Accelerating Computational Science Symposium 2012(ACSS 2012)

March 29-30, 2012

Not Your Father's Math Library
MAGMA for Dense Matrix Problems

Matrix Algebra on GPU and Multicore Architectures (MAGMA)
Parallel Linear Algebra for Scalable Multi-core Architectures (PLASMA)

Jack Dongarra & Stan Tomov
University of Tennessee
Oak Ridge National Laboratory

Major Changes to Algorithms/Software

- **Must rethink the design of our algorithms and software**
  - **Manycore and Hybrid architectures are disruptive technology**
    - **Similar to what happened with cluster computing and message passing**
  - **Rethink and rewrite the applications, algorithms, and software**
  - **Data movement is expensive**
  - **Flops are cheap**

Challenges for Software/Libraries

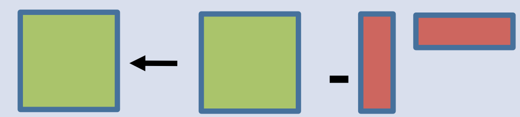
1. **Synchronization**
   - Break Fork-Join model
2. **Communication**
   - Use methods which have lower bound on communication
3. **Mixed precision methods**
   - SP:DP; 2x speed of ops and 2x speed for data movement
4. **Autotuning**
   - Today's machines are too complicated, build "smarts" into software to adapt to the hardware
5. **Fault resilient algorithms**
   - Implement algorithms that can recover from failures/bit flips

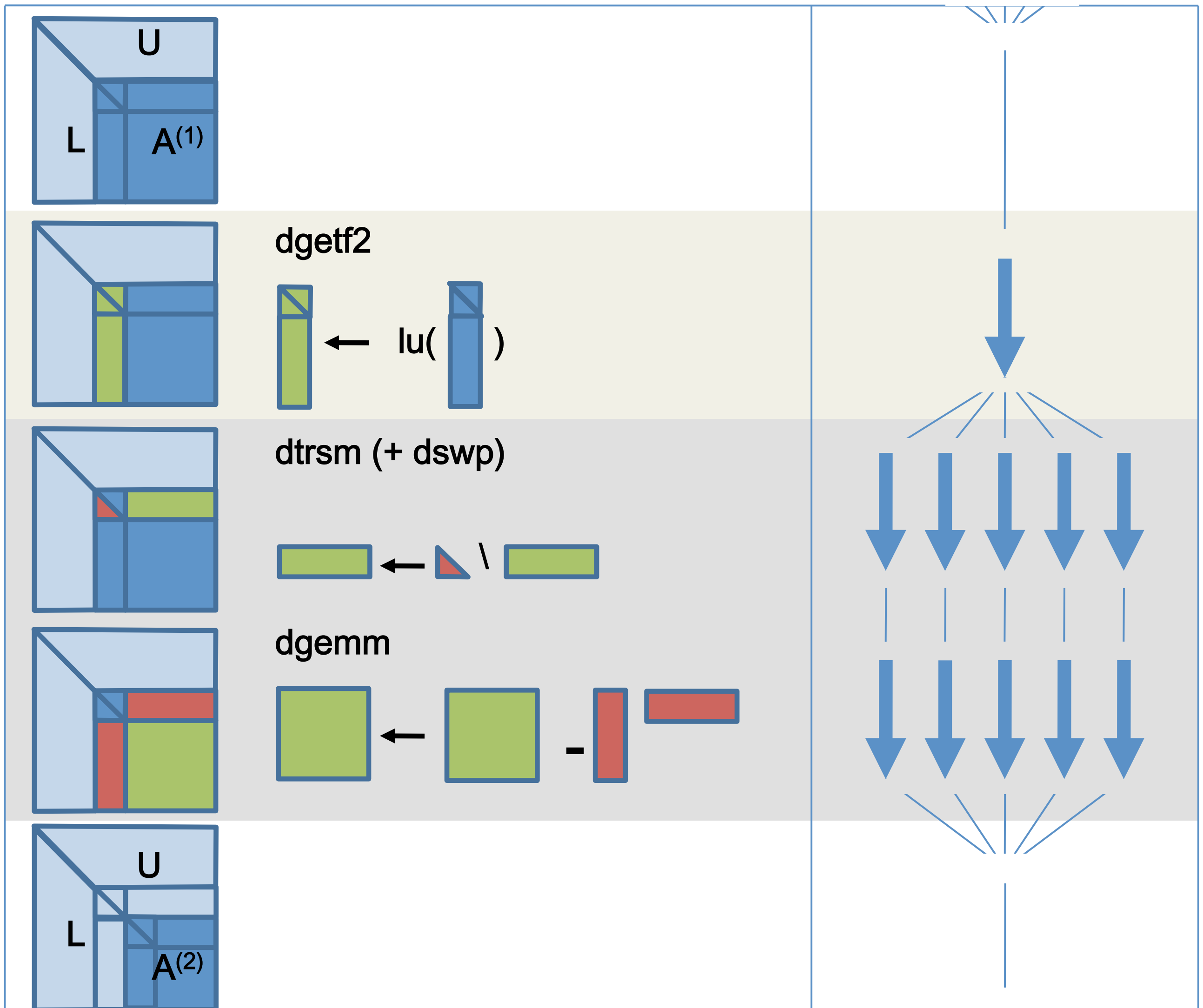
Fork-Join Parallelization of LU and QR.

Parallelize the update:

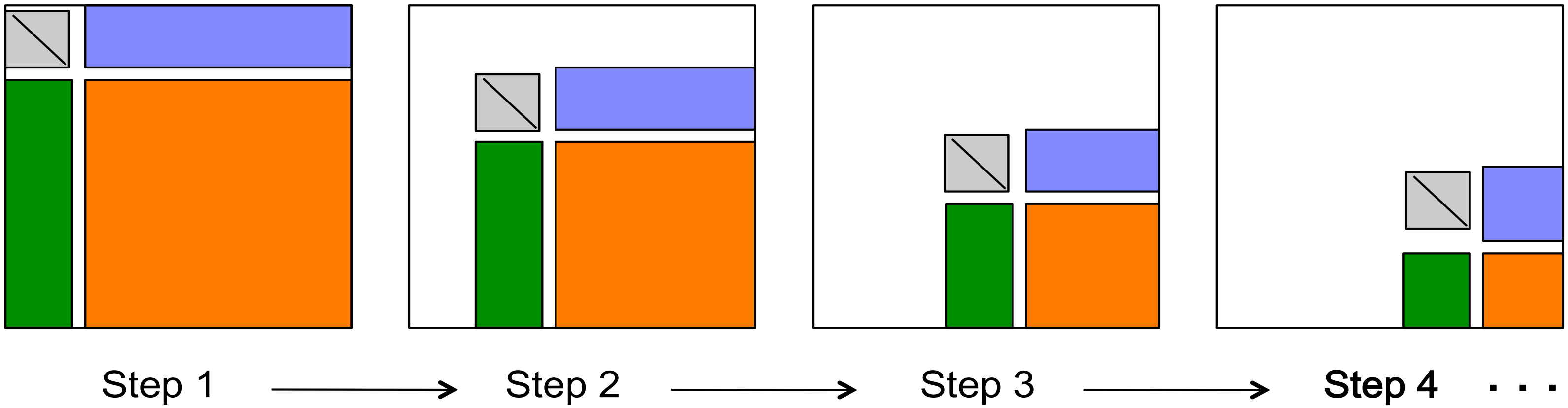
- Easy and done in any reasonable software.
- This is the $2/3n^3$ term in the FLOPs count.
- Can be done efficiently with LAPACK+multithreaded BLAS

