THE UNIVERSITY of TENNESSEE
NATIONAL LEADERSHIP IN HPC

SC12
Salt Lake City, Utah

MAGMA: Matrix Algebra on GPU and Multicore Architectures

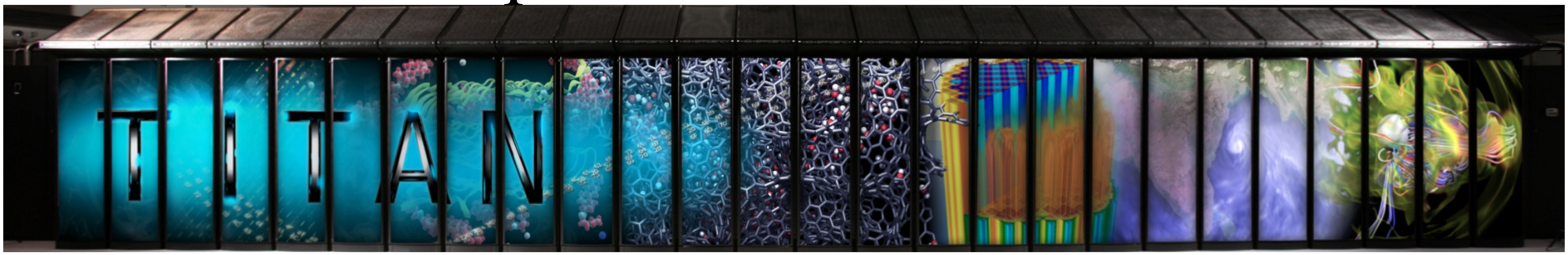
Jack Dongarra

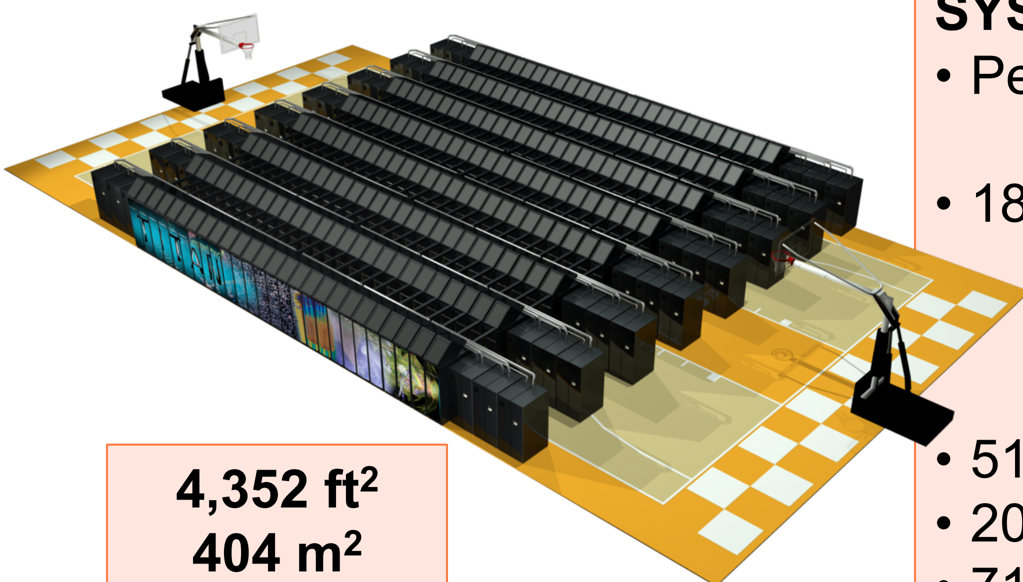
University of Tennessee, Knoxville
Oak Ridge National Laboratory

The MathWorks

NVIDIA.

ORNL's "Titan" Hybrid System: Cray XK7 with AMD Opteron and NVIDIA Tesla processors





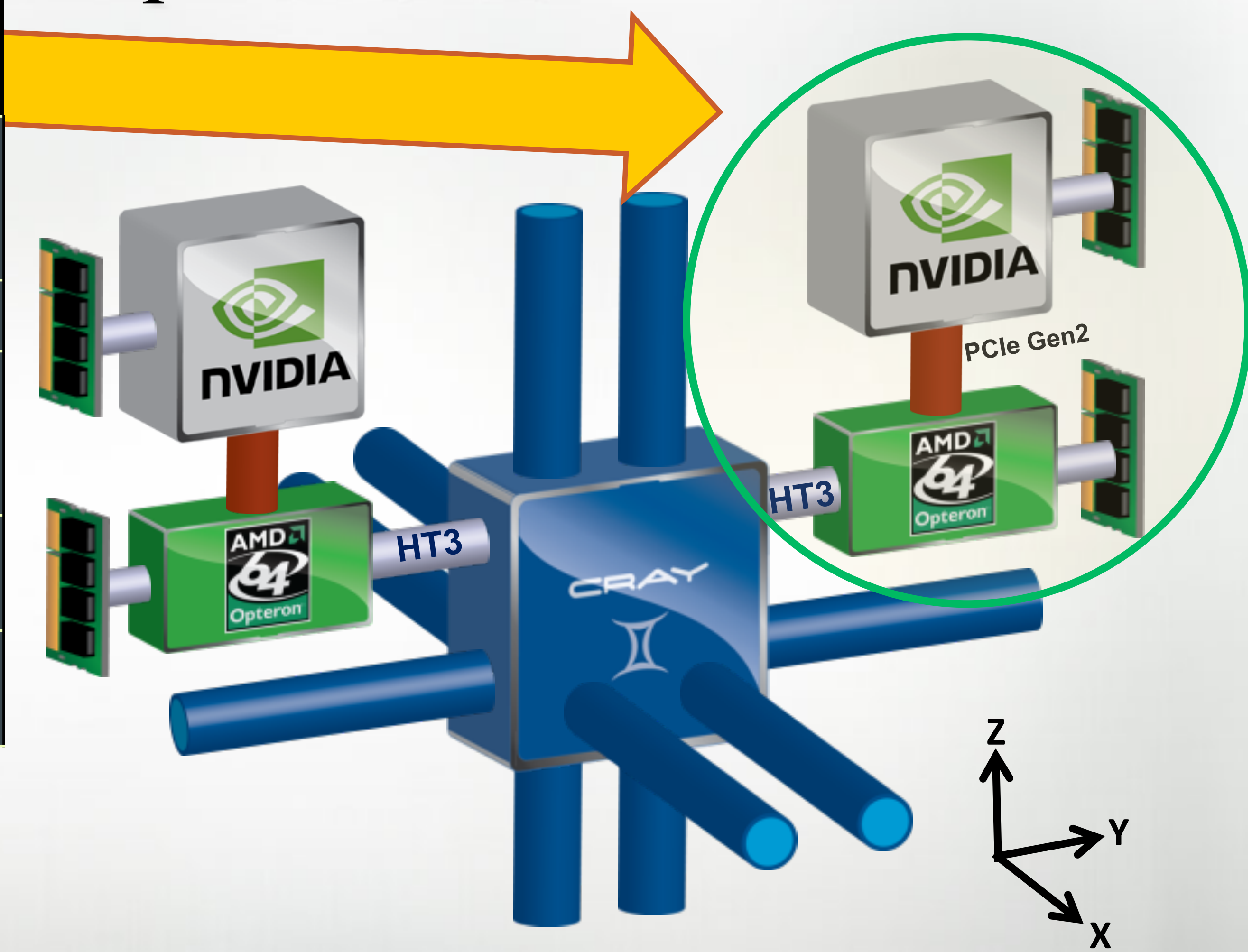
4,352 ft^2
404 m^2

SYSTEM SPECIFICATIONS:

- Peak performance of 27 PF
  - 24.5 Pflop/s GPU + 2.6 Pflop/s AMD
- 18,688 Compute Nodes each with:
  - 16-Core AMD Opteron CPU
  - NVIDIA Tesla "K20x" GPU
  - 32 + 6 GB memory
- 512 Service and I/O nodes
- 200 Cabinets
- 710 TB total system memory
- Cray Gemini 3D Torus Interconnect
- 9 MW peak power

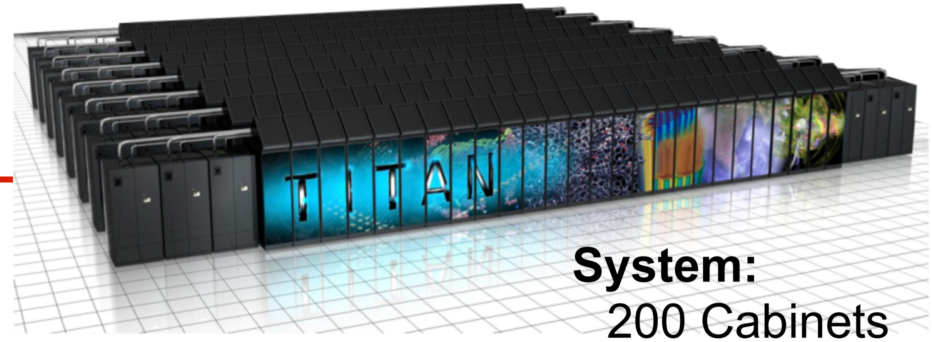
Cray XK7 Compute Node

XK7 Compute Node Characteristics
AMD Opteron 6274 Interlagos 16 core processor
Tesla K20x @ 1311 GF
Host Memory 32GB 1600 MHz DDR3
Tesla K20x Memory 6GB GDDR5
Gemini High Speed Interconnect



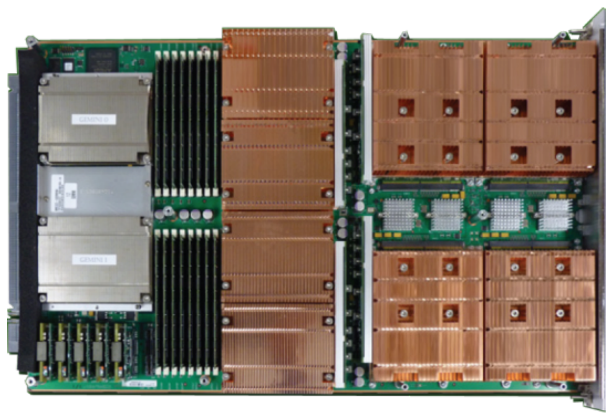
Slide courtesy of Cray, Inc.

