Architecture-Aware Algorithms and Software for Peta and Exascale Computing (Learning from the Past)

Jack Dongarra

University of Tennessee
Oak Ridge National Laboratory
University of Manchester

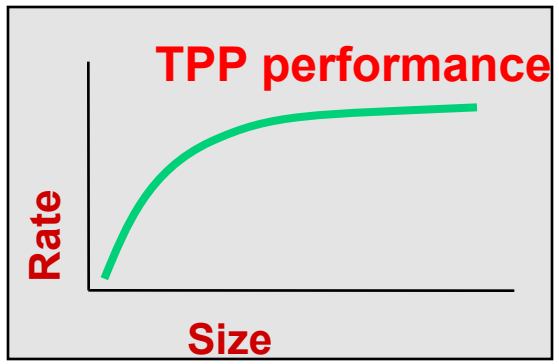
First …

- Thank a number of people who have helped with this work
  - Emmanuel Agullo, George Bosilca, Aurelien Bouteiller, Anthony Danalis, Jim Demmel, Tingxing "Tim" Dong, Mathieu Faverge, Azzam Haidar, Thomas Herault, Mitch Horton, Jakub Kurzak, Julien Langou, Julie Langou, Pierre Lemarinier, Piotr Luszczek, Hatem Ltaief, Stanimire Tomov, Asim YarKhan, …
- Much of what I will describe has been done before, at least in theory.

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

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

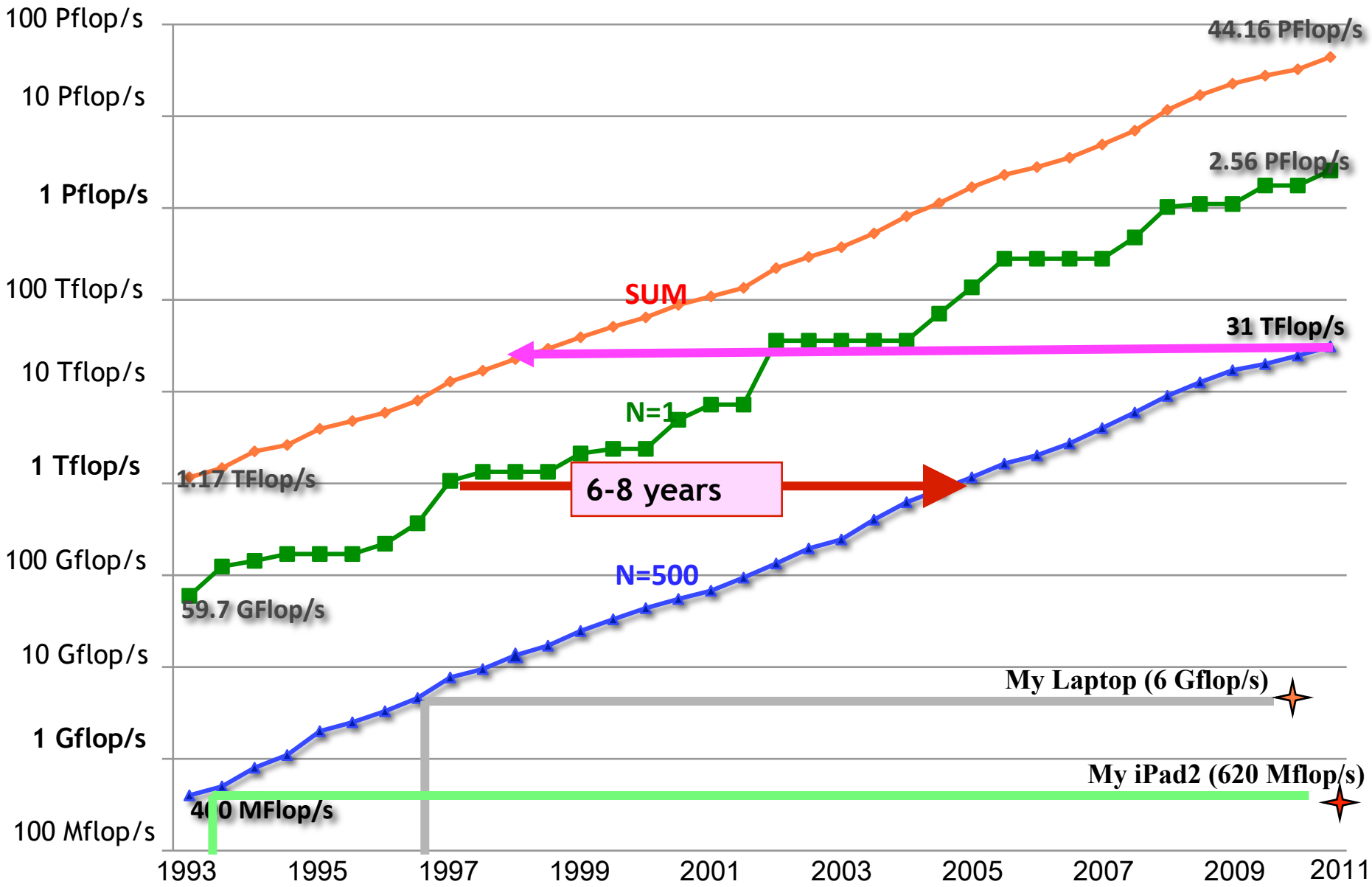
$Ax=b$, *dense problem*



- 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



36rd List: The TOP10

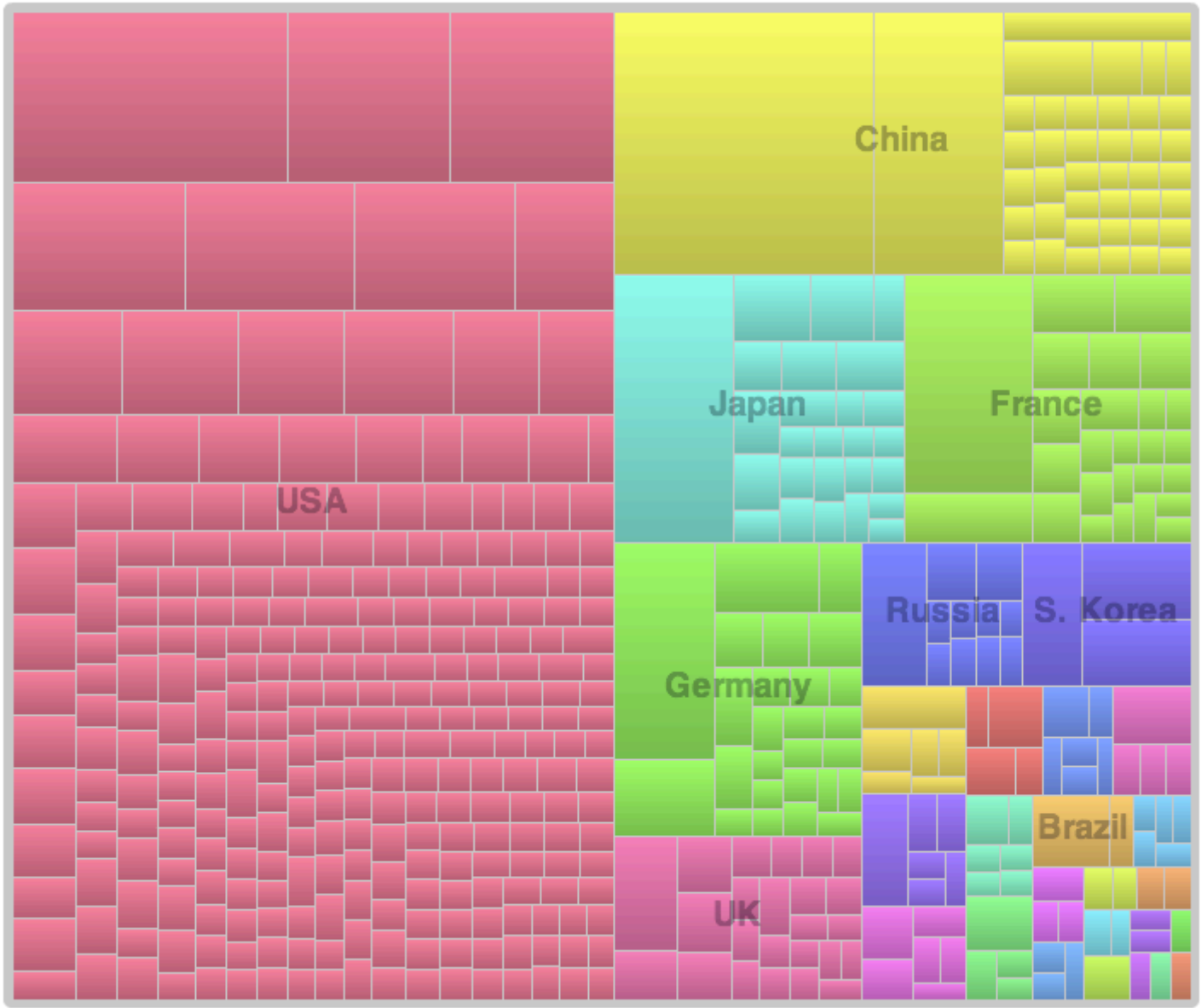
Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak
1	Nat. SuperComputer Center in Tianjin	Tianhe-1A, NUDT Intel + Nvidia GPU + custom	China	186,368	2.57	55
2	DOE / OS Oak Ridge Nat Lab	Jaguar, Cray AMD + custom	USA	224,162	1.76	75
3	Nat. Supercomputer Center in Shenzhen	Nebulea, Dawning Intel + Nvidia GPU + IB	China	120,640	1.27	43
4	GSIC Center, Tokyo Institute of Technology	Tusbame 2.0, HP Intel + Nvidia GPU + IB	Japan	73,278	1.19	52
5	DOE / OS Lawrence Berkeley Nat Lab	Hopper, Cray AMD + custom	USA	153,408	1.054	82
6	Commissariat a l'Energie Atomique (CEA)	Tera-10, Bull Intel + IB	France	138,368	1.050	84
7	DOE / NNSA Los Alamos Nat Lab	Roadrunner, IBM AMD + Cell GPU + IB	USA	122,400	1.04	76
8	NSF / NICS U of Tennessee	Kraken, Cray AMD + custom	USA	98,928	.831	81
9	Forschungszentrum Juelich (FZJ)	Jugene, IBM Blue Gene + custom	Germany	294,912	.825	82
10	DOE / NNSA LANL & SNL	Cielo, Cray AMD + custom	USA	107,152	.817	79

36rd List: The TOP10

Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	GFlops/ Watt
1	Nat. SuperComputer Center in Tianjin	Tianhe-1A, NUDT Intel + Nvidia GPU + custom	China	186,368	2.57	55	4.04	636
2	DOE / OS Oak Ridge Nat Lab	Jaguar, Cray AMD + custom	USA	224,162	1.76	75	7.0	251
3	Nat. Supercomputer Center in Shenzhen	Nebulea, Dawning Intel + Nvidia GPU + IB	China	120,640	1.27	43	2.58	493
4	GSIC Center, Tokyo Institute of Technology	Tsubame 2.1, HP Intel + Nvidia GPU + IB	Japan	73,278	1.19	52	1.40	850
5	DOE / OS Lawrence Berkeley Nat Lab	Hopper, Cray AMD + custom	USA	153,408	1.054	82	2.91	362
6	Commissariat a l'Energie Atomique (CEA)	Tera-10, Bull Intel + IB	France	138,368	1.050	84	4.59	229
7	DOE / NNSA Los Alamos Nat Lab	Roadrunner, IBM AMD + Cell GPU + IB	USA	122,400	1.04	76	2.35	446
8	NSF / NICS U of Tennessee	Kraken, Cray AMD + custom	USA	98,928	.831	81	3.09	269
9	Forschungszentrum Juelich (FZJ)	Jugene, IBM Blue Gene + custom	Germany	294,912	.825	82	2.26	365
10	DOE / NNSA LANL & SNL	Cielo, Cray AMD + custom	USA	107,152	.817	79	2.95	277
500	Computacenter LTD	HP Cluster, Intel + GigE	UK	5,856	.031	53		

Quiz: How Many of the Top500 systems use GPUs?