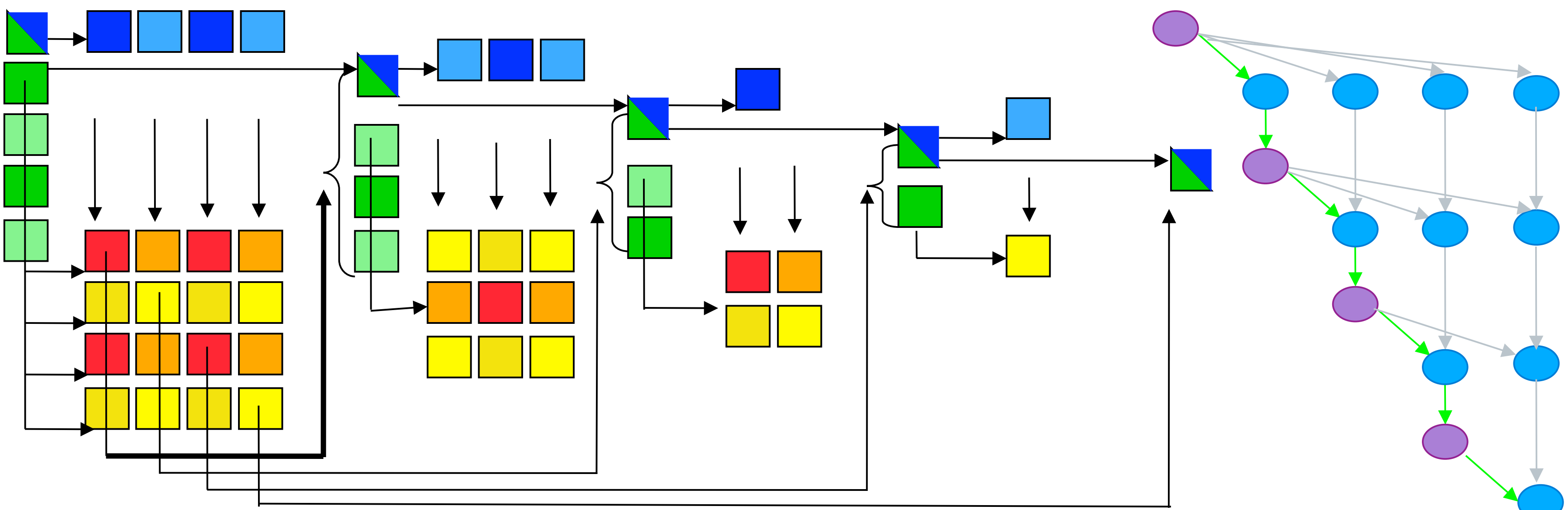
dgemm





1. Synchronization (in LAPACK LU)

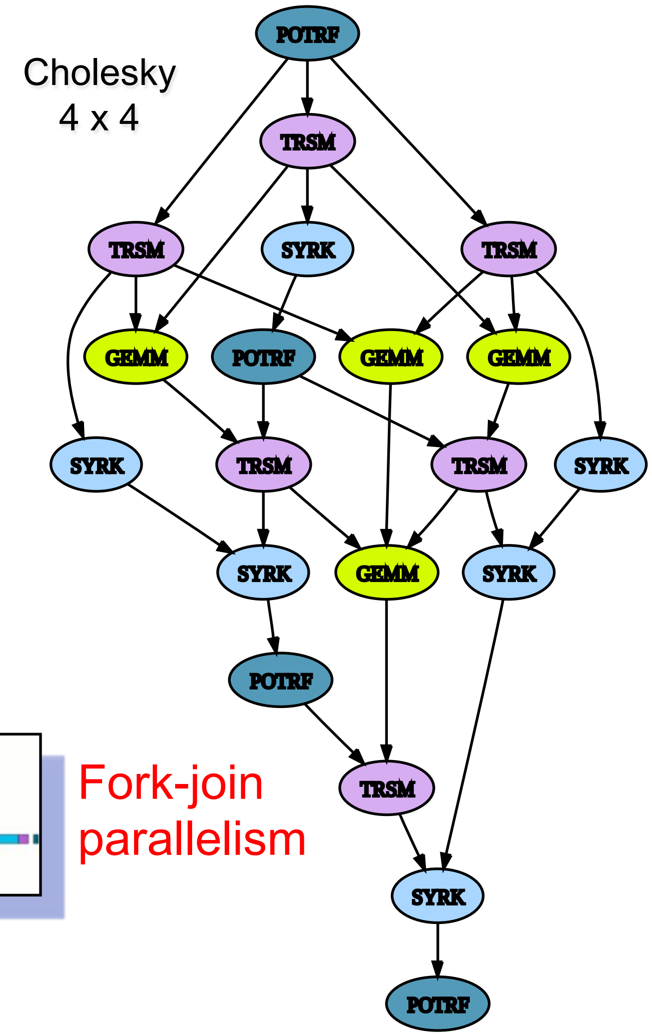


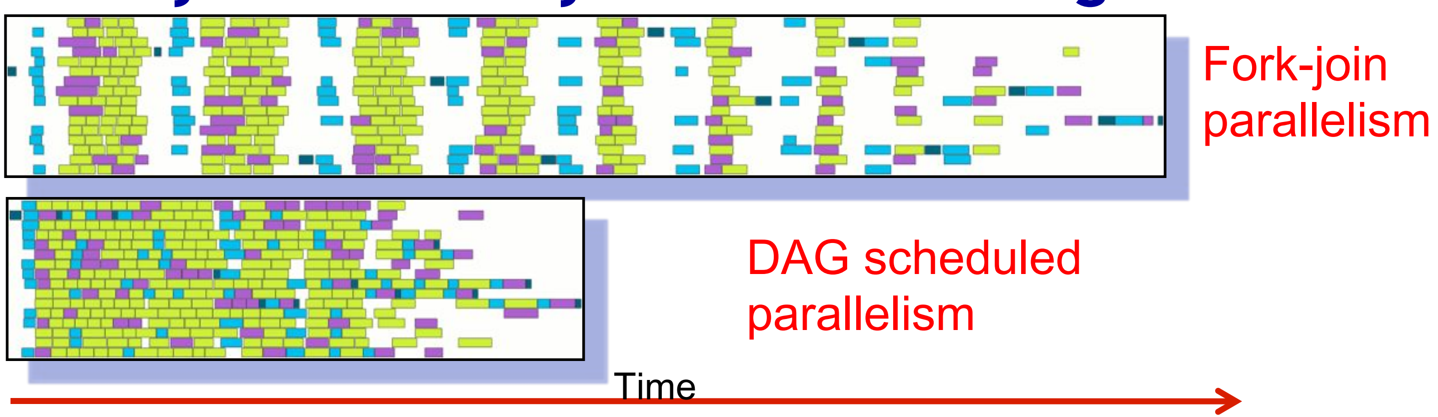


Allowing for delayed update, out of order, asynchronous, dataflow execution

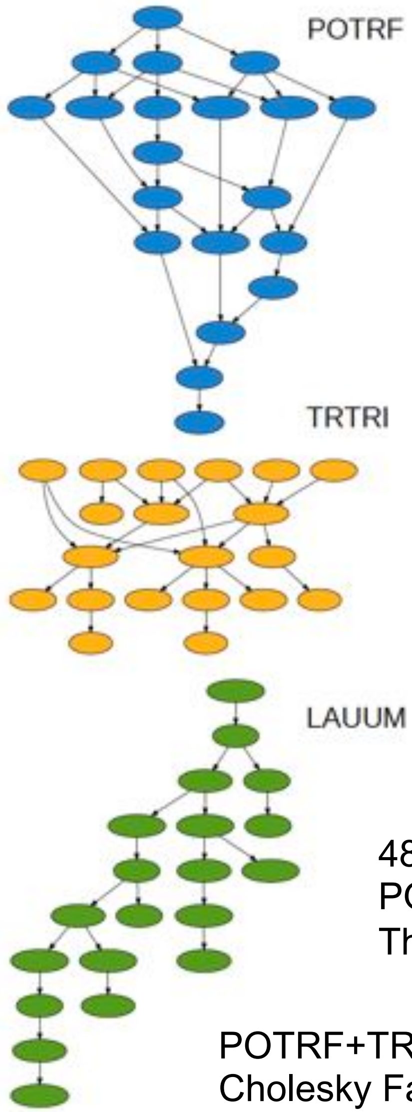
PLASMA/MAGMA: Parallel Linear Algebra s/w for Multicore/Hybrid Architectures

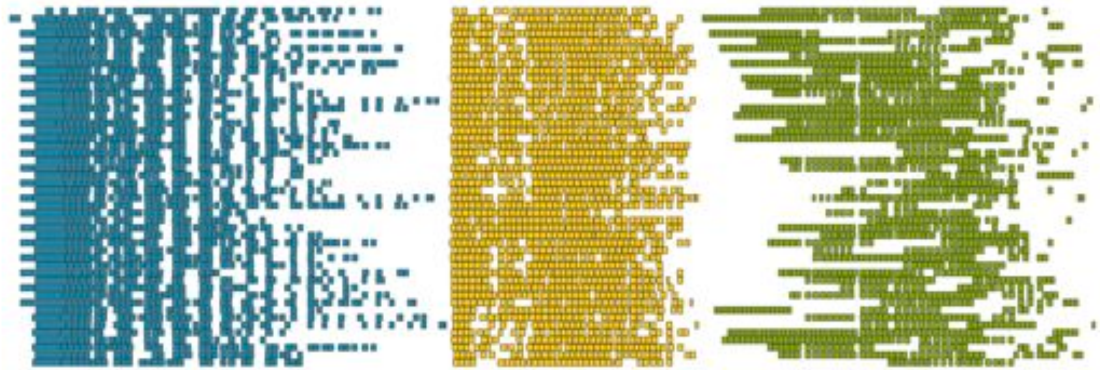
- **Objectives**
  - High utilization of each core
  - Scaling to large number of cores
