June 22 - 25, 2010

A Scalable High Performant Cholesky Factorization for Multicore with GPU Accelerators

H. Ltaief, S. Tomov, R. Nath, P. Du,
and J. Dongarra
University of Tennessee

VECPAR'10

ICL UT
INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

35rd List: The TOP10

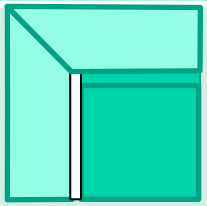
Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	MFlops /Watt
1	DOE / OS Oak Ridge Nat Lab	Jaguar / Cray Cray XT5 sixCore 2.6 GHz	USA	224,162	1.76	75	7.0	251
2	Nat. Supercomputer Center in Shenzhen	Nebulea / Dawning / TC3600 Blade, Intel X5650, Nvidia C2050 GPU	China	120,640	1.27	43	2.58	493
3	DOE / NNSA Los Alamos Nat Lab	Roadrunner / IBM BladeCenter QS22/LS21	USA	122,400	1.04	76	2.48	446
4	NSF / NICS / U of Tennessee	Kraken/ Cray Cray XT5 sixCore 2.6 GHz	USA	98,928	.831	81	3.09	269
5	Forschungszentrum Juelich (FZJ)	Jugene / IBM Blue Gene/P Solution	Germany	294,912	.825	82	2.26	365
6	NASA / Ames Research Center/NAS	Pleiades / SGI SGI Altix ICE 8200EX	USA	56,320	.544	82	3.1	175
7	National SC Center in Tianjin / NUDT	Tianhe-1 / NUDT TH-1 / IntelQC + AMD ATI Radeon 4870	China	71,680	.563	46	1.48	380
8	DOE / NNSA Lawrence Livermore NL	BlueGene/L IBM eServer Blue Gene Solution	USA	212,992	.478	80	2.32	206
9	DOE / OS Argonne Nat Lab	Intrepid / IBM Blue Gene/P Solution	USA	163,840	.458	82	1.26	363
10	DOE / NNSA Sandia Nat Lab	Red Sky / Sun / SunBlade 6275	USA	42,440	.433	87	2.4	180

#2 – National Supercomputer Center in Shenzhen - Dawning

- ✓ Nebulae
- ✓ Hybrid system, commodity + GPUs
- ✓ Theoretical peak 2.98 Pflop/s
- ✓ Linpack Benchmark at 1.27 Pflop/s
- ✓ 4640 nodes, each node:
  - ✓ 2 Intel 6-core Xeon5650 + Nvidia Fermi C2050 GPU (each 14 cores)
  - ✓ 120,640 cores
  - ✓ Infiniband connected

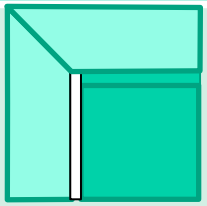
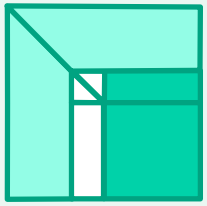
A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations

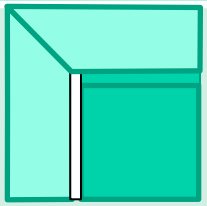
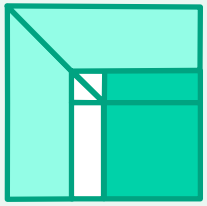
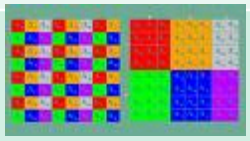
A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

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

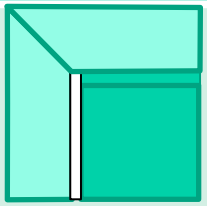
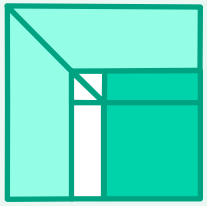
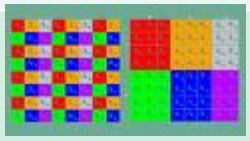
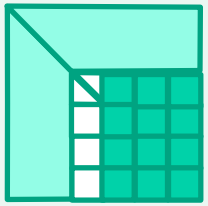
A New Generation of Software:

Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

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

A New Generation of Software:

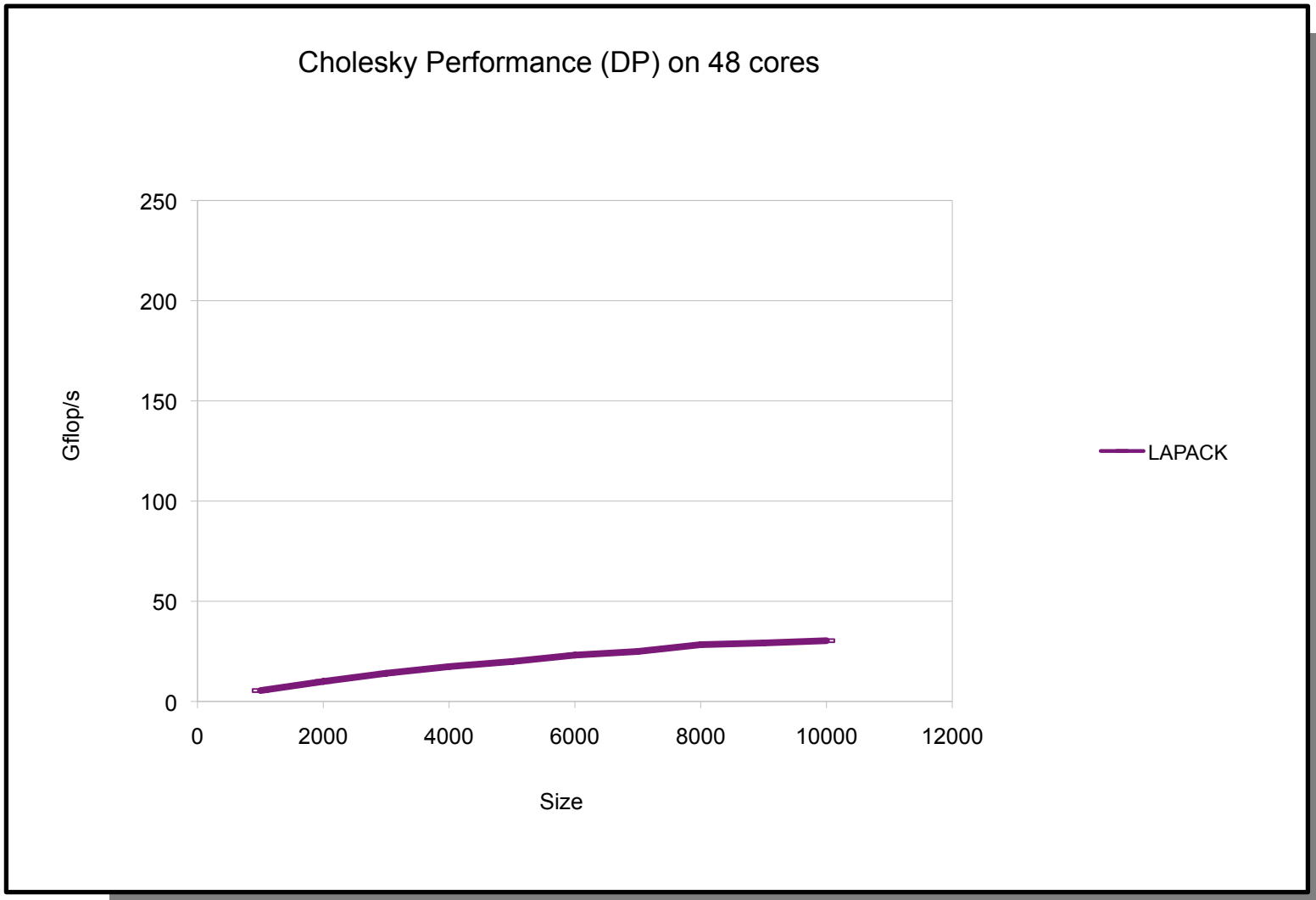
Parallel Linear Algebra Software for Multicore Architectures (PLASMA)

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
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels

What is PLASMA?

- Software library
- Dense linear algebra
  - Linear systems & least squares (LU, Cholesky, QR/LQ)
  - Eigenvalues & singular values – in research phase
- Multicore processors
  - Multi-socket multi-core
  - Shared memory systems
  - NUMA systems

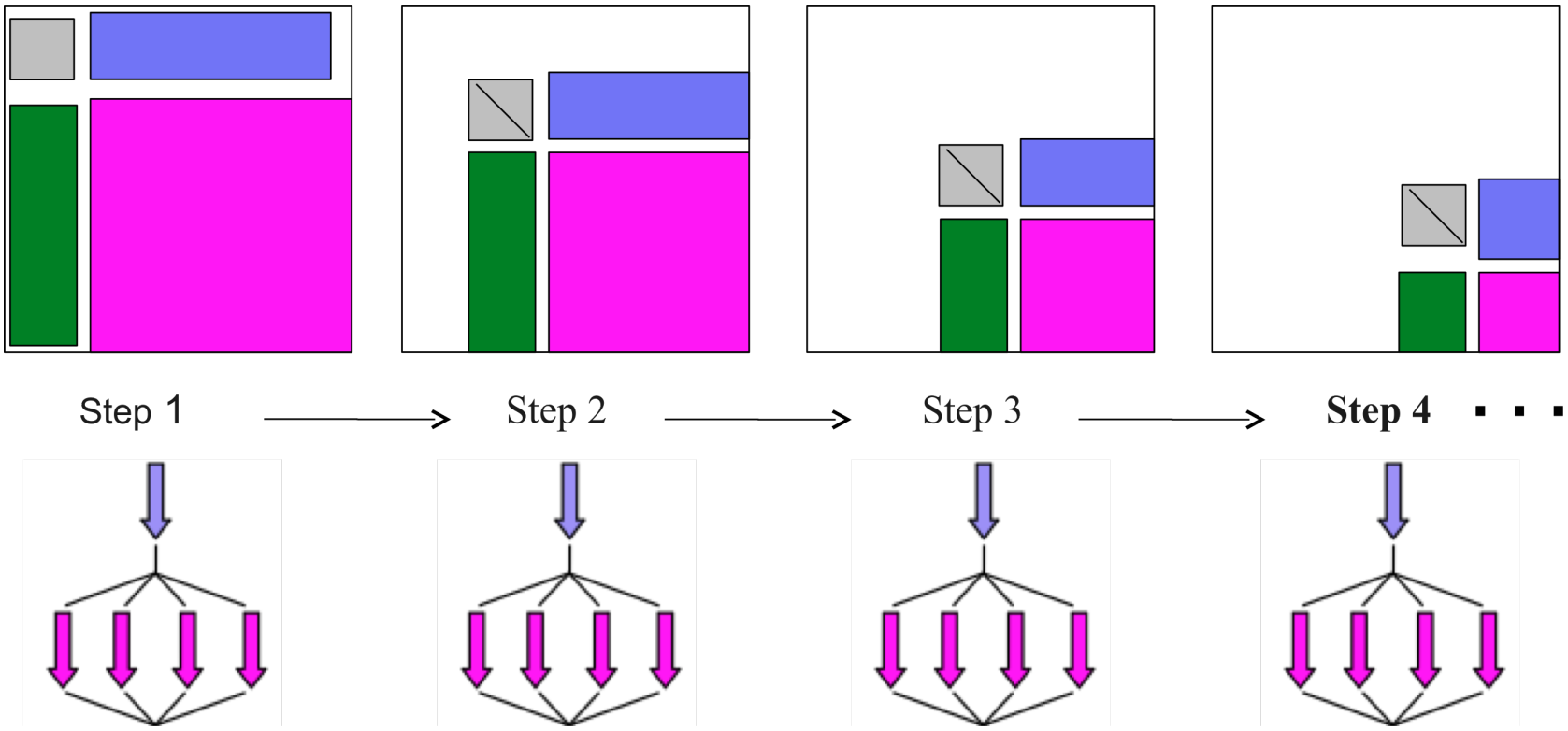
PLASMA: Motivations (1)