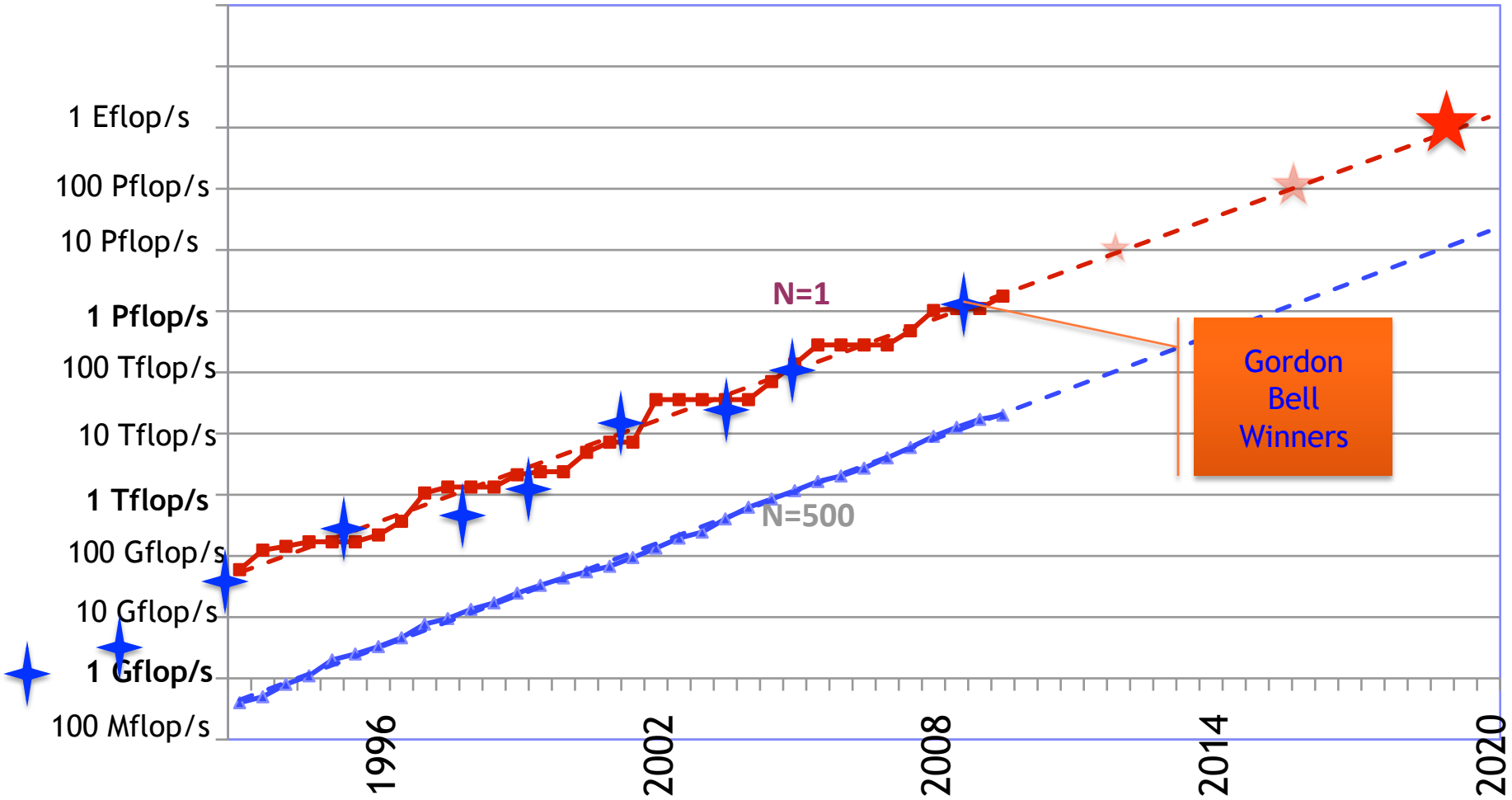
Countries Share



Absolute Counts

US:	274
China:	41
Germany:	26
Japan:	26
France:	26
UK:	25

Performance Development in Top500



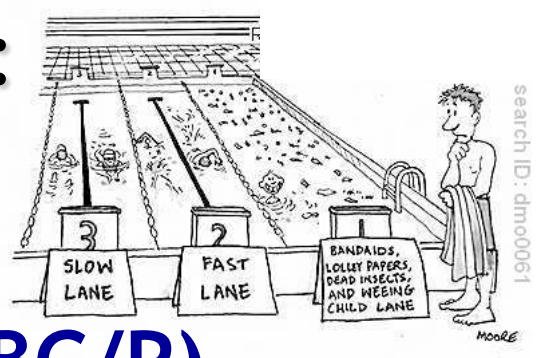
Potential System Architecture

Systems	2011
System peak	**2 Pflop/s**
Power	**7 MW**
System memory	0.3 PB
Node performance	125 GF
Node memory BW	25 GB/s
Node concurrency	12
Total Node Interconnect BW	3.5 GB/s
System size (nodes)	18,700
Total concurrency	225,000
Storage	15 PB
IO	0.2 TB
MTTI	days

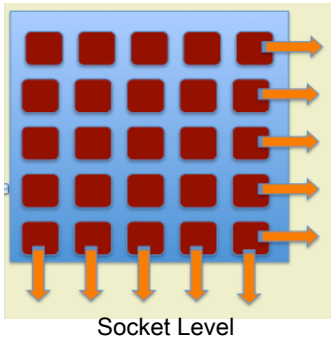
Potential System Architecture with a cap of $200M and 20MW

Systems	2011	2018	Difference Today & 2018
System peak	**2 Pflop/s**	**1 Eflop/s**	**O(1000)**
Power	**7 MW**	**~20 MW**	
System memory	0.3 PB	32 - 64 PB	O(100)
Node performance	125 GF	1,2 or 15TF	O(10) - O(100)
Node memory BW	25 GB/s	2 - 4TB/s	O(100)
Node concurrency	12	O(1k) or 10k	O(100) - O(1000)
Total Node Interconnect BW	3.5 GB/s	200-400GB/s	O(100)
System size (nodes)	18,700	O(100,000) or O(1M)	O(10) - O(100)
Total concurrency	225,000	O(billion)	O(10,000)
Storage	15 PB	500-1000 PB (>10x system memory is min)	O(10) - O(100)
IO	0.2 TB	60 TB/s (how long to drain the machine)	O(100)
MTTI	days	O(1 day)	- O(10)

Exascale (10^{18} Flop/s) Systems: Two Possible Swim Lanes

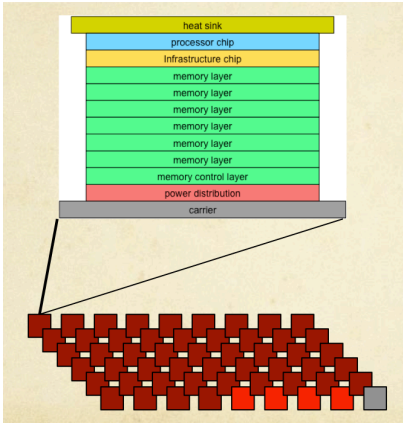


- **Light weight processors (think BG/P)**
  - ~1 GHz processor (10^9)
  - ~1 Kilo cores/socket (10^3)
  - ~1 Mega sockets/system (10^6)



Socket Level
Cores scale-out for planar geometry

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



Node Level
3D packaging

Commodity plus Accelerators

Commodity	Accelerator (GPU)
Intel Xeon	Nvidia C2050 "Fermi"
8 cores	448 CUDA cores
3 GHz	1.15 GHz
8 ops/cycle	448 ops/cycle
96 Gflop/s (DP)	515 Gflop/s DP

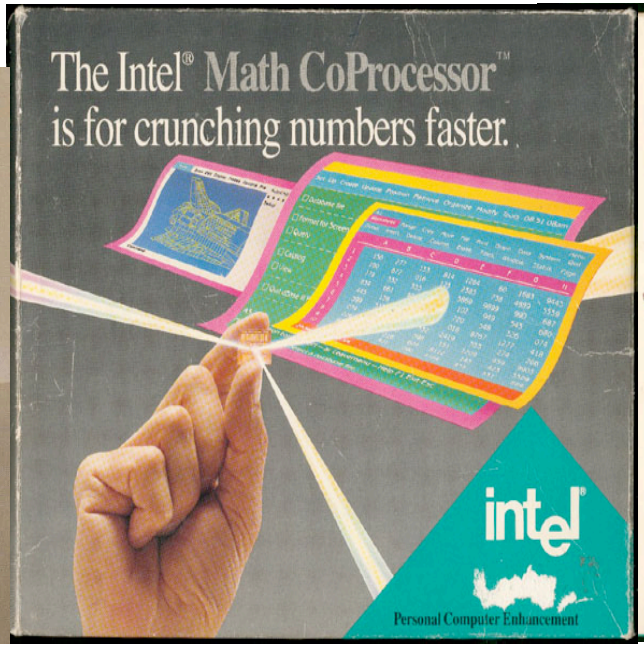
Quiz: How Many of the Top500 systems use GPUs?

Answer:
Today only 17 systems on the TOP500 use GPUs

Interconnect
PCI-X 16 lane
64 Gb/s
1 GW/s

We Have Seen This Before

- Floating Point Systems FPS-164 Scientific Computer (1976)
- Intel Math Co-processor (1980)
- Weitek Math Co-processor (1981)



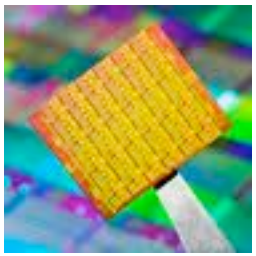
Balance Between Data Movement and Floating point

- FPS-164 and VAX (1976)
  - 11 Mflop/s; transfer rate 44 MB/s
  - Ratio of flops to bytes of data movement:
    1 flop per 4 bytes transferred
- Nvidia Fermi and PCI-X to host
  - 500 Gflop/s; transfer rate 8 GB/s
  - Ratio of flops to bytes of data movement:
    62 flops per 1 byte transferred

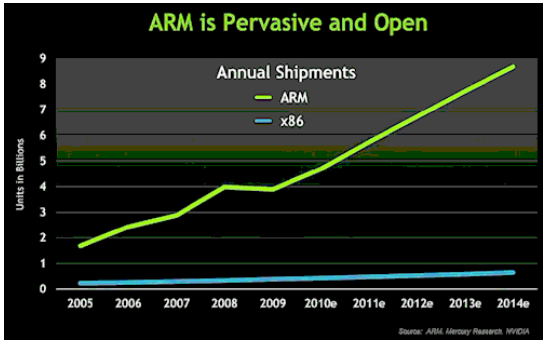
- Flop/s are cheap, so are provisioned in excess

Future Computer Systems

- Most likely be a hybrid design
  - Think standard multicore chips and accelerator (GPUs)
- Today accelerators are attached
- Next generation more integrated
- Intel's MIC architecture "Knights Ferry" and "Knights Corner" to come.
  - 48 x86 cores
- AMD's Fusion in 2012 - 2013
  - Multicore with embedded graphics ATI
- Nvidia's Project Denver plans to develop an integrated chip using ARM architecture in 2013.



AMD
The future is fusion



The High Cost of Data Movement

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

Approximate power costs (in picoJoules)

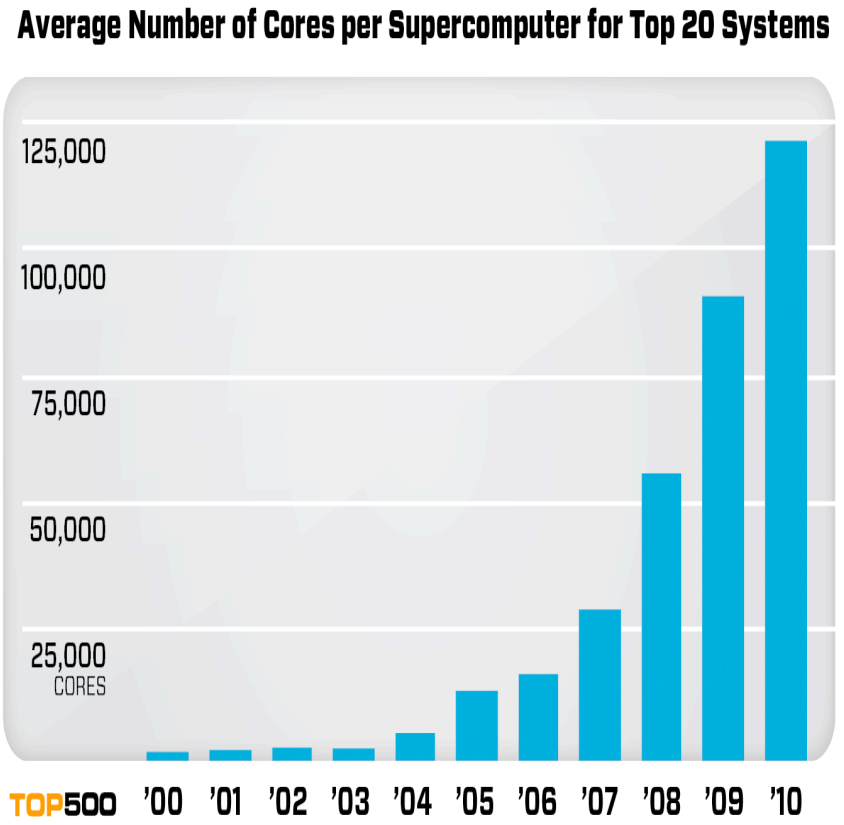
	2011
DP FMADD flop	100 pJ
DP DRAM read	4800 pJ
Local Interconnect	7500 pJ
Cross System	9000 pJ

Source: John Shalf, LBNL

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

Factors that Necessitate Redesign of Our Software

- Steepness of the ascent from terascale to petascale to exascale
- Extreme parallelism and hybrid design
  - Preparing for million/billion way parallelism
- Tightening memory/bandwidth bottleneck
  - Limits on power/clock speed implication on multicore
  - Reducing communication will become much more intense
  - Memory per core changes, byte-to-flop ratio will change
- Necessary Fault Tolerance
  - MTTF will drop
  - Checkpoint/restart has limitations
  - shared responsibility



Software infrastructure does not exist today

Major Changes to Software

- Must rethink the design of our software
  - Another disruptive technology
    - Similar to what happened with cluster computing and message passing
  - Rethink and rewrite the applications, algorithms, and software

Emerging Architectures

- Are needed by applications

- Applications are given (as function of time)
- Architectures are given (as function of time)

- Algorithms and software must be adapted or created to bridge to (hostile) architectures for the sake of the complex applications

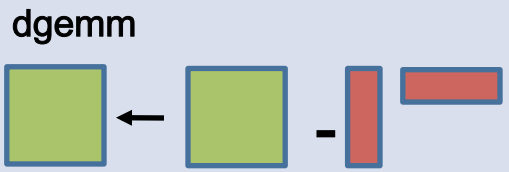
Exascale algorithms that expose and exploit multiple levels of parallelism