Titan:
Cray XK7 System

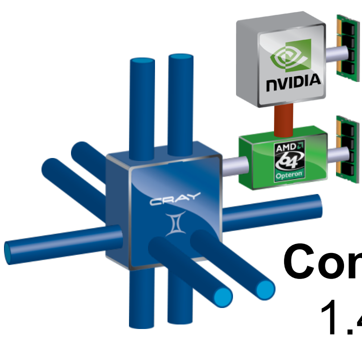


System:
200 Cabinets
18,688 Nodes
27 PF
710 TB

Cabinet:
24 Boards
96 Nodes
139 TF
3.6 TB



Board:
4 Compute Nodes
5.8 TF
152 GB

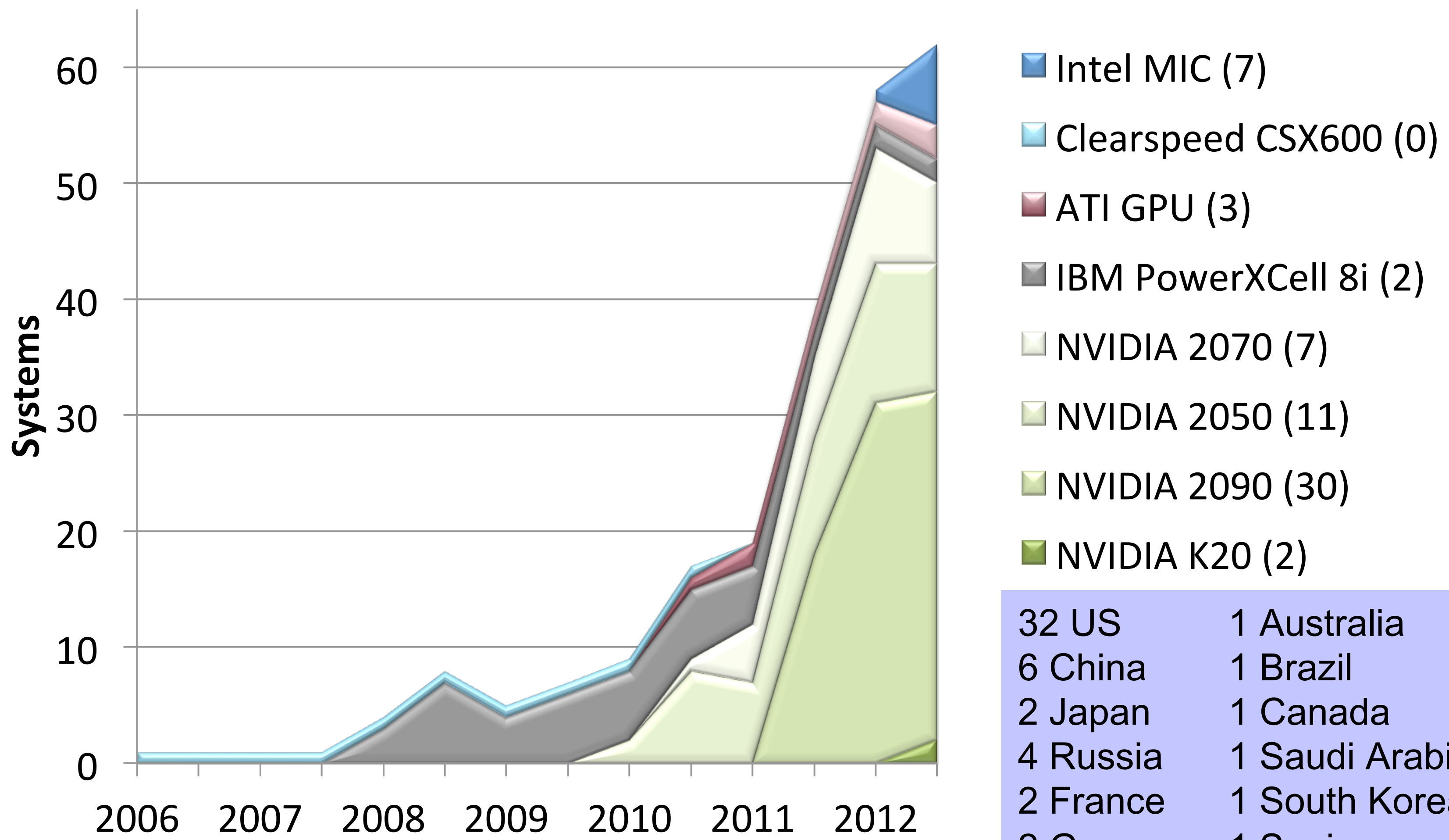


Compute Node:
1.45 TF
38 GB

November 2012: The TOP10

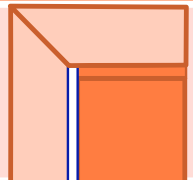
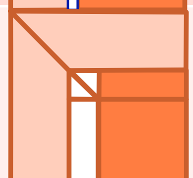
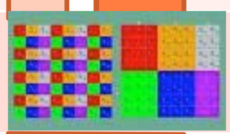
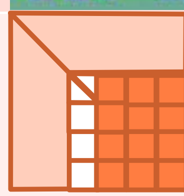
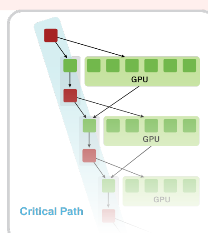
Rank	Site	Computer	Country	Cores	Rmax [Pflops]	% of Peak	Power [MW]	MFlops /Watt
1	DOE / OS Oak Ridge Nat Lab	Titan, Cray XK7 (16C) + Nvidia Kepler GPU (14c) + custom	USA	560,640	17.6	66	8.3	2120
2	DOE / NNSA L Livermore Nat Lab	Sequoia, BlueGene/Q (16c) + custom	USA	1,572,864	16.3	81	7.9	2063
3	RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx (8c) + custom	Japan	705,024	10.5	93	12.7	827
4	DOE / OS Argonne Nat Lab	Mira, BlueGene/Q (16c) + custom	USA	786,432	8.16	81	3.95	2066
5	Forschungszentrum Juelich	JuQUEEN, BlueGene/Q (16c) + custom	Germany	393,216	4.14	82	1.97	2102
6	Leibniz Rechenzentrum	SuperMUC, Intel (8c) + IB	Germany	147,456	2.90	90*	3.42	848
7	Texas Advanced Computing Center	Stampede, Dell Intel (8) + Intel Xeon Phi (61) + IB	USA	204,900	2.66	67	3.3	806
8	Nat. SuperComputer Center in Tianjin	Tianhe-1A, NUDT Intel (6c) + Nvidia Fermi GPU (14c) + custom	China	186,368	2.57	55	4.04	636
9	CINECA	Fermi, BlueGene/Q (16c) + custom	Italy	163,840	1.73	82	.822	2105
10	IBM	DARPA Trial System, Power7 (8C) + custom	USA	63,360	1.51	78	.358	422
500	Slovak Academy Sci	IBM Power 7	Slovak Rep	3,074	.077	81		

Accelerators (62 systems)



32 US	1 Australia
6 China	1 Brazil
2 Japan	1 Canada
4 Russia	1 Saudi Arabia
2 France	1 South Korea
2 Germany	1 Spain
1 India	1 Switzerland
2 Italy	1 Taiwan
2 Poland	1 UK

A New Generation of DLA Software

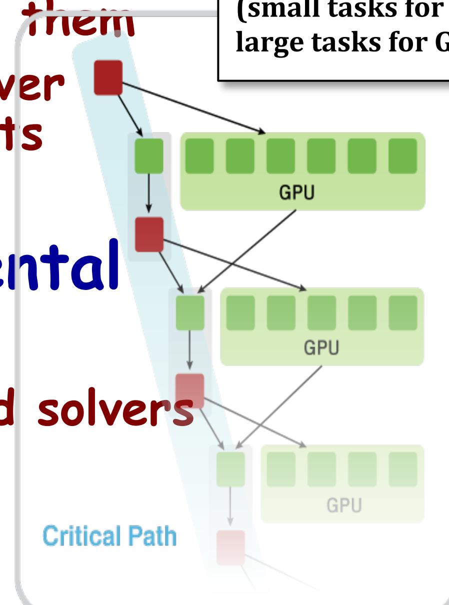
Software/Algorithms follow hardware evolution in time		
LINPACK (70's) (Vector operations)		Rely on - Level-1 BLAS operations
LAPACK (80's) (Blocking, cache friendly)		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) (Distributed Memory)		Rely on - PBLAS Mess Passing
PLASMA (00's) New Algorithms (many-core friendly)		Rely on - a DAG/scheduler - block data layout - some extra kernels
MAGMA Hybrid Algorithms (heterogeneity friendly)		Rely on - hybrid scheduler - hybrid kernels

Methodology overview

A methodology to use all available resources:

- MAGMA uses hybridization methodology based on
  - Representing linear algebra algorithms as collections of tasks and data dependencies among them
  - Properly scheduling tasks' execution over multicore and GPU hardware components
- Successfully applied to fundamental linear algebra algorithms
  - One- and two-sided factorizations and solvers
  - Iterative linear and eigensolvers
- Productivity
  - 1) High level; 2) Leveraging prior developments; 3) Exceeding in performance homogeneous solutions

Hybrid CPU+GPU algorithms (small tasks for multicores and large tasks for GPUs)

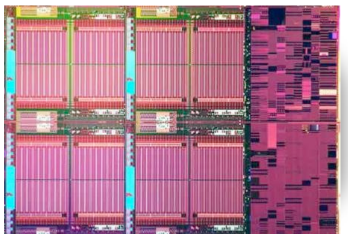


Commodity plus Accelerator Today

32 CUDA Cores/SMX

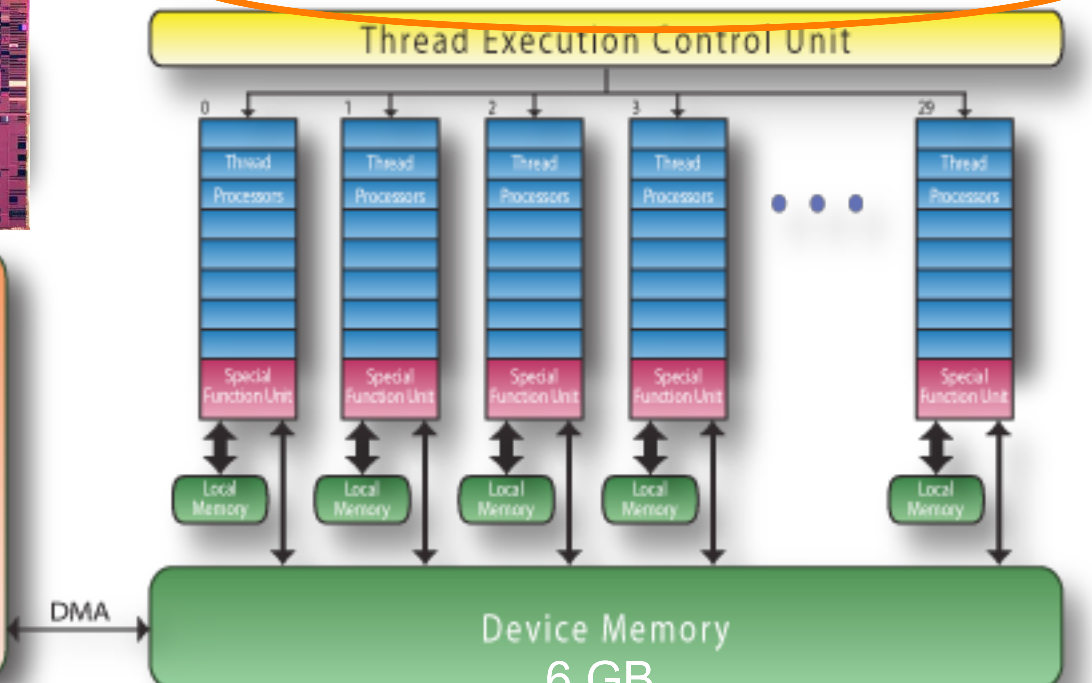
Commodity

Intel Xeon
8 cores
3 GHz
8*4 ops/cycle
96 Gflop/s (DP)