  - Synchronization reducing algorithms
- **Methodology**
  - Dynamic DAG scheduling (QUARK)
  - Explicit parallelism
  - Implicit communication
  - Fine granularity / block data layout
- **Arbitrary DAG with dynamic scheduling**

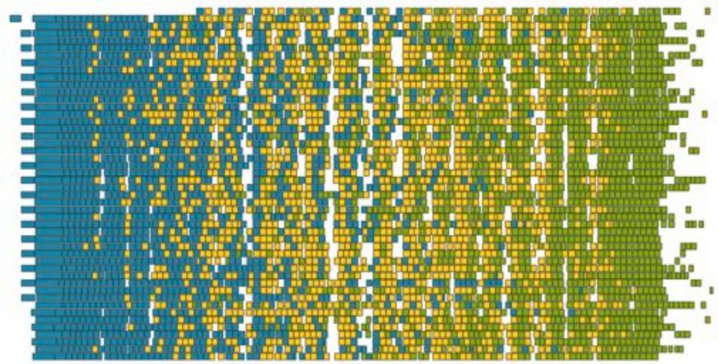


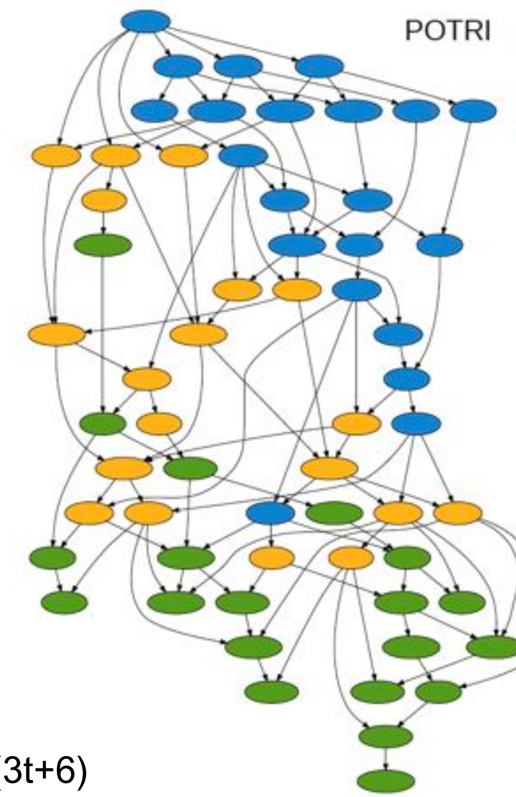


Pipelining: Cholesky Inversion
3 Steps: Factor, Invert L, Multiply L's









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

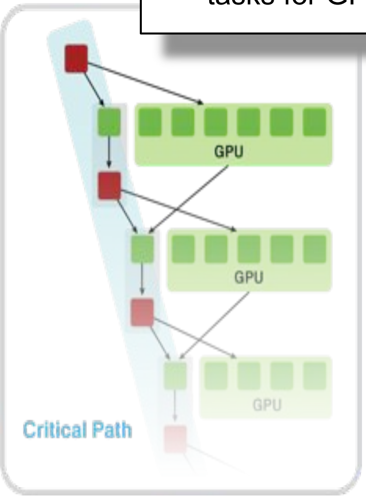
Pipelined: 18 (3t+6)