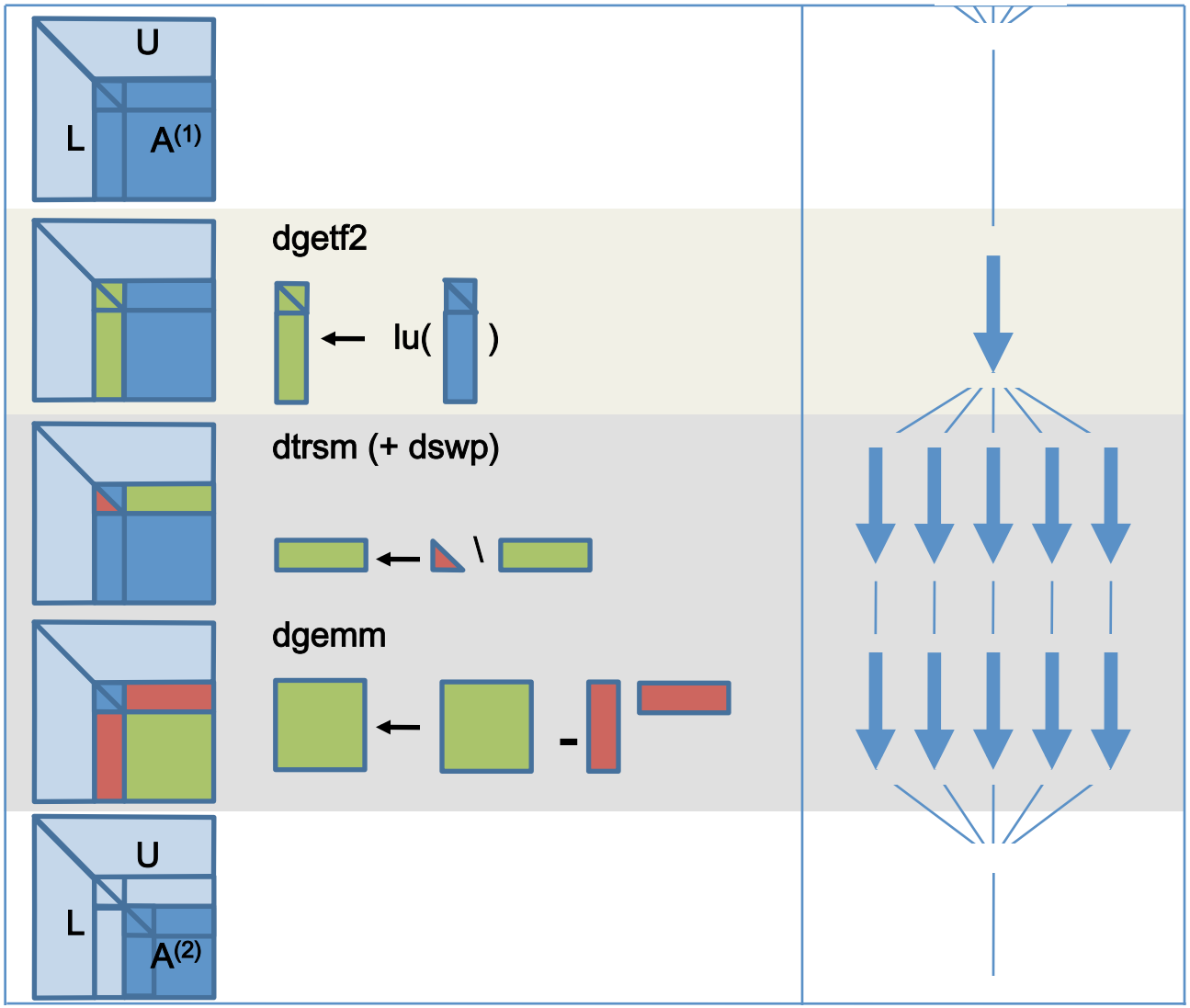
- Synchronization-reducing algorithms
  - Break Fork-Join model
- Communication-reducing algorithms
  - Use methods which have lower bound on communication
- Fault resilient algorithms
  - Implement algorithms that can recover from failures
- Mixed precision methods
  - 2x speed of ops and 2x speed for data movement
- Reproducibility of results
  - Today we can't guarantee this

Fork-Join Parallelization of LU and QR.

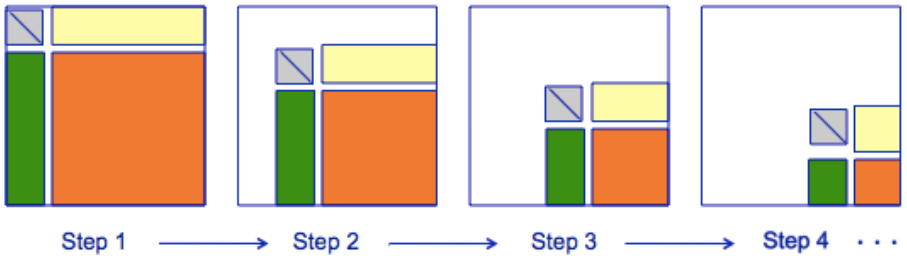
Parallelize the update:

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

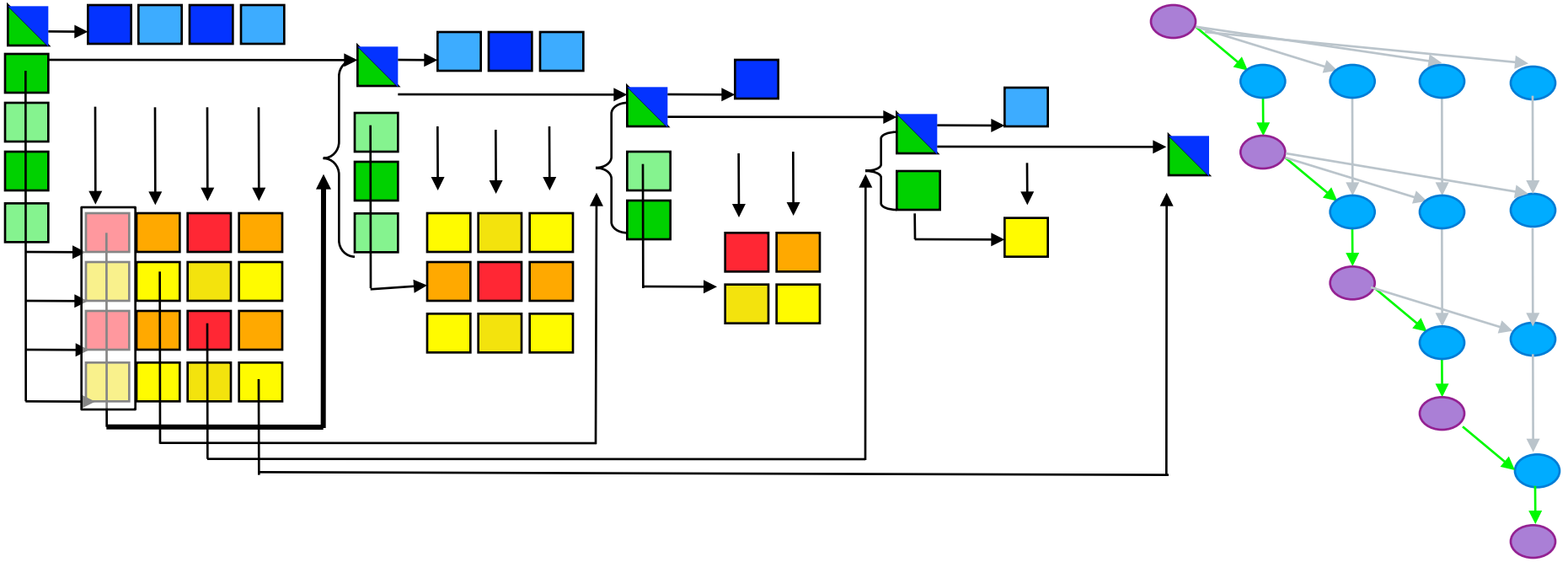




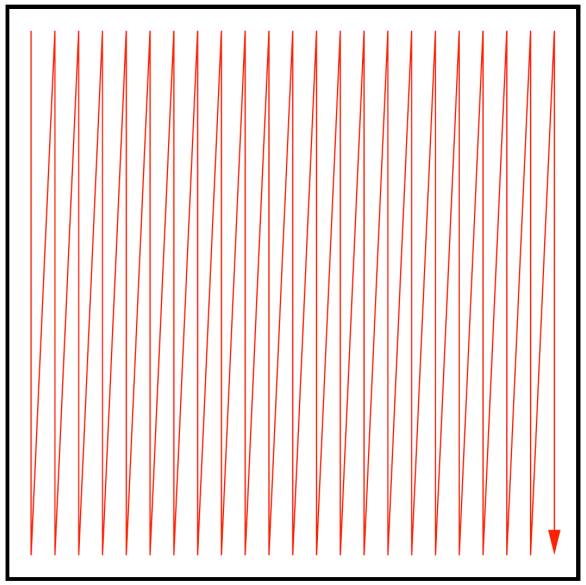
Parallel Tasks in LU/LL^T/QR

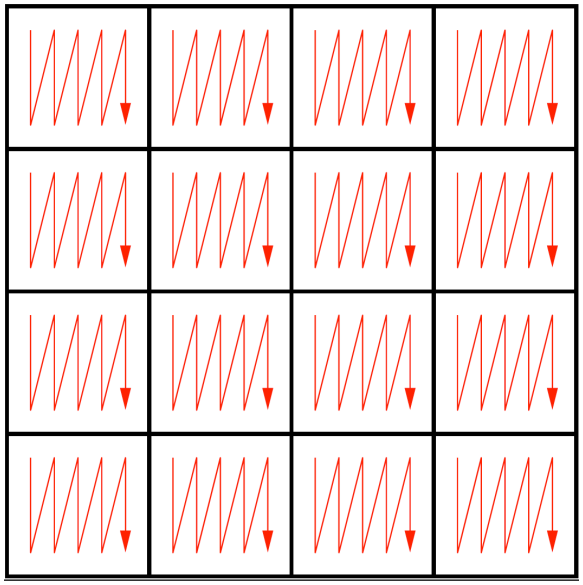


- Break into smaller tasks and remove dependencies



Data Layout is Critical

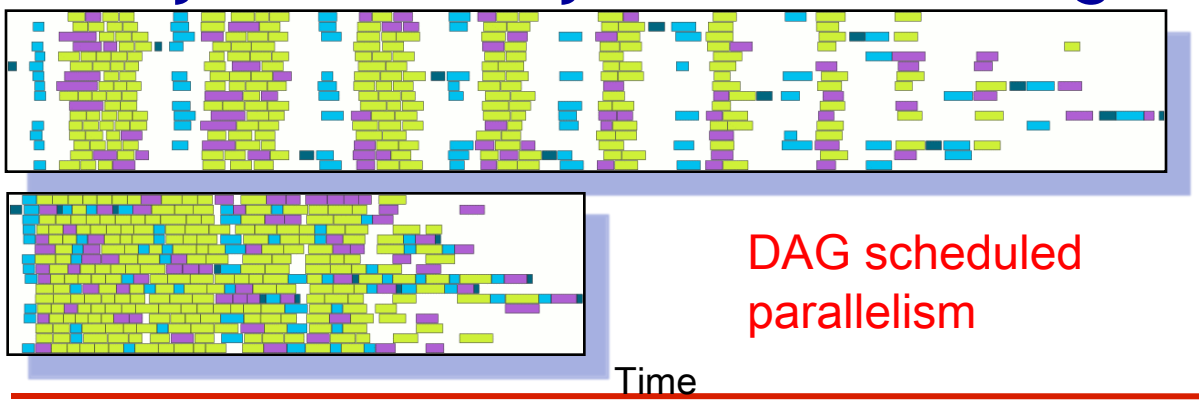


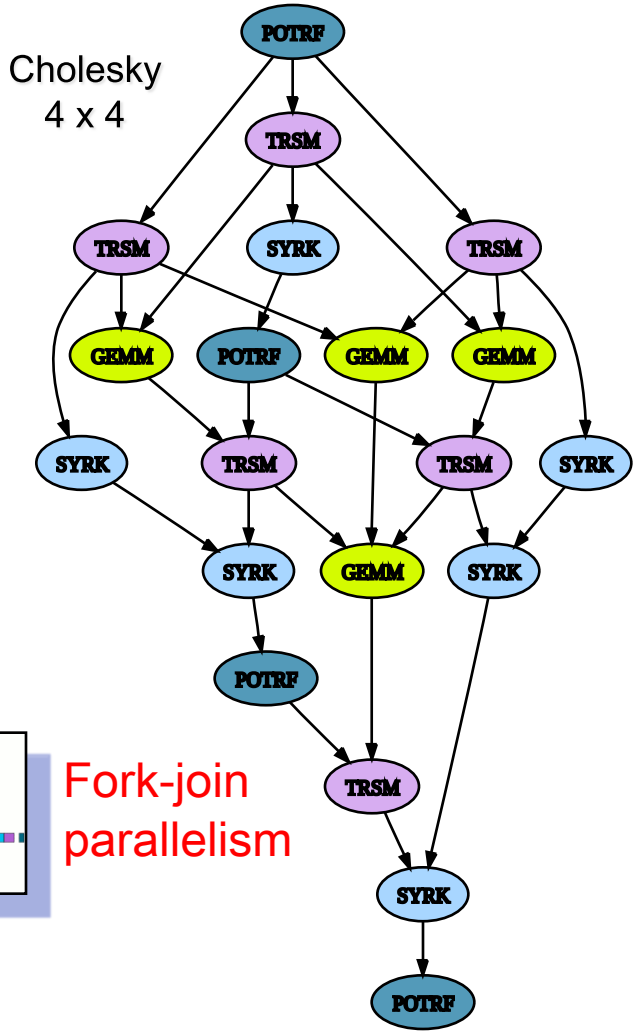


- Tile data layout where each data tile is contiguous in memory
- Decomposed into several fine-grained tasks, which better fit the memory of the small core caches

