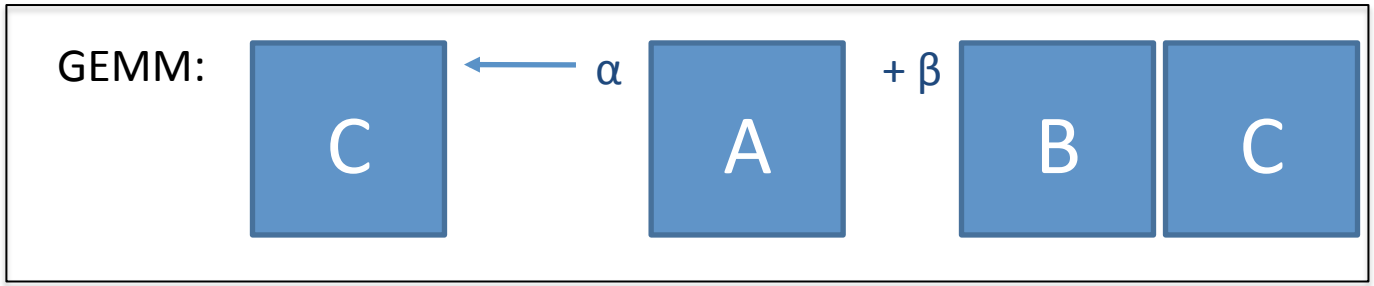
Performance for GEMV ≤ 5.7 Gflop/s

Performance for DOT ≤ 3.2 Gflop/s

Peak is 56 Gflop/s

We say that the operation is communication bounded. Very little reuse of data.

- Take two double precision vectors x and y of size n=500.



- Data size:
  - (500^2 double) * (8 Bytes / double) = 2 MBytes per matrix

  (Three matrices fit in cache (6 MBytes). OK.)

- Time to move the matrices in cache:
  - (6 MBytes) / (25.6 GBytes/sec) = **0.23 ms**

- Time to perform computation in GEMM:
  - ($2n^3$ flop) / (56 Gflop/sec) = **4.46 ms**

Matrix Matrix Operations

total_time ≥ max (time_comm , time_comp)

= max(0.23ms , 4.46ms) = 4.46ms

For this example, communication time is less than 6% of the computation time.

Performance = (2 x 500^3 flops)/4.69ms = 53.3 Gflop/s

There is a lots of data reuse in a GEMM; 2/3n per data element. Has good temporal locality.

If we assume total_time ≈ time_comm +time_comp, we get

Performance for GEMM ≈ 53.3 Gflop/sec

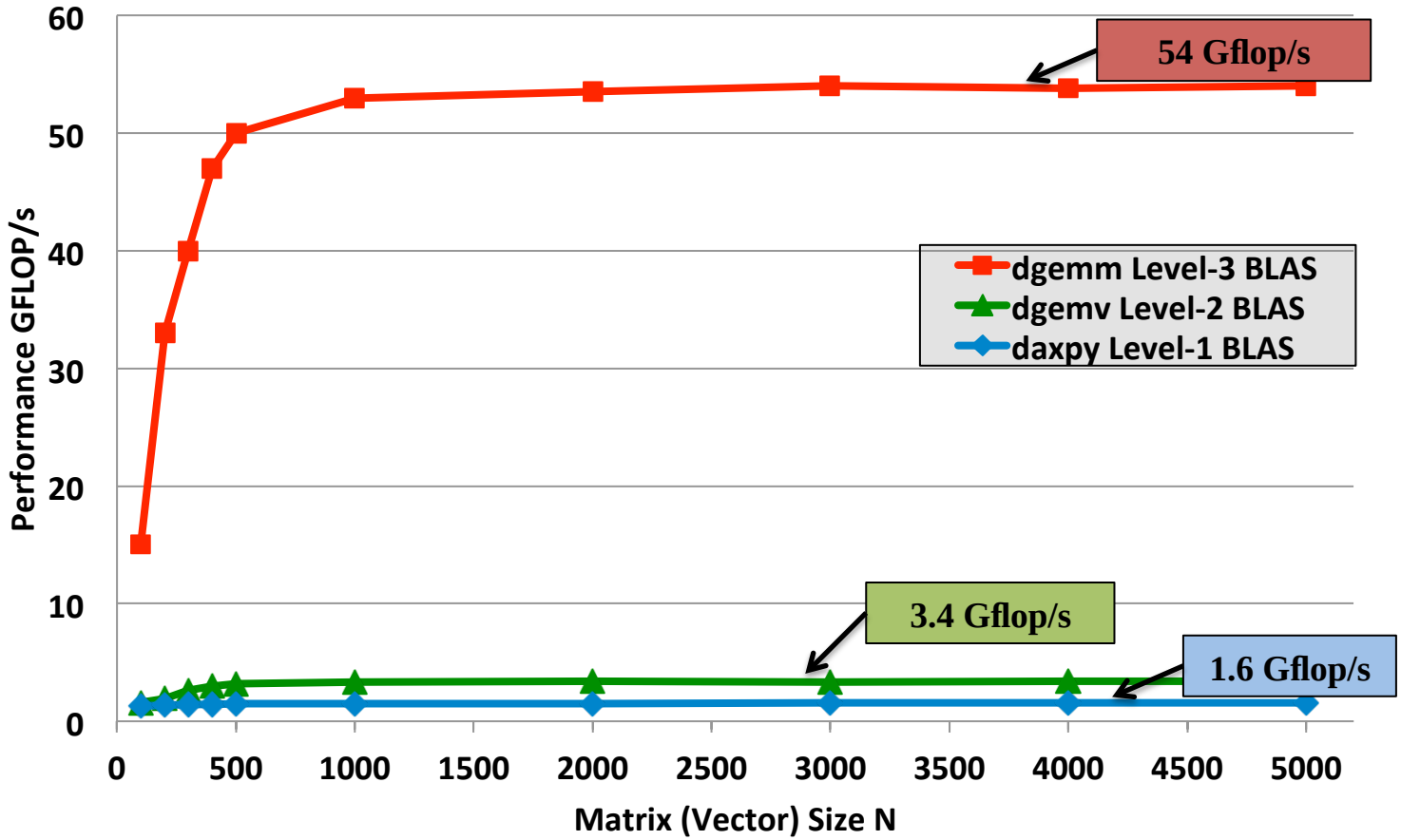
Performance for DOT ≤ 3.2 Gflop/s

Performance for GEMV ≤ 5.7 Gflop/s

(Out of 56 Gflop/sec possible, so that would be 95% peak performance efficiency.)

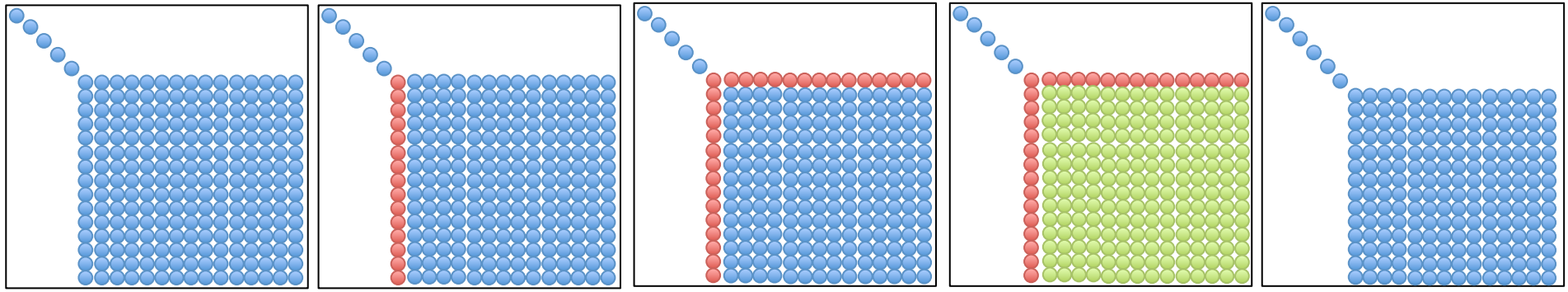
Level 1, 2 and 3 BLAS

1 core Intel Haswell i7-4850HQ, 2.3 GHz (Turbo Boost at 3.5 GHz);
Peak = 56 Gflop/s



1 core Intel Haswell i7-4850HQ, 2.3 GHz, Memory: DDR3L-1600MHz
6 MB shared L3 cache, and each core has a private 256 KB L2 and 64 KB L1.
The theoretical peak per core double precision is 56 Gflop/s per core.
Compiled with gcc and using Veclib

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



Factor column with Level 1 BLAS

Divide by Pivot row

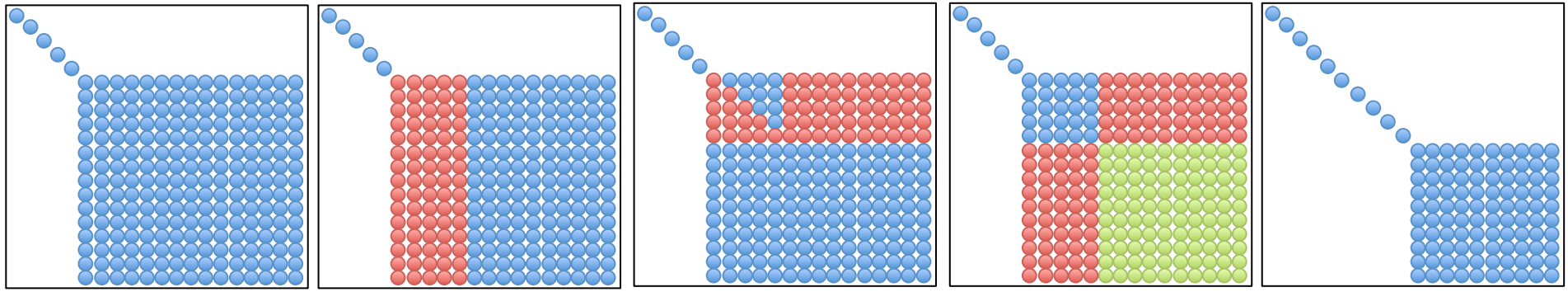
Schur complement update (Rank 1 update)

Next Step

Main points

- Factorization column (zero) mostly sequential due to memory bottleneck
- Level 1 BLAS
- Divide pivot row has little parallelism
- Rank -1 Schur complement update is the only easy parallelize task
- Partial pivoting complicates things even further
- Bulk synchronous parallelism (fork-join)
  - Load imbalance
  - Non-trivial Amdahl fraction in the panel
  - Potential workaround (look-ahead) has complicated implementation

The Standard LU Factorization LAPACK 1980's HPC of the Day: Cache Based SMP



Factor panel with Level 1,2 BLAS

Triangular update

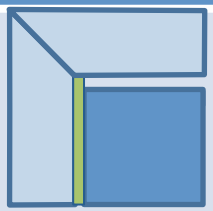
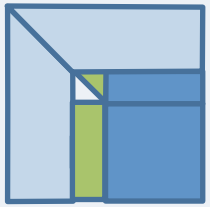
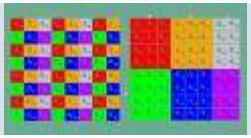
Schur complement update

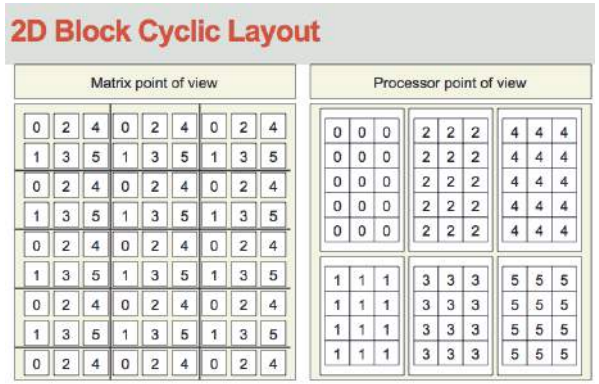
Next Step

Main points

- Panel factorization mostly sequential due to memory bottleneck
- Triangular solve has little parallelism
- Schur complement update is the only easy parallelize task
- Partial pivoting complicates things even further
- Bulk synchronous parallelism (fork-join)
  - Load imbalance
  - Non-trivial Amdahl fraction in the panel
  - Potential workaround (look-ahead) has complicated implementation