Hybrid Algorithms

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

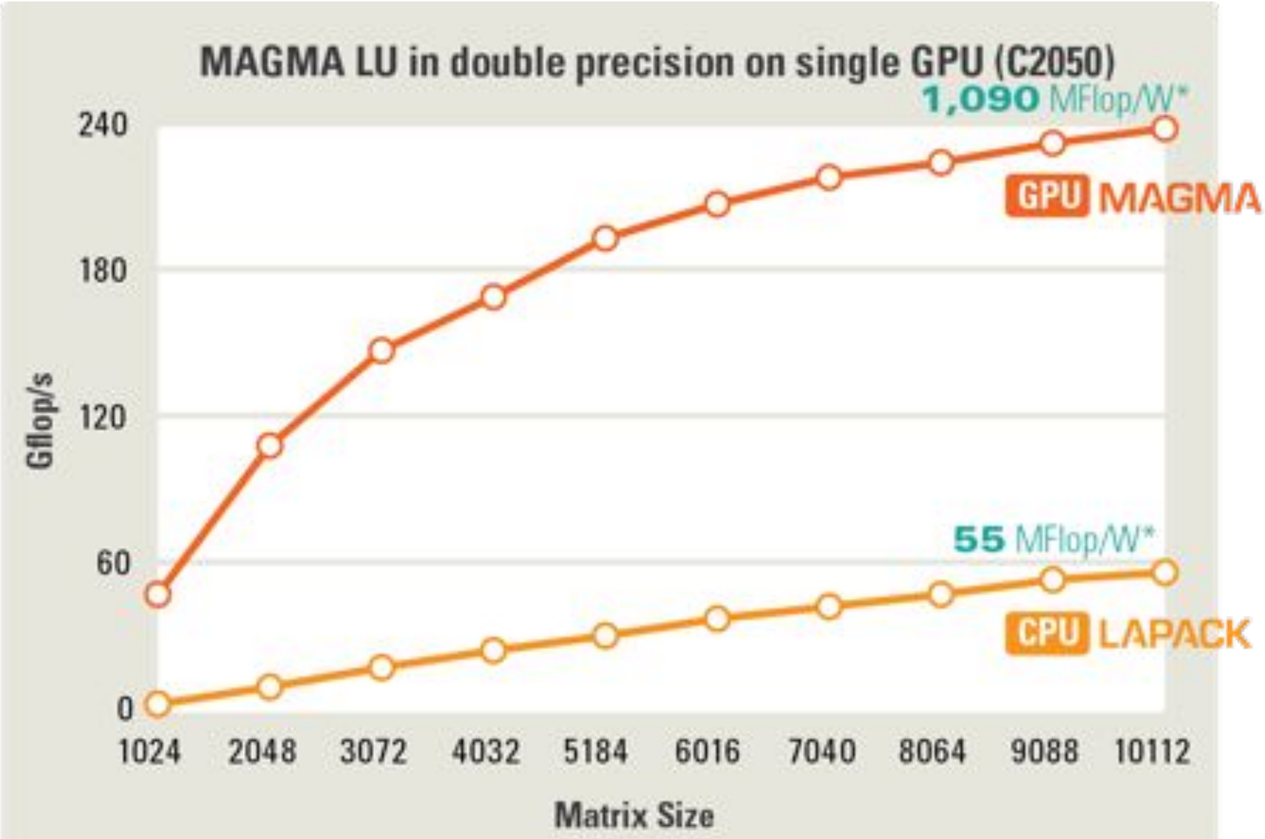
Hybrid CPU+GPU algorithms
(small tasks for multicores and large tasks for GPUs)



- **Successfully applied to fundamental linear algebra algorithms**
  - One and two-sided factorizations and solvers
  - Iterative linear and eigen-solvers

- **Productivity**
  - High-level
  - Leveraging prior developments
  - Exceeding in performance homogeneous solutions

MAGMA Performance (single GPU)