Accelerator (GPU)

Nvidia M2090 “Fermi”
512 “Cuda cores”
1.3 GHz
512 ops/cycle
665 Gflop/s (DP) or 1302 Gflop/s (SP)



Interconnect
PCI-X 16 lane
64 Gb/s (8 GB/s)

Commodity plus Accelerator Today

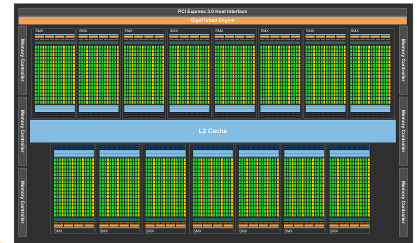
Commodity

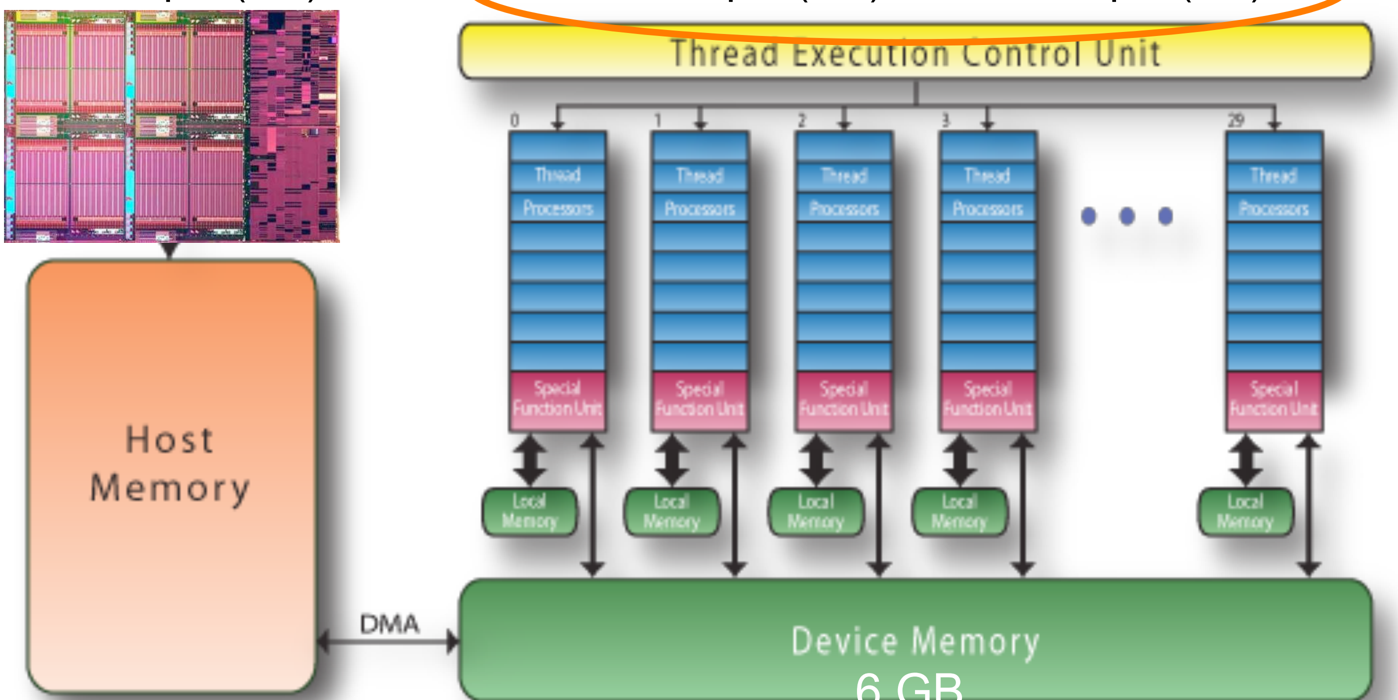
Intel Xeon
8 cores
3 GHz
8*4 ops/cycle
96 Gflop/s (DP)

Accelerator (GPU)

Nvidia K20X “Kepler”
2688 “Cuda cores”
.732 GHz
2688*2/3 ops/cycle
1.31 Tflop/s (DP) or 3.62 Tflop/s (SP)

192 Cuda cores/SMX





Interconnect
PCI-X 16 lane
64 Gb/s (8 GB/s)

MAGMA: LAPACK for GPUs

- **MAGMA**
  - Matrix algebra for GPU and multicore architecture
  - To provide LAPACK/ScaLAPACK on hybrid architectures
  - http://icl.cs.utk.edu/magma/
- **MAGMA 1.3**
  - For NVIDIA CUDA GPUs on shared memory systems
  - 80+ hybrid algorithms have been developed (total of 320+ routines)
    - One-sided factorizations and linear system solvers
    - Two-sided factorizations and eigenproblem solvers
    - A subset of BLAS and auxiliary routines in CUDA
- **MAGMA developers & collaborators**
  - UTK, UC Berkeley, UC Denver, INRIA (France), KAUST (Saudi Arabia)
  - Community effort, similar to LAPACK/ScaLAPACK

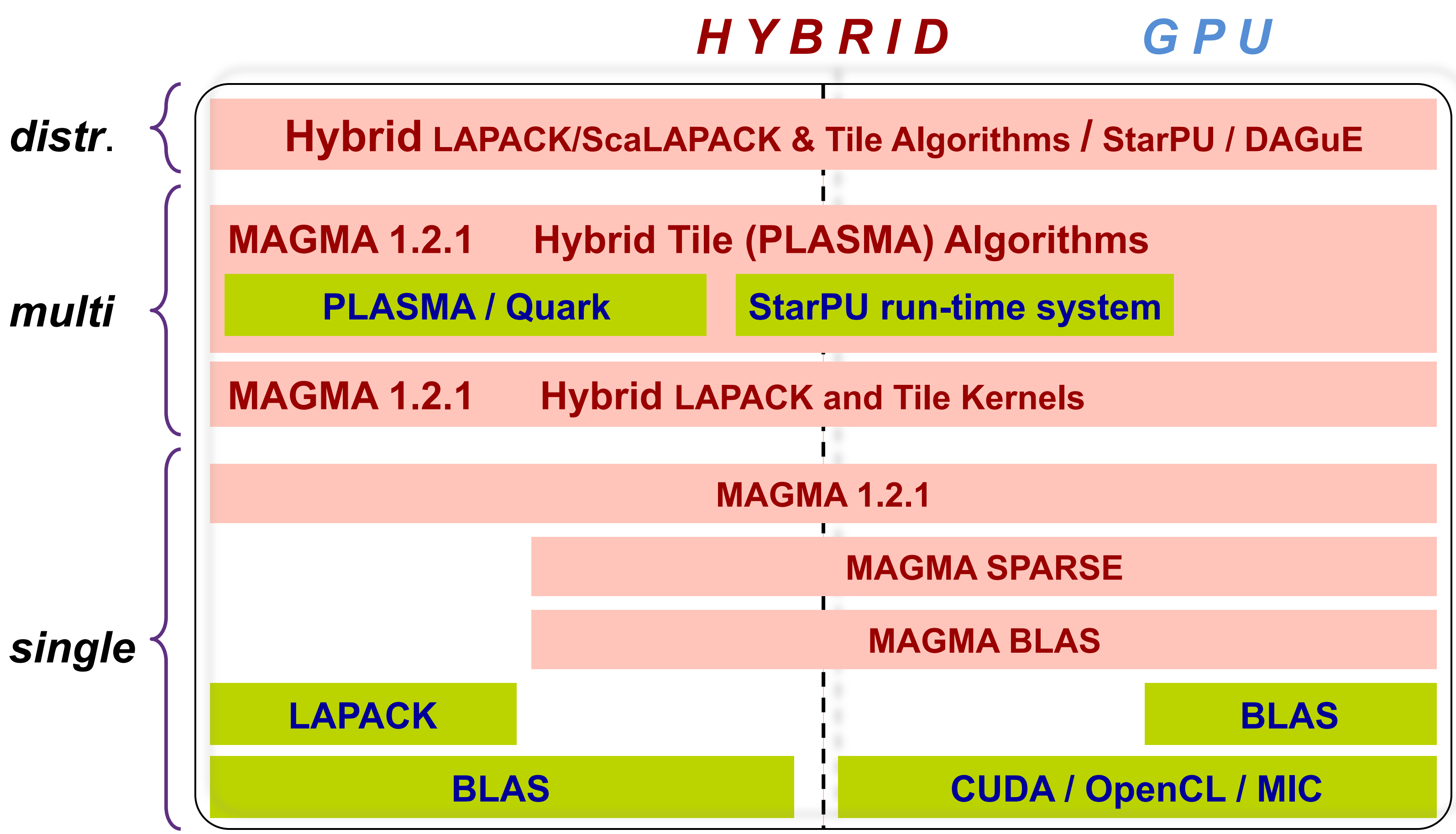
MAGMA Functionality

- **80+ hybrid algorithms** have been developed (total of 320+ routines)
  - Every algorithm is in 4 precisions (s/c/d/z)
  - There are 3 mixed precision algorithms (zc & ds)
  - These are hybrid algorithms, expressed in terms of BLAS
- **MAGMA BLAS**
  - A subset of GPU BLAS, optimized for Tesla and Fermi GPUs

MAGMA 1.1 ROUTINES & FUNCTIONALITIES	SINGLE GPU	MULTI-GPU STATIC	MULTI-GPU DYNAMIC
One-sided Factorizations (LU, QR, Cholesky)	✓	✓	✓
Linear System Solvers	✓		✓
Linear Least Squares (LLS) Solvers	✓		✓
Matrix Inversion	✓		✓
Singular Value Problem (SVP)	✓		
Non-symmetric Eigenvalue Problem	✓		
Symmetric Eigenvalue Problem	✓		
Generalized Symmetric Eigenvalue Problem	✓		