Last Generations of DLA Software

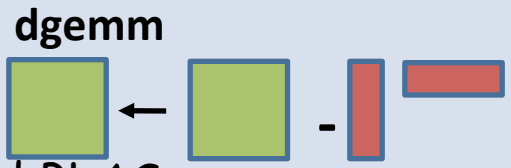
Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing

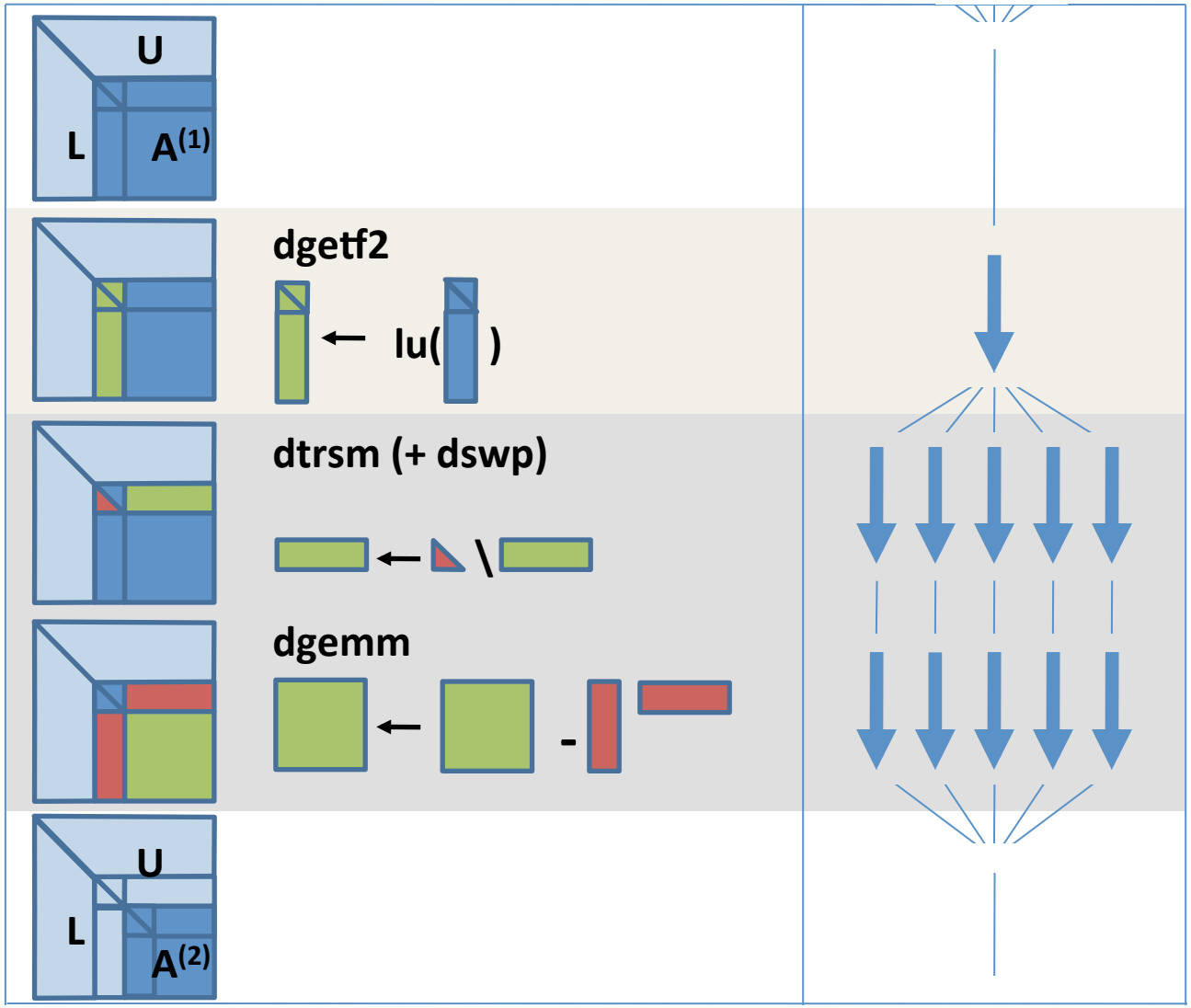


Parallelization of LU and QR.

Parallelize the update:

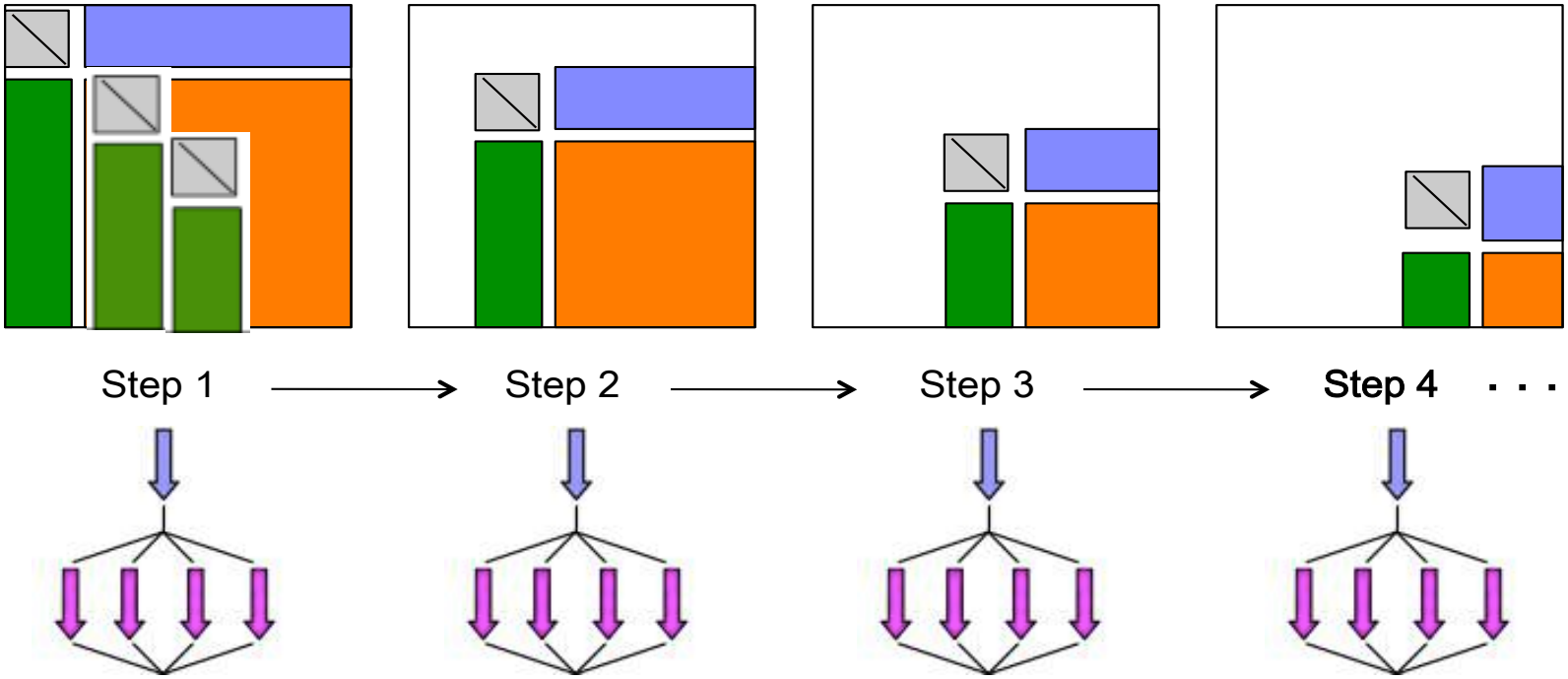
- Easy and done in any reasonable software.
- This is the $2/3n^3$ term in the FLOPs count.
- Can be done efficiently with LAPACK+multithreaded BLAS

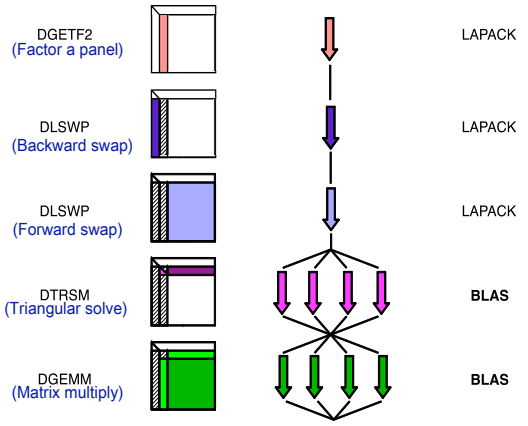




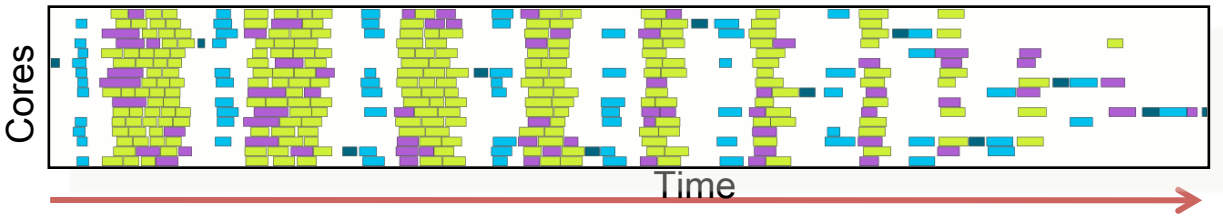
Fork - Join parallelism
Bulk Sync Processing

Synchronization (in LAPACK LU)





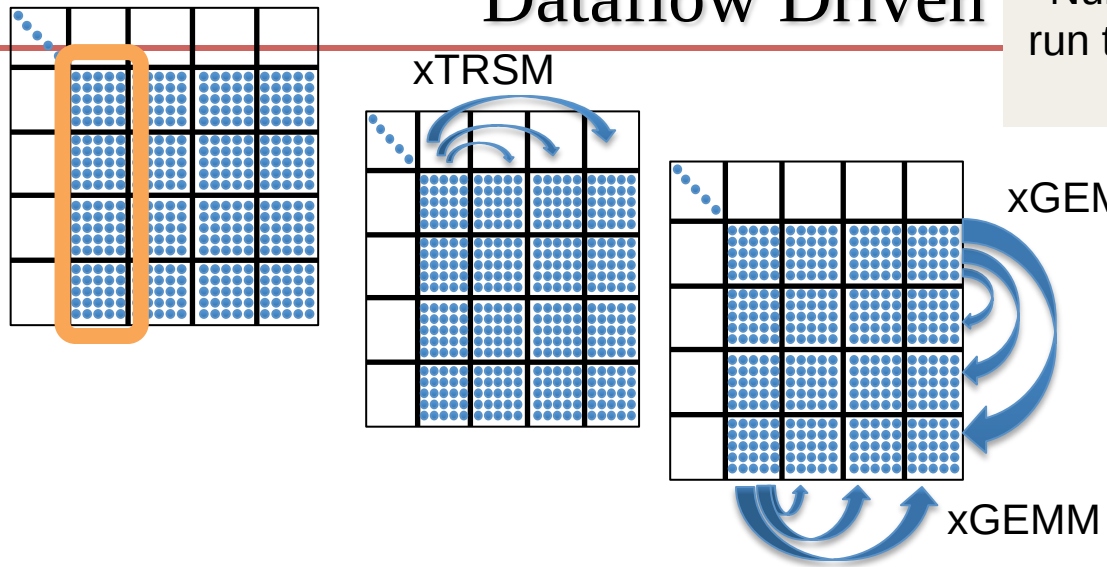
- fork join
- bulk synchronous processing



PLASMA LU Factorization

Dataflow Driven

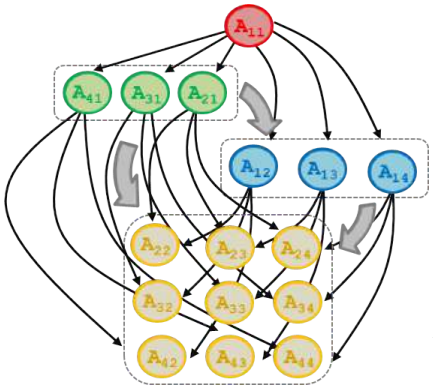
Numerical program generates tasks and run time system executes tasks respecting data dependences.