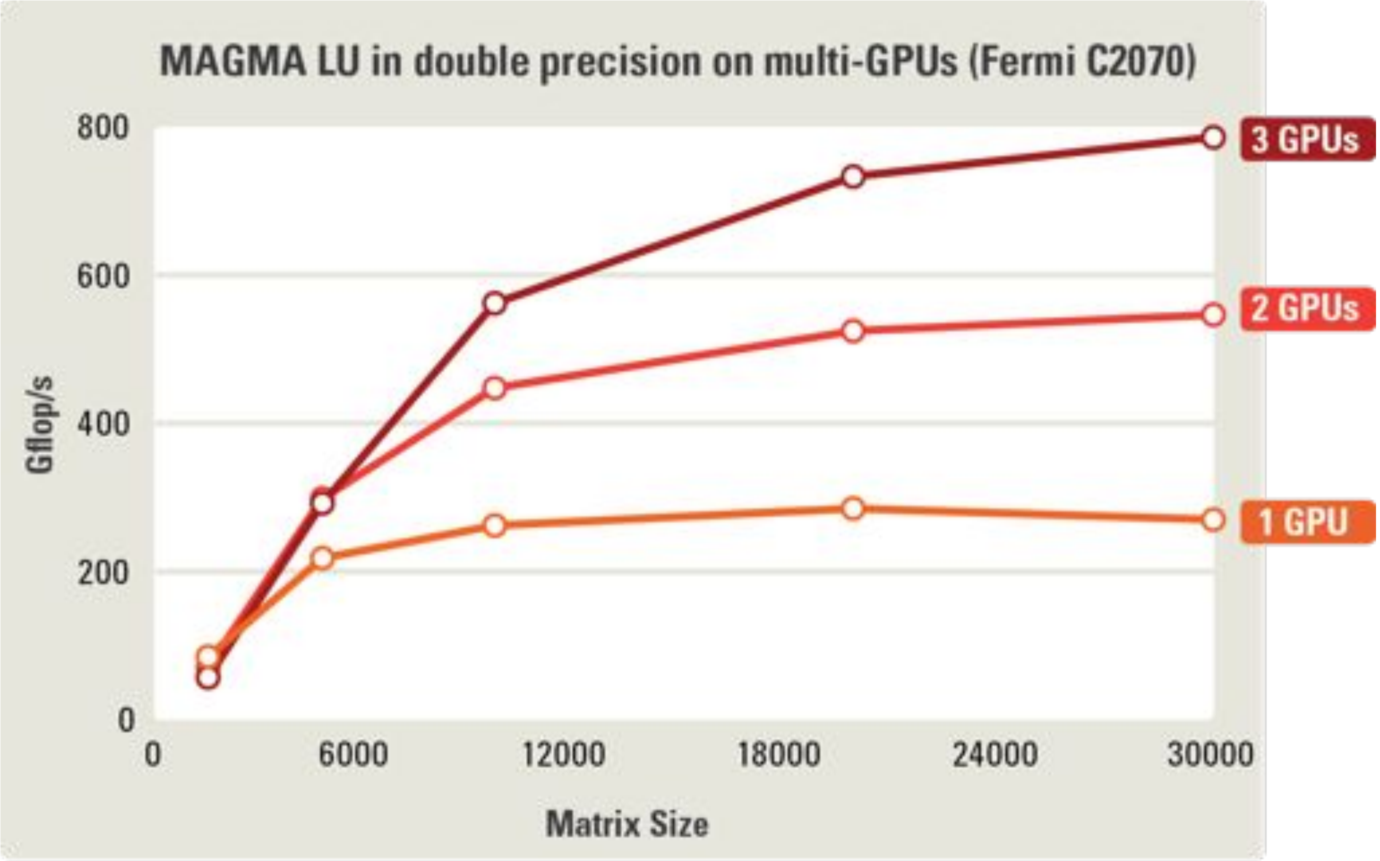


GPU Fermi C2050 (448 CUDA Cores @ 1.15 GHz)
+ Intel Q9300 (4 cores @ 2.50 GHz)
DP peak **515** + **40** GFlop/s
Power * ~**220** W

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

MAGMA Performance (scaling)



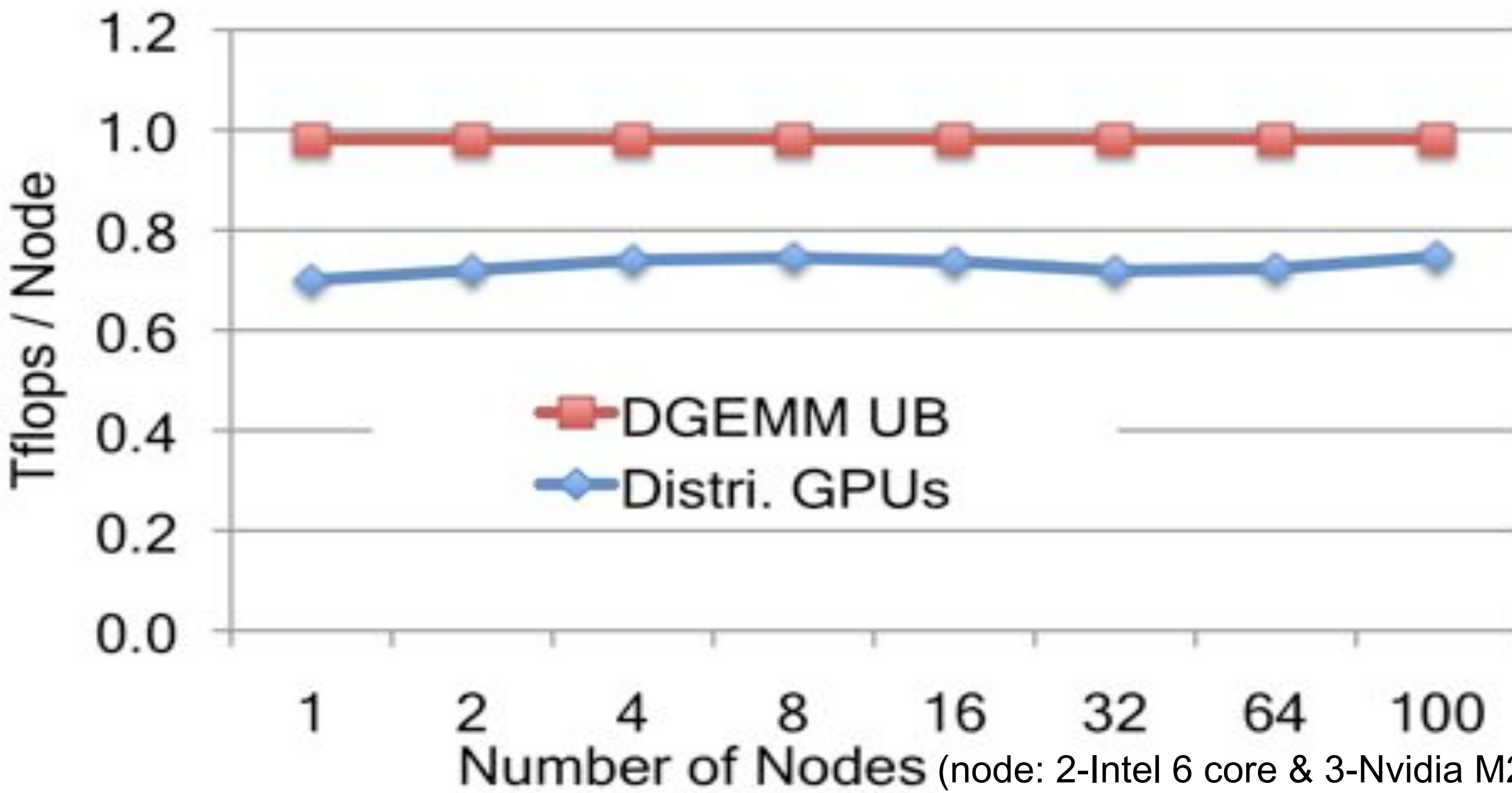
Keeneland system, using one node

3 NVIDIA GPUs (M2070 @ 1.1 GHz, 5.4 GB)

2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

Cholesky Factorization (DP)

- Weak scalability on many nodes (NSF Keeneland system;
  node: 2-Intel 6 core & 3-Nvidia M2070)
- Input size: 34560, 46080, 69120, 92160, 138240, 184320, 276480, 460800



(node: 2-Intel 6 core & 3-Nvidia M2070)

The Need for HP Linear Algebra

Electronic structure calculations

- Density functional theory

  Many-body Schrödinger equation (exact but exponential scaling)

$$\{-\sum_i \frac{1}{2}\nabla_i^2 + \sum_{i,j}\frac{1}{|r_i - r_j|} + \sum_{i,J}\frac{Z}{|r_i - R_J|}\}\Psi(r_1,..r_N) = E\Psi(r_1,..r_N)$$

