Efficient Eigensolver Algorithms on Accelerator Based Architectures

Stan Tomov

w/ Azzam Haidar, Piotr Luszczek, Ichitaro Yamazaki, Mark Gates, and Jack Dongarra

Innovative Computing Laboratory
Department of Computer Science
University of Tennessee, Knoxville

SIAM LA 2015
Atlanta, GA
October 29, 2015

ICL UT INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Outline

- Motivation
- Eigenvalue and SVD problems
- Main challenges
  - Parallelism
  - Heterogeneity
  - Communication
- One and two-stage algorithms
- Conclusions and future directions

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Linear algebra in applications

Dense Linear Algebra (DLA) is needed in a wide variety of science and engineering applications:

- **Linear systems:** Solve $Ax = b$
  - Computational electromagnetics, material science, applications using boundary integral equations, airflow past wings, fluid flow around ship and other offshore constructions, and many more
- **Least squares:** Find x to minimize $\| Ax - b \|$
  - Computational statistics (e.g., linear least squares or ordinary least squares), econometrics, control theory, signal processing, curve fitting, and many more
- **Eigenproblems:** Solve $Ax = \lambda x$
  - Computational chemistry, quantum mechanics, material science, face recognition, PCA, data-mining, marketing, Google Page Rank, spectral clustering, vibrational analysis, compression, and many more
- **Singular Value Decomposition (SVD):** $A = U \Sigma V^*$ ($Au = \sigma v$ and $A^*v = \sigma u$)
  - Information retrieval, web search, signal processing, big data analytics, low rank matrix approximation, total least squares minimization, pseudo-inverse, and many more
- **Many variations depending on structure of A**
  - A can be symmetric, positive definite, tridiagonal, Hessenberg, banded, sparse with dense blocks, etc.
- **DLA is crucial to the development of sparse solvers**

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

General Overview: the Eigen-problem algorithms

1. Singular Value Decomposition $A = U\Sigma V^T$
   - Bi-Diagonalization Reduction + solve + back transformation.

2. Symmetric Eigenvalue Problem $Ax = \lambda x$
   - Tri-Diagonalization Reduction + solve + back transformation.

3. Non-symmetric Eigenvalue Problem $Ax = \lambda x$
   - Hessenburg Reduction + solve + back transformation.

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

General Overview: the Eigen-problem algorithms

- **Three phases:**
  - **Reduction phase – reduce *A* to condensed form**

A = Q H Q*

Nonsymmetric Eigenvalue problem

A = Q T Q*

Symmetric Eigenvalue problem

A = Q B P*

Singular value Decomposition problem

  - **Solution phase – compute the eigenvalues, eigenvectors and singular value of the condensed matrix**

H = E T E*
==> A = Q (E T E*) Q*

T = E Λ E*
==> A = Q (E Λ E*) Q*

B = X S Y*
==> A = Q (X S Y*) P*

  - **Back transformation phase – Update the computed eigenvectors and orthogonal matrix**

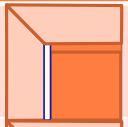
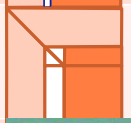
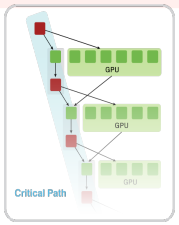
Z = Q E

Z = Q E

U = Q X
V* = Y* P*

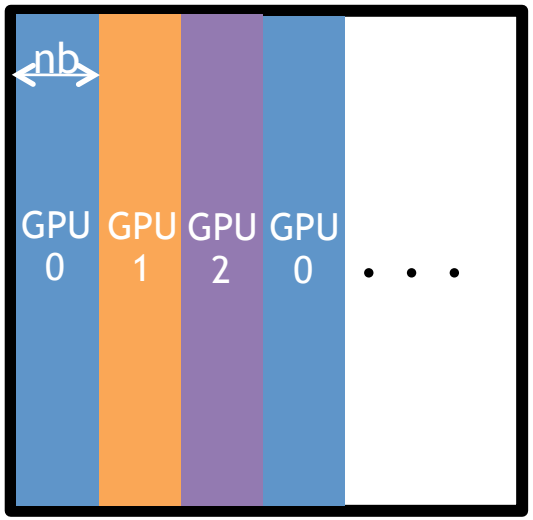
Major challenges to software on heterogeneous architectures

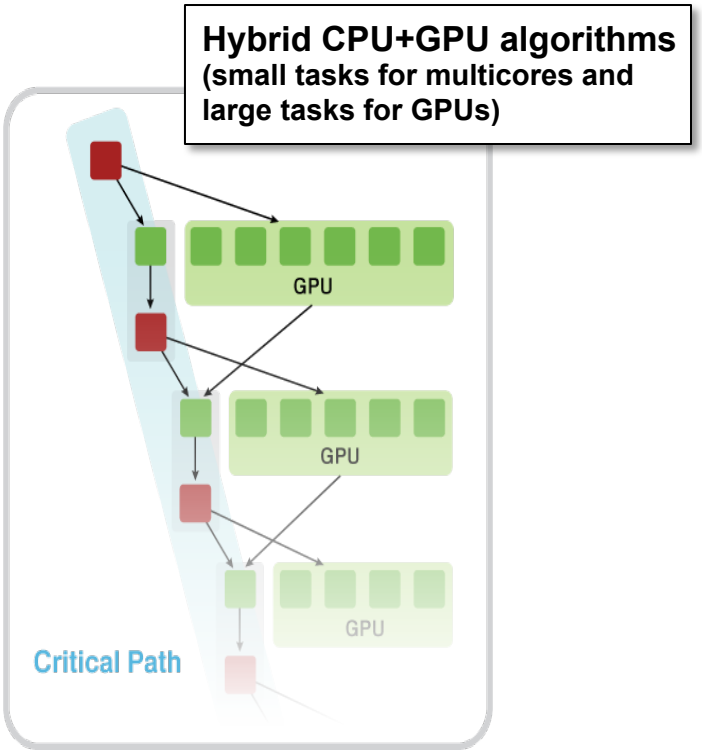
- **Parallelism**
  - Explosion in parallelism; stalled clock rates
    [2x1,664 CUDA cores in K80]
- **Heterogeneity**
  - A few heavyweight and many lightweight cores
  - Multiple processor architectures, memory systems, etc.
- **Communications**
  - Growing gap between peak performance and bandwidth

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 (for nested parallelism)