Sparse / Dense Matrix System

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

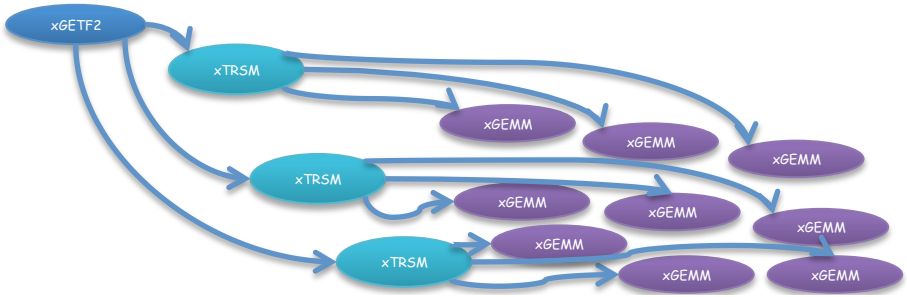
DAG-based factorization



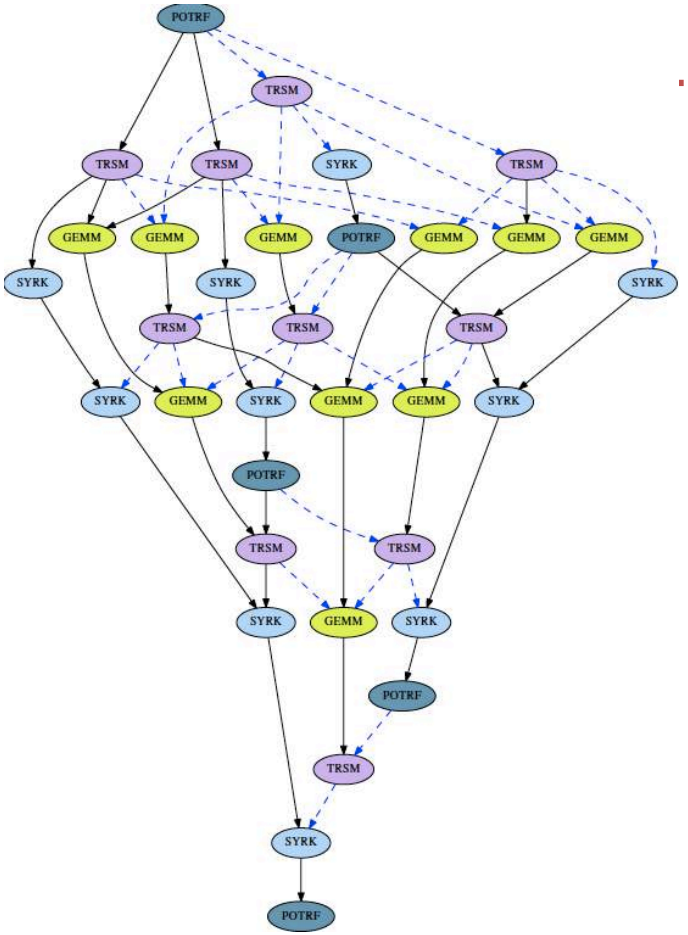
Batched LA

- **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



QUARK



A runtime environment for the dynamic execution of precedence-constraint tasks (DAGs) in a multicore machine

- Translation
- If you have a serial program that consists of computational kernels (tasks) that are related by data dependencies, QUARK can help you execute that program (relatively efficiently and easily) in parallel on a multicore machine

QUARK

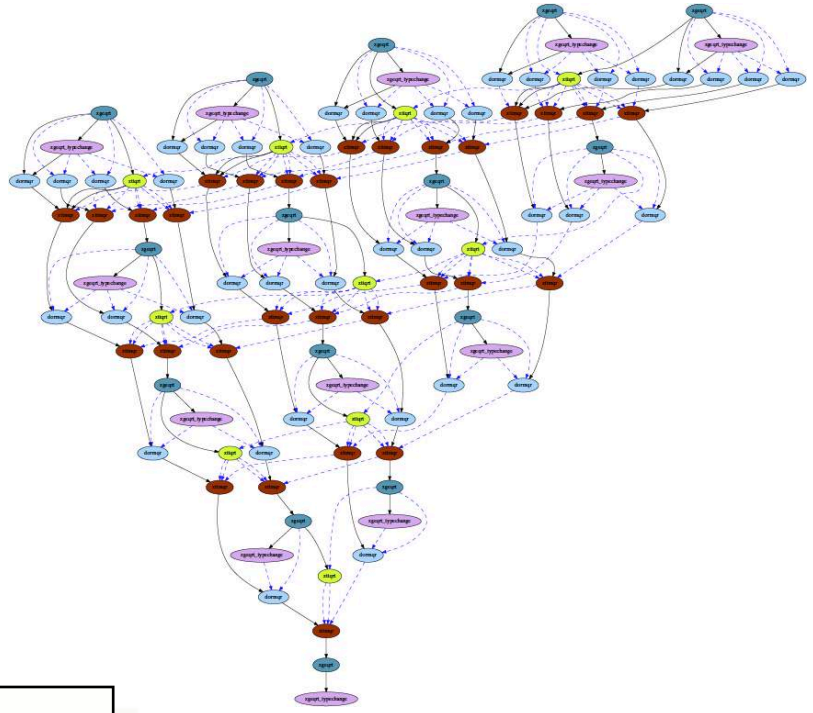
Shared Memory Superscalar Scheduling

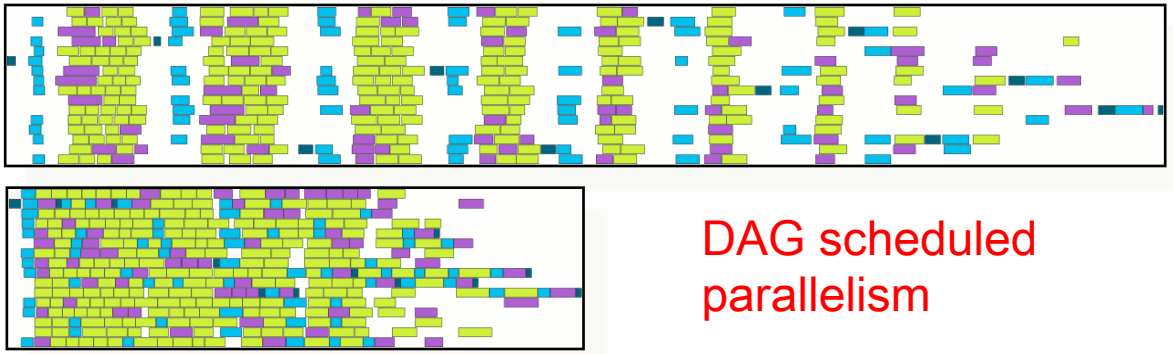
```
FOR k = 0..TILES-1
   A[k][k] ← DPOTRF(A[k][k])
   FOR m = k+1..TILES-1
      A[m][k] ← DTRSM(A[k][k], A[m][k])
   FOR m = k+1..TILES-1
      A[m][m] ← DSYRK(A[m][k], A[m][m])
      FOR n = k+1..m-1
         A[m][n] ← DGEMM(A[m][k], A[n][k], A[m][n])
```

definition – pseudocode

The Purpose of a QUARK Runtime

- 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

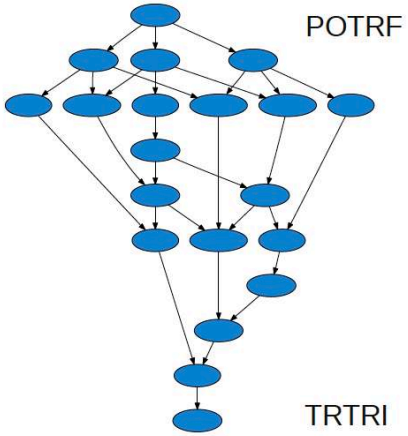


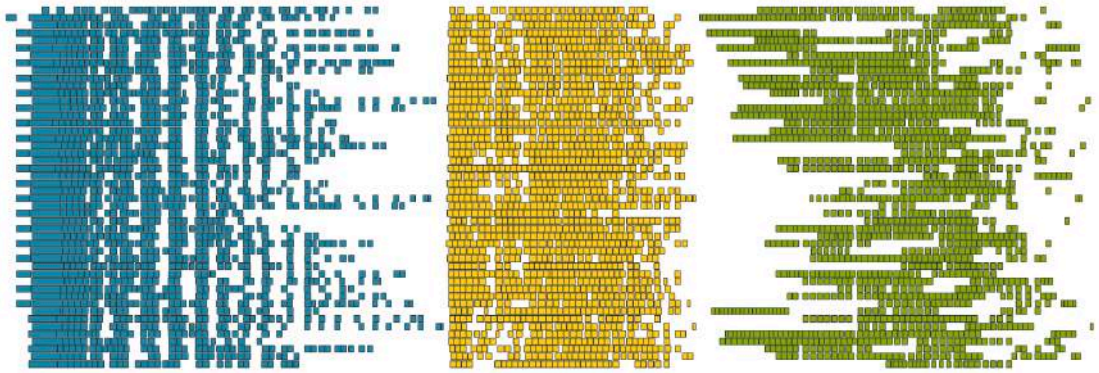


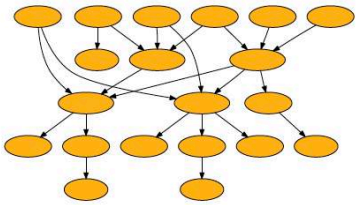
DAG scheduled parallelism

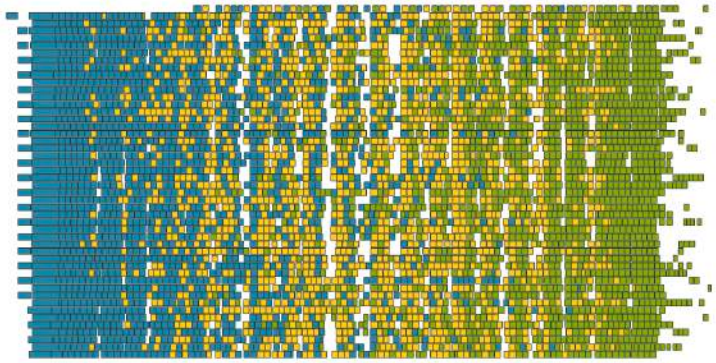
Fork-join parallelism
Notice the synchronization penalty in the presence of heterogeneity.