PLASMA: Motivations (2)

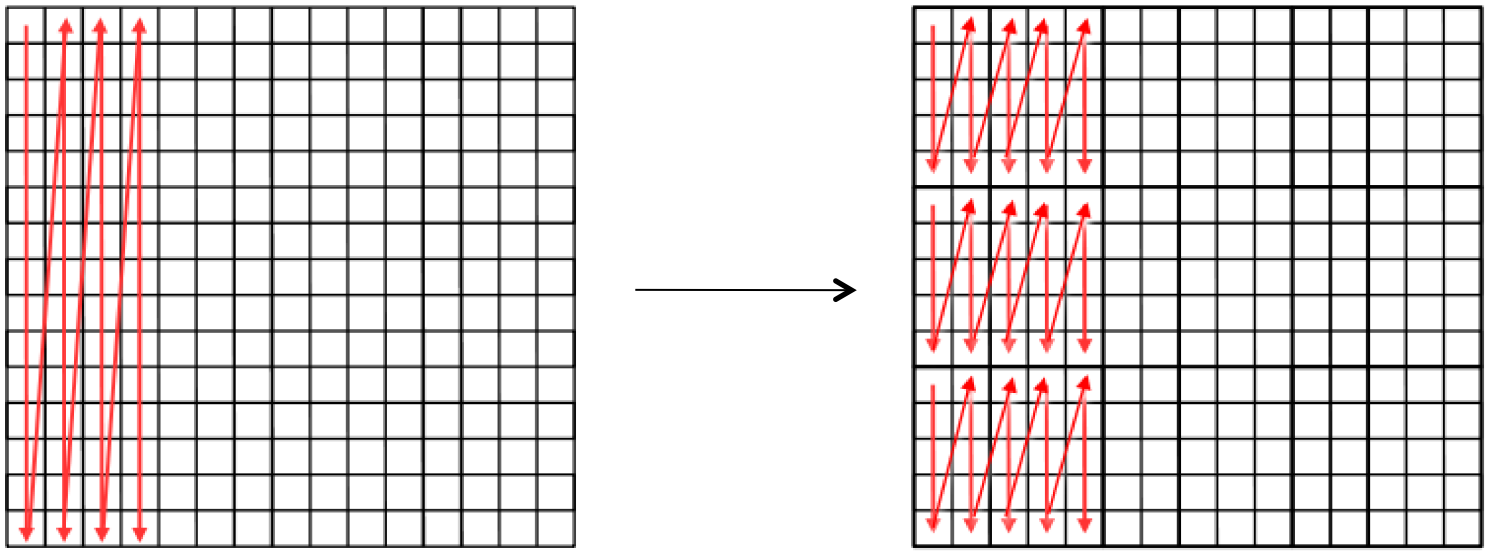
- ✓ Major Changes to Software
- ✓ Must rethink the design of our software
  - ✓ Another disruptive technology
    - ✓ Similar to what happened with cluster computing and message passing
    - ✓ Rethink and rewrite the applications, algorithms, and software
- ✓ Numerical libraries for example will change
  - ✓ For example, both LAPACK and ScaLAPACK will undergo major changes to accommodate this

PLASMA: Motivations (3)



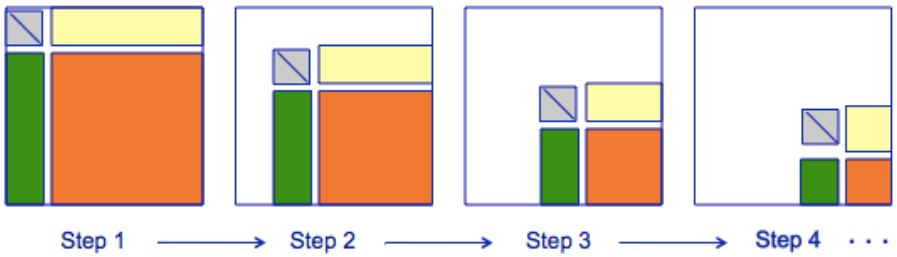
- Fork-join parallelism bottleneck.

PLASMA: Motivations (4)