SINGLE GPU	Hybrid LAPACK algorithms with static scheduling and LAPACK data layout
MULTI-GPU STATIC	Hybrid LAPACK algorithms with 1D block cyclic static scheduling and LAPACK data layout
MULTI-GPU DYNAMIC	Tile algorithms with StarPU scheduling and tile matrix layout

MAGMA Software Stack

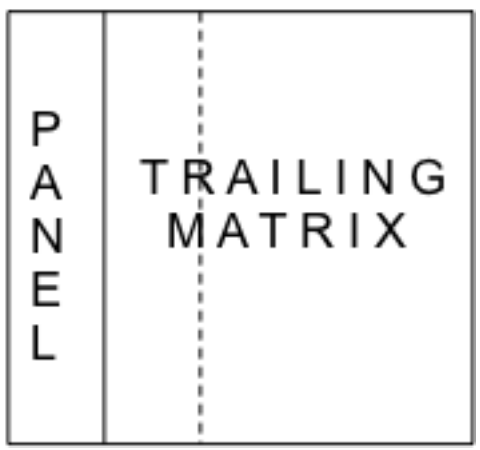


Linux, Windows, Mac OS X | C/C++, Fortran | Matlab, Python

Hybrid Algorithms

One-Sided Factorizations (LU, QR, and Cholesky)

- **Hybridization**
  - **Panels (Level 2 BLAS) are factored on CPU using LAPACK**
  - **Trailing matrix updates (Level 3 BLAS) are done on the GPU using "look-ahead"**



A Hybrid Algorithm Example

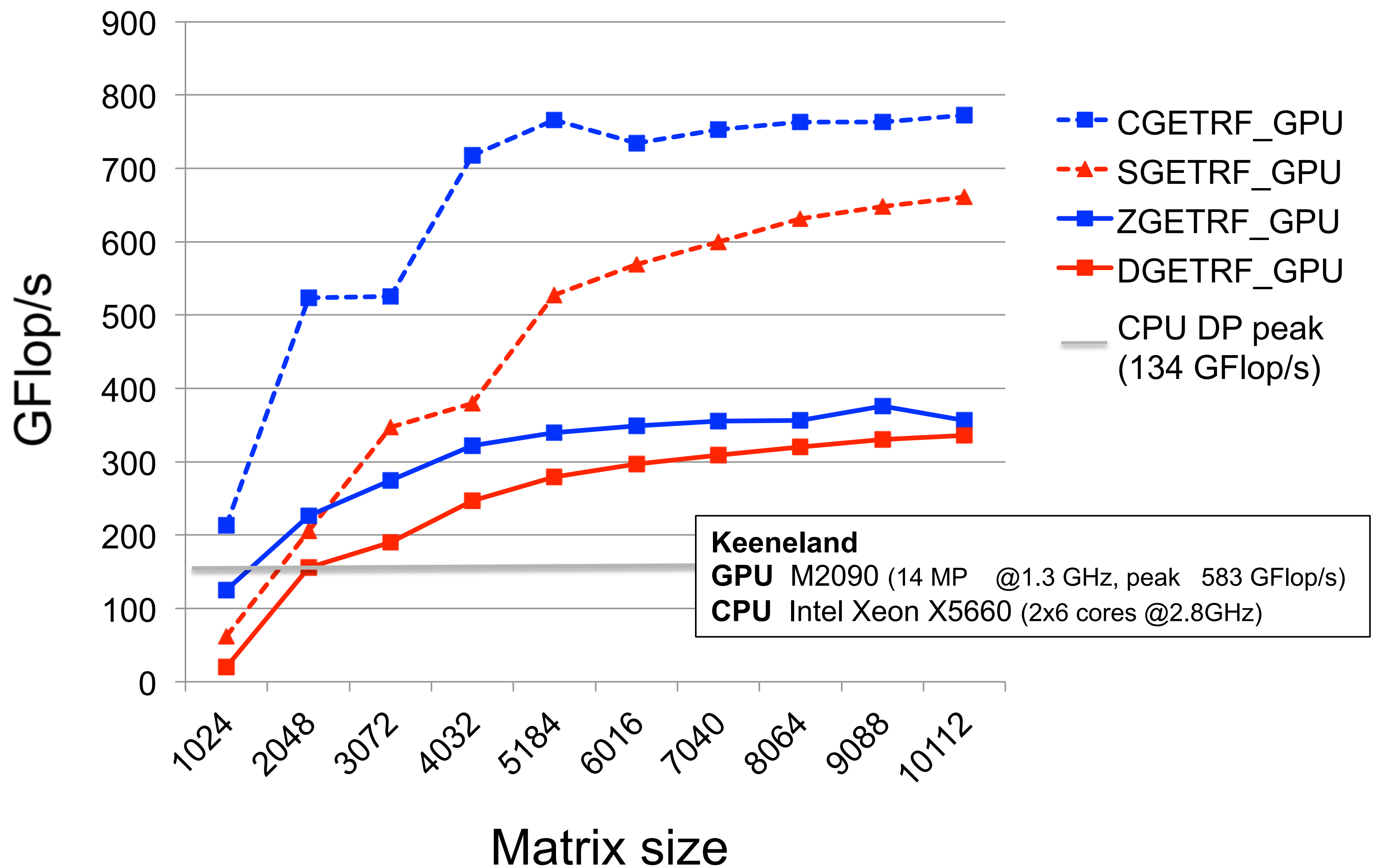
- Left-looking hybrid Cholesky factorization in MAGMA

```
1   for ( j=0; j<n; j += nb) {
2       jb = min(nb, n – j);
3       magma_zherk( MagmaUpper, MagmaConjTrans,
                jb, j, m_one, dA(0, j), ldda, one, dA(j, j), ldda, queue );
4       magma_zgetmatrix_async( jb, jb, dA(j,j), ldda, work, 0, jb, queue, &event );
5       if ( j+jb < n )
6           magma_zgemm( MagmaConjTrans, MagmaNoTrans, jb, n-j-jb, j, mz_one,
                    dA(0, j ), ldda, dA(0, j+jb), ldda, z_one,  dA(j, j+jb), ldda, queue );
7       magma_event_sync( event );
8       lapackf77_zpotrf( MagmaUpperStr, &jb, work, &jb, info );
9       if ( *info != 0 )
10          *info += j;
11      magma_zsetmatrix_async( jb, jb, work, 0, jb, dA(j,j), ldda, queue, &event );
12      if ( j+jb < n ) {
13          magma_event_sync( event );
14          magma_ztrsm( MagmaLeft, MagmaUpper, MagmaConjTrans, MagmaNonUnit,
                    jb, n-j-jb, z_one, dA(j, j), ldda, dA(j, j+jb), ldda, queue );
        }
    }
```

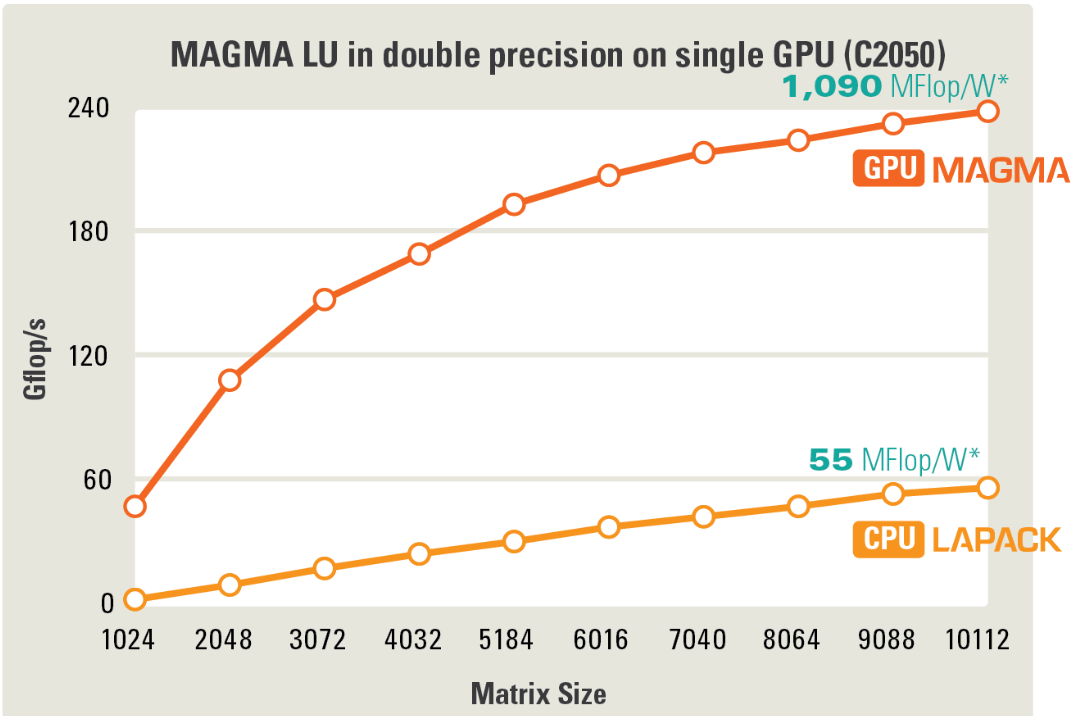
- The difference with LAPACK – the 4 additional lines in red
- Line 8 (done on CPU) is overlapped with work on the GPU (from line 6)

Multiple precision support

Performance of the LU factorization in various precisions



LU Factorization (single GPU)



GPU Fermi C2050 (448 CUDA Cores @ 1.15 GHz)
+ Intel Q9300 (4 cores @ 2.50 GHz)
DP peak **515** + **40** GFlop/s
Power * ~**220** W

CPU AMD Istanbul
[8 sockets x 6 cores (48 cores) @2.8GHz]
DP peak **538** GFlop/s
Power * ~**1,022** W

* Computation consumed power rate (total system rate minus idle rate), measured with KILL A WATT PS, Model P430

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.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} - x_i = -\frac{f(x_i)}{f'(x_i)}$$

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.