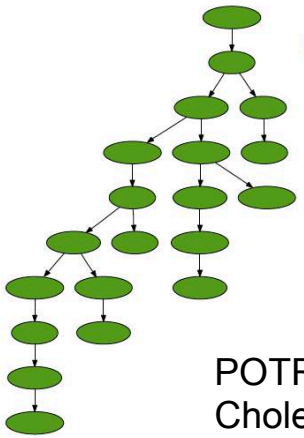
Pipelining: Cholesky Inversion
3 Steps: Factor, Invert L, Multiply L's

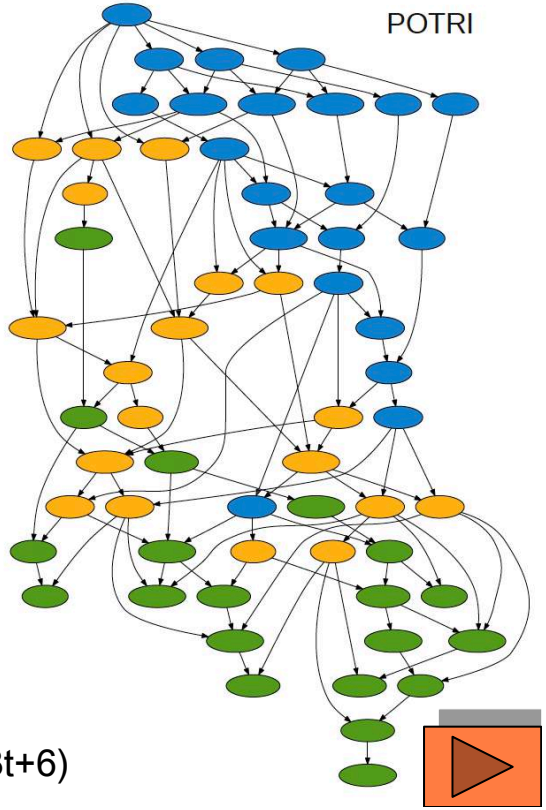










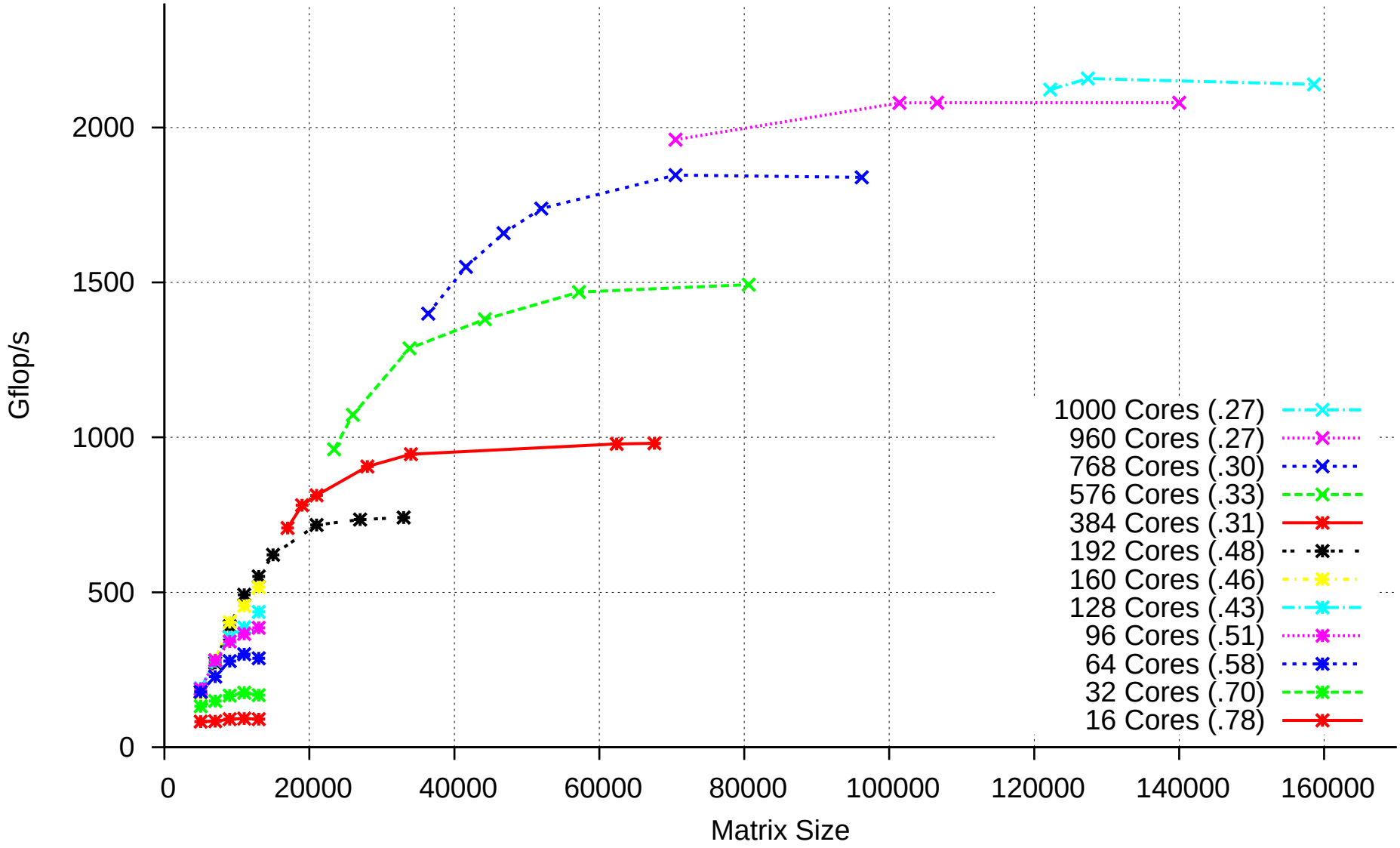


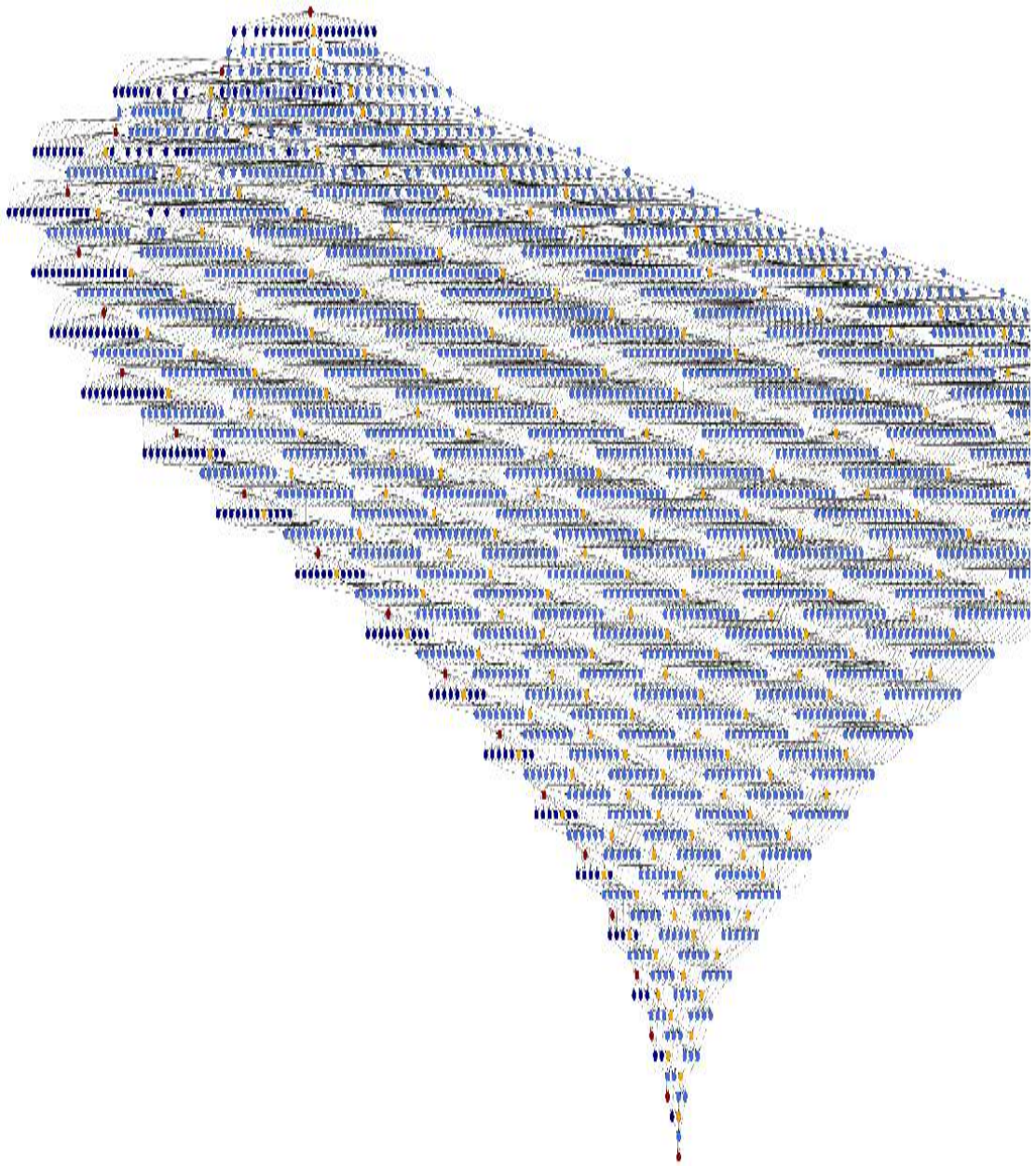
48 cores
POTRF, TRTRI and LAUUM.
The matrix is 4000 x 4000,tile size is 200 x 200,

POTRF+TRTRI+LAUUM: 25 (7t-3)
Cholesky Factorization alone: 3t-2

Pipelined: 18 (3t+6)

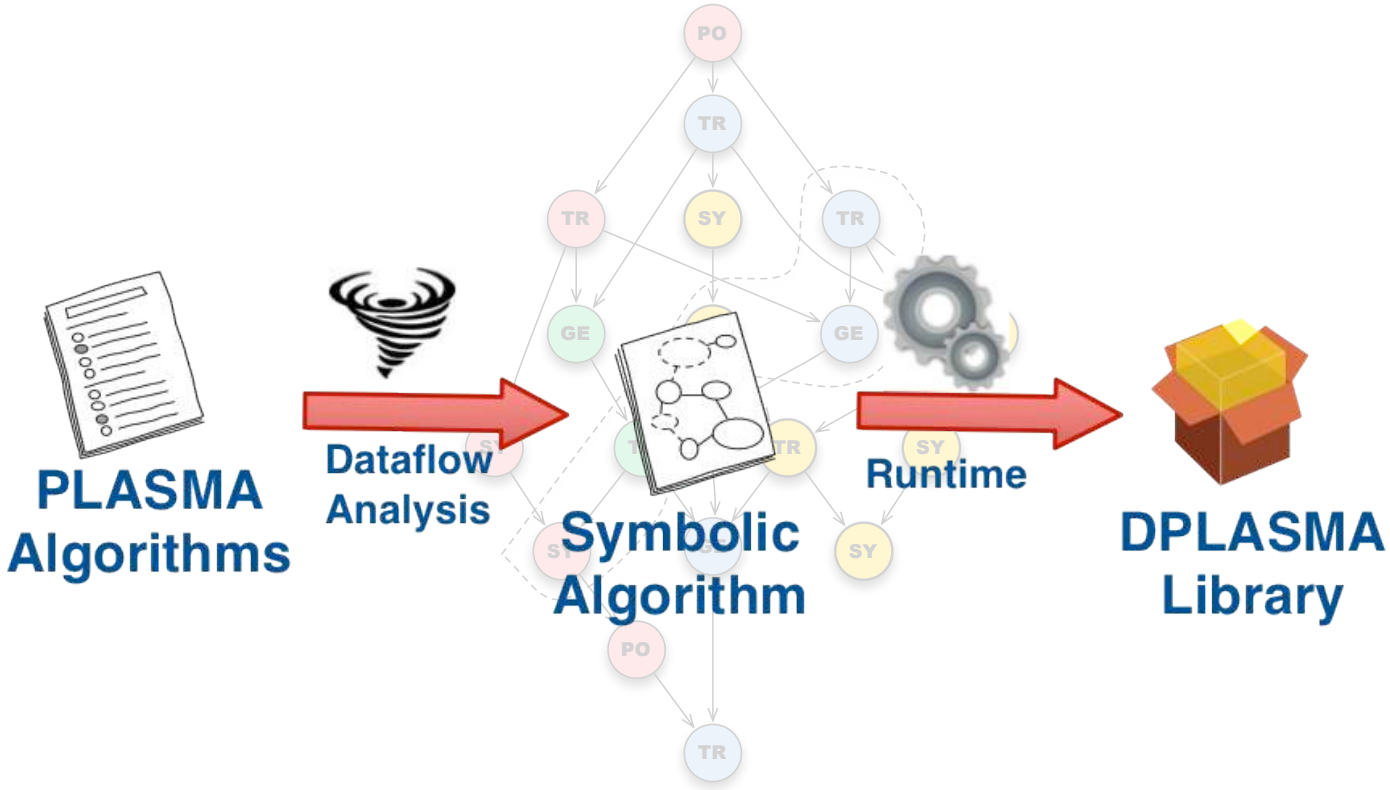
Performance of PLASMA Cholesky, Double Precision
Comparing Various Numbers of Cores (Percentage of Theoretical Peak)
1024 Cores (64 x 16-cores) 2.00 GHz Intel Xeon X7550, 8,192 Gflop/s Peak (Double Precision) [nautilus]
Static Scheduling