Parallelism

- **Algorithms expressed in terms of BLAS**
- **Task based approach with event/data driven execution**
  - Algorithms as DAGs
  - Runtime system schedules the execution (static, QUARK, PaRSEC, OpenMP4, SMPSs, StarPU, etc.)
- **1-D block-cyclic distribution on single node** (or 2-D block-cyclic on distributed memory systems)





Heterogeneity

Left-looking hybrid Cholesky

From sequential LAPACK

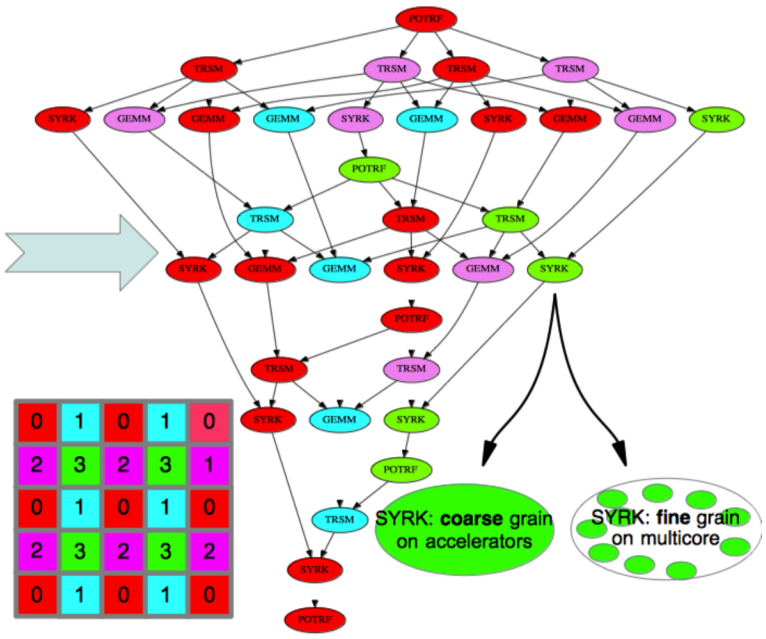
```
for( j=0, j<n; j+=nb) {
    jb = min(nb, n-j);
    zherk( MagmaUpper
           jb, j, one, dA(0
    if (j+jb < n)
      zgemm( MagmaCo
             dA(0,j), ldd
    zpotrf( MagmaUpper
    if (info != 0)
       *info += j;
    If (j+jb) < n) {
      ztrsm( MagmaLeft,
             jb, n
    }
}
```

to parallel hybrid MAGMA

```
1  for( j=0, j<n; j+=nb) {
2      jb = min(nb, n-j);
3      magma_zherk( MagmaUpper, MagmaConjTra
                    jb, j, one, dA(0,j), ldda, one, dA
4      magma_zgetmatrix_async( jb, jb, dA(j,j), ldd
5      if (j+jb < n)
6        magma_zgemm( MagmaConjTrans, Magma
                      dA(0,j), ldda, dA(0,j+jb), ldc
7      magma_event_sync( event );
8      zpotrf( MagmaUpperStr, &jb, work, &jb, info);
9      if (info != 0)
10       *info += j;
11     magma_zsetmatrix_async(jb, jb, work, jb, dA
12     If (j+jb) < n) {
13       magma_event_sync( event );
14       magma_ztrsm( MagmaLeft, MagmaUpper,
                      jb, n-j-jb, one, dA(j,j), ldda,
       }
   }
```

MAGMA MIC runtime environment

- Dynamic MAGMA is DAG-based approach
- Also, working very well for distributed systems



[A. Haidar, A. YarKhan, C. Cao, P. Luszczek, S. Tomov, and J. Dongarra, **"Flexible Linear Algebra Development and Scheduling with Cholesky Factorization"**, 17th IEEE International Conference on High Performance Computing and Communications, New York, August 2015.]

Note:

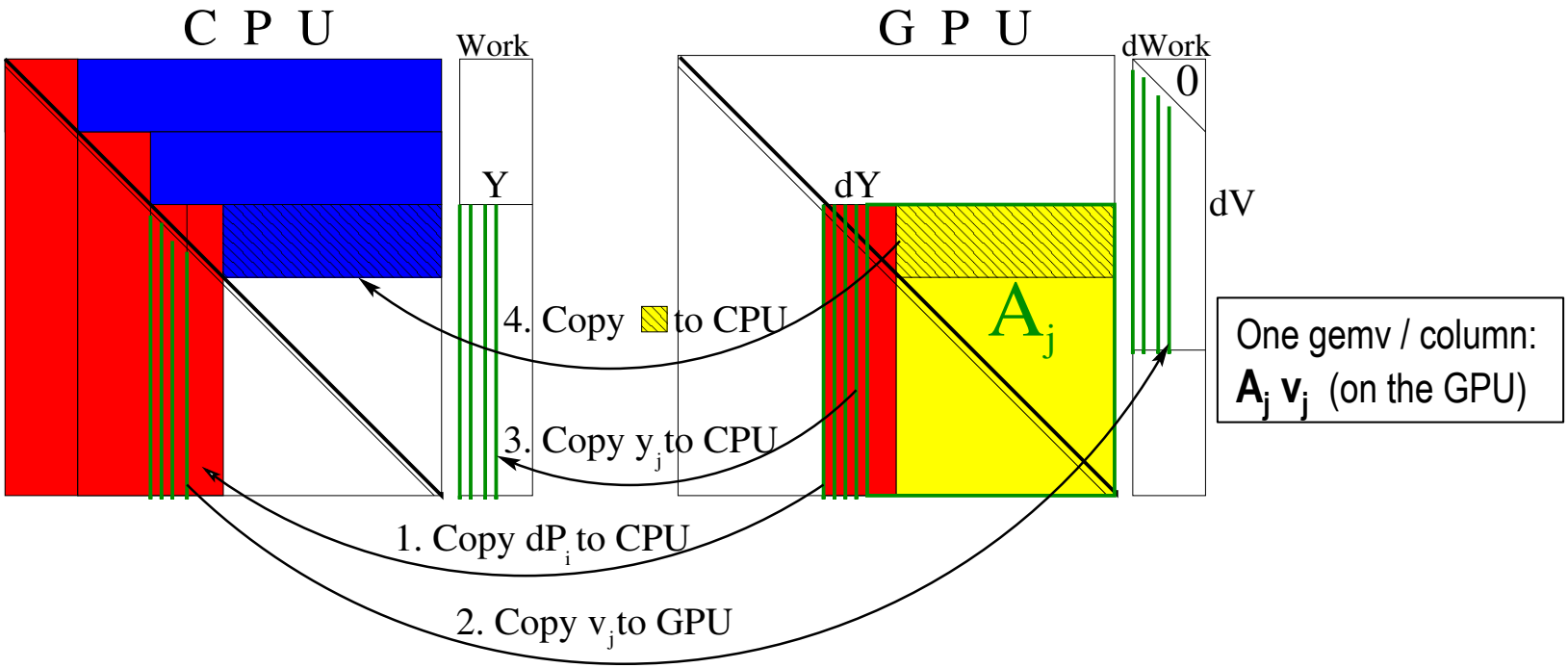
- MAGMA and LAPACK look similar
- Difference is lines in red, specifying data transfers and dependencies
- Differences can be hidden in a dynamic scheduler making the top level representation of MAGMA MIC algorithms almost identical to LAPACK

Heterogeneity ...