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

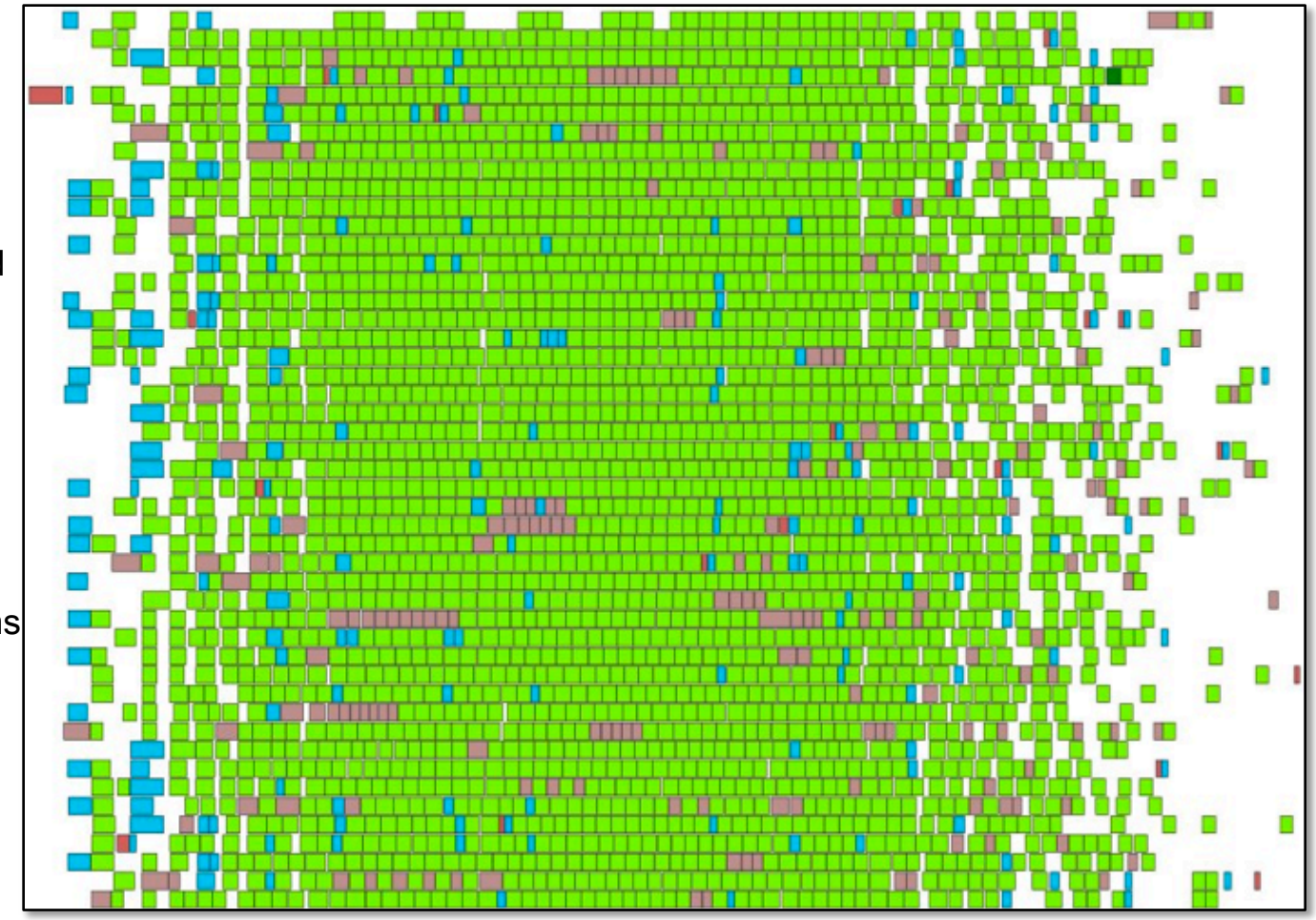




Synchronization Reducing Algorithms

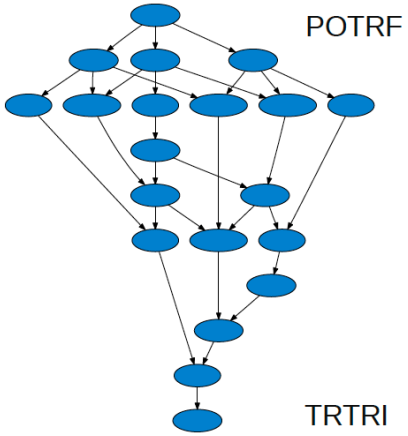
- Regular trace
- Factorization steps pipelined
- Stalling only due to natural load imbalance
- Dynamic
- Out of order execution
- Fine grain tasks
- Independent block operations

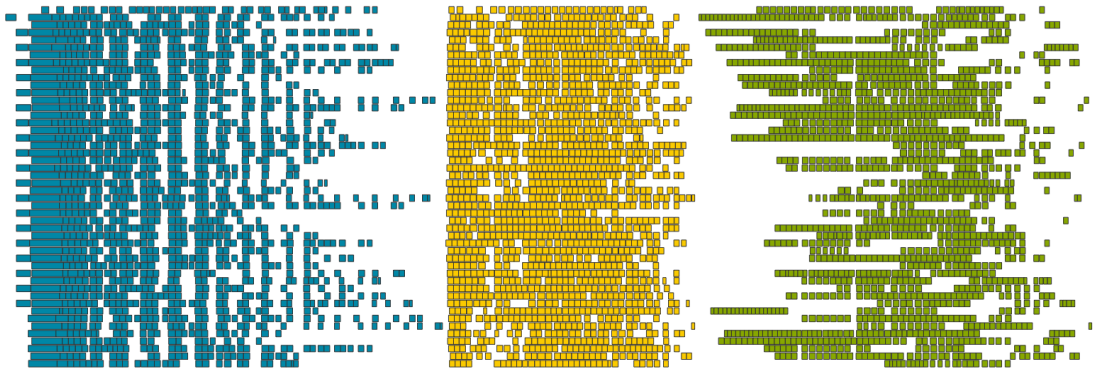
The colored area over the rectangle is the efficiency

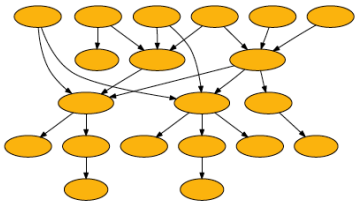


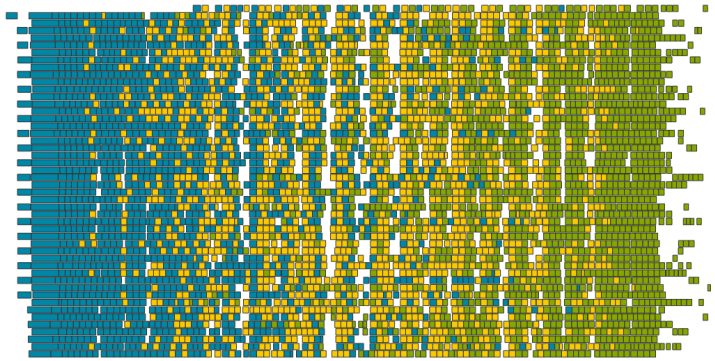
Tile LU factorization; Matrix size 4000x4000, Tile size 200
8-socket, 6-core (48 cores total) AMD Istanbul 2.8 GHz

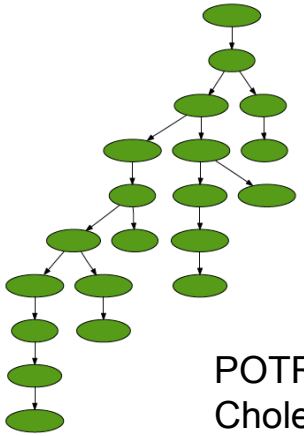
Pipelining: Cholesky Inversion
3 Steps: Factor, Invert L, Multiply L's

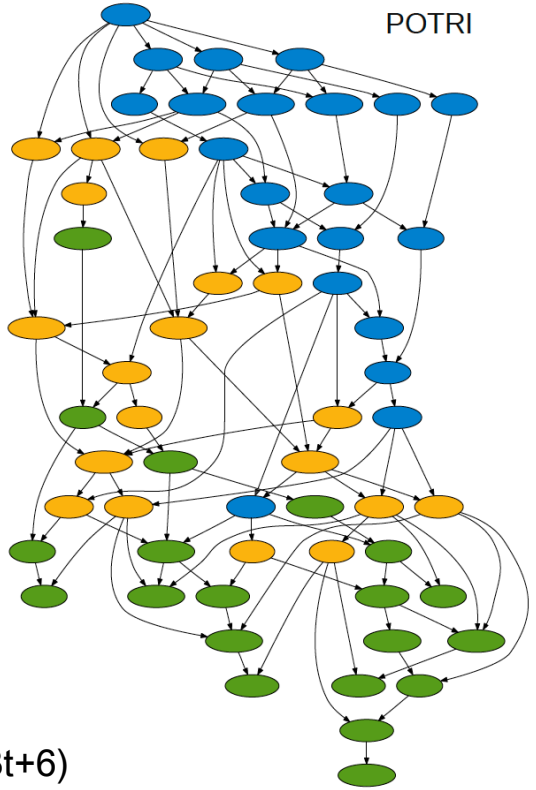












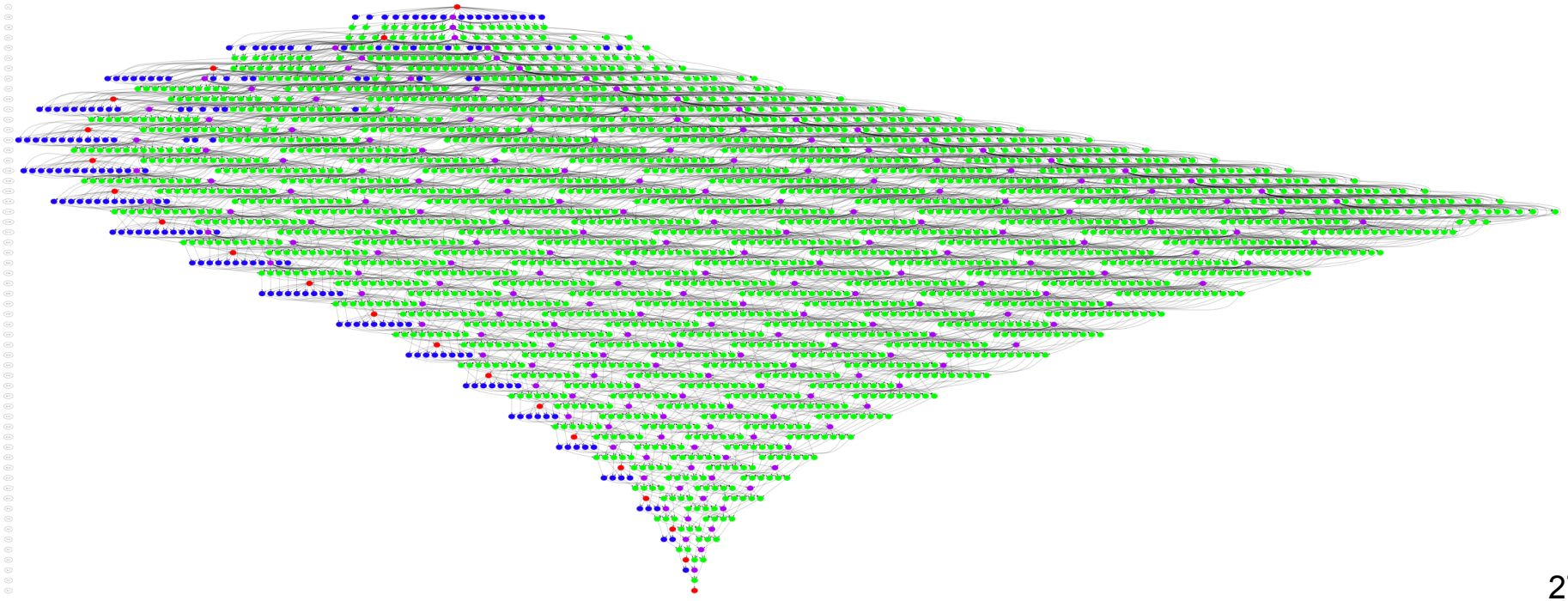
48 cores
POTRF, TRTRI and LAUUM.
The matrix is 4000 x 4000,tile size is 200 x 200,

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

Pipelined: 18 (3t+6)

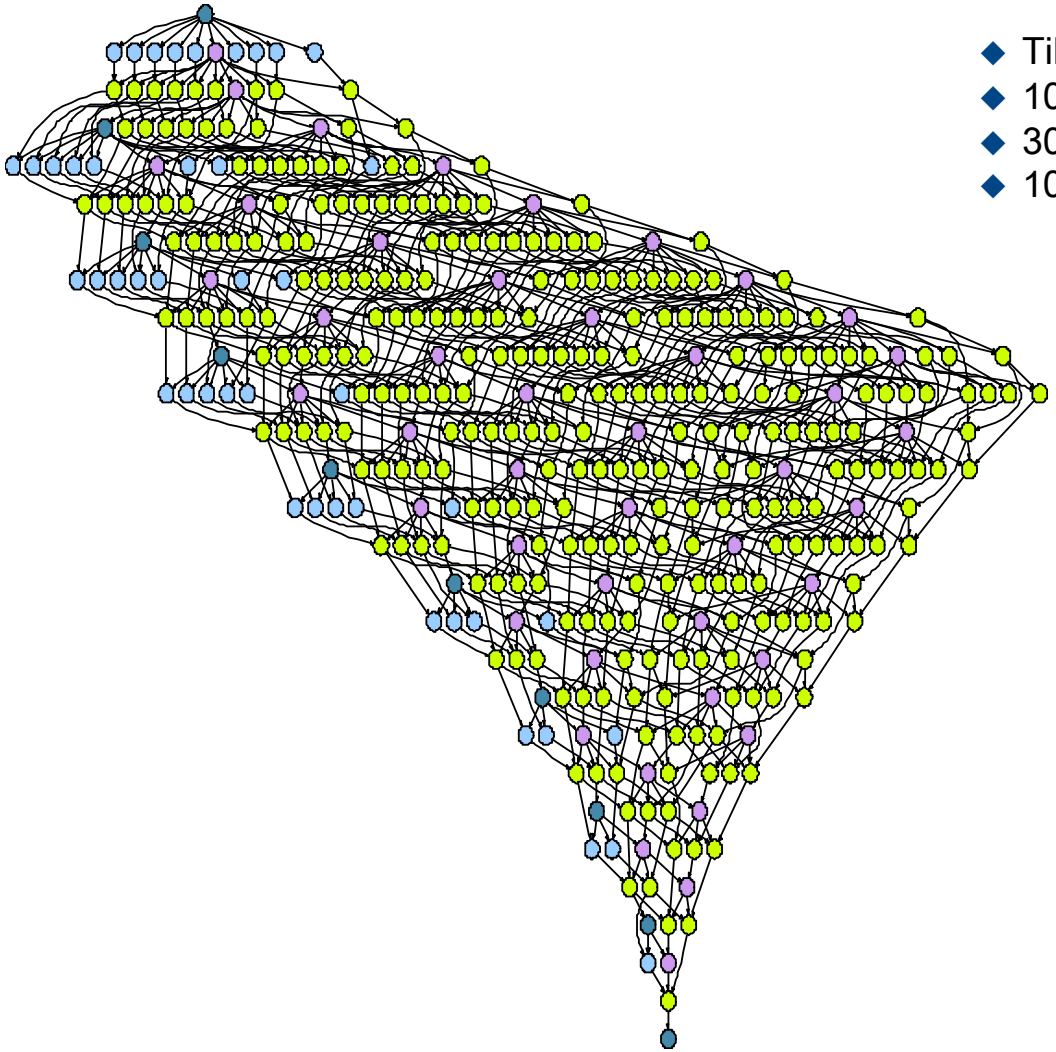
Big DAGs: No Global Critical Path

- DAGs get very big, very fast
  - So windows of active tasks are used; this means no global critical path
  - Matrix of NBxNB tiles; NB^3 operation
    - NB=100 gives 1 million tasks