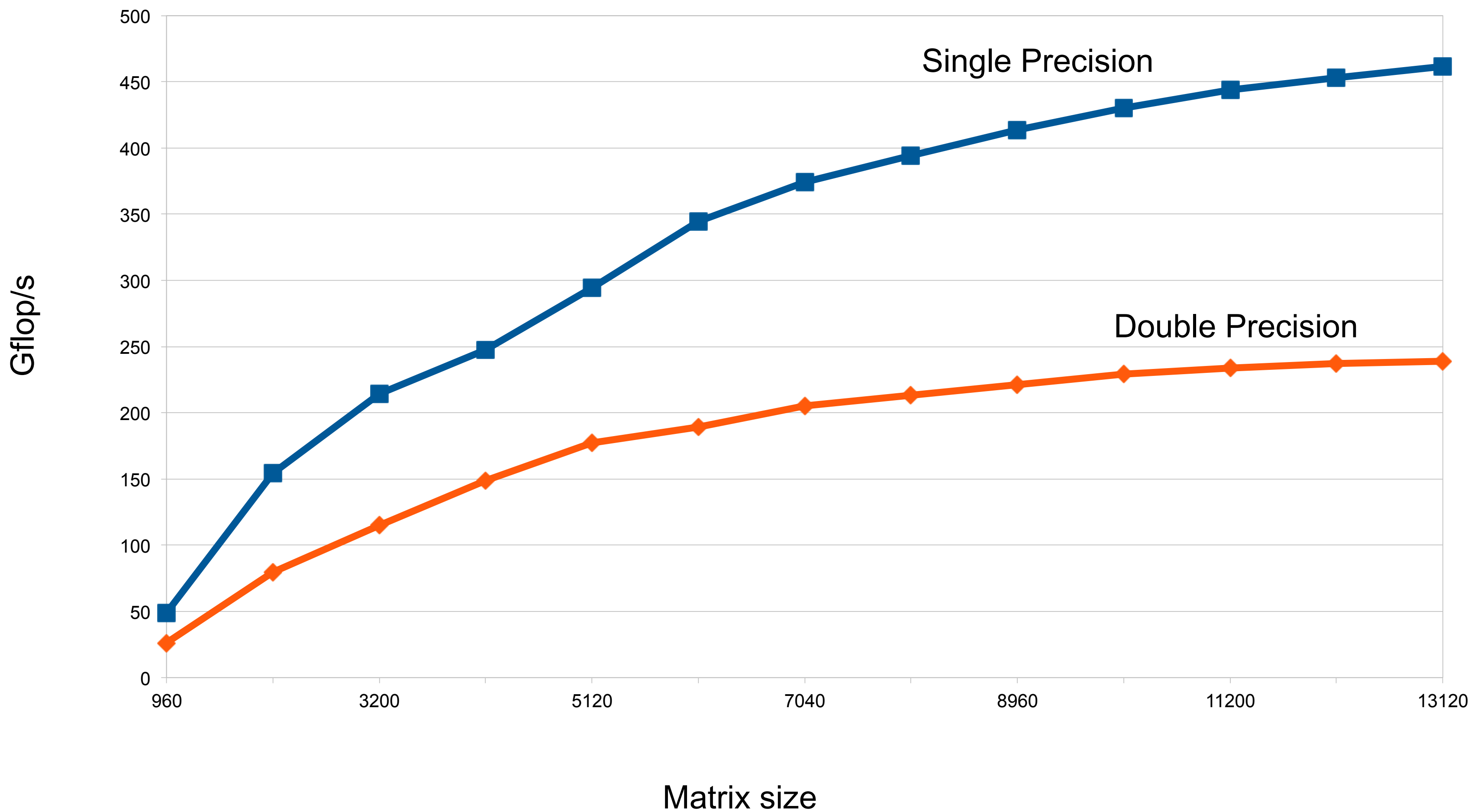
```
L U = lu(A)                    SINGLE    O(n^3)
x = L\(U\b)                    SINGLE    O(n^2)
r = b – Ax                     DOUBLE    O(n^2)
WHILE || r || not small enough
      z = L\(U\r)              SINGLE    O(n^2)
      x = x + z                DOUBLE    O(n^1)
      r = b – Ax         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

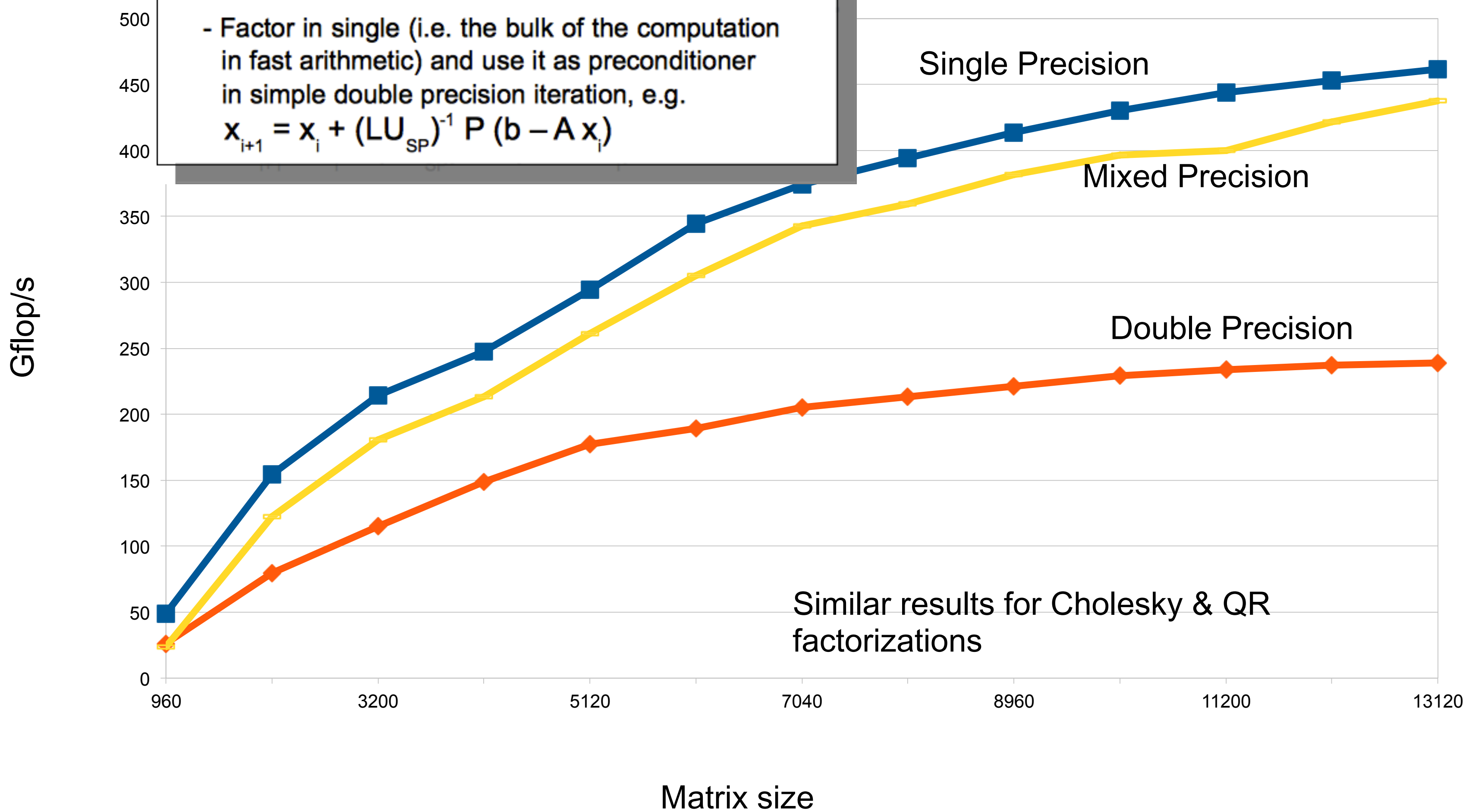
FERMI Tesla C2050: 448 CUDA cores @ 1.15GHz
SP/DP peak is 1030 / 515 GFlop/s



Ax = b

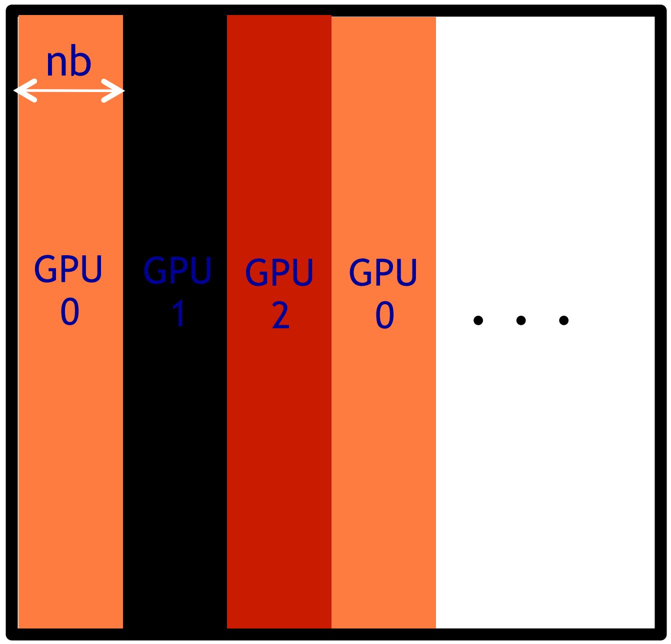
FERMI Tesla C2050: 448 CUDA cores @ 1.15GHz
SP/DP peak is 1030 / 515 GFlop/s

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

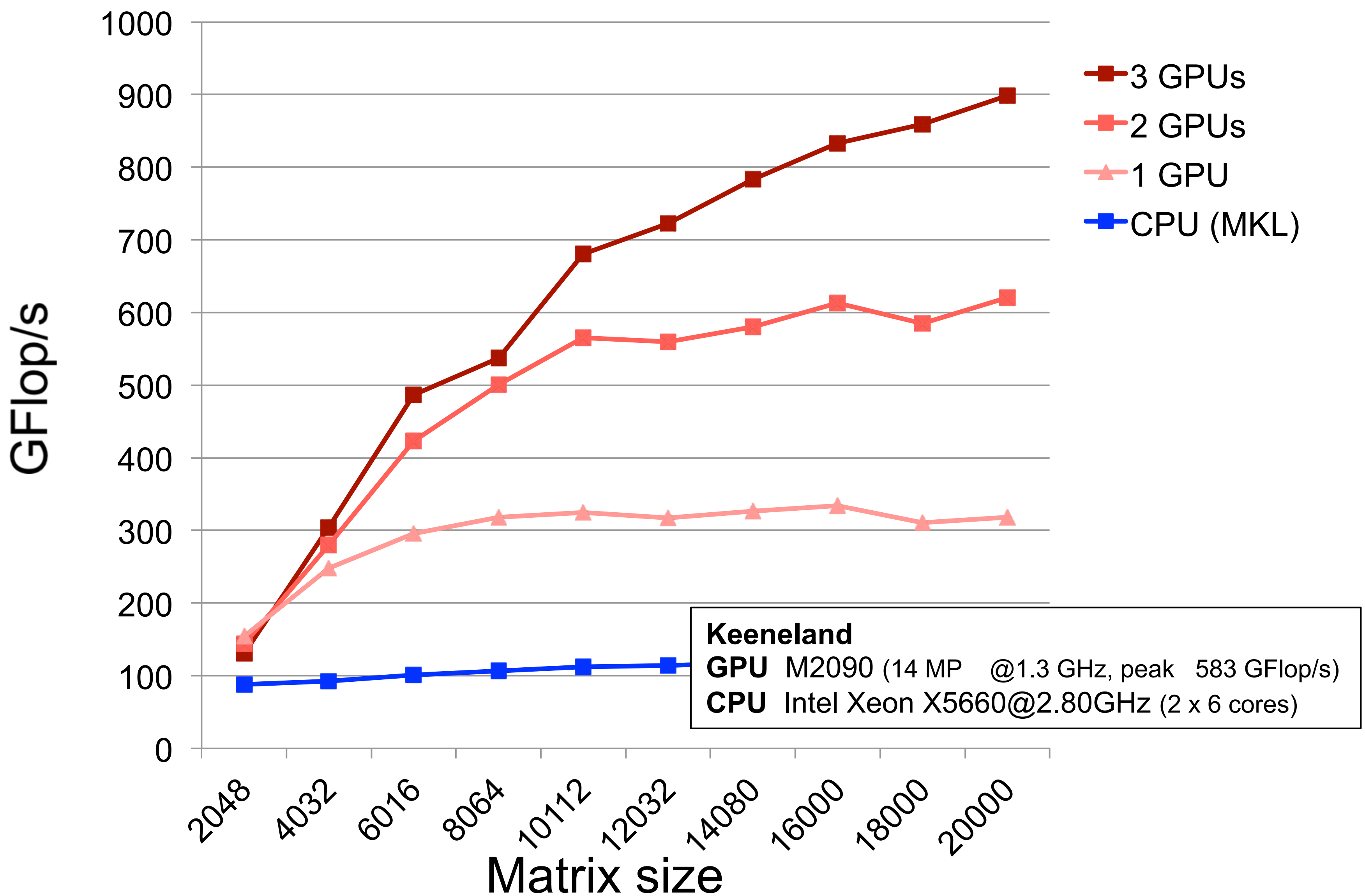


MultiGPU Support

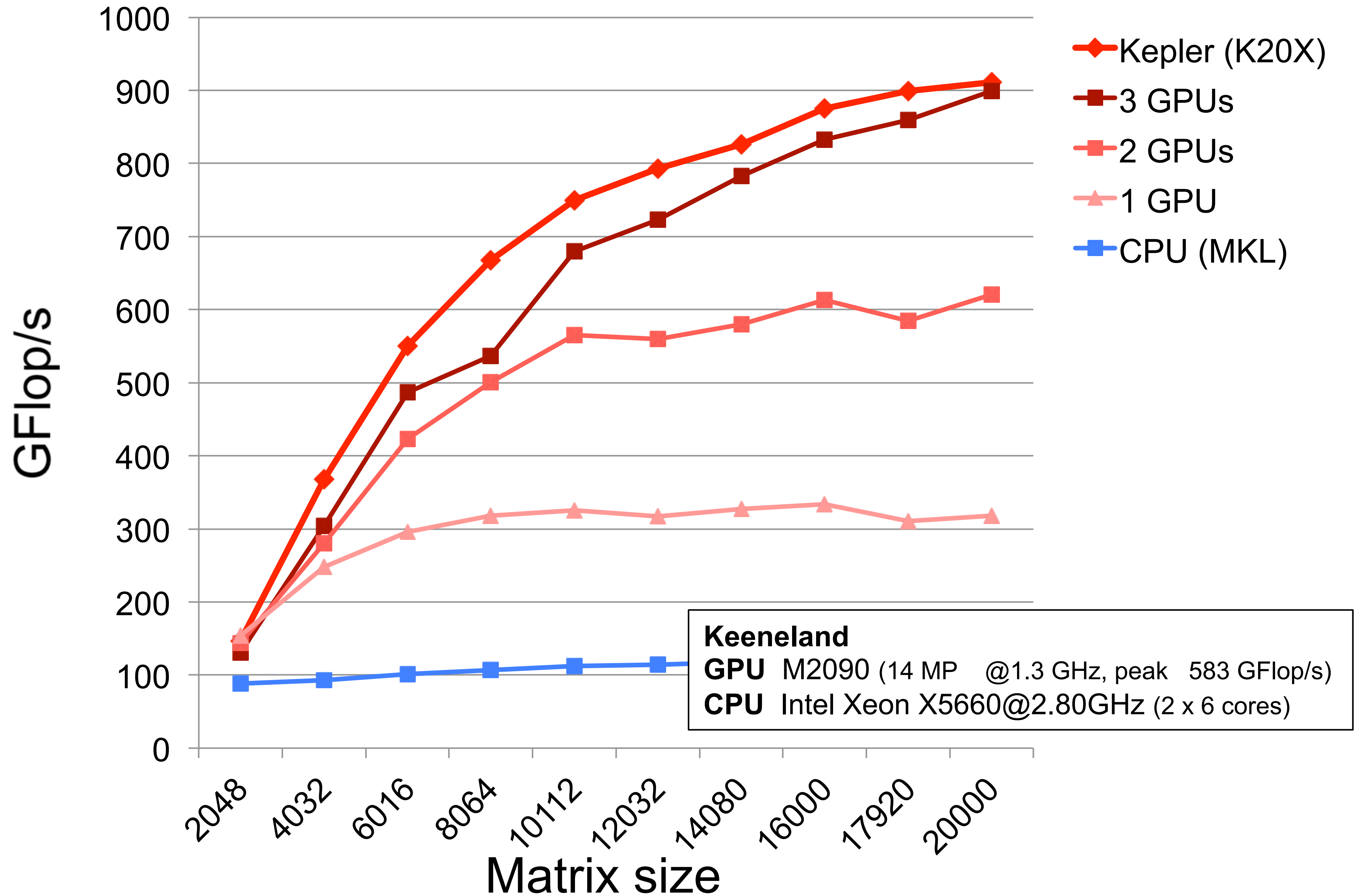
- **Data distribution**
  - 1-D block-cyclic distribution
- **Algorithm**
  - GPU holding current panel is sending it to CPU
  - All updates are done in parallel on the GPUs
  - Look-ahead is done with GPU holding the next panel



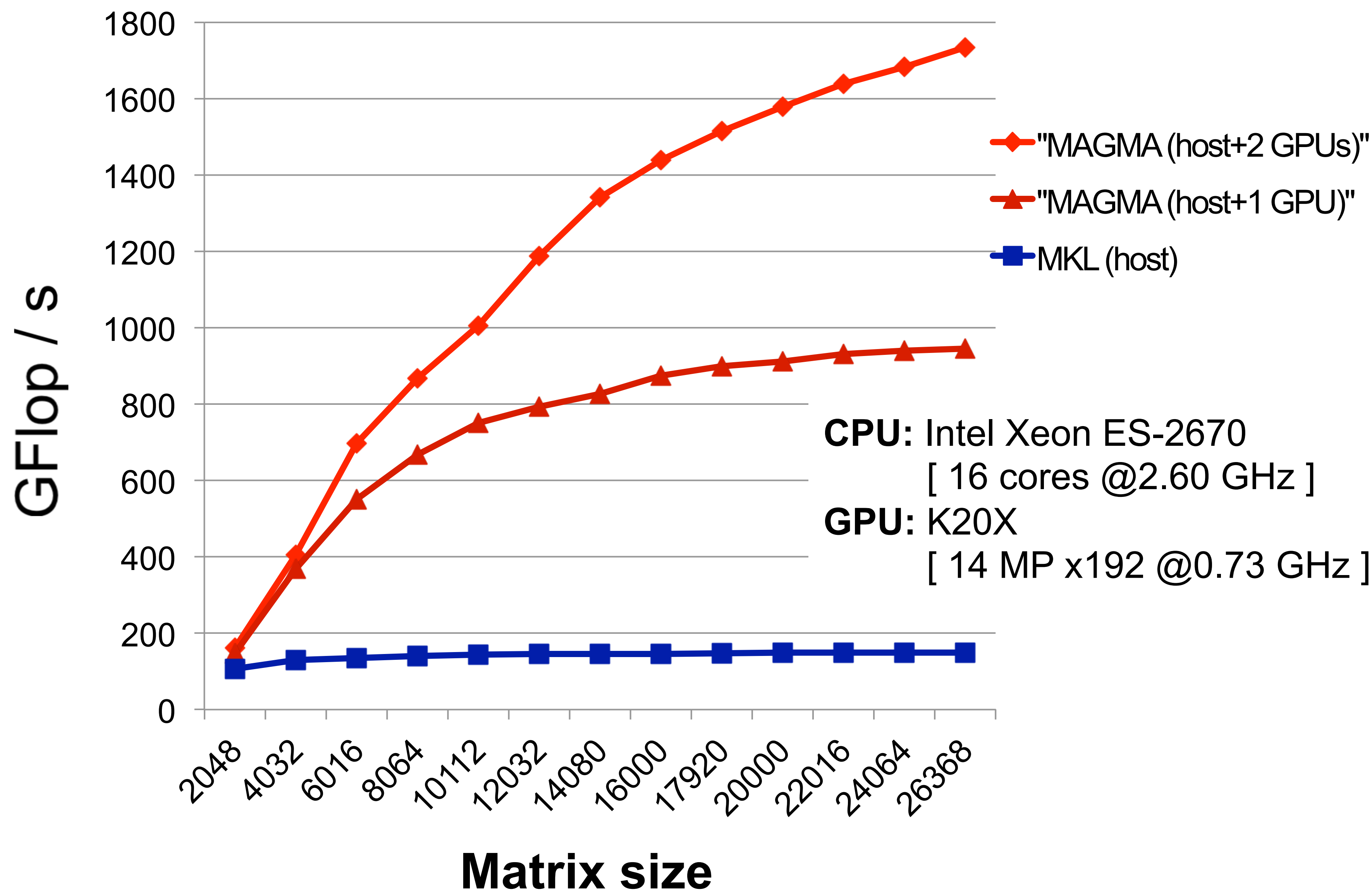
LU Factorization on multiGPUs in DP



LU Factorization on multiGPUs in DP