- **Example:**
  - **Panels are in general scheduled on the CPU**
  - **For the eigen-problems and SVD panels can be hybrid**

Hybrid panel in the Hessenberg reduction

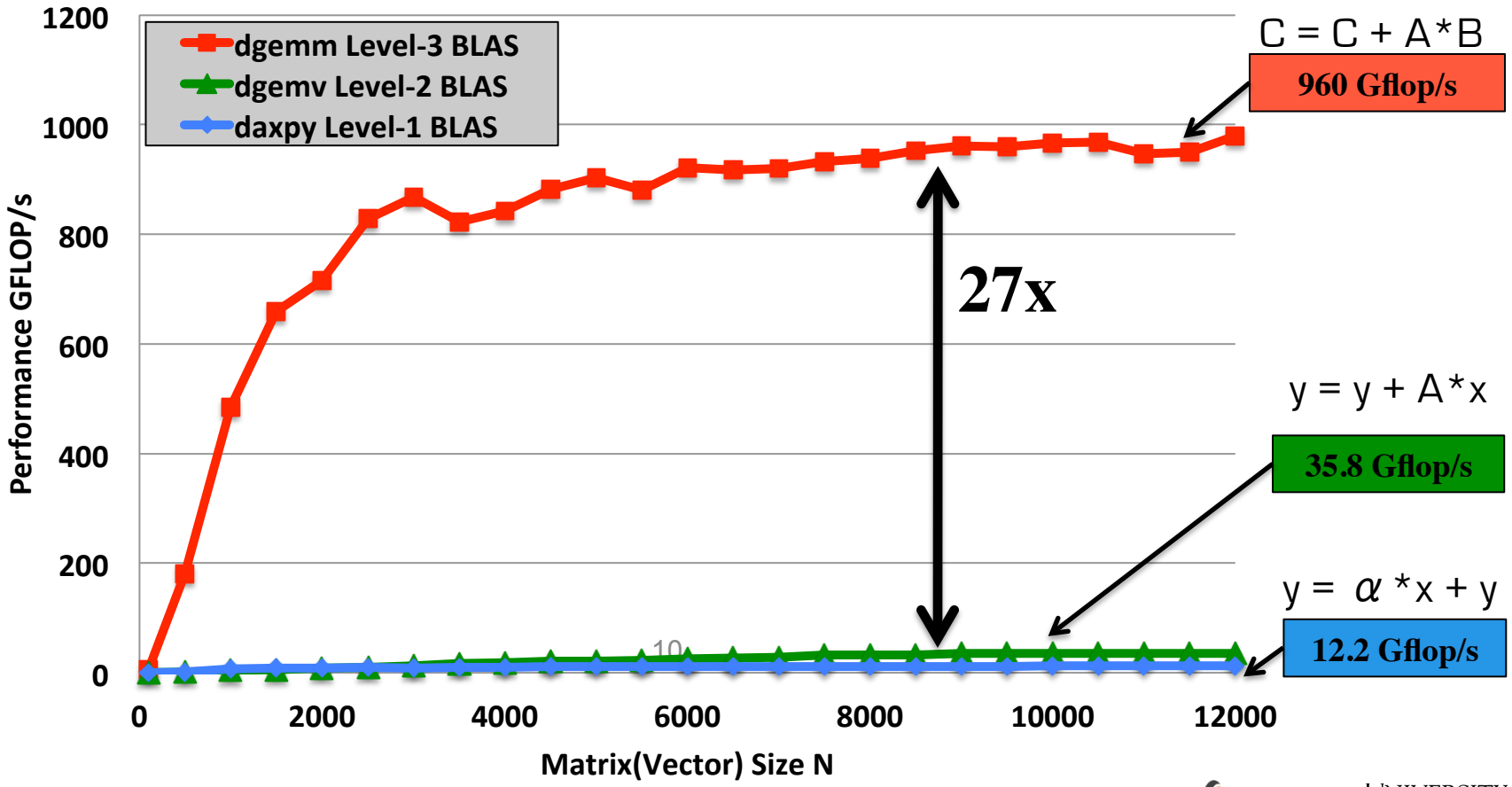
(Next householder vector from the panel factorization is sent to GPU and multiplied by trailing matrix)