PLASMA Local Scheduling

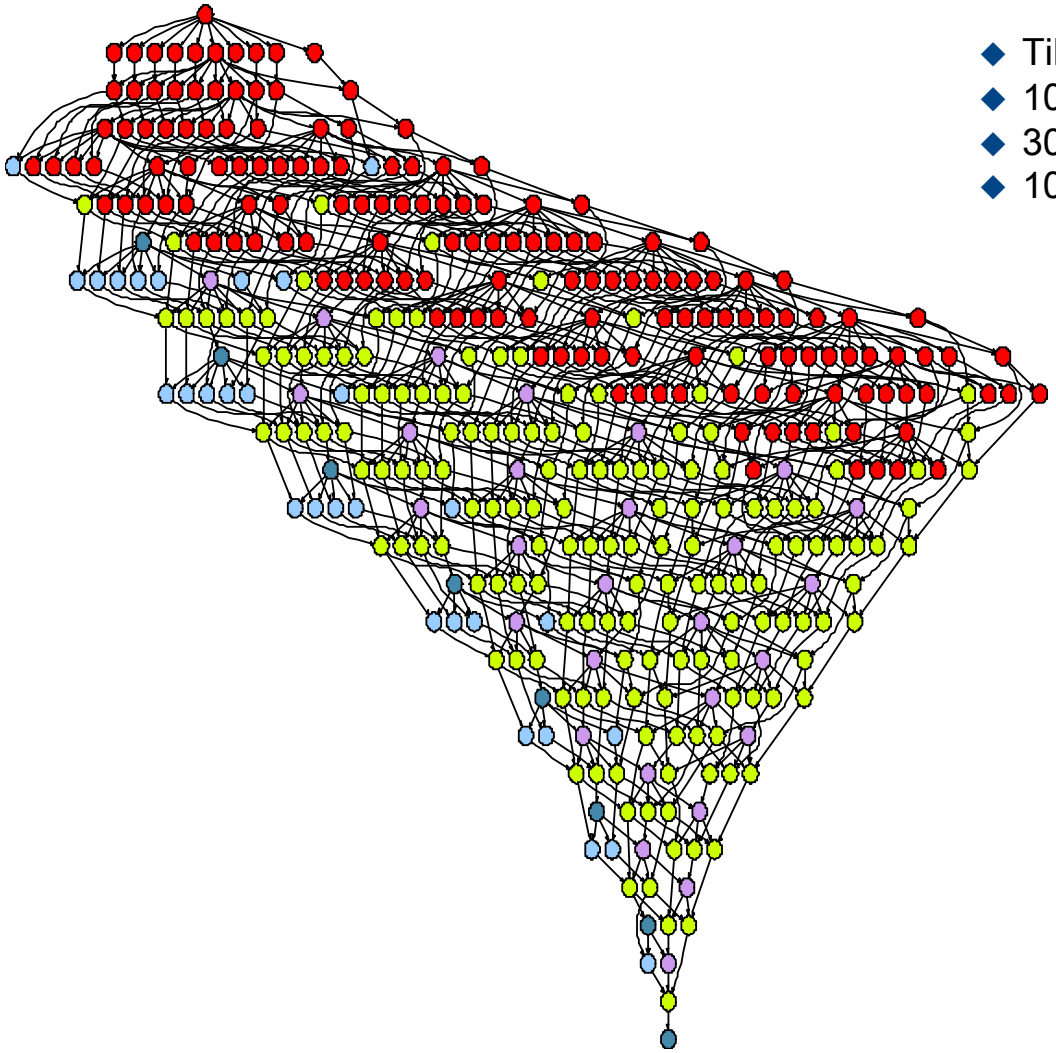
Dynamic Scheduling: Sliding Window



- Tile LU factorization
- 10 x 10 tiles
- 300 tasks
- 100 task window

PLASMA Local Scheduling

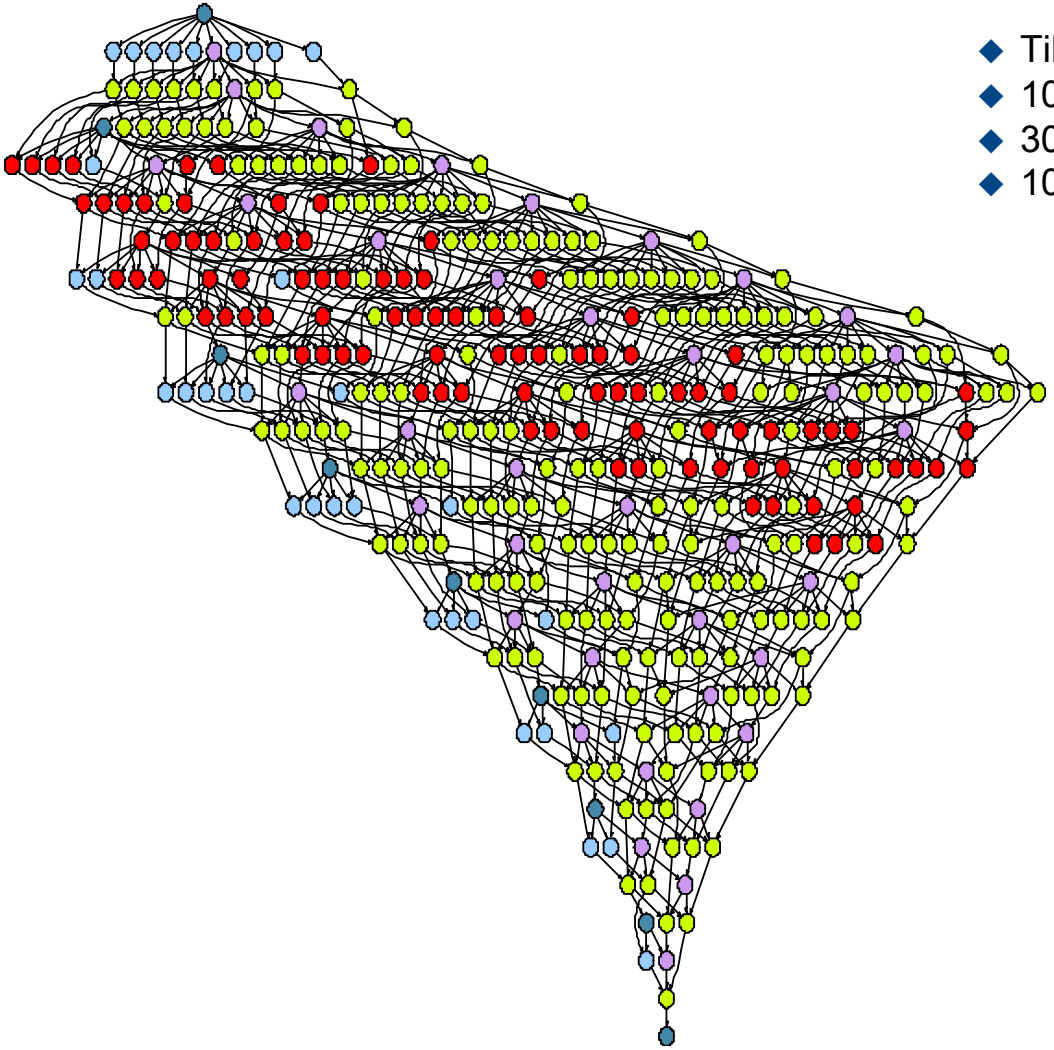
Dynamic Scheduling: Sliding Window



- Tile LU factorization
- 10 x 10 tiles
- 300 tasks
- 100 task window

PLASMA Local Scheduling

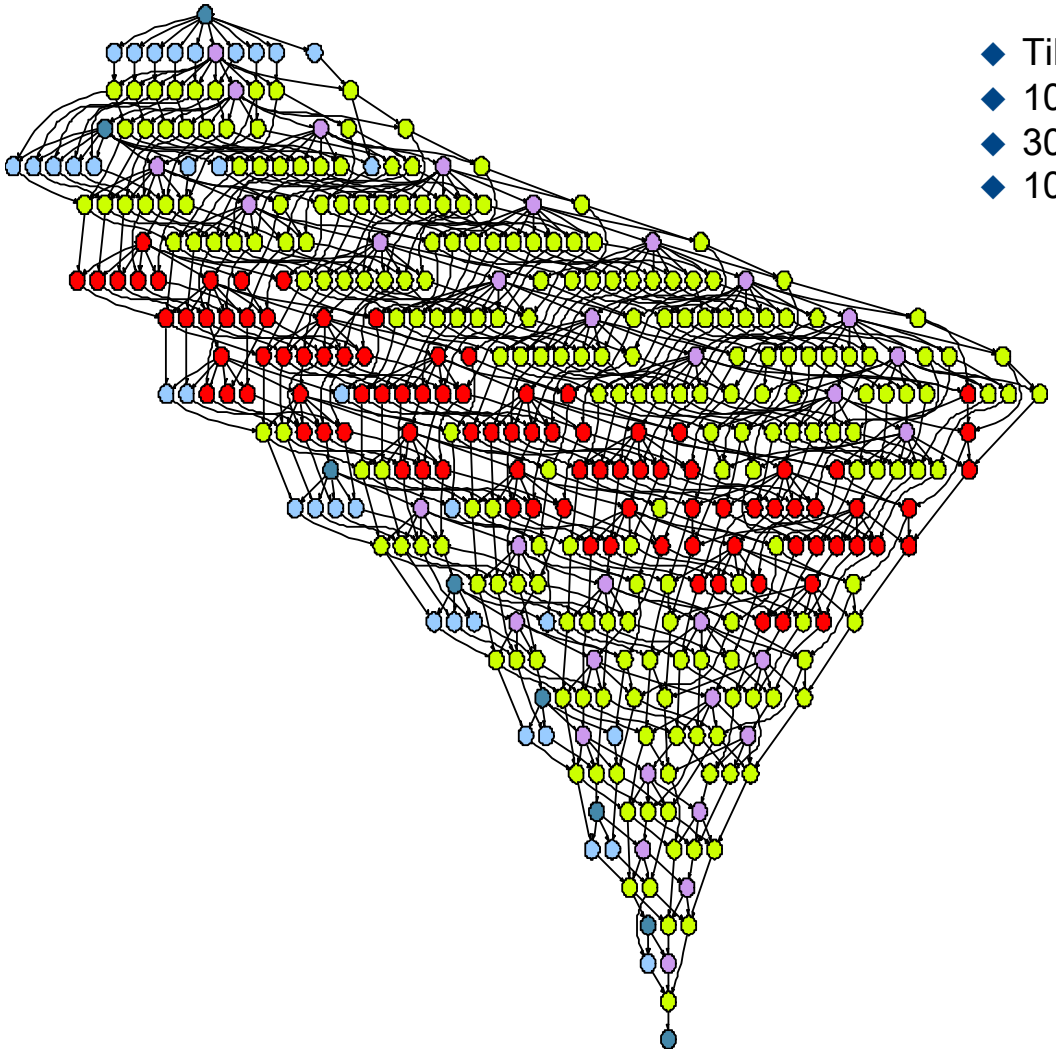
Dynamic Scheduling: Sliding Window



- Tile LU factorization
- 10 x 10 tiles
- 300 tasks
- 100 task window

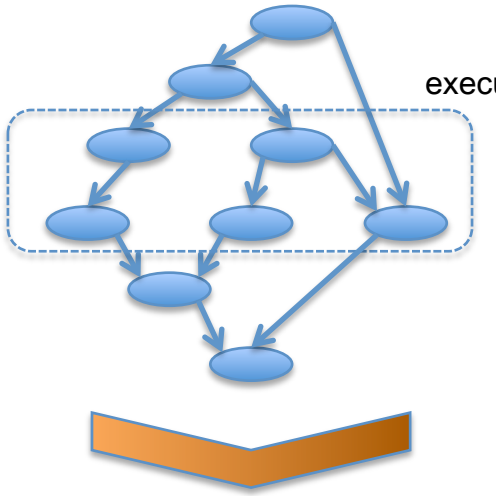
PLASMA Local Scheduling

Dynamic Scheduling: Sliding Window



- Tile LU factorization
- 10 x 10 tiles
- 300 tasks
- 100 task window

PLASMA
(On Node)



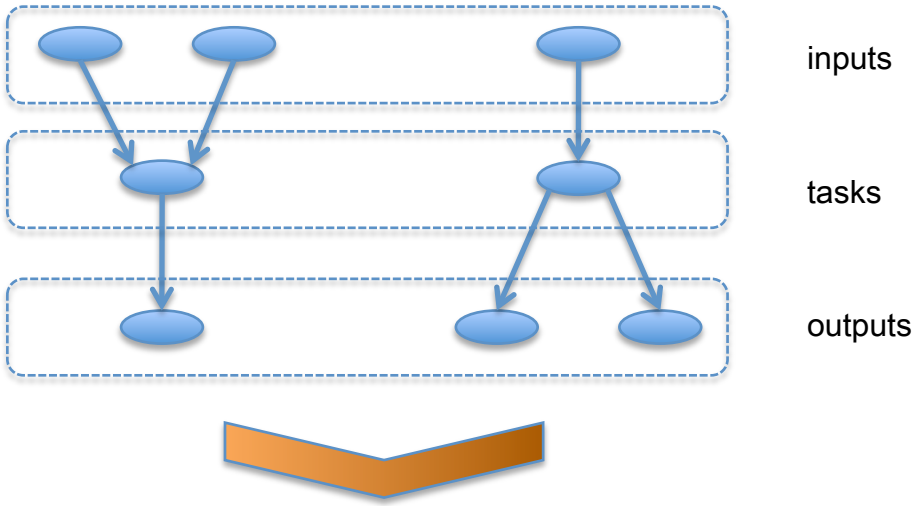
QUARK

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)