- Nuclei fixed, generating external potential (system dependent, non-trivial)
- N is number of electrons

Kohn Sham Equation: The many body problem of interacting electrons is reduced to non-interacting electrons (single particle problem) with the same electron density and a different effective potential (cubic scaling).

$$\{-\frac{1}{2}\nabla^2 + \int\frac{\rho(r')}{|r-r'|}dr' + \sum_J\frac{Z}{|r-R_J|} + V_{xc}\}\psi_i(r) = E_i\psi_i(r)$$

$$\rho(r) = \sum_i |\psi_i(r)|^2 = |\Psi(r_1,..r_N)|^2$$

- V_{xc} represents effects of the Coulomb interactions between electrons
- ρ is the density (of the original many-body system)

V_{xc} is not known except special cases ⇒ use approximation, e.g. Local Density Approximation (LDA) where V_{xc} depends only on ρ

- A model leading to self-consistent iteration computation with need for HP LA (e.g, **diagonalization** and **orthogonalization)**

Generalized Hermitian Eigen-Problem

$$A\,x = \lambda\,B\,x$$

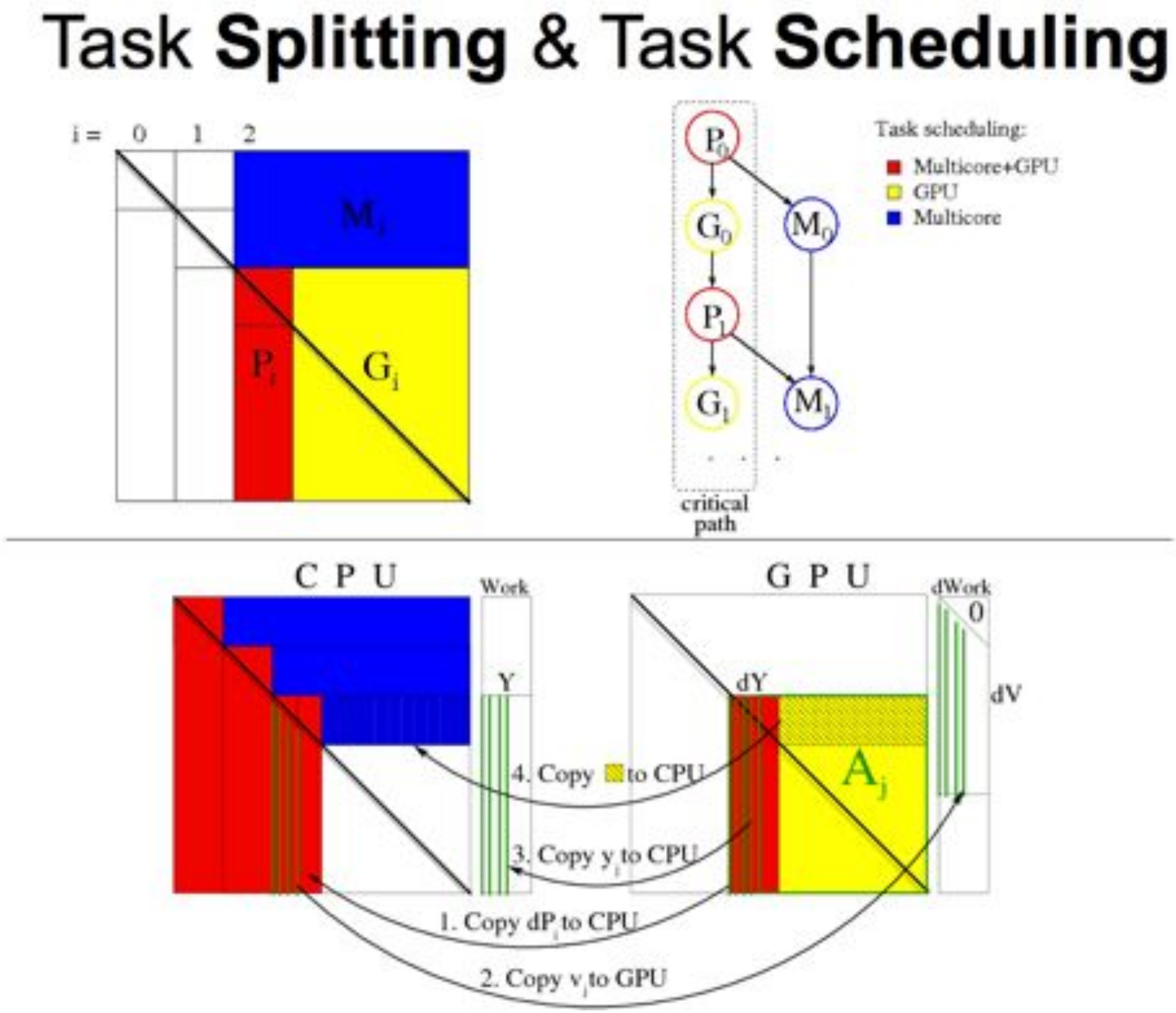
- Compute Cholesky factorization of
  $B = LL^H$

- Transform the problem to a standard eigenvalue problem
  $\tilde{A} = L^{-1}AL^{-H}$