- DAG too large to be generated ahead of time
  - Generate it dynamically
  - Merge parameterized DAGs with dynamically generated DAGs
- HPC is about distributed heterogeneous resources
  - Have to be able to move the data across multiple nodes
  - The scheduling cannot be centralized
  - Take advantage of all available resources with minimal intervention from the user
- Facilitate the usage of the HPC resources
  - Languages
  - Portability

DPLASMA: Going to Distributed Memory

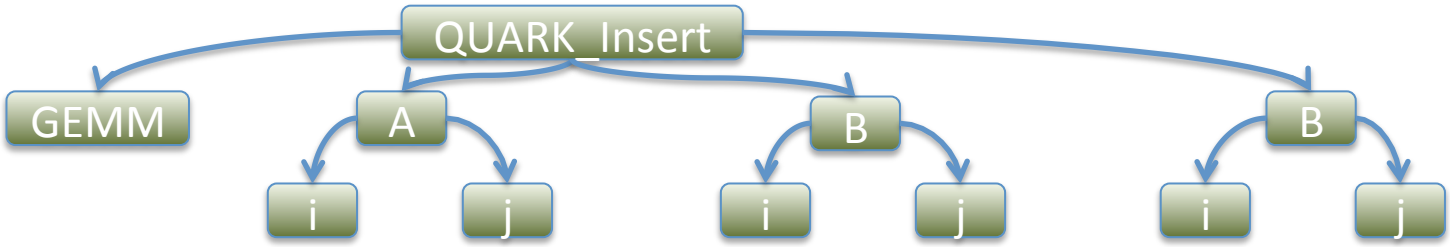


Start with PLASMA

```
for i,j = 0..N
```

```
QUARK_Insert( GEMM,  A[i, j],INPUT,   B[j, i],INPUT,  C[i,i],INOUT )
QUARK_Insert( TRSM,  A[i, j],INPUT,   B[j, i],INOUT )
```

Parse the C source code to Abstract Syntax Tree



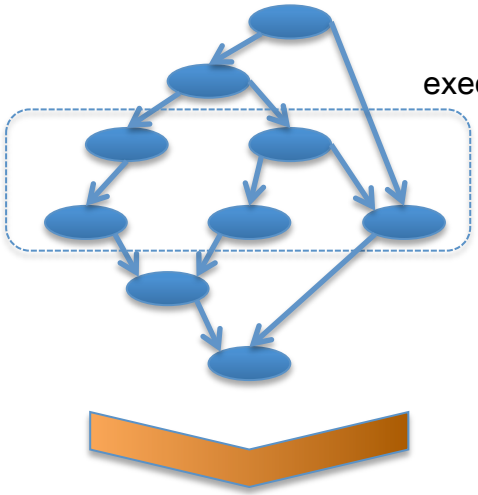
Analyze dependencies with Omega Test

```
{ 1 < i < N : GEMM(i, j) => TRSM(j) }
```

Loops & array references have to be affine

Generate Code which has the Parameterized DAG

PLASMA
(On Node)

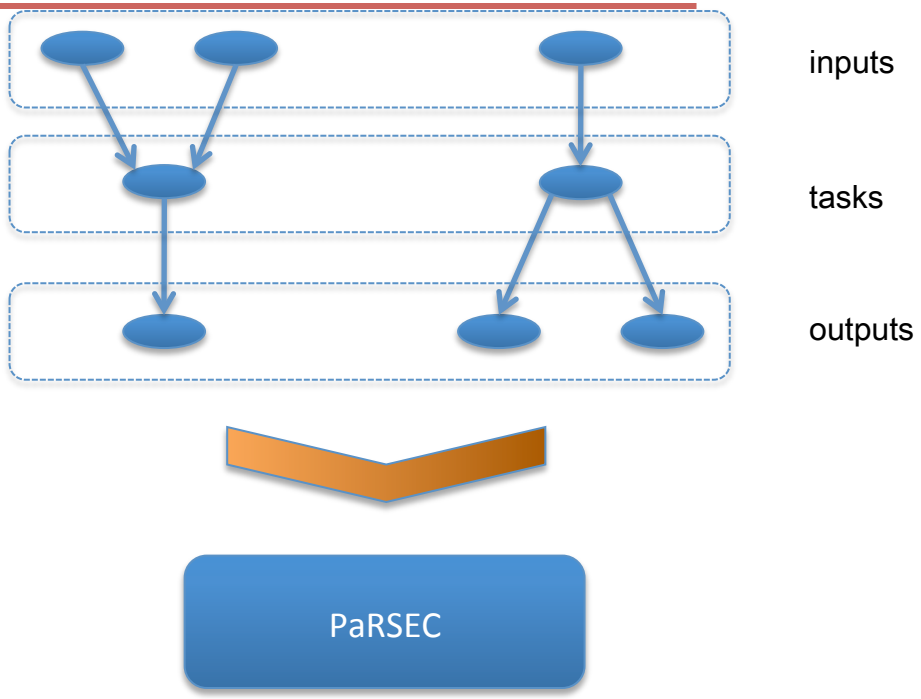


Number of tasks in DAG:

$O(n^3)$

Cholesky: 1/3 n^3
LU: 2/3 n^3
QR: 4/3 n^3

DPLASMA
(Distributed System)



Number of tasks in parameterized DAG:

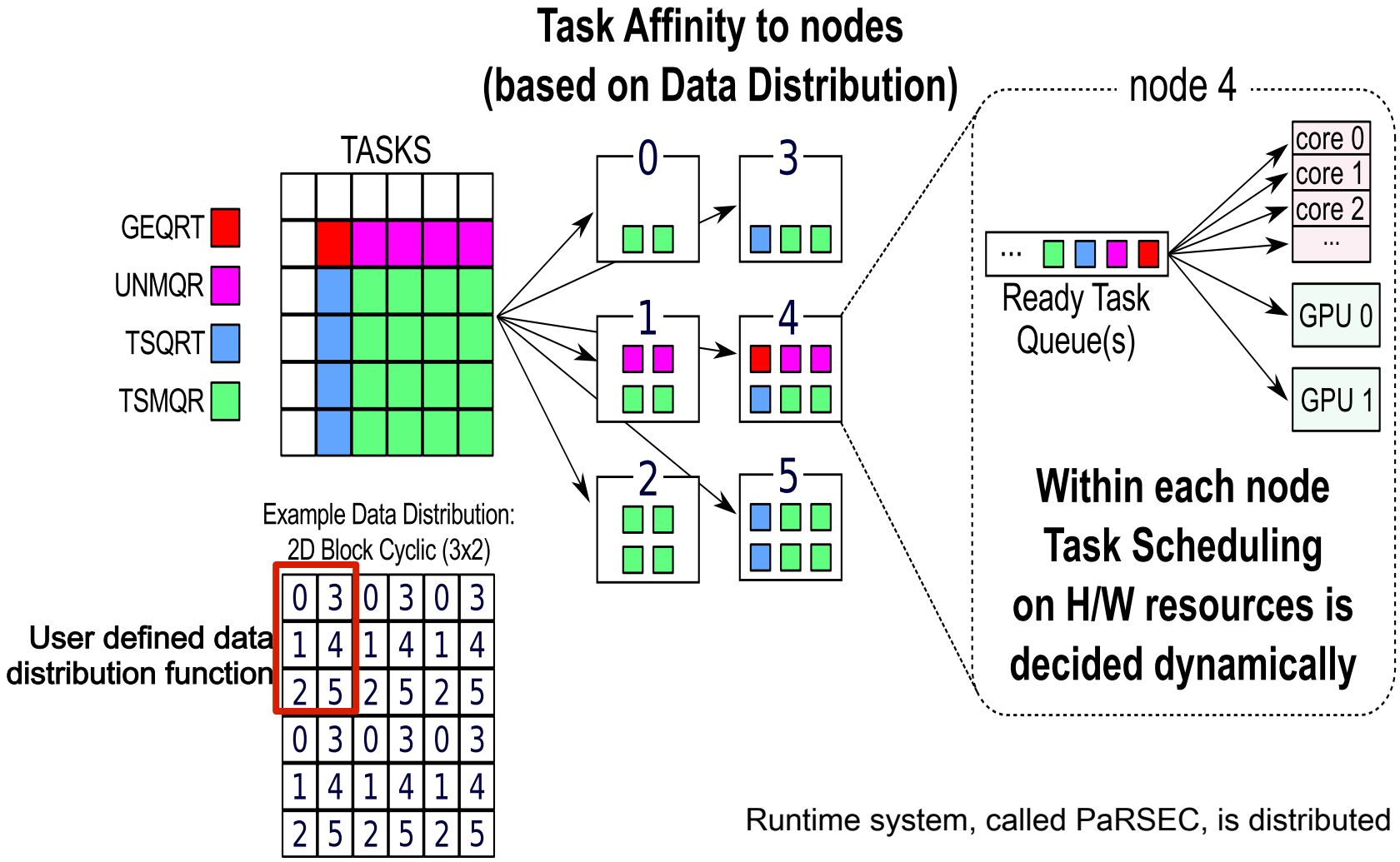
O(1)

Cholesky: 4 (POTRF, SYRK, GEMM, TRSM)
LU: 4 (GETRF, GESSM, TSTRF, SSSSM)
QR: 4 (GEQRT, LARFB, TSQRT, SSRFB)

DAG: Conceptualized & Parameterized

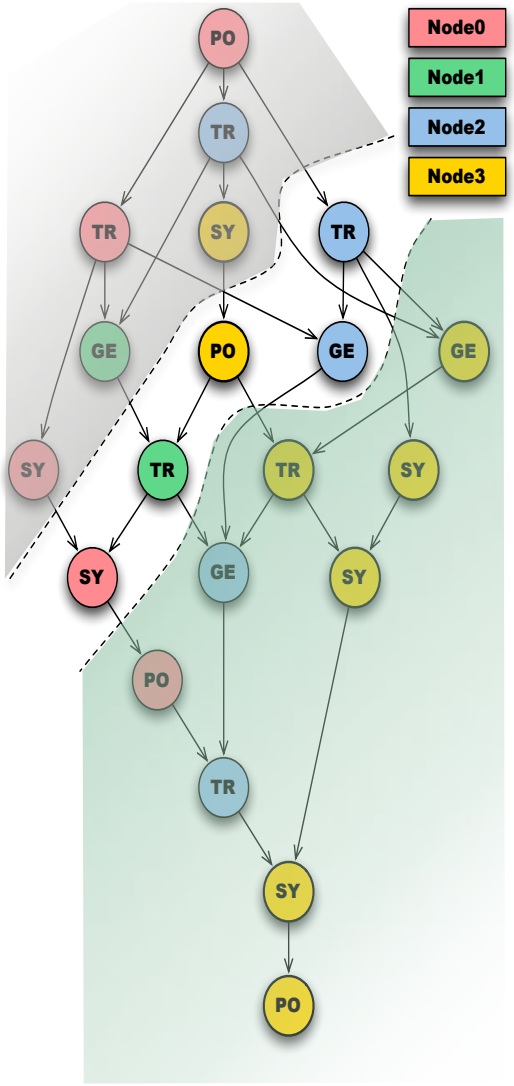
small enough to store on each core in every node = Scalable