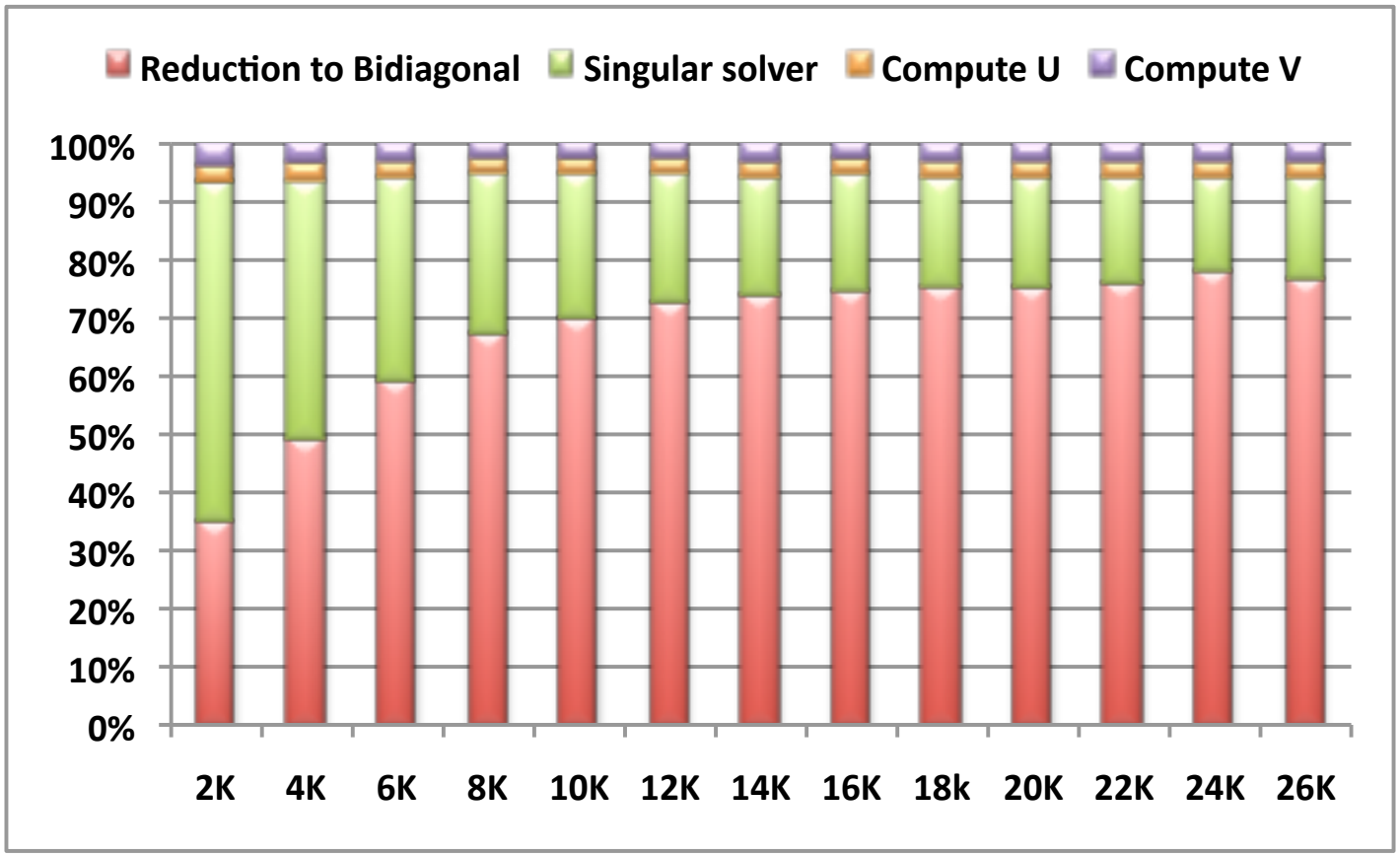
Communication
Not all flops are equal (some need more data movements)

Level 1, 2 and 3 BLAS

60 core **Intel Xeon Phi** KNC 7120, 1.23 GHz, Peak DP = 1208 Gflop/s



Profiling DGESDD (CPU)
Classical Reduction and D&C



Three Phases:

1. **Reduction**
2. **Solver**
3. **Back transform**

Back-trans $4n^3$

Solver $\frac{8n^3}{3}$

Reduction $\frac{8n^3}{3}$

Total flops $\approx 9n^3$

- **DGESDD – computes singular value decomposition of *m x n* matrix**
- **Reduction to Bidiagonal form(DGEBRD) – ≈ 75% of the total computation cost**

Communication … Bidiagonal reduction

Performance Bound Analysis

- **Total cost for bidiagonal reduction (1st phase of SVD)**

$$\approx 4n_b \sum_{n_b}^{n/n_b} l^2 + 4n_b \sum_{2n_b}^{\frac{n-n_b}{n_b}} k^2$$

$$\approx \frac{8}{3}n^3$$

$$\approx \frac{4}{3}n^3_{\text{gemv}} + \frac{4}{3}n^3_{\text{gemm}}$$

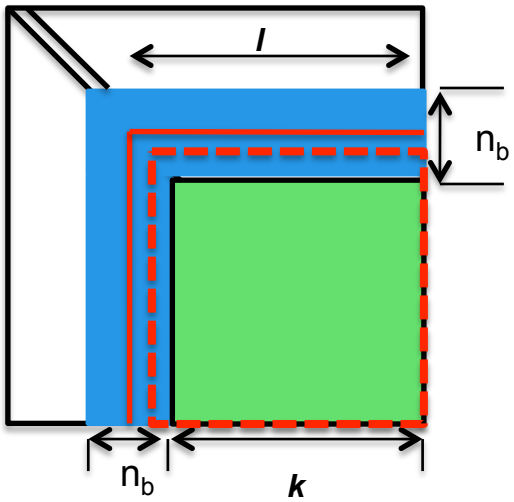
- **Performance upper bound**

$$P_{max} = \frac{number\ of\ operations}{minimum\ time\ t_{min}}$$

$$= \frac{\frac{8}{3}n^3}{\frac{4}{3}n^3 * \frac{1}{P_{gemv}} + \frac{4}{3}n^3 * \frac{1}{P_{gemm}}}$$