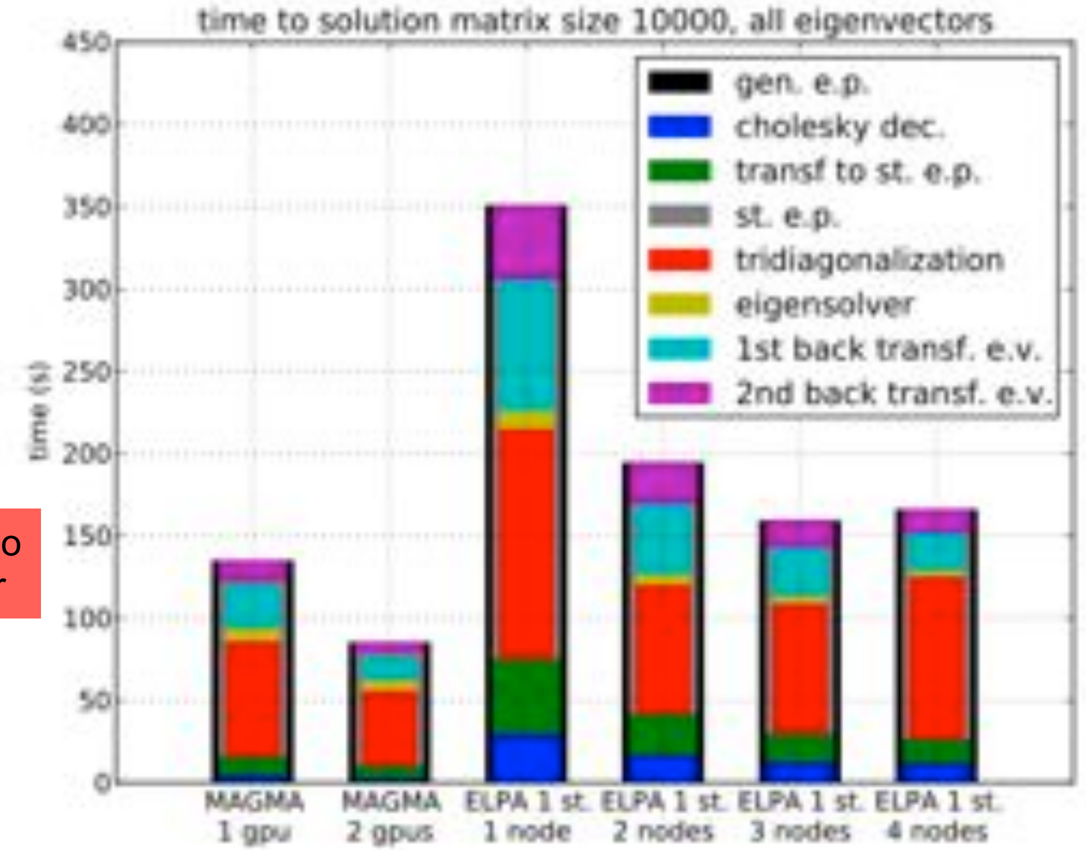
- Solve Hermitian standard Eigenvalue problem
  $\tilde{A}y = \lambda y$

- Transform back the eigenvectors
  $x = L^{-H}\,y$

Hybridization detail



Performance Comparison of Generalized Eigenproblem in Double Complex Precision



Time shown, so lower is better

- **Test system:**
  2 x Intel X5650 (6 core), 2 x Nvidia M2090

Thomas Schulthess
Raffaele Solcà

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

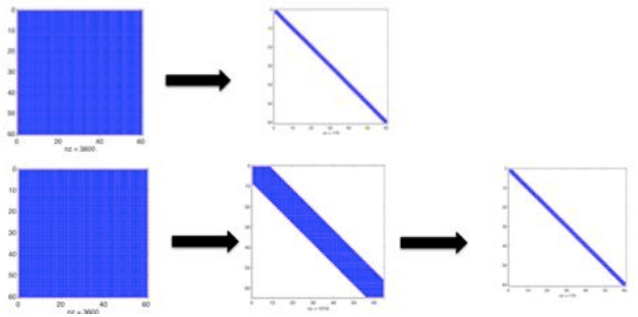
ELPA solver from:

Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations ☆

T. Auckenthaler[a], V. Blum[b], H.-J. Bungartz[a], T. Huckle[a], R. Johanni[c], L. Krämer[d], B. Lang[d,*], H. Lederer[c], P.R. Willems[d]

[a] Fakultät für Informatik, Technische Universität München, D-85748 Garching, Germany
[b] Fritz-Haber-Institut der Max-Planck-Gesellschaft, D-14195 Berlin, Germany
[c] Rechenzentrum Garching der Max-Planck-Gesellschaft am Max-Planck-Institut für Plasmaphysik, D-85748 Garching, Germany
[d] Fachbereich C, Bergische Universität Wuppertal, D-42097 Wuppertal, Germany

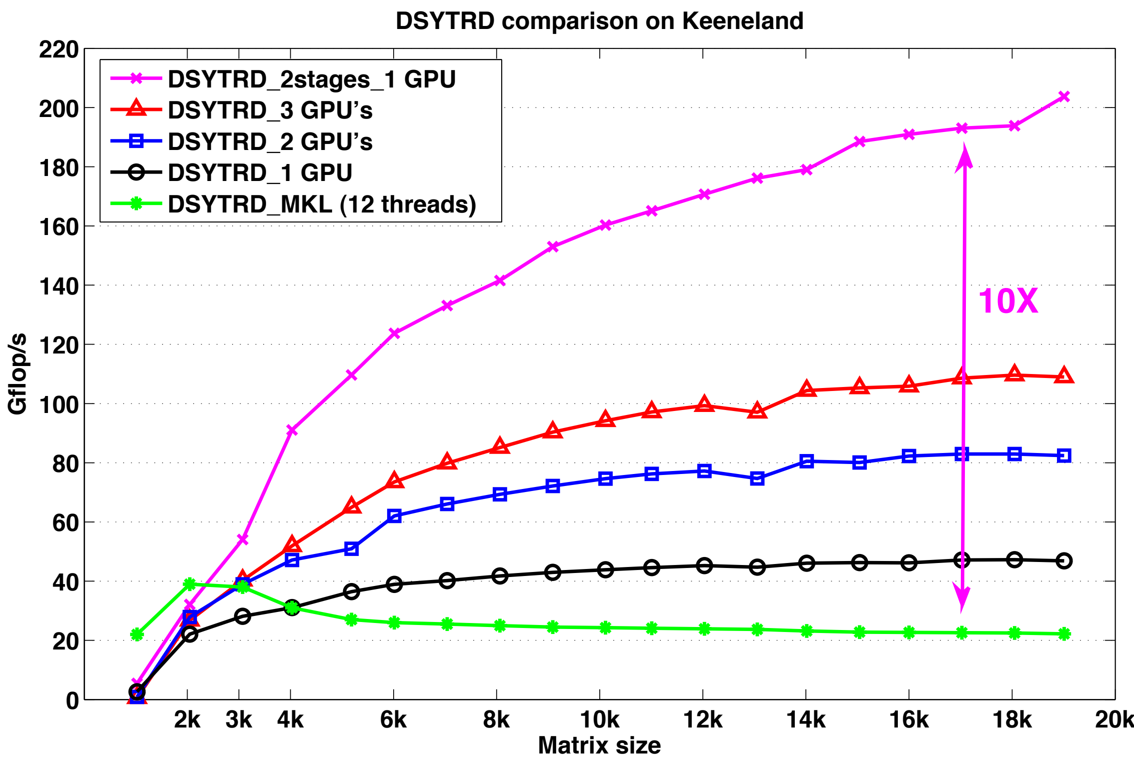
Two-Stage Approach to Tridiagonal Form (Communication Reducing)



- **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]

Performance results

Tridiagonalization in double precision on Fermi



- **Communication reducing**
- **Developed routines for multiGPUs obtaining scalable performance**
- **The new algorithm (2 stages approach) on a Keeneneland node bring a speedup of ~ 10 X**