Task Affinity in DPLASMA

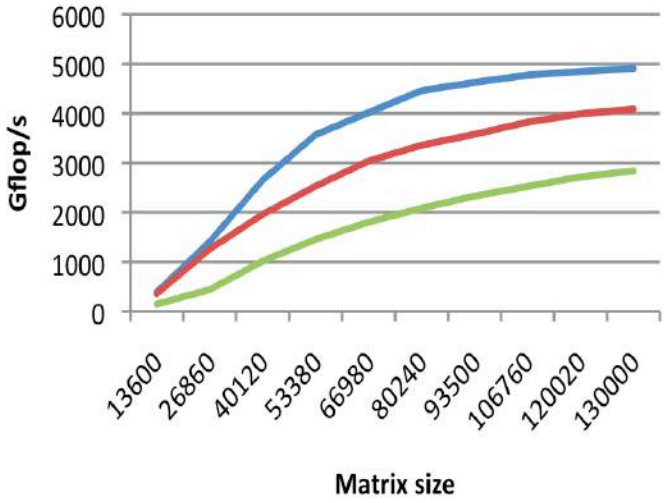


Runtime system, called PaRSEC, is distributed

Runtime DAG scheduling



- Every node has the symbolic DAG representation
  - Only the (node local) frontier of the DAG is considered
  - Distributed Scheduling based on remote completion notifications
- Background remote data transfer automatic with overlap
- NUMA / Cache aware Scheduling
  - Work Stealing and sharing based on memory hierarchies

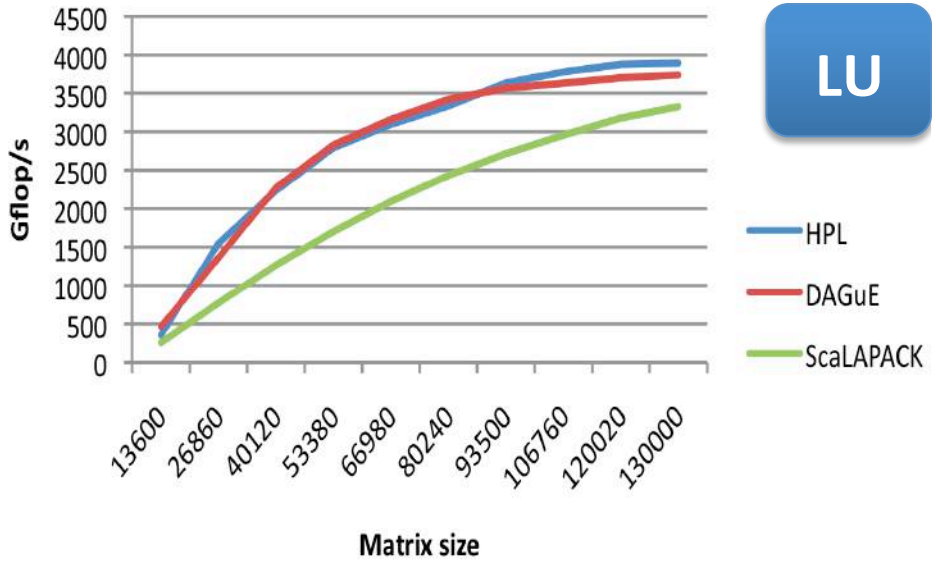


Cholesky

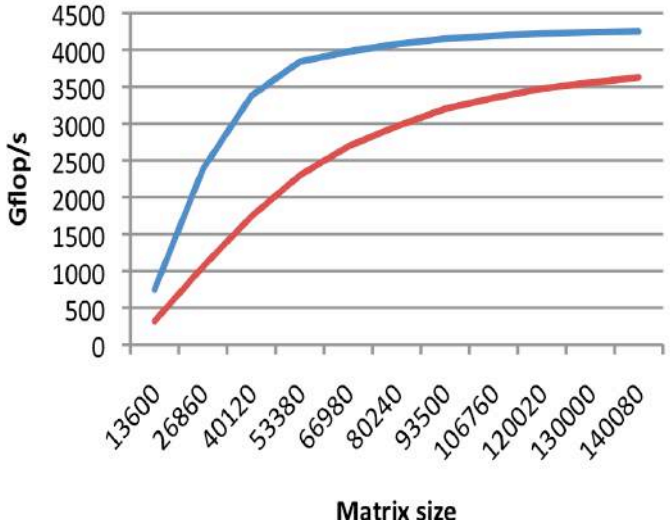
DAGuE
DSBP
ScaLAPACK

DSBP = Distributed Square Block Packed

81 nodes
Dual socket nodes
Quad core Xeon L5420
Total 648 cores at 2.5 GHz
ConnectX InfiniBand DDR 4x



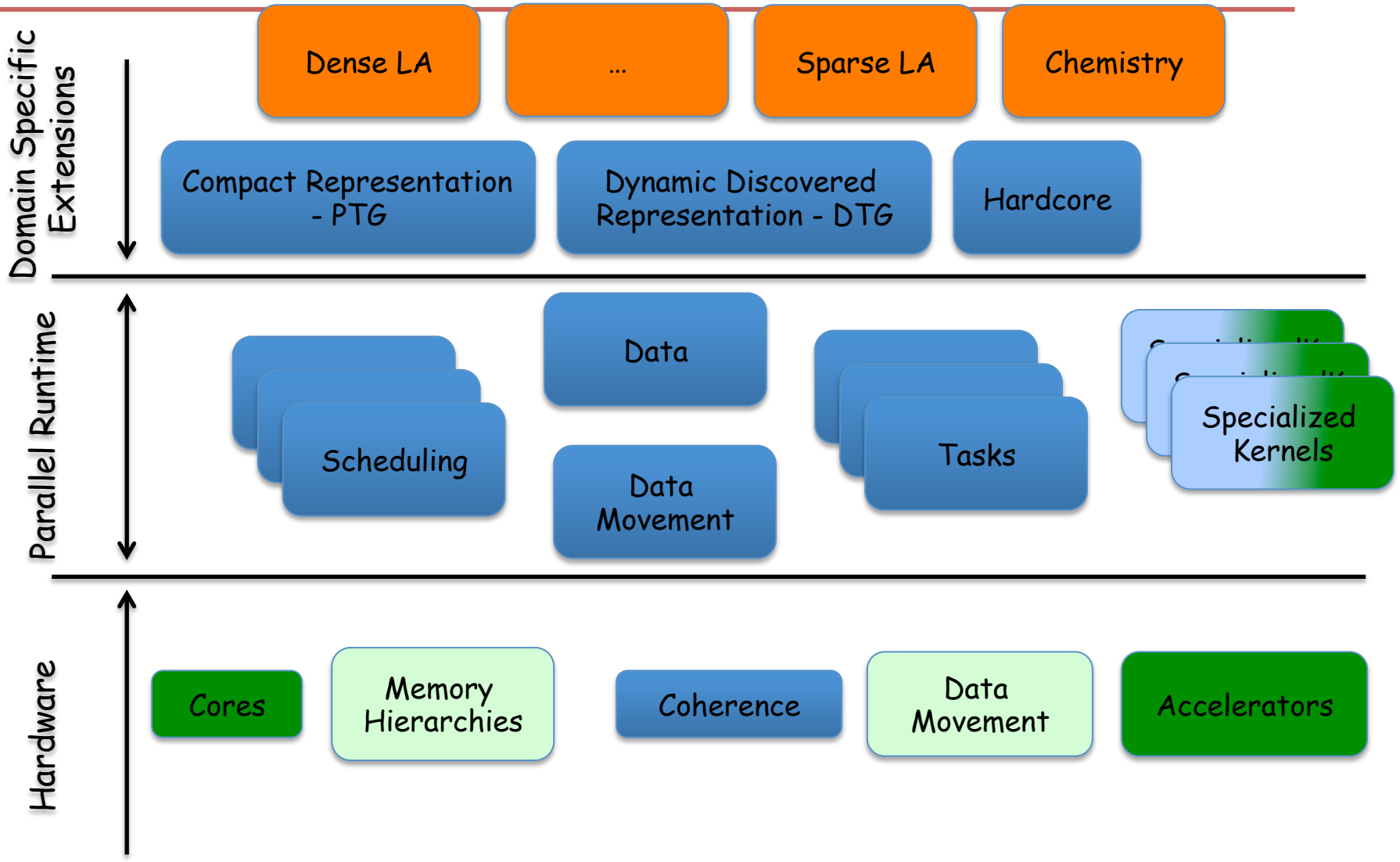
LU



QR

DAGuE
ScaLAPACK

The PaRSEC framework

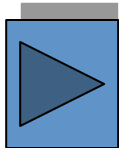


Other Systems

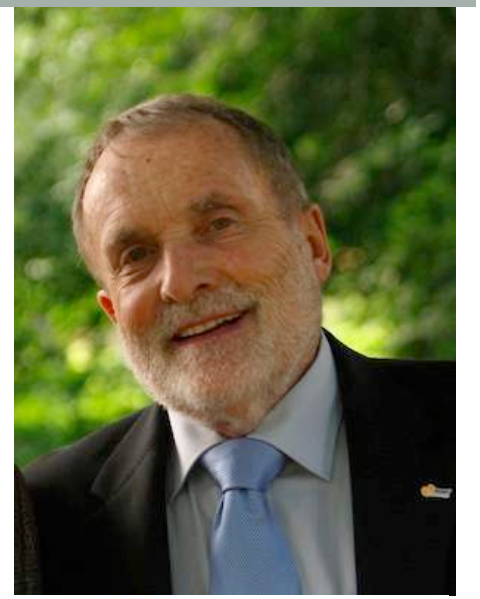
	PaRSEC	SMPss	StarPU	Charm ++	FLAME	QUARK	Tblas	PTG
Scheduling	Distr. (1/core)	Repl (1/node)	Repl (1/node)	Distr. (Actors)	w/ SuperMatrix	Repl (1/node)	Centr.	Centr.
Language	Internal or Seq. w/ Affine Loops	Seq. w/ add_task	Seq. w/ add_task	Msg-Driven Objects	Internal (LA DSL)	Seq. w/ add_task	Seq. w/ add_task	Internal
Accelerator	GPU	GPU	GPU		GPU	GPU		
Availability	Public	Public	Public	Public	Public	Public	Not Avail.	Not Avail.

Early stage: ParalleX
Non-academic: Swarm, MadLINQ, CnC

All projects support Distributed and Shared Memory
(QUARK with QUARKd; FLAME with Elemental)



TOP500



- In 1986 Hans Meuer started a list of supercomputer around the world, they were ranked by peak performance.
- Hans approached me in 1992 to put together our lists into the “TOP500”.
- The first TOP500 list was in June 1993.