$$= \frac{2 * P_{gemm} * P_{gemv}}{P_{gemm} + P_{gemv}}$$

$$\approx 2P_{gemv}$$
$$when \quad P_{gemm} \gg P_{gemv}.$$

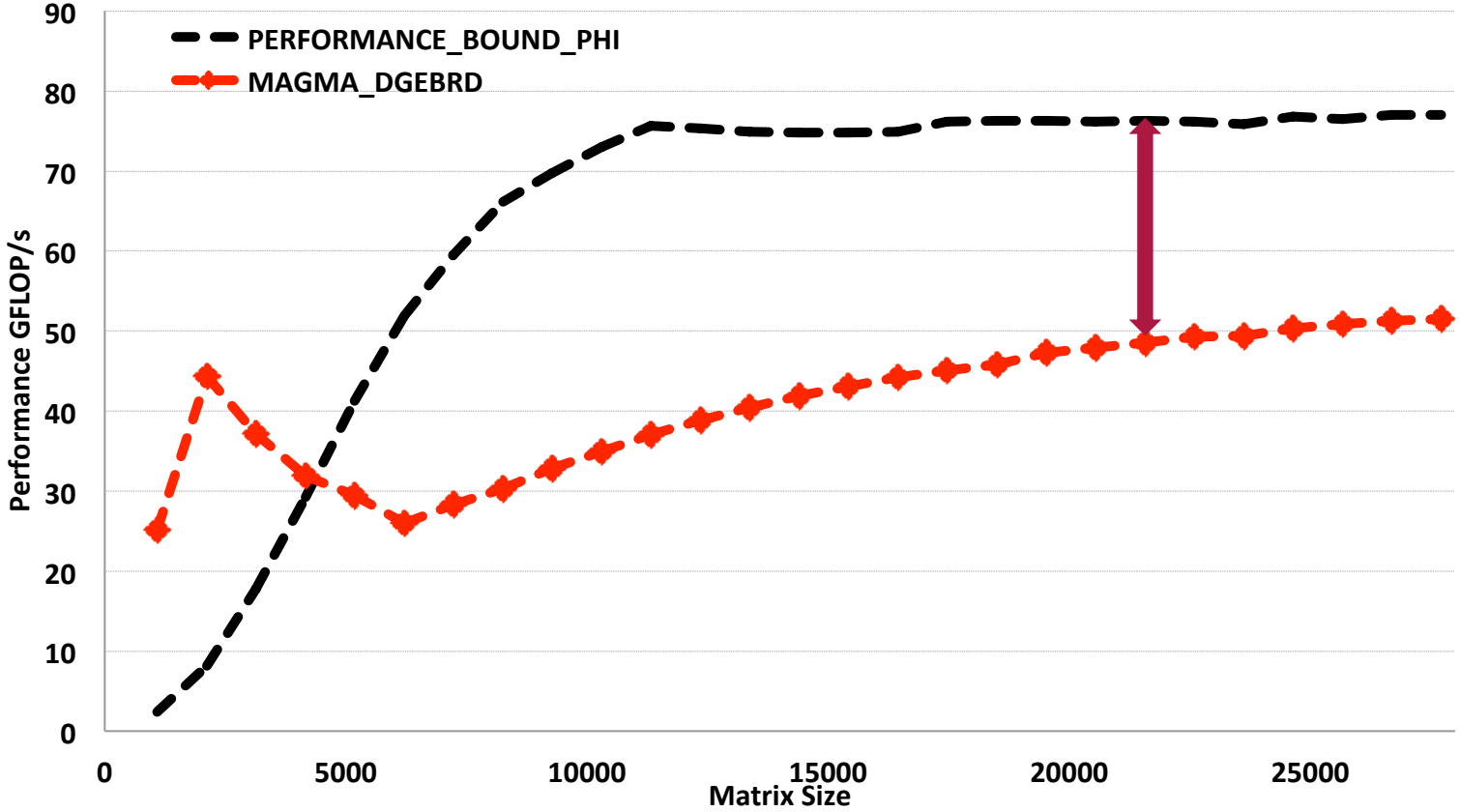


Trailing Matrix Update – Level 3 BLAS update **GEMM**

Panel - Level 2 BLAS update **GEMV**

Panel

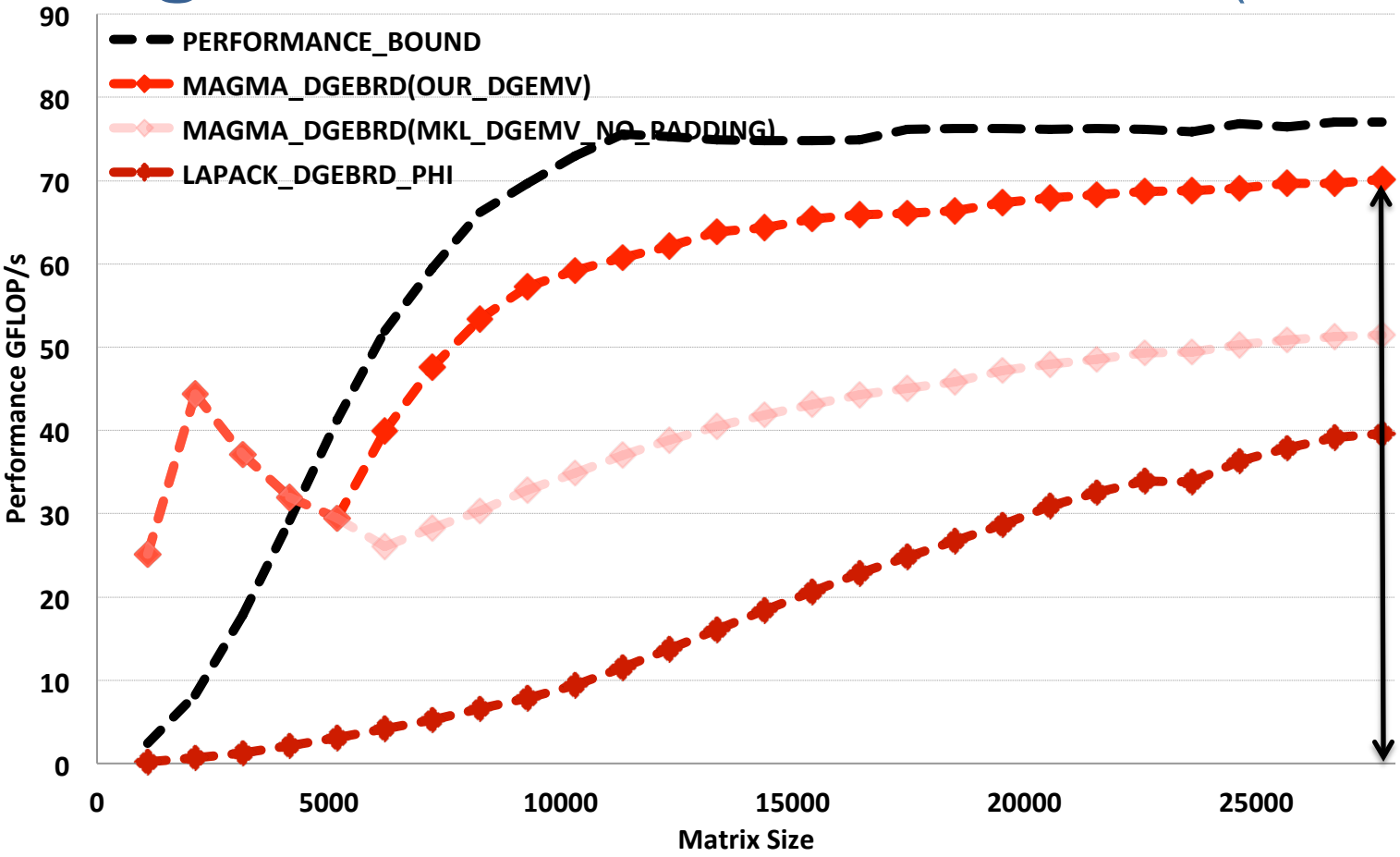
Communication ...
Bidiagonal reduction – Performance (Xeon Phi)



- **Bottleneck:**
  - Accelerator implementation reaches **60% of the performance bound** (Theoretical peak > 1 Tflop/s)
  - The gap indicates that DGEMV inside DGEBRD does not perform well.
- **Observations:**

  MKL's DGEMV performance depends on both, the size and the memory alignment of the matrix

Communication ...
Bidiagonal reduction – Performance (Xeon Phi)