DAGuE

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)

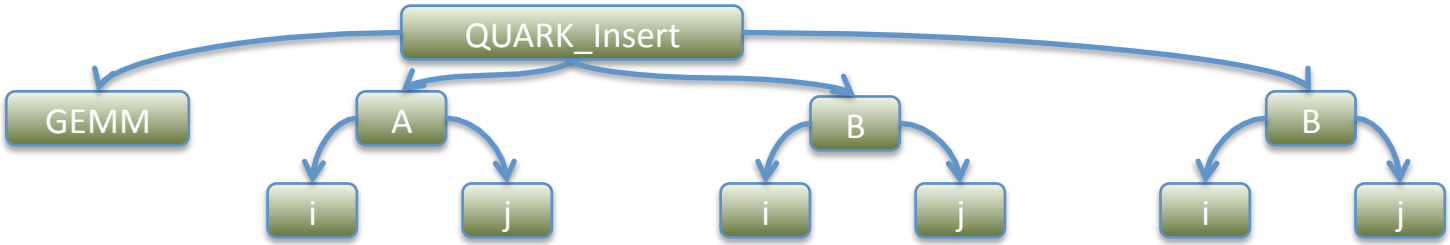
DAG: Conceptualized & Parameterized

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

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



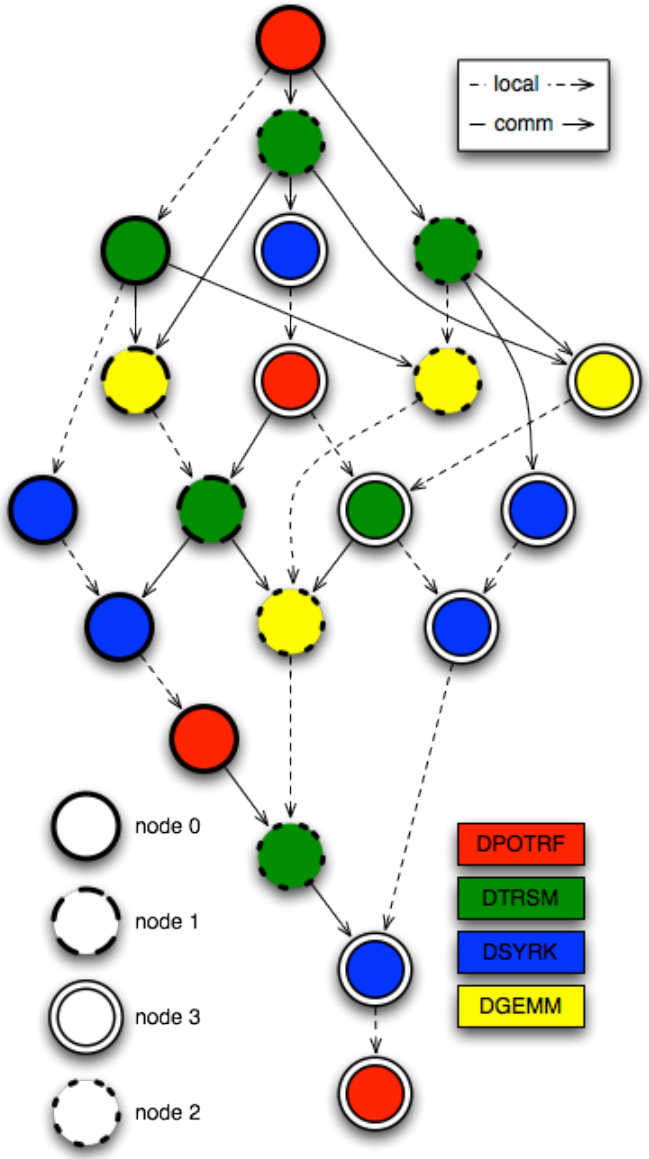
Analyze dependencies with Omega Test

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

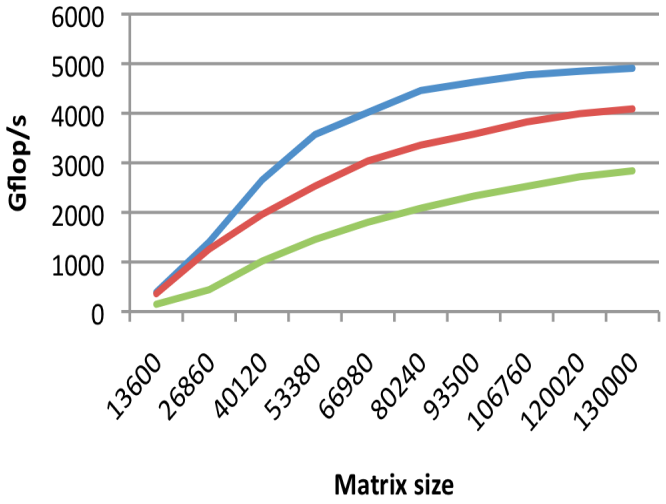
Loops & array references have to be affine

Generate Code which has the Parameterized DAG

Example: Cholesky 4x4



* RT is using the symbolic information from the compiler to make scheduling, message passing, & RT decisions
* Data distribution: regular, irregular
* Task priorities
* No left looking or right looking, more adaptive or opportunistic

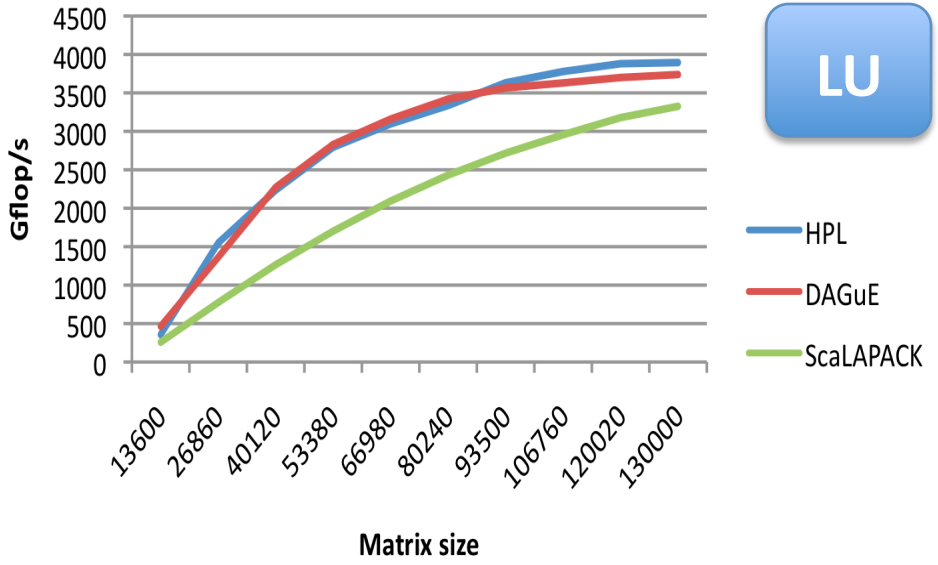


Cholesky

DAGuE
DSBP
ScaLAPACK

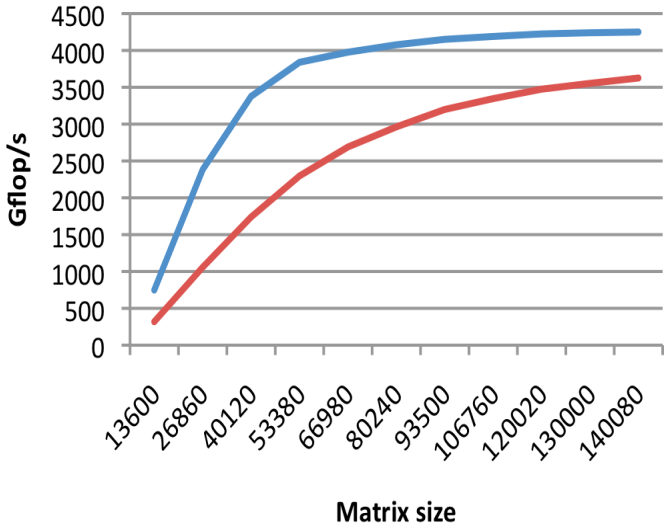
DSBP = Distributed Square Block Packed

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



LU

HPL
DAGuE
ScaLAPACK



QR

DAGuE
ScaLAPACK

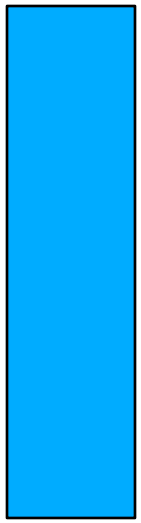
Communication Avoiding Algorithms

- Goal: Algorithms that communicate as little as possible
- Jim Demmel and company have been working on algorithms that obtain a provable minimum communication. (M. Anderson yesterday)
- Direct methods (BLAS, LU, QR, SVD, other decompositions)
  - Communication lower bounds for *all* these problems
  - Algorithms that attain them (*all* dense linear algebra, some sparse)
- Iterative methods – Krylov subspace methods for Ax=b, Ax=λx
  - Communication lower bounds, and algorithms that attain them (depending on sparsity structure)
- For QR Factorization they can show:

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

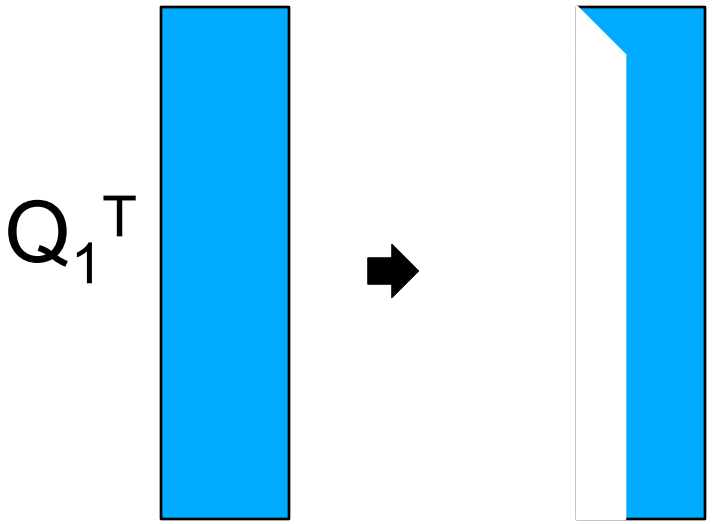
Standard QR Block Reduction

- **We have a *m x n* matrix *A* we want to reduce to upper triangular form.**



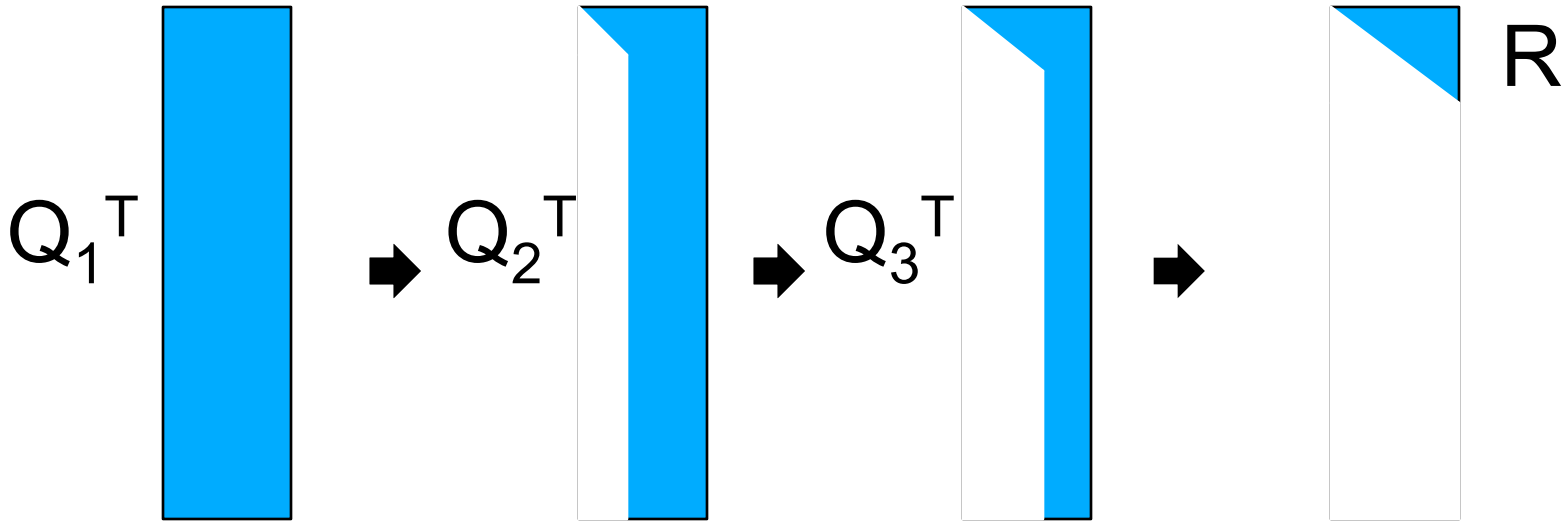
Standard QR Block Reduction

- **We have a *m x n* matrix *A* we want to reduce to upper triangular form.**



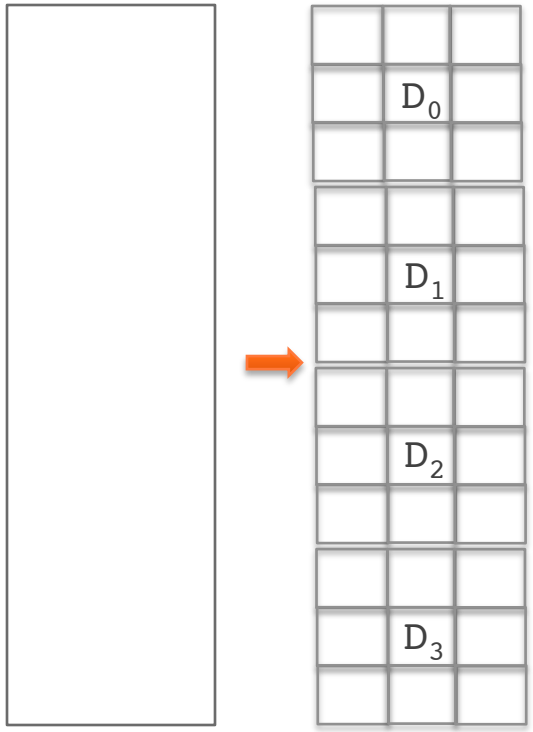
Standard QR Block Reduction

- **We have a *m x n* matrix *A* we want to reduce to upper triangular form.**



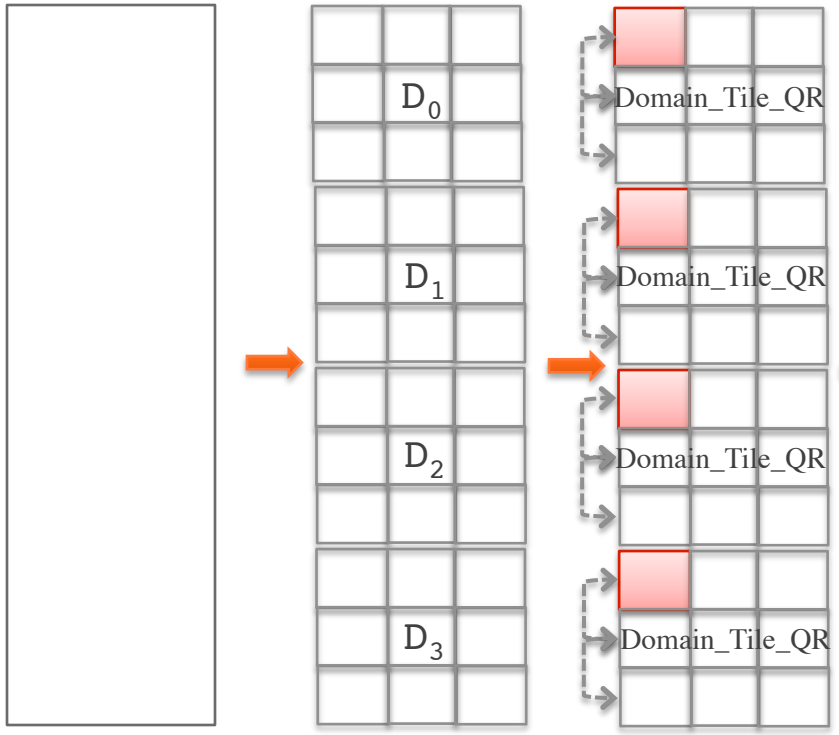
$$A = Q_1 Q_2 Q_3 R = QR$$

Communication Avoiding QR Example



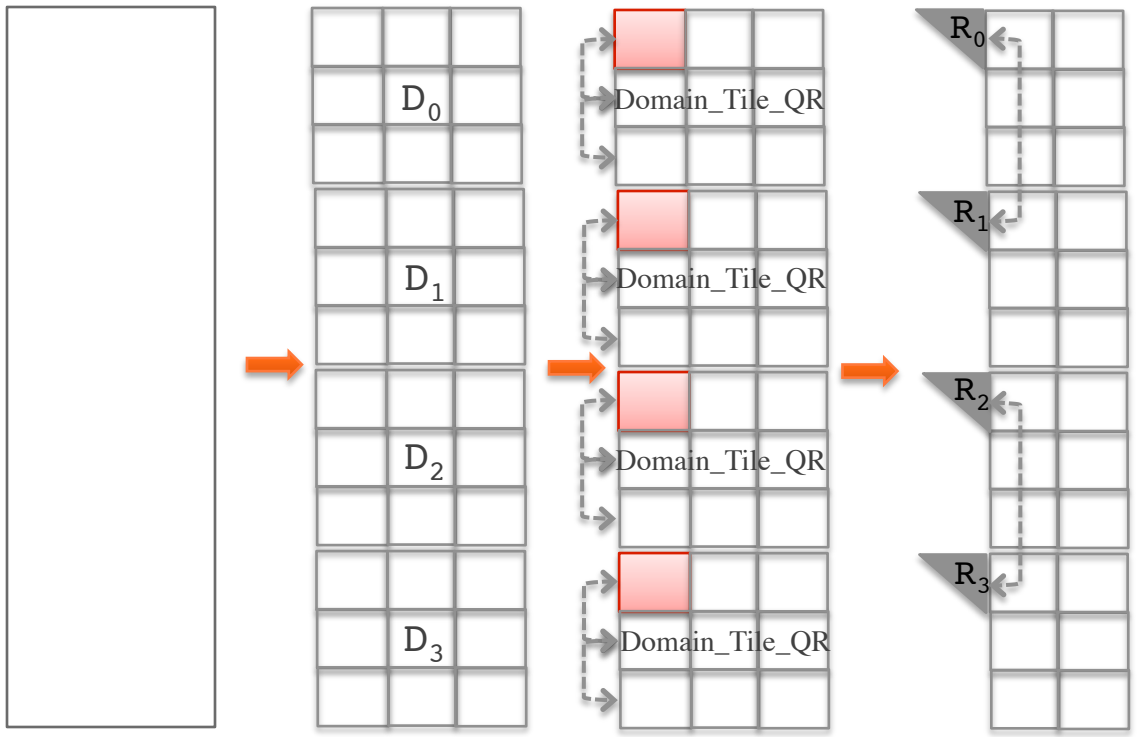
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Avoiding QR Example



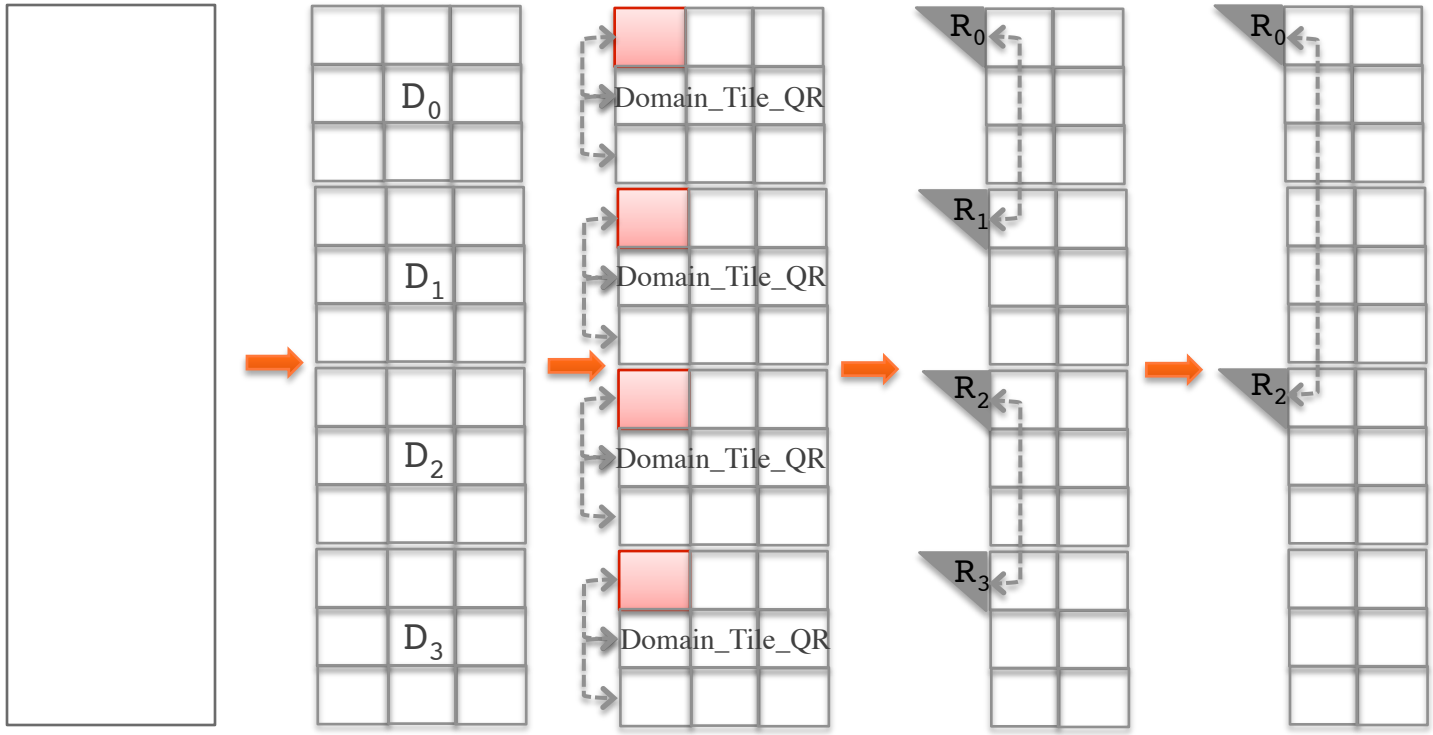
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Avoiding QR Example



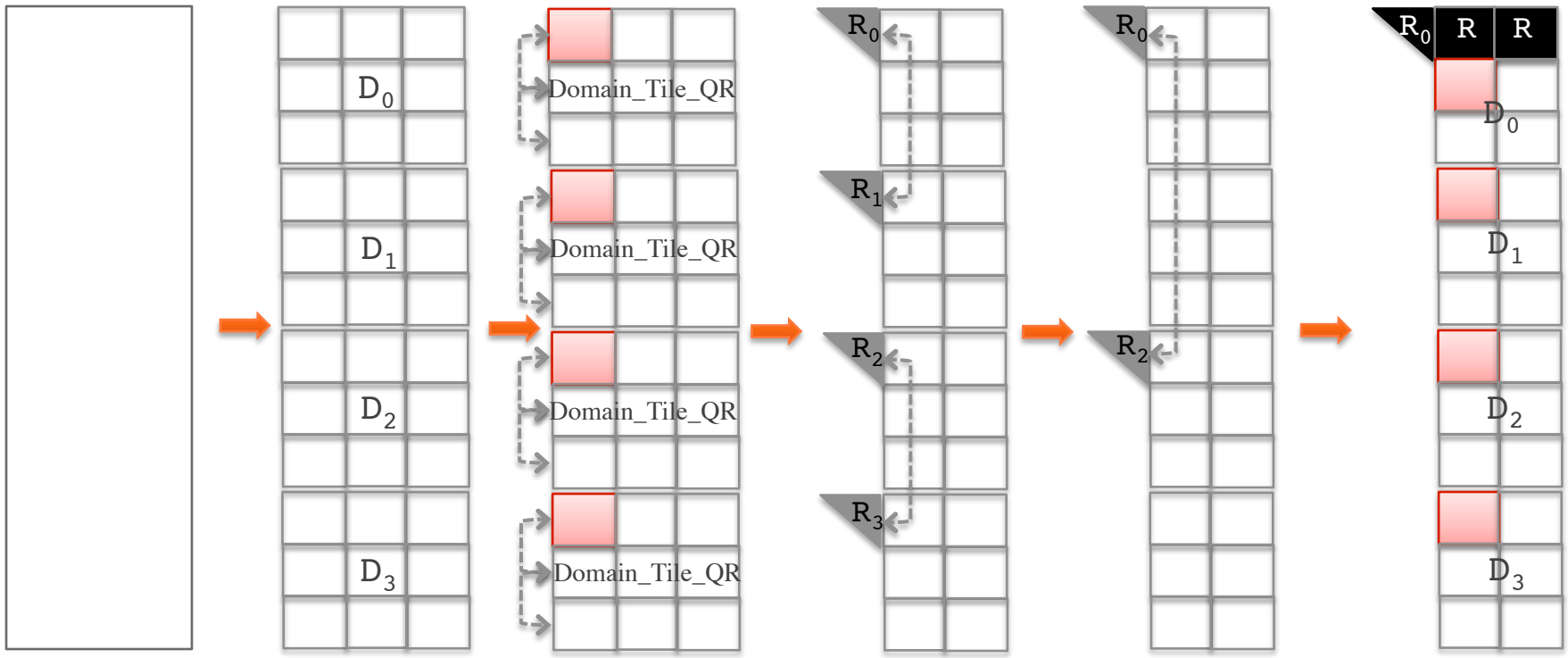
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Avoiding QR Example



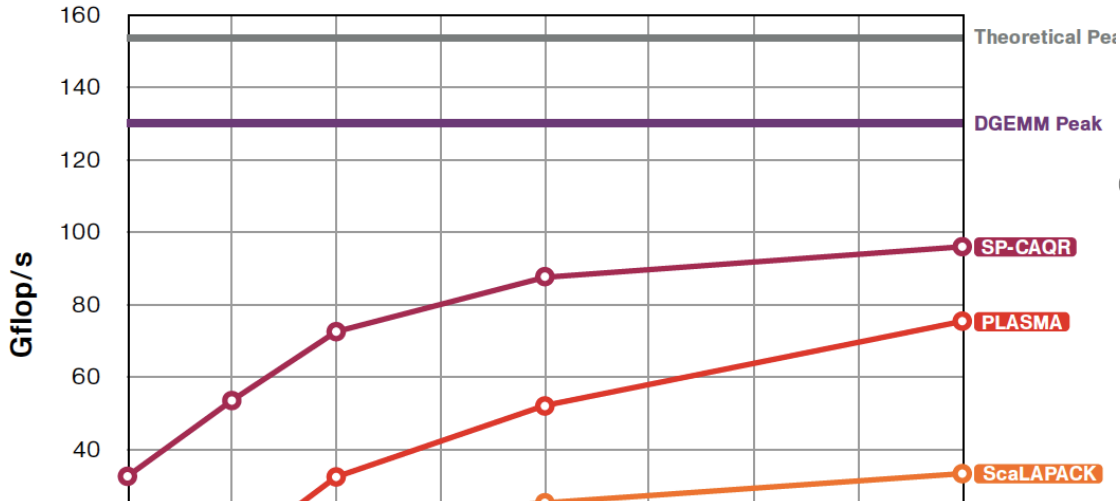
A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