LU Factorization on two Keplers in DP



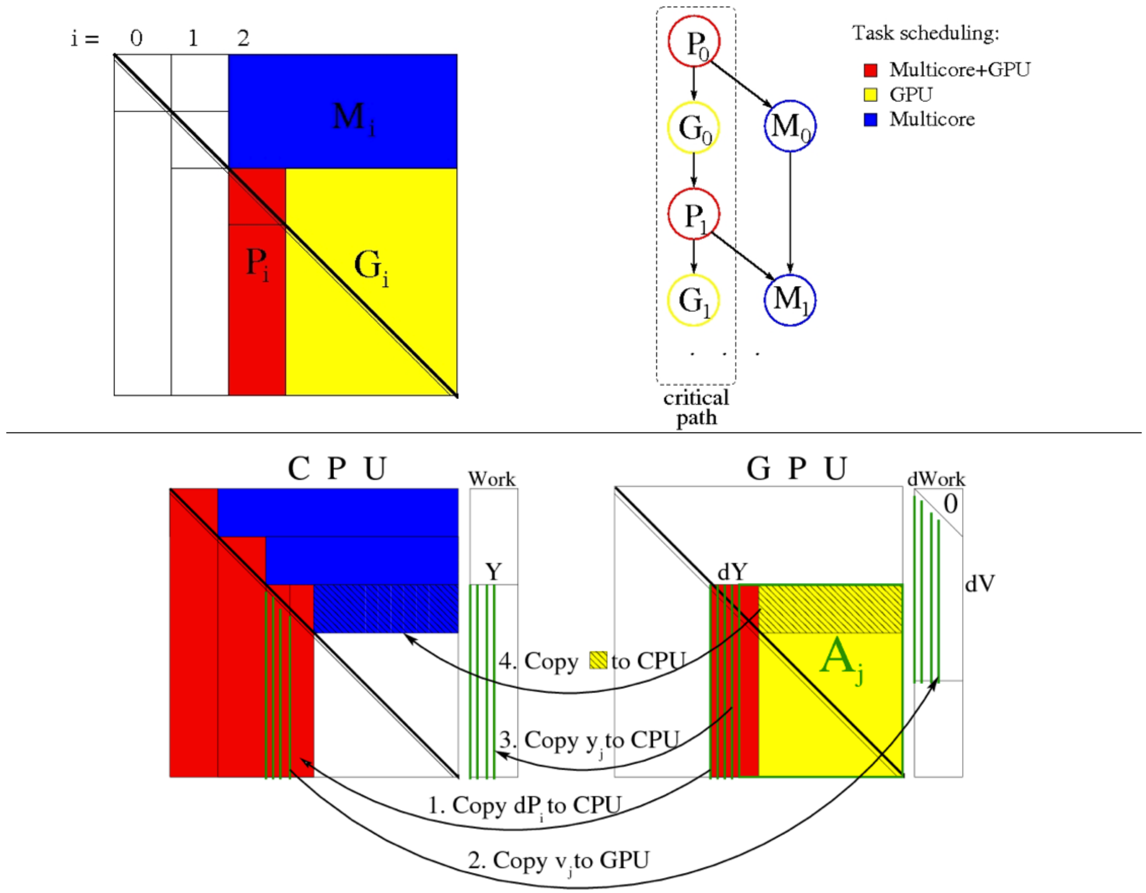
Hybrid Algorithms

Two-Sided Factorizations (to bidiagonal, tridiagonal, and upper Hessenberg forms) for eigen- and singular-value problems

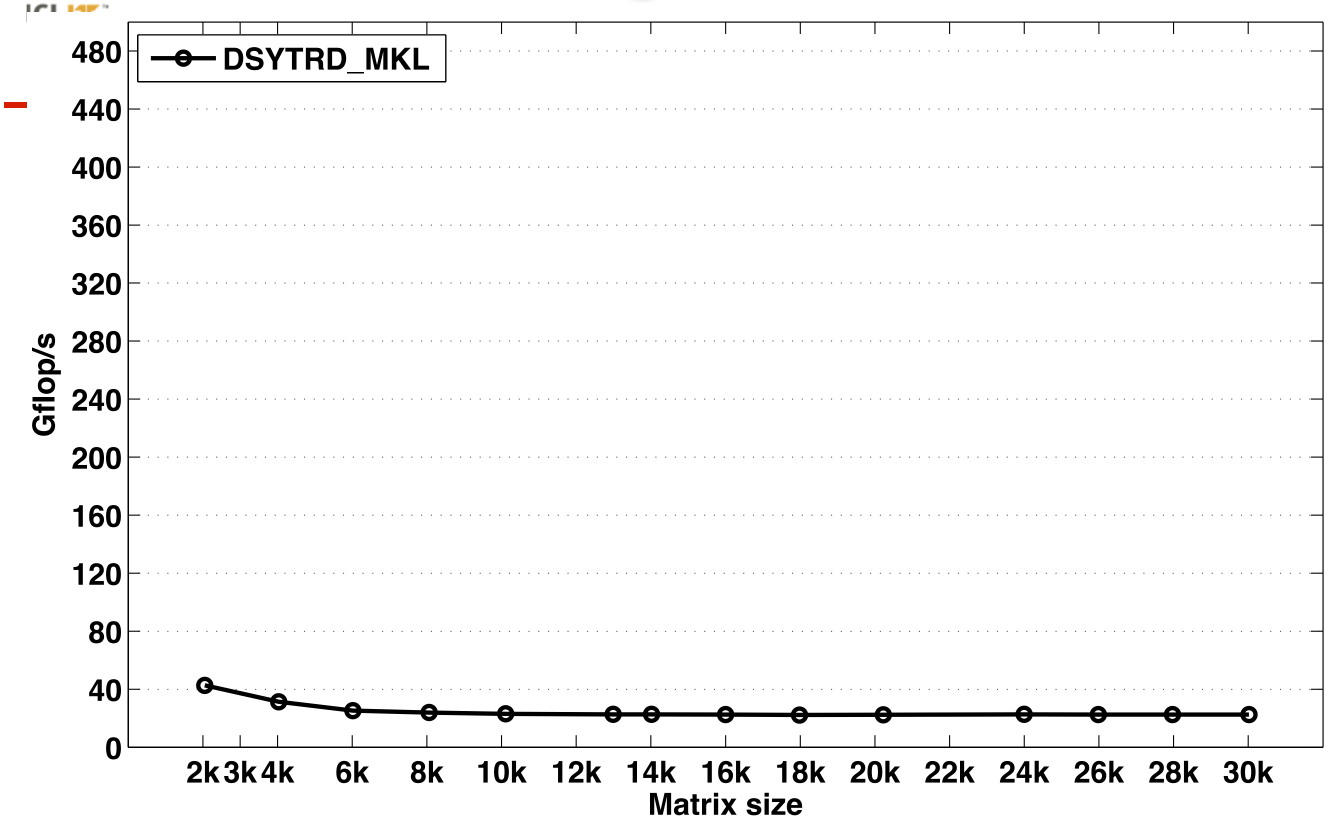
- **Hybridization**
- **Trailing matrix updates (Level 3 BLAS) are done on the GPU (similar to the one-sided factorizations)**
- **Panels (Level 2 BLAS) are hybrid**
- **Operations with memory footprint restricted to the panel are done on CPU**
- **The time consuming matrix-vector products involving the entire trailing matrix are done on the GPU**

Hybrid Two-Sided Factorizations

Task **Splitting** & Task **Scheduling**



Toward fast Eigensolver



flops formula: **$n^3/3$*time**

Higher is faster

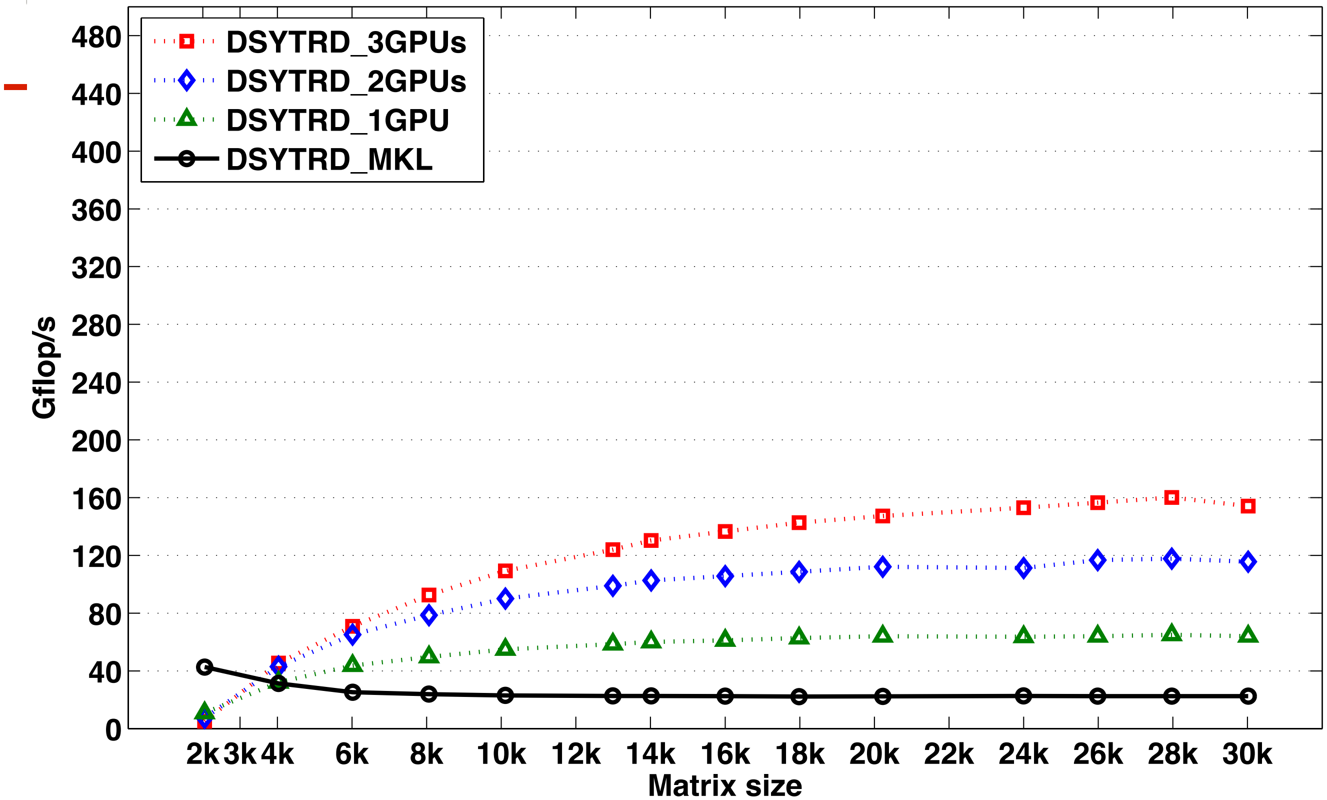
Keeneland system, using one node
3 NVIDIA GPUs (M2090@ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

Characteristics

- Too many Blas-2 op,
- Relies on panel factorization,
- ➔Bulk sync phases,
- ➔Memory bound algorithm.

A. Haidar, S. Tomov, J. Dongarra, T. Schulthess, and R. Solca, ***A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks*****, ICL Technical report, 03/2012.**