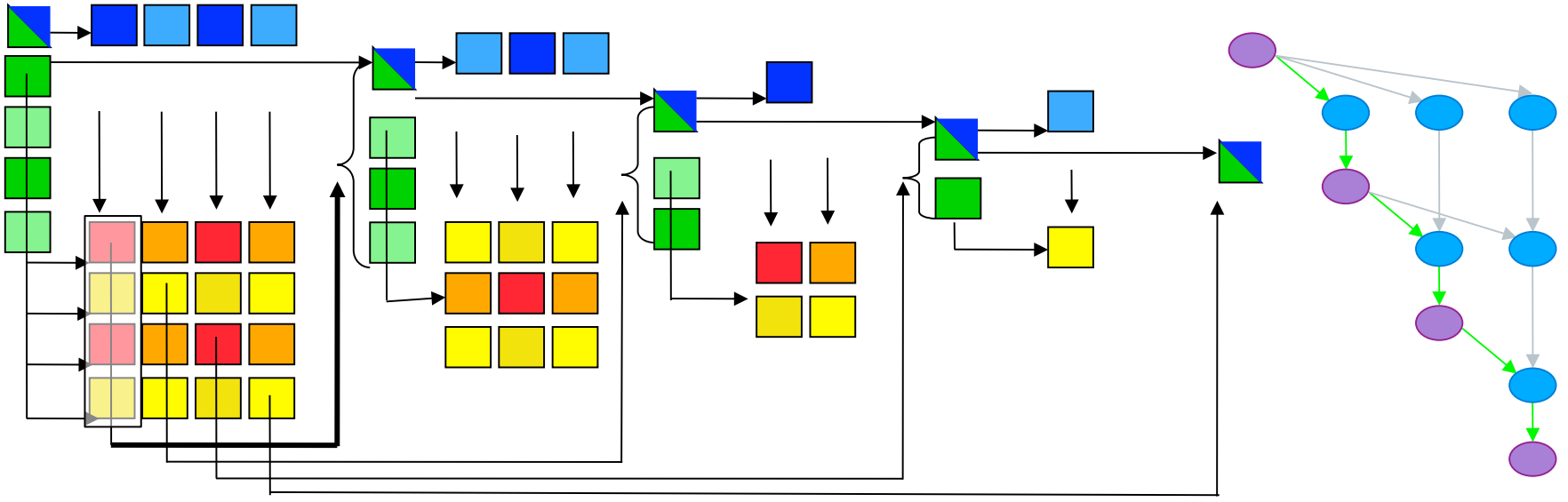


- Expensive Access to Memory. (Lapack Data Layout)
- Tile Algorithms. (Block Data Layout)

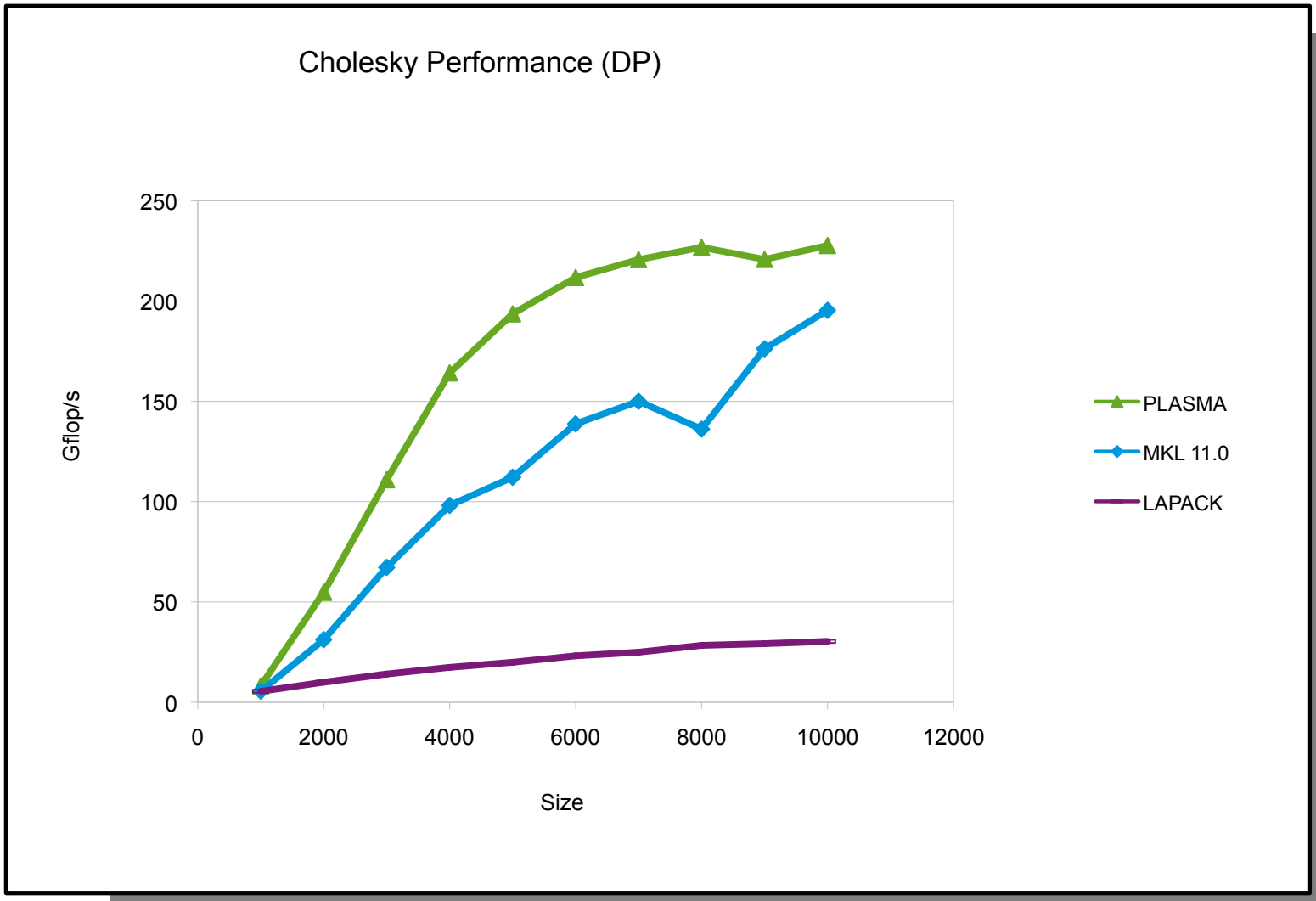
PLASMA: Motivations (5)



Break into smaller tasks and remove dependencies



PLASMA Performance (Cholesky, 48 cores)



PLASMA (Redesign LAPACK/ScaLAPACK)

Parallel Linear Algebra Software for Multicore Architectures

- ✓ **Asychronicity**
  - ✓ Avoid fork-join (Bulk sync design)
- ✓ **Dynamic Scheduling**
  - ✓ Out of order execution
- ✓ **Fine Granularity**
  - ✓ Independent block operations
- ✓ **Locality of Reference**
  - ✓ Data storage – Block Data Layout

Lead by UTK, UCB, and UCD similar to LAPACK/ScaLAPACK as a community effort

Challenges of using GPUs

- ✓ **High** levels of **parallelism**
  Many GPU cores, serial kernel execution
  [e.g. 240 in the Nvidia Tesla; up to 512 in *Fermi* – to have concurrent kernel execution]

- ✓ **Hybrid/heterogeneous architectures**
  Match algorithmic requirements to architectural strengths
  [e.g. small, non-parallelizable tasks to run on CPU, large and parallelizable on GPU]