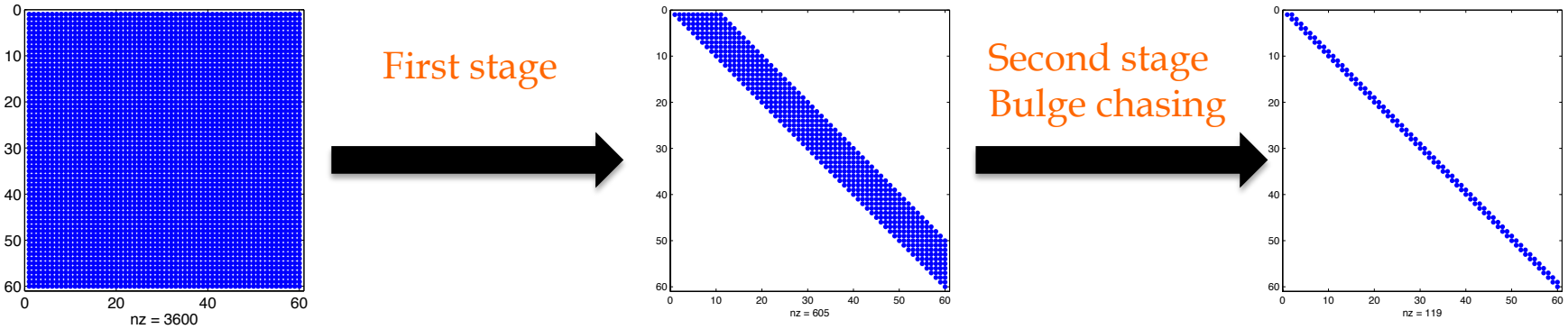
- Our implementation is within 90% of the performance bound
- Hierarchical communications (blocking + prefetching for both L1 and L2 cache) is essential here

Communication … Bidiagonal reduction

- The only way to do better – design a new algorithm to communicate less

Communication … Bidiagonal reduction

- The only way to do better – design a new algorithm to communicate less
- Recent work on developing new "2-stage" algorithms



$$\text{flops} \approx \sum_{s=1}^{\frac{n-n_b}{n_b}} 2n_b^3 + (nt-s)3n_b^3 + (nt-s)\tfrac{10}{3}n_b^3 + (nt-s)\times(nt-s)5n_b^3$$

$$+ \sum_{s=1}^{\frac{n-n_b}{n_b}} 2n_b^3 + (nt-s-1)3n_b^3 + (nt-s-1)\tfrac{10}{3}n_b^3 + (nt-s)\times(nt-s-1)5n_b^3$$

$$\approx \tfrac{10}{3}n^3 + \tfrac{10n_b}{3}n^2 + \tfrac{2n_b}{3}n^3$$

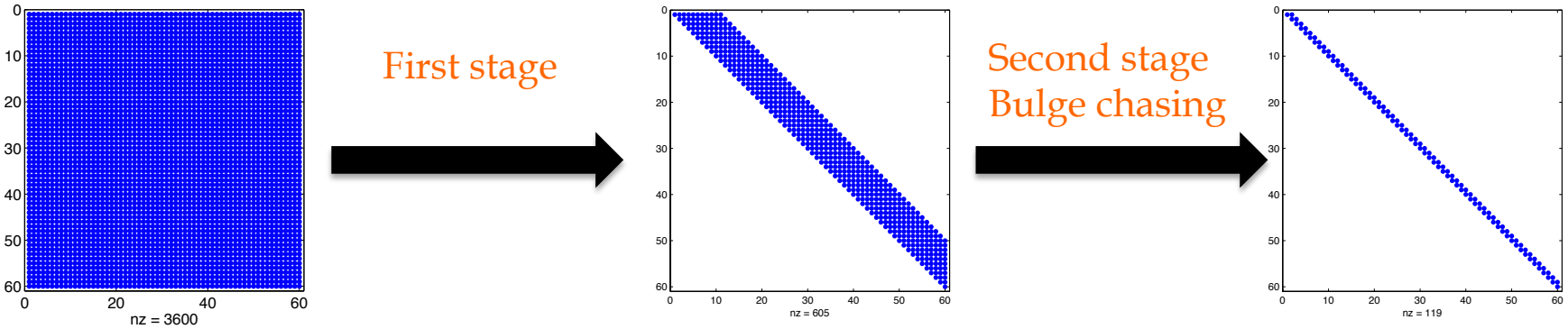
$$\approx \tfrac{10}{3}n^3(\text{gemm})_{\text{first stage}}$$

$$\text{flops} = 6\times n_b \times n^2(\text{gemv})_{\text{second stage}}$$

More Flops, original did 8/3 n ^{3}

Communication ... Bidiagonal reduction

- The only way to do better – design a new algorithm to communicate less
- Recent work on developing new "2-stage" algorithms



$$\textbf{speedup} = \frac{\textbf{time of one-stage}}{\textbf{time of two-stage}}$$

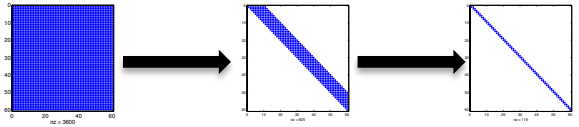
$$= \frac{4n^3/3P_{gemv} + 4n^3/3P_{gemm}}{10n^3/3P_{gemm} + 6n_b n^2/P_{gemv}}$$