Toward fast Eigensolver



flops formula: **$n^3/3$*time**

Higher is faster

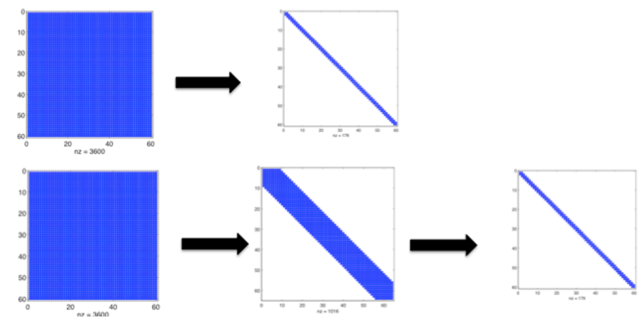
Keeneland system, using one node
3 NVIDIA GPUs (M2090@ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

Characteristics

- Blas-2 GEMV moved to the GPU,
- Accelerate the algorithm by doing all BLAS-3 on GPU,
- ➔Bulk sync phases,
- ➔Memory bound algorithm.

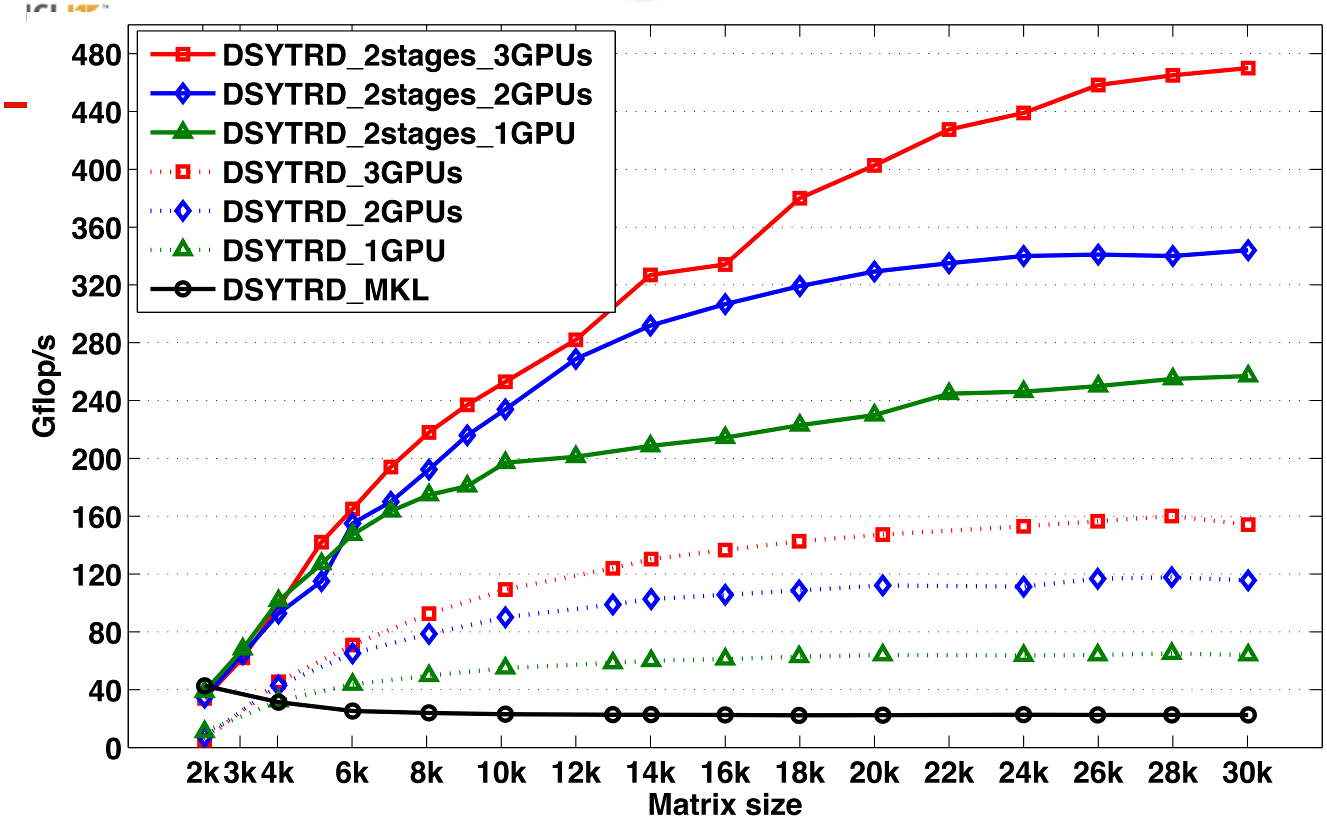
A. Haidar, S. Tomov, J. Dongarra, T. Schulthess, and R. Solca, *A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks*, ICL Technical report, 03/2012.

Two-Stage Approach to Tridiagonal Form (Communication Reducing)



- **Reduction to band**
  - On multicore + GPUs
  - Performance as in the one-sided factorizations [derived from fast Level 3 BLAS]
- **Band to tridiagonal**
  - Leads to “irregular” (bulge chasing) computation
    - Done very efficiently on multicore !
  - GPUs are used to assemble the orthogonal Q from the transformations [needed to find the eigenvectors]

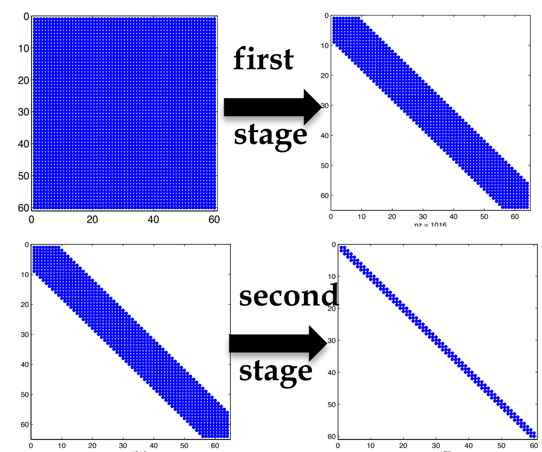
Toward fast Eigensolver