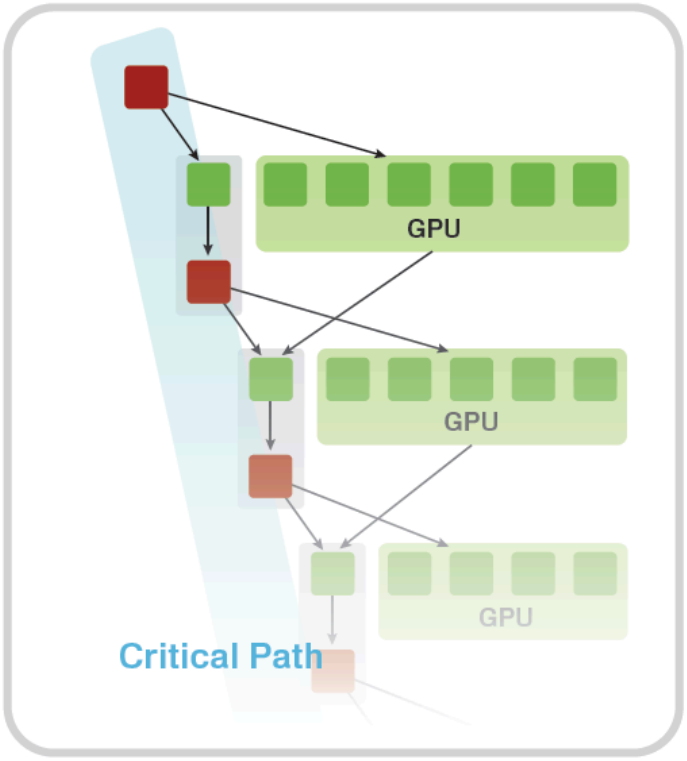
- ✓ **Compute *VS* communication gap**
  Exponentially growing gap; persistent challenge
  [Processor speed improves 59%, memory bandwidth 23%, latency 5.5%]
  [on all levels, e.g. a GPU Tesla C1070 (4 x C1060) has compute power of O(1,000) Gflop/s but GPUs communicate through the CPU using O(1) GB/s connection]

MAGMA Software

- ✓ Available through MAGMA's homepage http://icl.cs.utk.edu/magma/
- ✓ Included are the 3 one-sided matrix factorizations
- ✓ Iterative Refinement Algorithm (Mixed Precision)
- ✓ Standard (LAPACK) data layout and accuracy
- ✓ This release is intended for single GPU
- ✓ Coming soon: 2-sided reductions

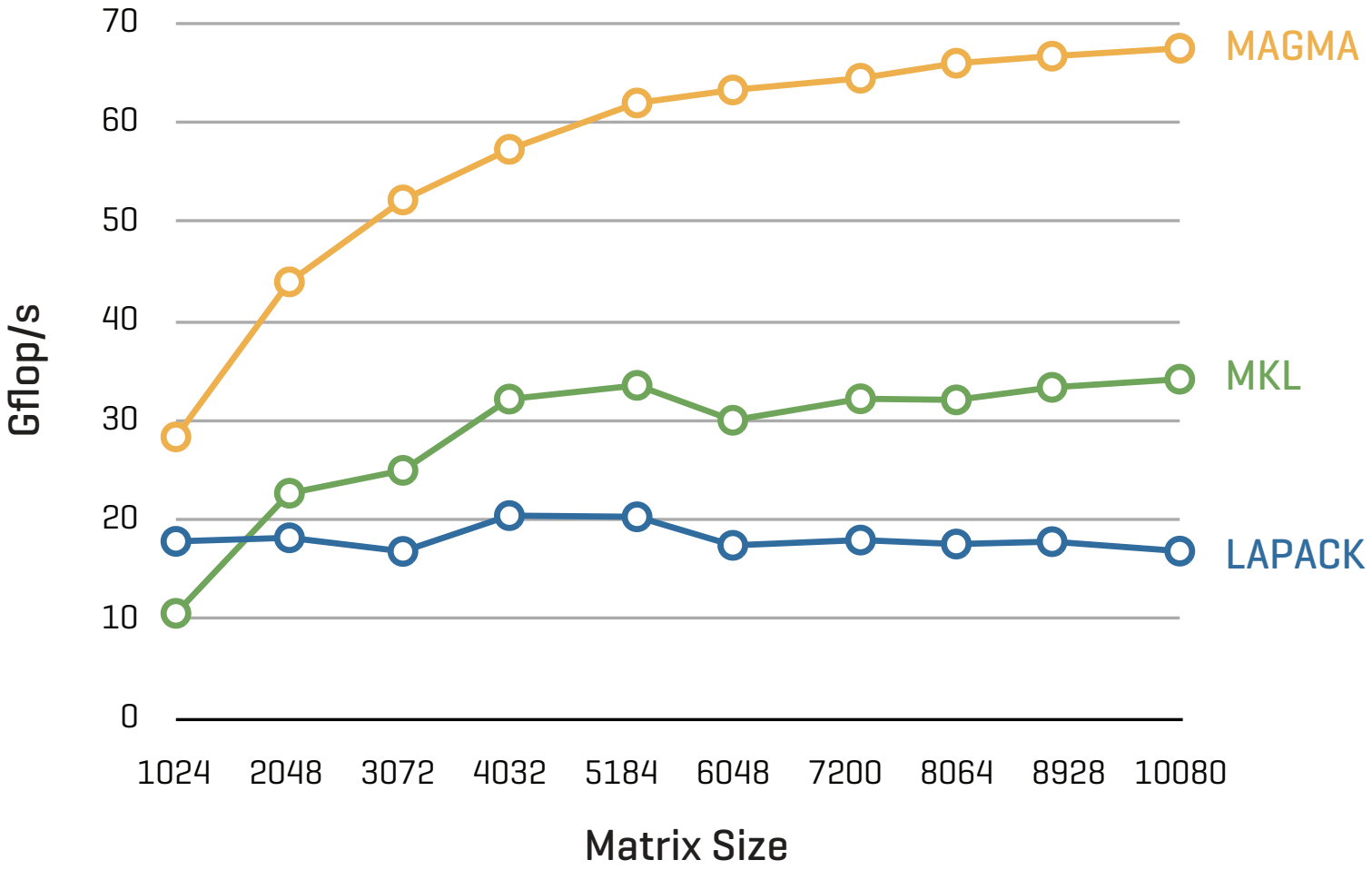
MAGMA Approach

- ✓ **Look ahead** technique to pipeline between steps of the computation



- ✓ **1 CPU** factorizes the panel, **1 GPU** updates the trailing submatrix.