TOP500® SUPERCOMPUTER SITES

Rank	Site	System	Cores	Rmax (GFlop/s)	Rpeak (GFlop/s)	Power (kW)
1	Los Alamos National Laboratory United States	CM-5/1024 Thinking Machines Corporation	1,024	59.7	131.0	
2	Minnesota Supercomputer Center United States	CM-5/544 Thinking Machines Corporation	544	30.4	69.6	
3	National Security Agency United States	CM-5/512 Thinking Machines Corporation	512	30.4	65.5	
4	NCSA United States	CM-5/512 Thinking Machines Corporation	512	30.4	65.5	
5	NEC Japan	SX-3/44R NEC	4	23.2	25.6	
6	Atmospheric Environment Service (AES)	SX-3/44	4	20.0	22.0	

HPL - Bad Things

- LINPACK Benchmark is 37 years old
  - TOP500 (HPL) is 22 years old
- Floating point-intensive performs $O(n^3)$ floating point operations and moves $O(n^2)$ data.
- No longer so strongly correlated to real apps.
- Reports Peak Flops (although hybrid systems see only 1/2 to 2/3 of Peak)
- Encourages poor choices in architectural features
- Overall usability of a system is not measured
- Used as a marketing tool
- Decisions on acquisition made on one number
- Benchmarking for days wastes a valuable resource

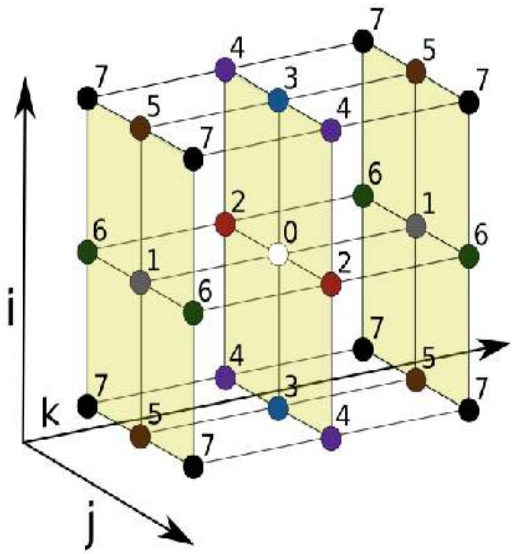
Proposal: HPCG

- High Performance Conjugate Gradient (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

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

Model Problem Description

- Synthetic discretized 3D PDE (FEM, FVM, FDM).
- Single heat diffusion model.
- Zero Dirichlet BCs, Synthetic RHS s.t. solution = 1.
- Local domain: $(n_x \times n_y \times n_z)$
- Process layout: $(np_x \times np_y \times np_z)$
- Global domain: $(n_x * np_x) \times (n_y * np_y) \times (n_z * np_z)$
- Sparse matrix:
  - 27 nonzeros/row interior.
  - 7 – 18 on boundary.
  - Symmetric positive definite.



27-point stencil operator

HPL vs. HPCG: Bookends

- Some see HPL and HPCG as “bookends” of a spectrum.
  - Applications teams know where their codes lie on the spectrum.
  - Can gauge performance on a system using both HPL and HPCG numbers.
- Problem of HPL execution time still an issue:
  - Need a lower cost option. End-to-end HPL runs are too expensive.
  - Work in progress.

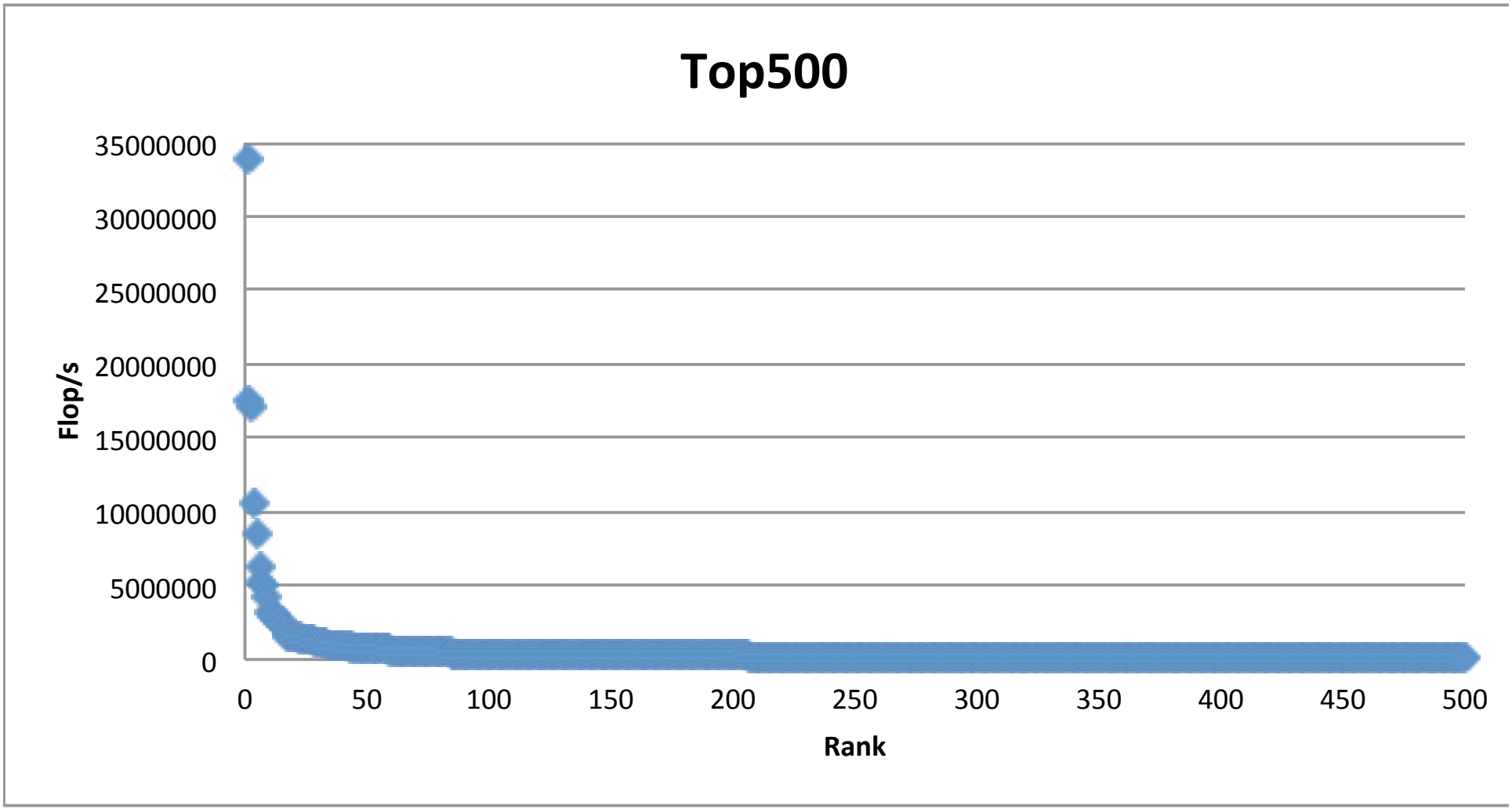
  - Began last year with about 20 results, today have 41 systems.
    - Not interested in collecting 500 systems

HPCG Results, July 2015

HPCG Rank	Site	Computer	Cores	HPL Rmax (Pflops)	HPL Rank	HPCG (Pflops)	HPCG /HPL	% of Peak
1	NSCC / Guangzhou	Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom	3,120,000	33.9	1	.580	1.7%	1.1%
2	RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx 8C + Custom	705,024	10.5	4	.461	4.4%	4.1%
3	DOE/OS Oak Ridge Nat Lab	Titan, Cray XK7 AMD 16C + Nvidia Kepler GPU 14C + Custom	560,640	17.6	2	.322	1.8%	1.2%
4	DOE/OS Argonne Nat Lab	Mira BlueGene/Q, Power BQC 16C 1.60GHz + Custom	786,432	8.59	5	.167	1.9%	1.7%
5	NASA Ames	Pleiades, SGI ICE X, Intel 2.6,2.8,2.5 GHz+ IB	186,288	4.09	11	.132	3.2%	2.7%
6	Swiss CSCS	Piz Daint, Cray XC30, Xeon 8C + Nvidia Kepler 14C + Custom	115,984	6.27	6	.125	2.0%	1.6%
7	KAUST	Shaheen II,Cray XC40, Xeon 16C 2.3GHz + Custom	196,608	5.54	7	.114	2.1%	1.6%
8	Texas Advanced Computing Center	Stampede, Dell Intel 8c + Intel Xeon Phi 61c + IB	522,080	5.17	8	.097	1.9%	1.0%
9	Leibniz Rechenzentrum	SuperMUC, Intel 8C + IB	147,456	2.90	20	.0833	2.9%	2.6%
10	DOE/OS LBNL	Edison, Cray XC30, Xeon, 12c, 2,4GHz + Custom	133,824	1.66	33	.0786	4.8%	3.1%