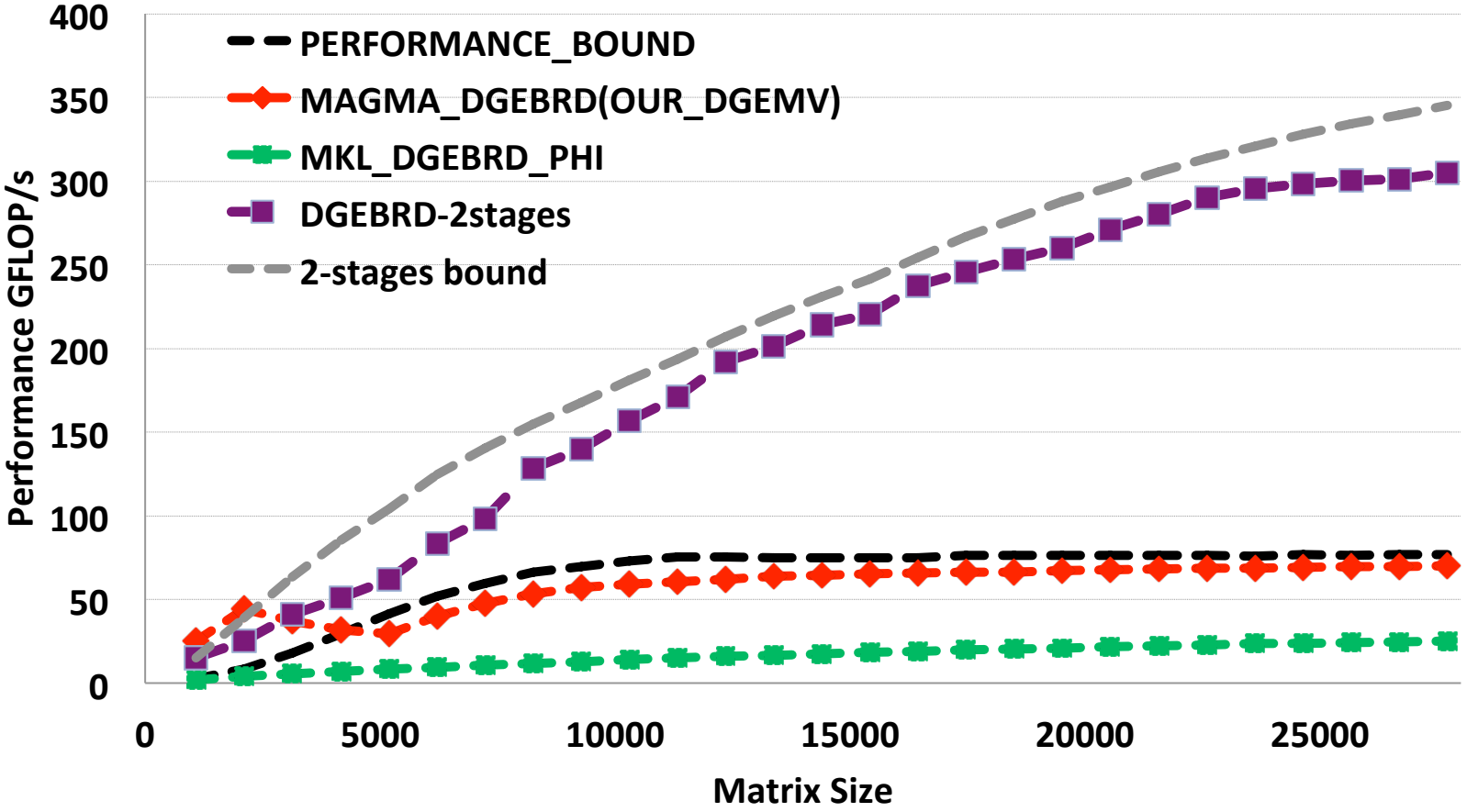
$$\Longrightarrow \frac{84}{70} \leq \textbf{Speedup} \leq \frac{84}{15}$$

$$\Longrightarrow 1.2 \leq \textbf{Speedup} \leq 5.6$$

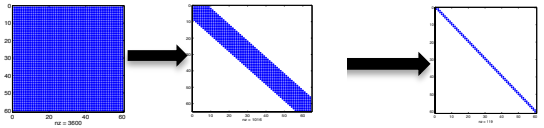
if P_{gemm} is about 20x P_{gemv} and $120 \leq n_b \leq 240$.