flops formula: **n³/3*time**

Higher is faster

Keeneland system, using one node
3 NVIDIA GPUs (M2090@ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

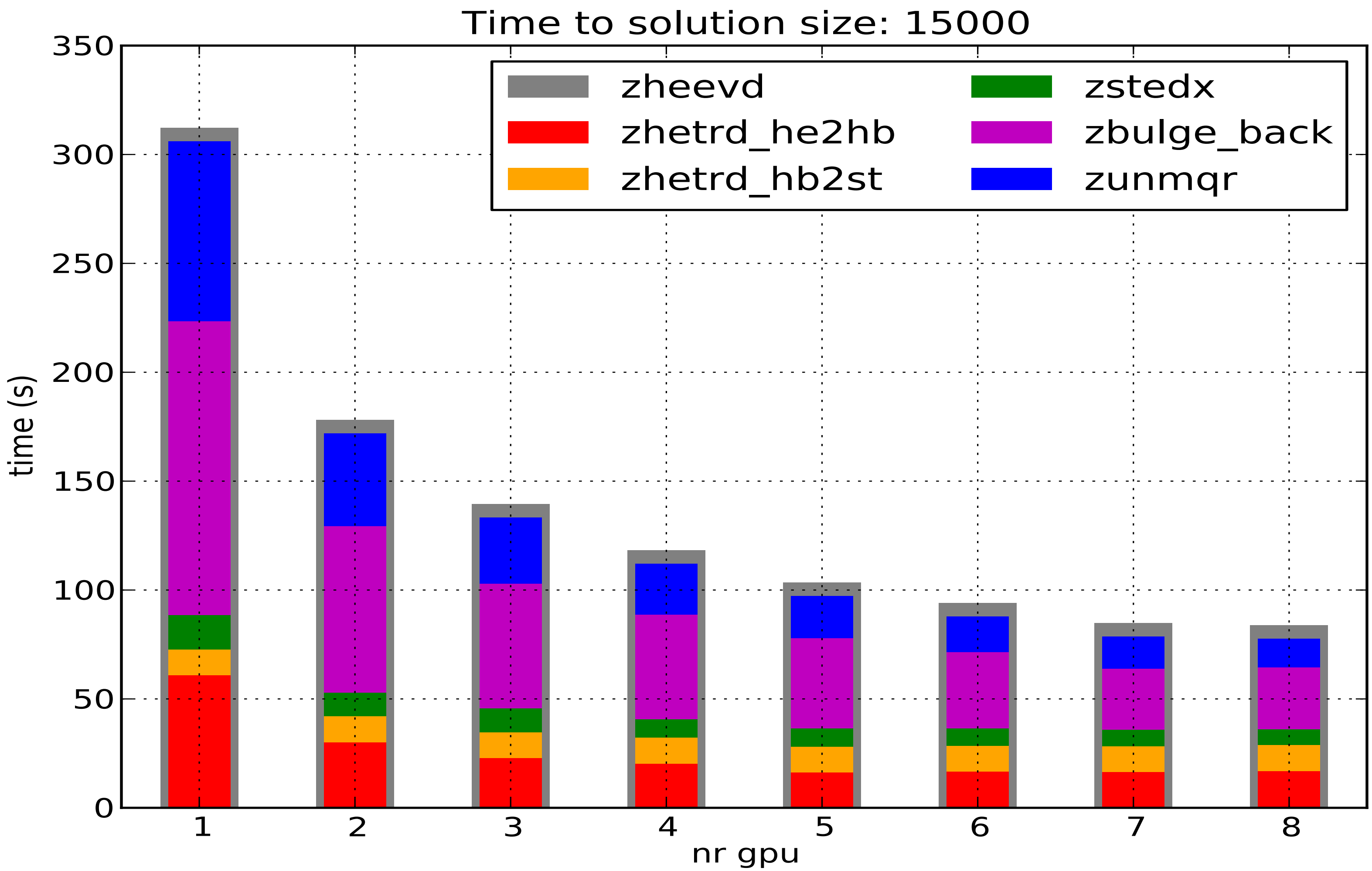


Characteristics

- Stage 1: BLAS-3, increasing computational intensity,
- Stage 2: BLAS-1.5, new cache friendly kernel,
- 4X/12X faster than standard approach,
- Bottelneck: if all Eigenvectors are required, it has 1 back transformation extra cost.

A. Haidar, S. Tomov, J. Dongarra, T. Schulthess, and R. Solca, *A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks*, ICL Technical report, 03/2012.

Strong scaling full Eigensolver/Eigevectors



CSCS system, using one node
8 NVIDIA GPUs (M2090@ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

Contact

Stan Tomov
tomov@eecs.utk.edu
Innovative Computing Laboratory
University of Tennessee, Knoxville

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

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

Collaborating Partners
University of Tennessee, Knoxville
University of California, Berkeley
University of Colorado, Denver
INRIA, France (StarPU team)
KAUST, Saudi Arabia

MAGMA

PLASMA