MAGMA Performance



[for more performance data, see http://icl.cs.utk.edu/magma]

MAGMA Status

MAGMA 0.2

- LU, QR, Cholesky (S, C, D, Z)
- Linear solvers
  - In working precision, based on LU, QR, and Cholesky
  - Mixed-precision iterative refinement
- CPU and GPU interfaces
- Two-sided factorizations
  - Reduction to upper Hessenberg form for the general eigenvalue problem
- MAGMA BLAS
  - Routines critical for MAGMA (GEMM, SYRK, TRSM, GEMV, SYMV, etc.)

Unreleased

- Bidiagonal two-sided reduction for SVD
- Tridiagonal two-sided for the symmetric eigenvalue problem
- Divide & Conquer for the symmetric eigenvalue problem
- GEMM for FERMI
- Cholesky and QR for multiGPUs on MAGNUM tiles
- GMRES and PCG

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 Larrabee? Now called "Knights Corner" and "Knights Ferry" to come.
  - ✓ 48 x86 cores
- ✓ AMD's Fusion in 2011 – 2013
  - ✓ Multicore with embedded graphics ATI
- ✓ Nvidia's plans?

- Next hardware generation is **Hybrid**.
- PLASMA runs **ONLY** on homogeneous multicore systems.
- MAGMA runs **ONLY** on 1CPU-1GPU configuration.

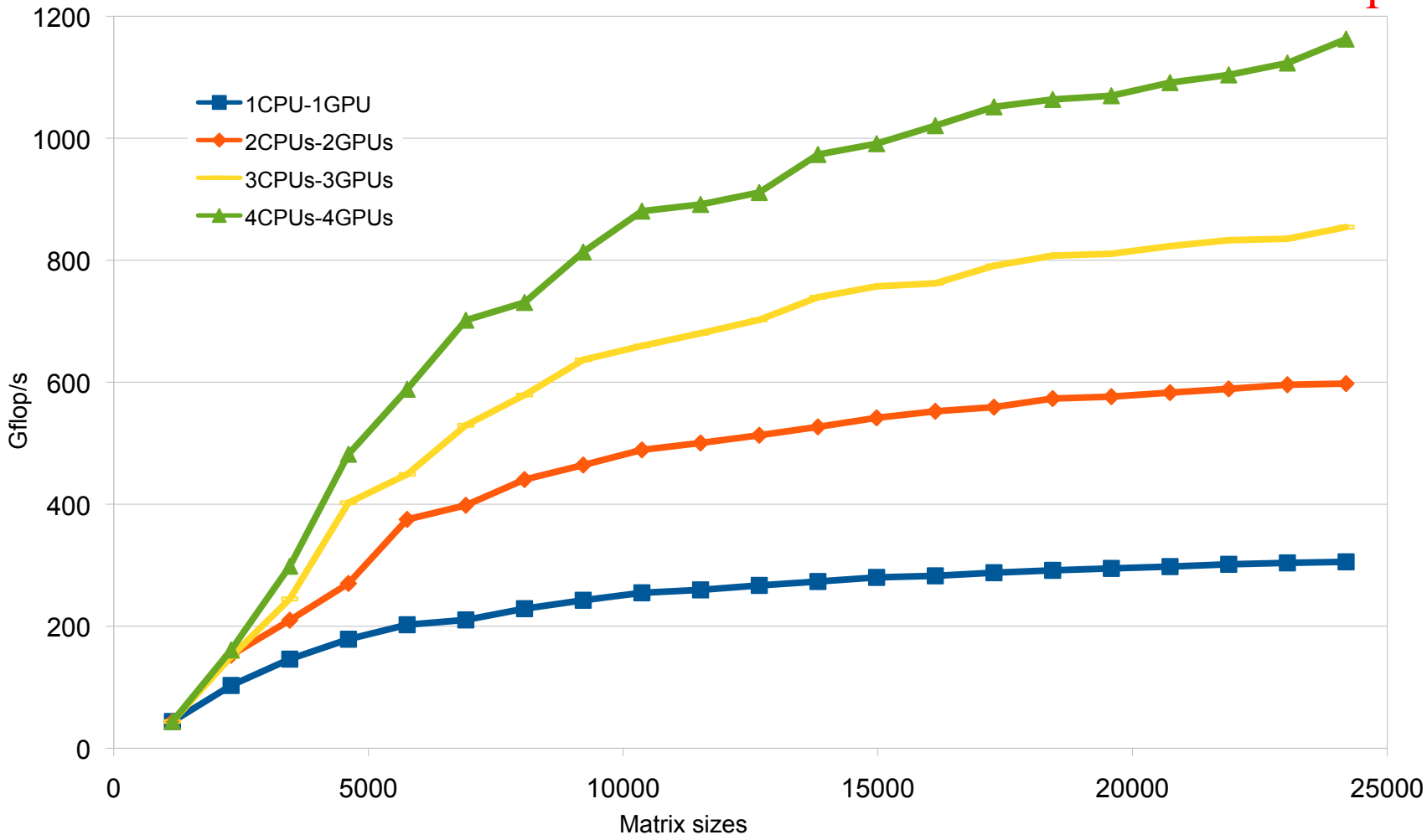
- **Idea**: Combine the strengths from both projects to target **multicore + multiGPUs**
  - Benefit from the MAGMA hybrid computation.
  - Benefit from the PLASMA Parallel thread execution.
  - Move toward Multiple CPUs – Multiple GPUs environment.
  - May require to write new high performance kernels for the GPU.
- Test case: LL^T