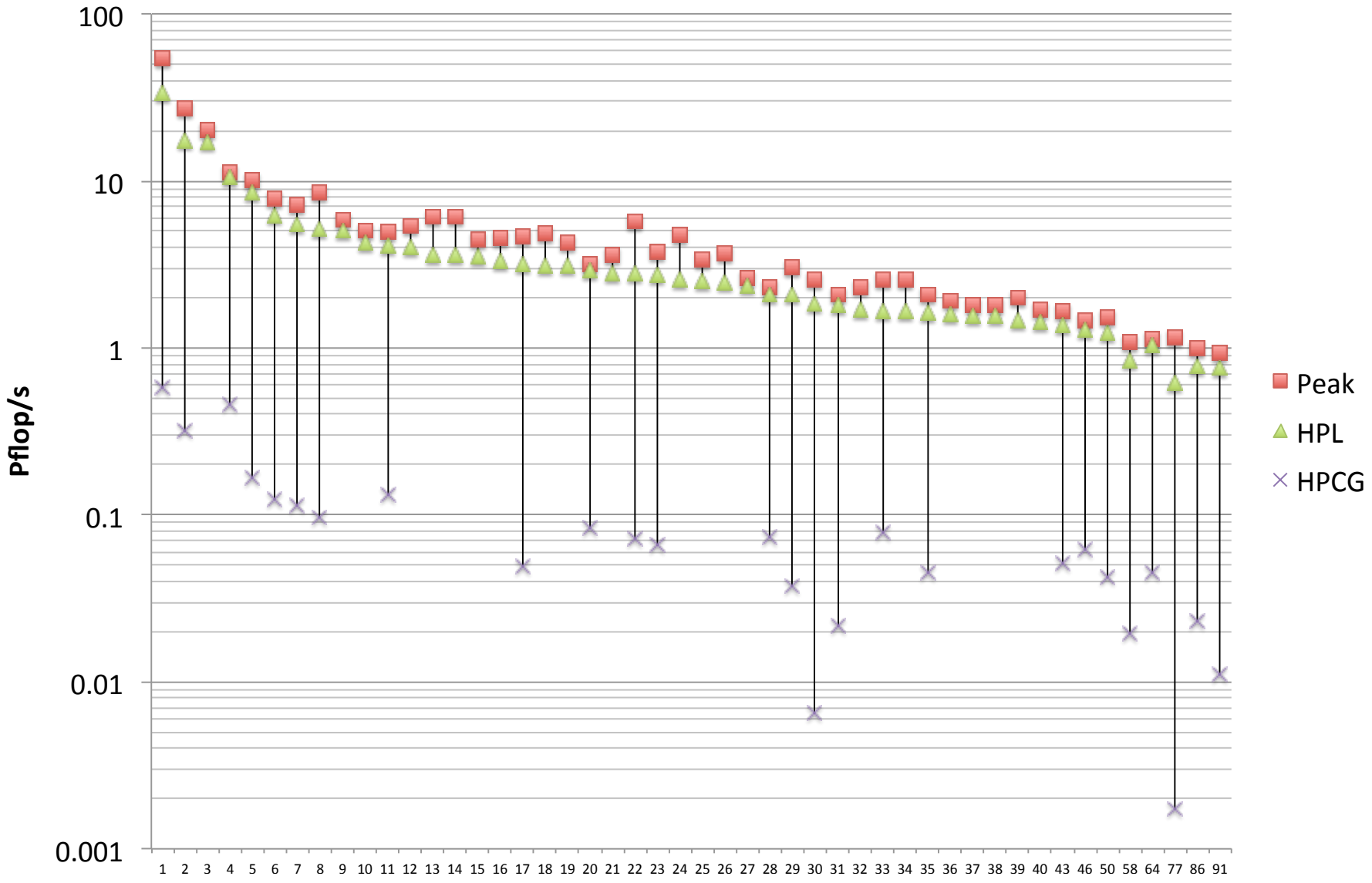
HPCG Results, July 2015

HPCG Rank	Site	Computer	Cores	HPL Rmax (Pflops)	HPL Rank	HPCG (Pflops)	HPCG/ HPL	% of Peak
11	Plasma Simulator	Fujitsu FX100, Sparc64 Xifx 32C + custom	82,944	2.38	27	.073	3.1%	2.8%
12	GSIC Center TiTech	Tsubame 2.5 Xeon 6C, 2.93GHz + Nvidia K20x + IB	76,032	2.79	22	.0725	2.6%	1.3%
13	HLRS/Universitaet Stuttgart	Hornet Cray XC40, Xeon 2.5GHz + custom	94,656	2.76	23	.066	2.4%	1.7%
14	Max-Planck	iDataPlex Xeon 10C, 2.8GHz + IB	65,320	1.28	46	.061	4.8%	4.2%
15	Earth Simulator	NEC SX-ACE 4C, 1 GHz + custom	8,192	0.487		.058	12%	11%
16	CEA/TGCC-GENCI	Curie thin nodes Bullx B510 Intel Xeon 8C 2.7 GHz + IB	77,184	1.36	43	.051	3.8%	3.1%
17	Exploration and Production Eni S.p.A.	HPC2, Intel Xeon 10C 2.8 GHz + Nvidia Kepler 14C + IB	62,640	3.00	17	.049	1.6%	1.2%
18	Grand Equipement National de Calcul Intensif	Occigen Bullx Xeon 12C 2.6Ghz + IB	50,544	1.63	35	.045	2.8%	2.2%
19	Oakleaf-FX	PIMEHPC FX10, Sparc64 16C, 1.85 GHz + custom	76,800	1.04	64	.0448	4.3%	3.9%
20	IFERC/F4E	Helios Bullx B510 Intel Xeon 8C 2.7 GHz + IB	70,560	1.24	50	.0426	3.4%	2.8%

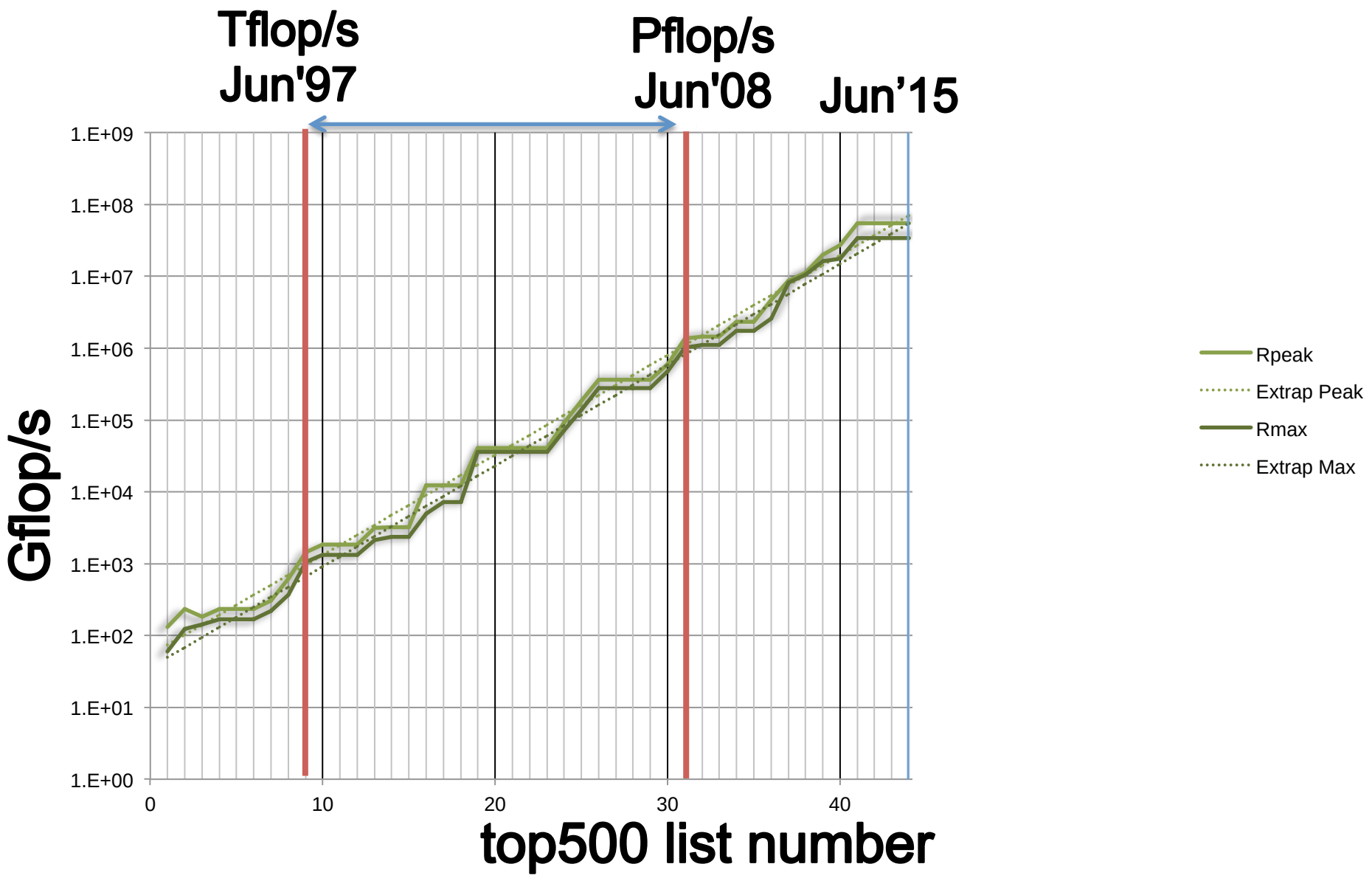

Top500
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
Flop/s
0
50
100
150
200
250
300
350
400
450
500
Rank

Peak, HPL Pflop/s

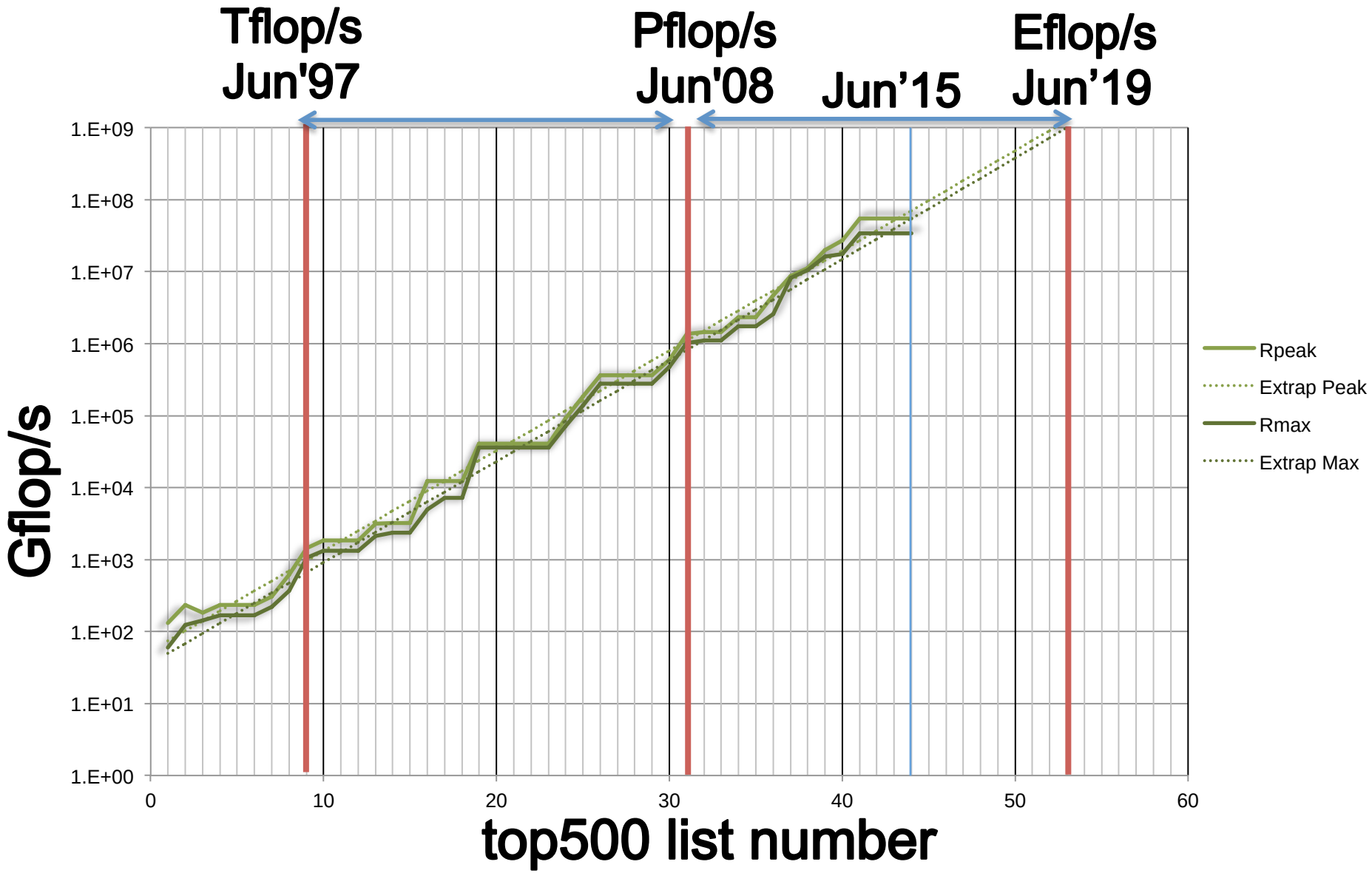
Peak, HPL, HPCG Pflop/s



Top500 List 54 Edition



Top500 List 54 Edition



Today's #1 System

Systems	2015 Tianhe-2
System peak	**55 Pflop/s**
Power	**18 MW** **(3 Gflops/W)**
System memory	1.4 PB (1.024 PB CPU + .384 PB CoP)
Node performance	3.43 TF/s (.4 CPU +3 CoP)
Node concurrency	24 cores CPU + 171 cores CoP
Node Interconnect BW	6.36 GB/s
System size (nodes)	16,000
Total concurrency	3.12 M 12.48M threads (4/core)
MTTF	Few / day

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

Systems	2015 Tianhe-2
System peak	**55 Pflop/s**
Power	**18 MW** **(3 Gflops/W)**
System memory	1.4 PB (1.024 PB CPU + .384 PB CoP)
Node performance	3.43 TF/s (.4 CPU +3 CoP)
Node concurrency	24 cores CPU + 171 cores CoProc
Node Interconnect BW	6.36 GB/s
System size (nodes)	16,000
Total concurrency	3.12 M 12.48M threads (4/core)
MTTF	Few / day

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

Systems	2015 Tianhe-2	2020-2023	Difference Today & Exa
System peak	**55 Pflop/s**	**1 Eflop/s**	**~20x**
Power	**18 MW** **(3 Gflops/W)**	**~20 MW** **(50 Gflops/W)**	**O(1)** **~15x**
System memory	1.4 PB (1.024 PB CPU + .384 PB CoP)	32 - 64 PB	~50x
Node performance	3.43 TF/s (.4 CPU +3 CoP)	1.2 or 15TF/s	O(1)
Node concurrency	24 cores CPU + 171 cores CoProc	O(1k) or 10k	~5x - ~50x
Node Interconnect BW	6.36 GB/s	200-400 GB/s	~40x
System size (nodes)	16,000	O(100,000) or O(1M)	~6x - ~60x
Total concurrency	3.12 M 12.48M threads (4/core)	O(billion)	~100x
MTTF	Few / day	Many / day	O(?)

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.

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**

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

MAGMA **PLASMA**