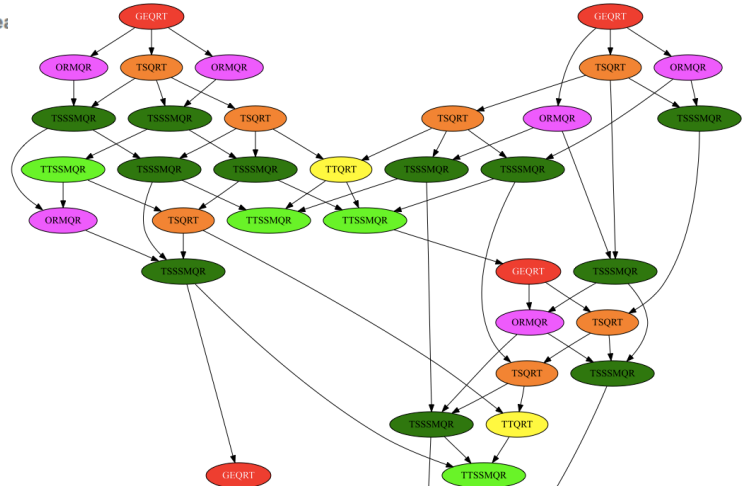
Communication Avoiding QR Example

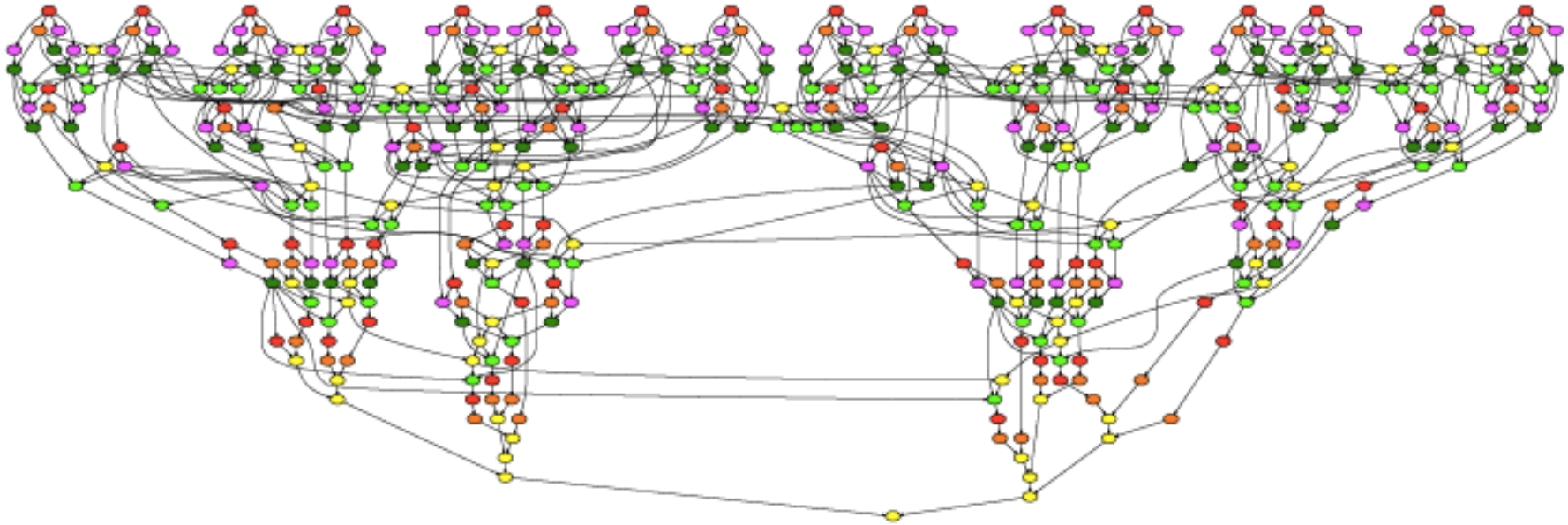


A. Pothen and P. Raghavan. Distributed orthogonal factorization. In *The 3rd Conference on Hypercube Concurrent Computers and Applications, volume II, Applications,* pages 1610–1620, Pasadena, CA, Jan. 1988. ACM. Penn. State.

Communication Reducing QR Factorization







Mixed Precision Methods

- Mixed precision, use the lowest precision required to achieve a given accuracy outcome
  - Improves runtime, reduce power consumption, lower data movement
  - Reformulate to find correction to solution, rather than solution; Δx rather than x.

Idea Goes Something Like This…

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

Mixed-Precision Iterative Refinement

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

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

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

Mixed-Precision Iterative Refinement

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

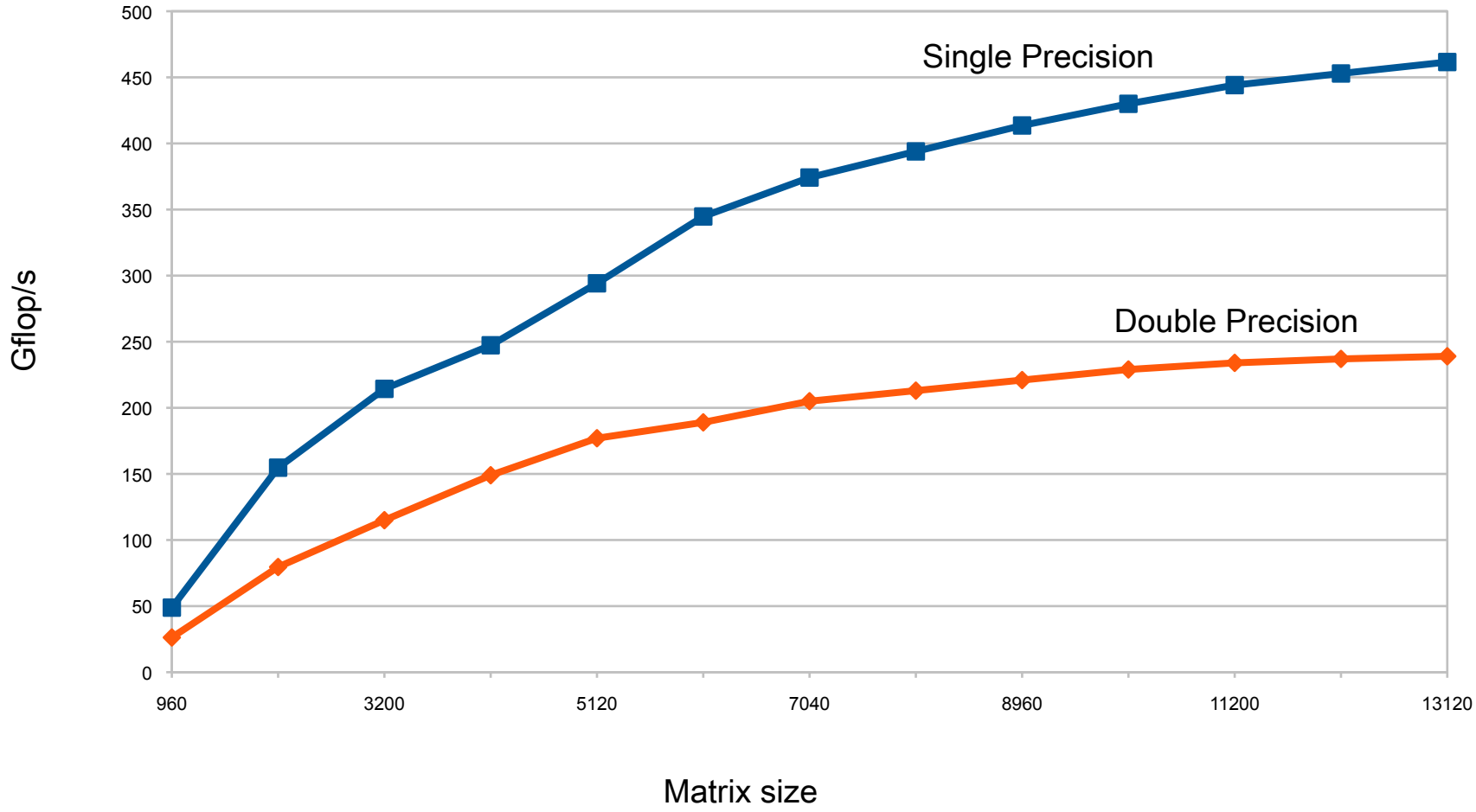
L U = lu(A)	SINGLE	$O(n^3)$
x = L\(U\b)	SINGLE	$O(n^2)$
r = b – Ax	DOUBLE	$O(n^2)$
WHILE || r || not small enough		
z = L\(U\r)	SINGLE	$O(n^2)$
x = x + z	DOUBLE	$O(n^1)$
r = b – Ax	DOUBLE	$O(n^2)$
END		

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

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

Ax = b

FERMI Tesla C2050: 448 CUDA cores @ 1.15GHz
SP/DP peak is 1030 / 515 GFlop/s

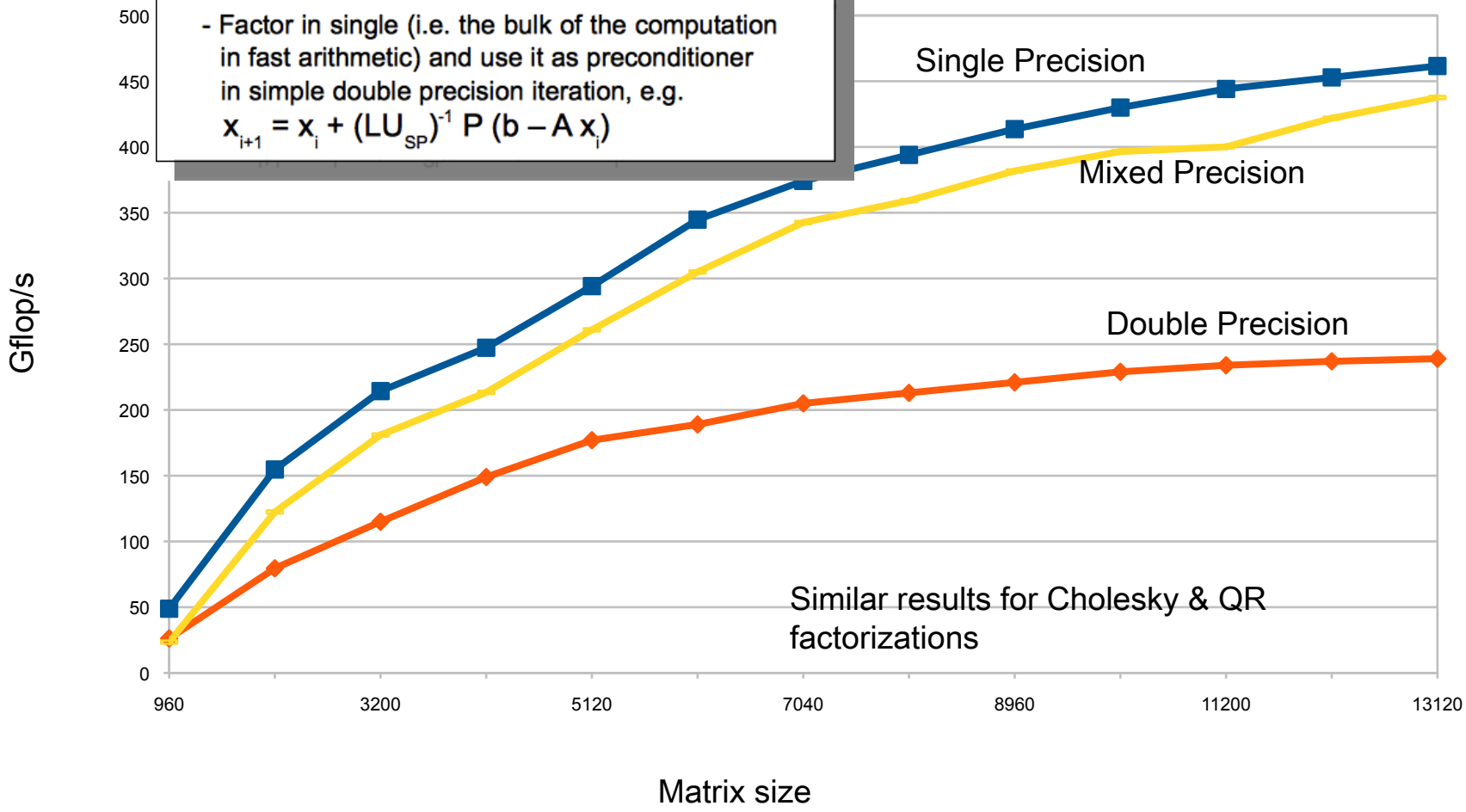


Ax = b

FERMI Tesla C2050: 448 CUDA cores @ 1.15GHz
SP/DP peak is 1030 / 515 GFlop/s

- **Direct solvers**
  - Factor and solve in working precision
- **Mixed Precision Iterative Refinement**
  - Factor in single (i.e. the bulk of the computation in fast arithmetic) and use it as preconditioner in simple double precision iteration, e.g.

$$x_{i+1} = x_i + (LU_{SP})^{-1} P (b - A x_i)$$



Reproducibility

- For example $\sum x_i$ when done in parallel can't guarantee the order of operations.
- Lack of reproducibility due to floating point nonassociativity and algorithmic adaptivity (including autotuning) in efficient production mode
- Bit-level reproducibility may be unnecessarily expensive most of the time
- Force routine adoption of uncertainty quantification
  - Given the many unresolvable uncertainties in program inputs, bound the error in the outputs in terms of errors in the inputs

Conclusions

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

Last …

- Thank a number of people who have helped with this work
  - Emmanuel Agullo, George Bosilca, Aurelien Bouteiller, Anthony Danalis, Jim Demmel, Tingxing "Tim" Dong, Mathieu Faverge, Azzam Haidar, Thomas Herault, Mitch Horton, Jakub Kurzak, Julien Langou, Julie Langou, Pierre Lemarinier, Piotr Luszczek, Hatem Ltaief, Stanimire Tomov, Asim YarKhan
- More details tomorrow Session 21 on Numerical Algorithms

55

Published in the January 2011 issue of The International Journal of High Performance Computing Applications

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

- *Alan Turing (1912—1954)*

- www.exascale.org