Cholesky on multicore + multi-GPUs (4)

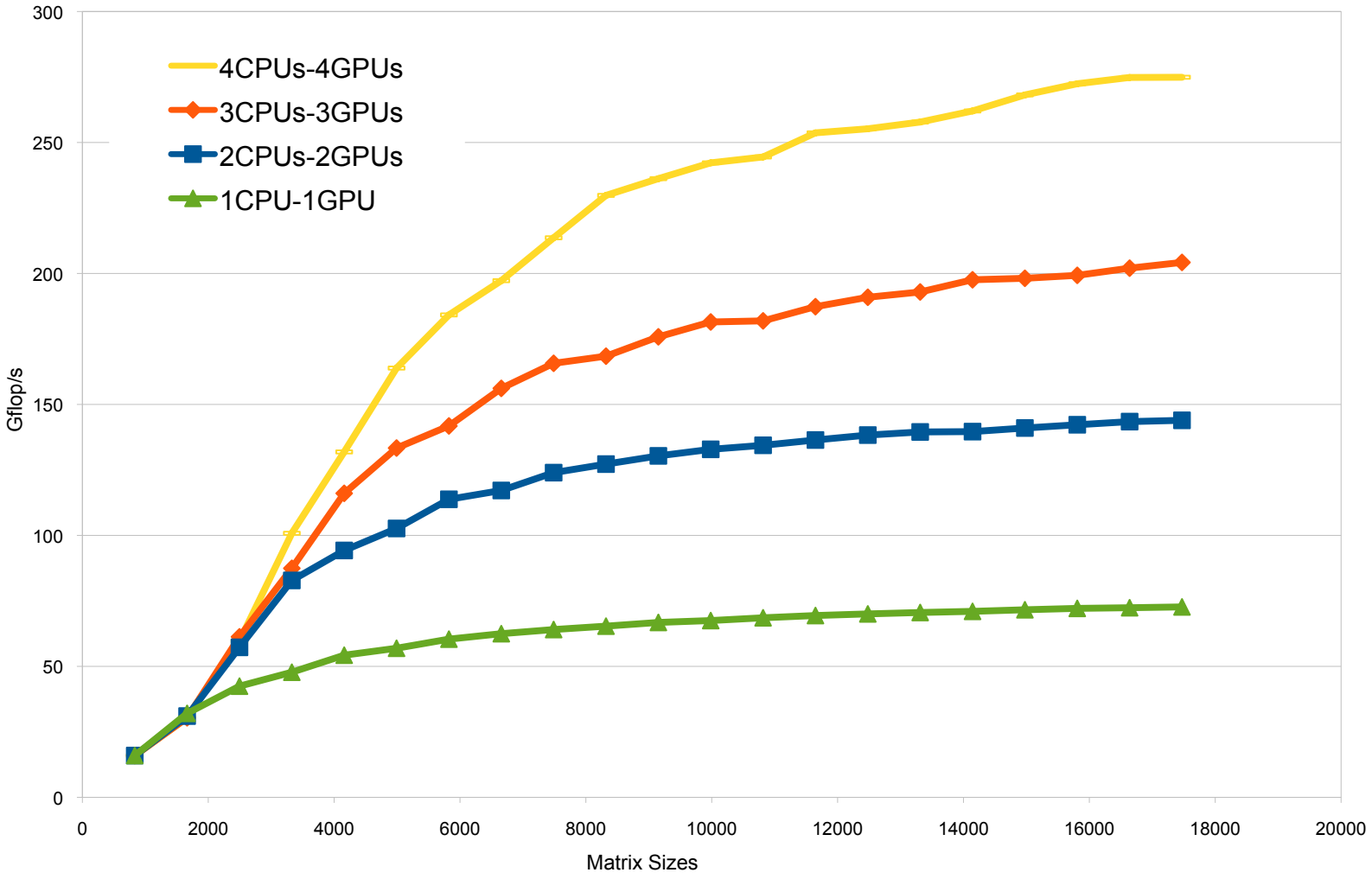
- **Hardware**
  - HOST: Two-dual core AMD Opteron 1.8GHz, 2GB memory
  - DEVICE:
    - 4 GPU TESLA C1070 1.44GHz
    - 240 computing cores per GPU
    - 4GB memory per GPU
    - Single precision floating point performance (PEAK): 3.73 Tflop/s
    - Memory bandwidth: 408 GB/s
    - System interface: PCIexpress

Single Precision - Cholesky on Multicore + Multi GPUs

1.2 Tflop/s !!!



DP Cholesky with Multiple GPUs



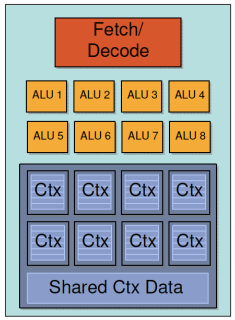
Thanks!

QUESTIONS?