Keeneland system, using one node

3 NVIDIA GPUs (M2070@ 1.1 GHz, 5.4 GB)

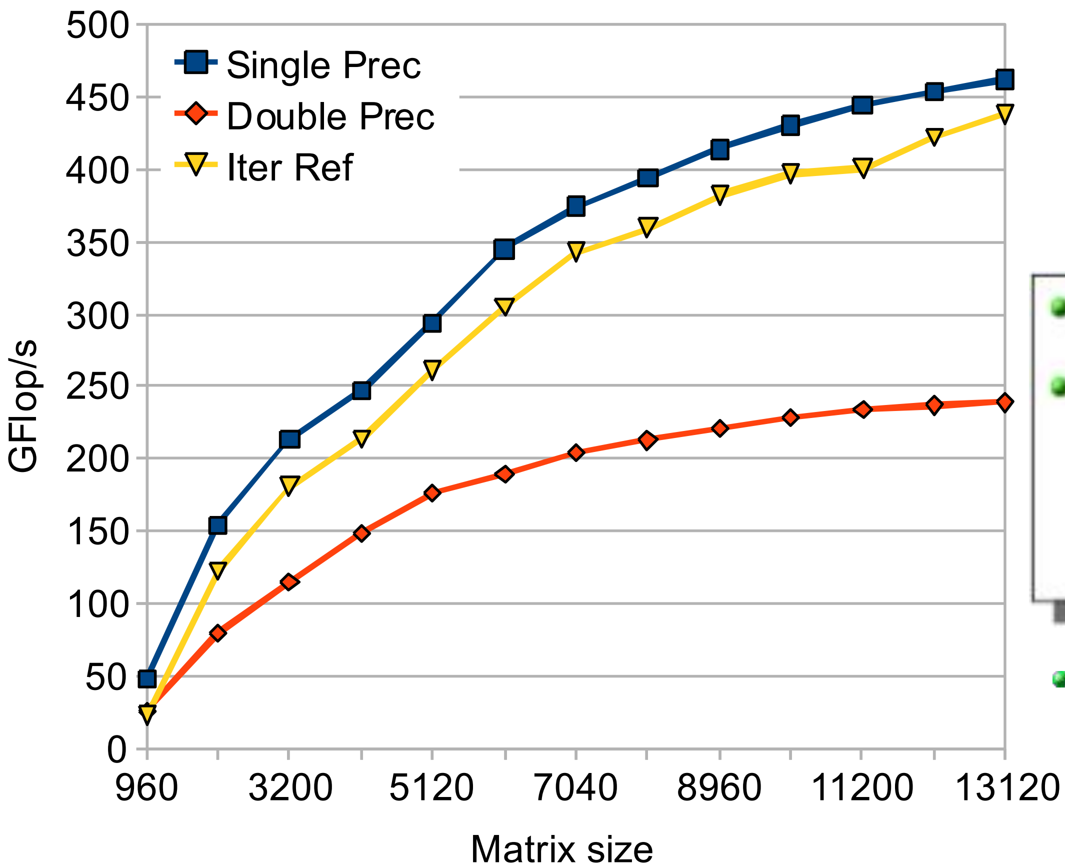
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

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].**

Mixed Precision Solvers

MAGMA LU-based solvers on Fermi (C2050)



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

- Similar results for Cholesky & QR

COMPUTATIONAL ROUTINES IN MAGMA 1.1

	MATRIX	OPERATION	ROUTINE	INTERFACES CPU	GPU
LINEAR EQUATIONS	GE	LU	{sdcz}getrf	✓	✓
		Solve	{sdcz}getrs		✓
		Invert	{sdcz}getri		✓
	SPD/HPD	Cholesky	{sdcz}potrf	✓	✓
		Solve	{sdcz}potrs		✓
		Invert	{sdcz}potri		✓
	TR	Invert	{sdcz}trtri	✓	
ORTHOGONAL FACTORIZATIONS	GE	QR	{sdcz}geqrf	✓	
		Generate Q	{sd}orgqr	✓	✓
			{cz}ungqr	✓	✓
		Multiply matrix by Q	{sd}ormqr	✓	✓
			{cz}unmqr	✓	✓
	GE	LQ factorization	{sdcz}gelqf	✓	✓
		QL factorization	{sdcz}geqlf	✓	
		Multiply matrix by Q	{sd}ormql	✓	✓
			{cz}unmql	✓	✓
STANDARD EVP	GE	Hessenberg reduction	{sdcz}gehrd	✓	
		Generate Q	{sd}orghr	✓	
			{cz}unghr	✓	
	SY/HE	Tridiagonalization	{sd}sytrd	✓	
			{cz}hetrd	✓	
		Generate Q	{sd}orgtr		✓
			{cz}ungtr		✓
		Multiply by Q	{sd}ormtr	✓	✓
			{cz}unmtr	✓	✓
SVD	GE	Bidiagonalization	{sdcz}gebrd	✓	
GENERALIZED EVP	SPD/HPD	Reduction to standard	{sd}sygst	✓	✓
		form	{cz}hegst	✓	✓

DRIVER ROUTINES IN MAGMA 1.1

	MATRIX	OPERATION	ROUTINE	INTERFACES CPU	GPU
LINEAR EQUATIONS	GE	Solve using LU	{sdcz}gesv	✓	✓
		Solve using MP	{zc,ds}gesv		✓
	SPD/HPD	Solve using Cholesky	{sdcz}posv	✓	✓
		Solve using MP	{zc,ds}posv		✓
LLS	GE	Solve LLS using QR	{sdcz}geqrs		✓
		Solve using MP	{zc,ds}geqrsv		✓
STANDARD EVP	GE	Compute e-values,	{sdcz}geev	✓	
		optionally e-vectors			
	SY/HE	Computes all e-values,	{sd}syevd	✓	✓
		optionally e-vectors	{cz}heevd	✓	✓
		Range (D&C)	{cz}heevdx		✓
		Range (B & I It.)	{cz}heevx	✓	✓
		Range (MRRR)	{cz}heevr	✓	✓
STAND. SVP	GE	Compute SVD,	{sdcz}gesvd	✓	
		optionally s-vectors		✓	
GENERALIZED EVP	SPD/HPD	Compute all e-values,	{sd}sygvd	✓	
		optionally e-vectors	{cz}hegvd	✓	
		Range (D&C)	{cz}hegvdx	✓	
		Range (B & I It.)	{cz}hegvx	✓	
		Range (MRRR)	{cz}hegvr	✓	

Collaborators / Support

- **MAGMA team**
  http://icl.cs.utk.edu/magma/
- **PLASMA team**
  http://icl.cs.utk.edu/plasma
- **DAGuE team**
  http://icl.cs.utk.edu/dague/
- **Collaborating partners**

University of Tennessee, Knoxville
University of California, Berkeley
University of Colorado, Denver

INRIA, France
KAUST, Saudi Arabia