NVIDIA GeForce GTX 280

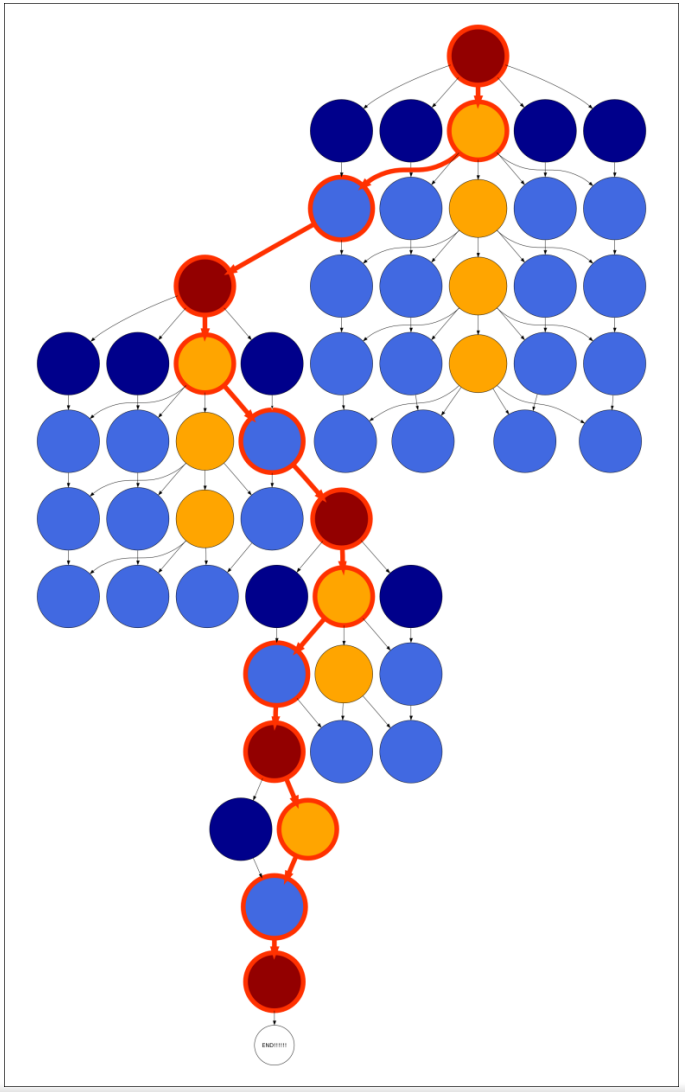
- ✓ NVIDIA-Speak
  - ✓ 240 CUDA cores (ALUs)
- ✓ Generic speak
  - ✓ 30 processing cores
    - 8 CUDA Cores (SIMD functional units) per core
  - ✓ 1 mul-add (2 flops) + 1 mul per functional unit (3 flops/cycle)
  - ✓ Best case theoretically 240 mul-adds + 240 muls per cycle
    - 1.3 GHz clock
    - 30 * 8 * (2 + 1) * 1.33 = 933 Gflop/s peak
  - ✓ Best case reality: 240 mul-adds per clock
    - Just able to do the mul-add so 2/3 or 624 Gflop/s
  - ✓ All this is single precision
    - Double precision is 78 Gflop/s peak (Factor of 8 from SP; exploit mixed prec)
  - ✓ In SP SGEMM performance 375 Gflop/s

Processing Core



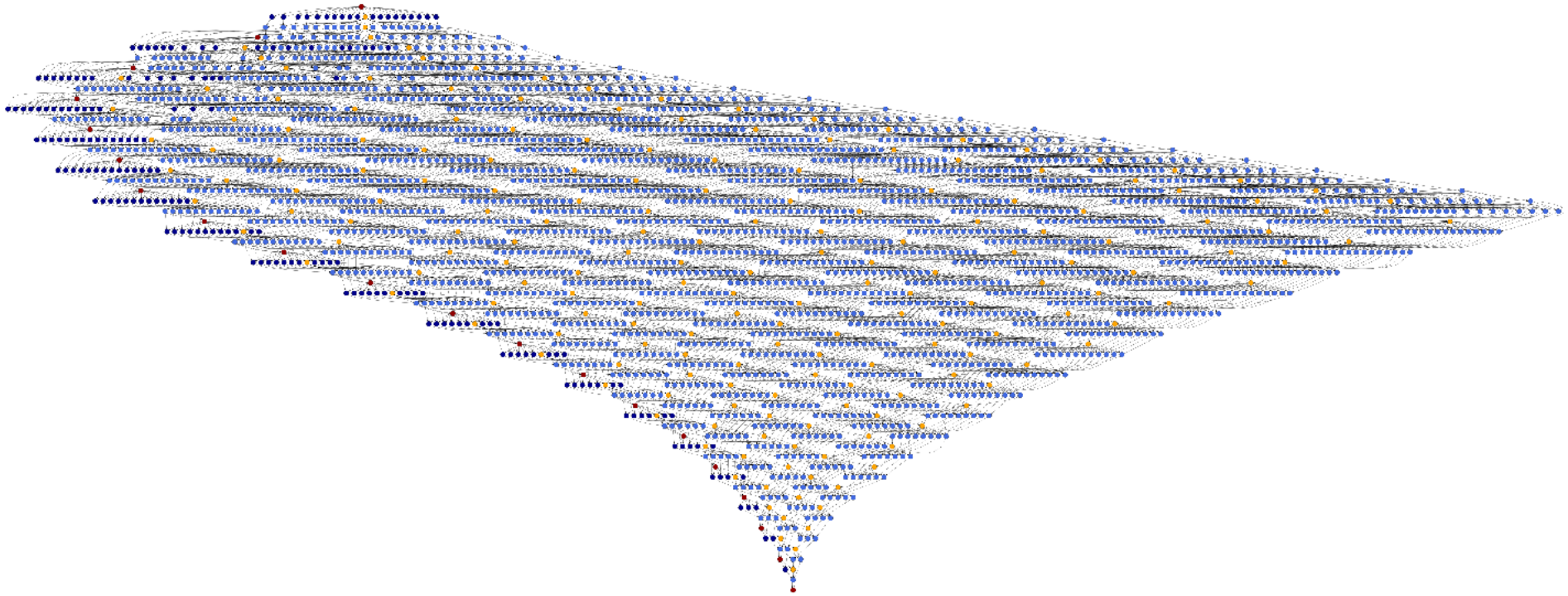
If We Had A Small Matrix Problem

- We would generate the DAG, find the critical path and execute it.
- DAG too large to generate ahead of time
  - Not explicitly generate
  - Dynamically generate the DAG as we go
- Machines will have large number of cores in a distributed fashion
  - Will have to engage in message passing
  - Distributed management
  - Locally have a run time system



The DAGs are Large

Here is the DAG for a factorization on a 20 x 20 matrix



For a large matrix say O(10^6) the DAG is huge

Many challenges for the software