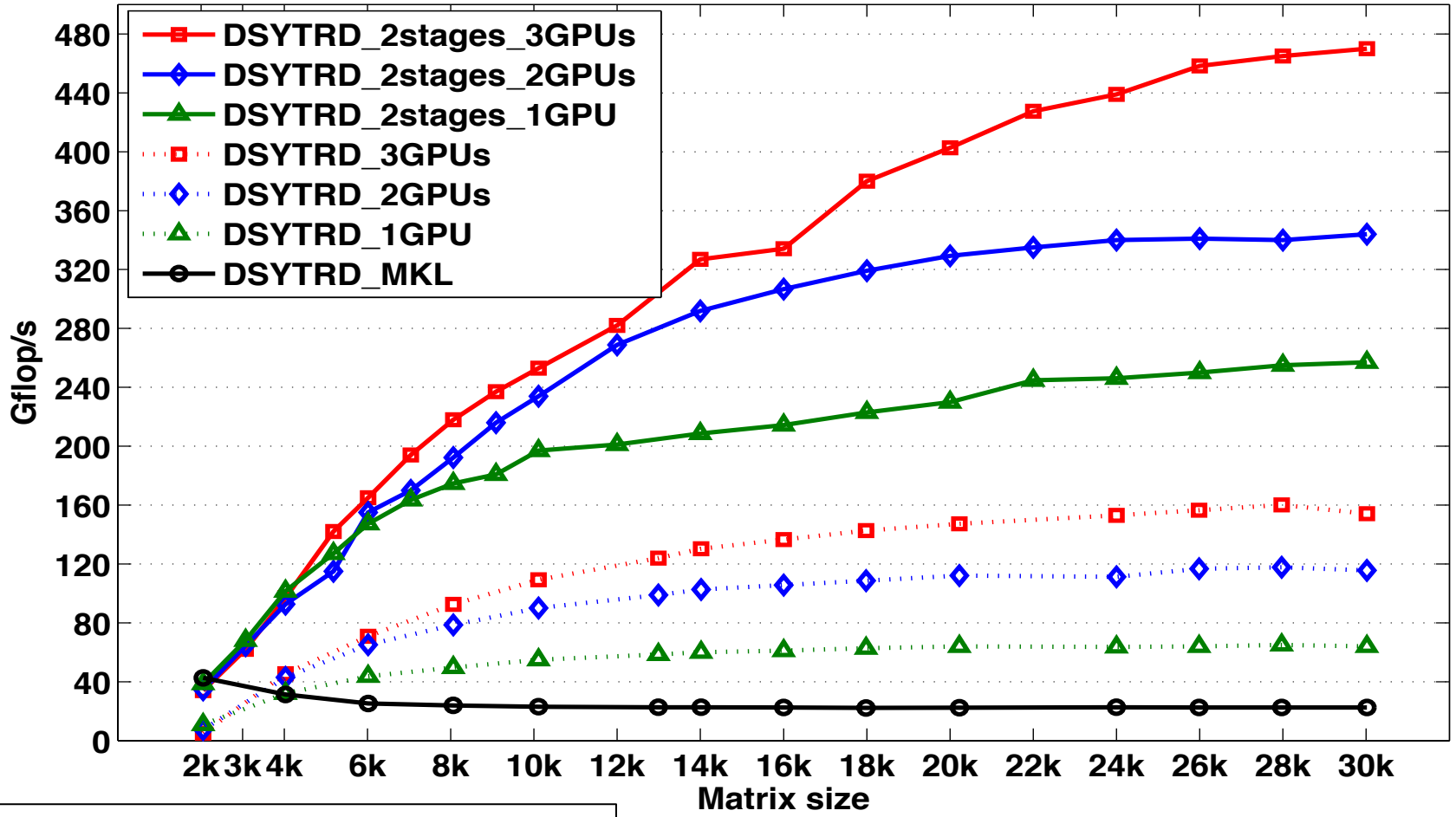
Communication ... Bidiagonal reduction





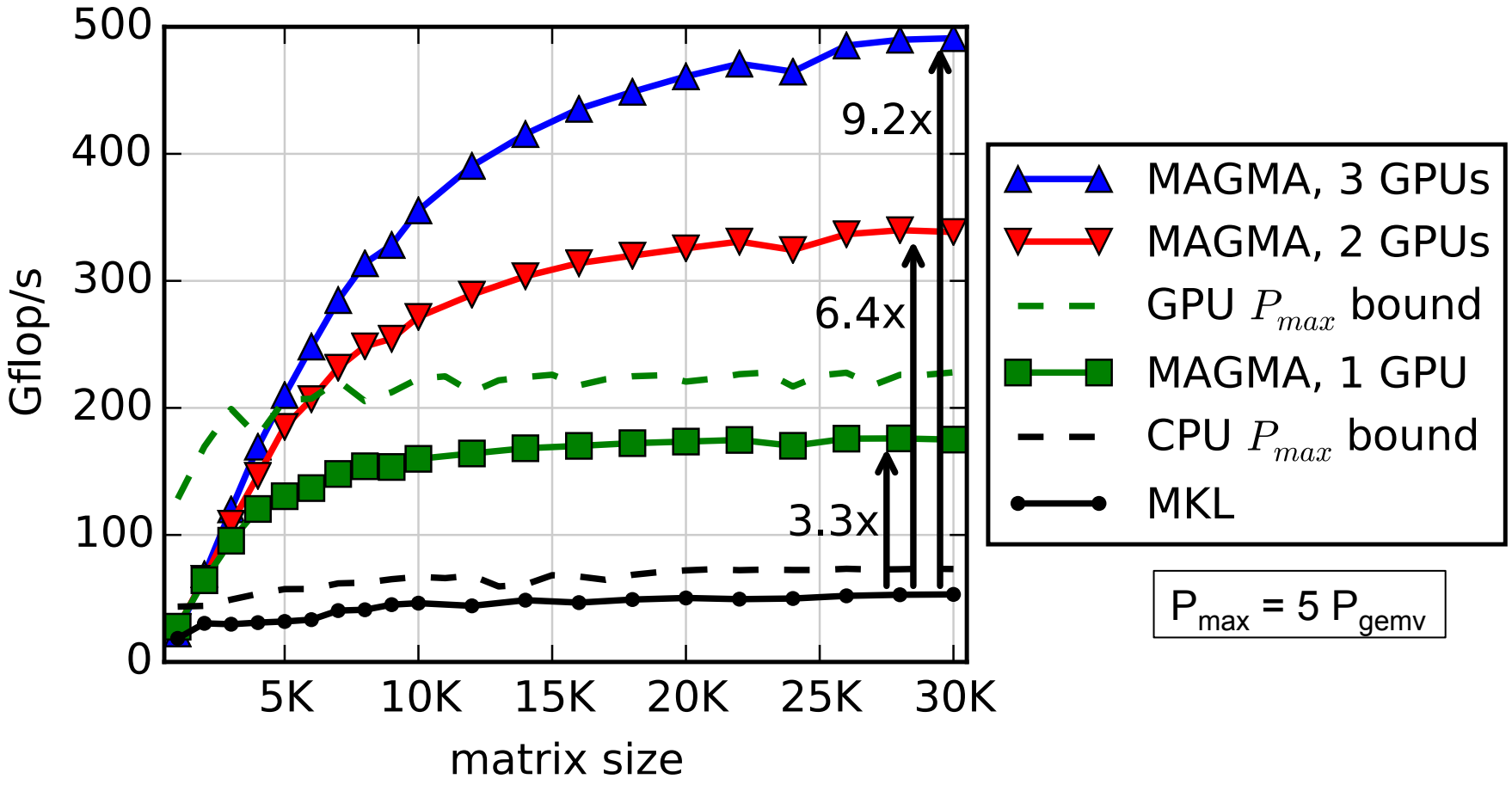
Communication ... Tridiagonal reduction





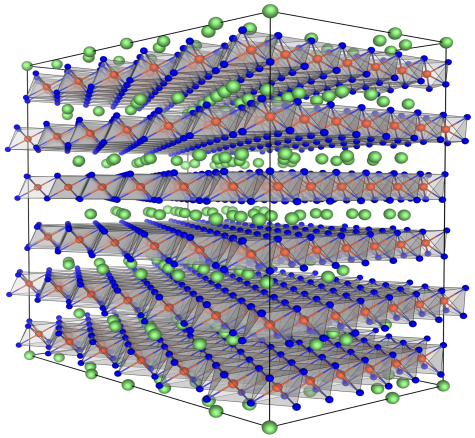
3 NVIDIA GPUs (M2090@ 1.1 GHz, 5.4 GB)
2 x 6 Intel Cores (X5660 @ 2.8 GHz, 23 GB)

Communication ... Hessenberg reduction (dgehrd)



3 NVIDIA GPUs (K40@ 876 MHz)
2 x 8 Intel Cores (E5-2670 @ 2.6 GHz)

Distributed memory systems
Symmetric eigensolvers with Thomas Schulthess et al. CSCS



	Setup H20	Solve HC=4OC	The rest	total
28x28(2R:4T) ScaLAPACK	463.7	3839.7	61.6	4365.0
28x28(2R:4T) ELPA	471.7	1199.3	60.7	1731.7
14x14(1R:8T) MAGMA	166.7	911.9	79.9	1158.5

MAGMA is **1.5 times faster** than CPU

MAGMA is **2 times more energy efficient**

Cray XC30:
- 8-core Intel Xeon E5-2670 Sandy Bridge socket
- Nvidia K20X

- **R. Solca, A. Kozhevnikov, A. Haidar, S. Tomov, T. Schulthess, and J. Dongarra,** *Efficient implementation of quantum material simulations on distributed CPU-GPU systems*. SC'15, Best Paper Award finalist, Austin, TX, November 15-20, 2015.

- **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,* International Journal of High Performance Computing Applications , vol. 28, no 2, pp. 196—209, May 2014.

- **A. Haidar, R. Solca, S. Tomov, T. Schulthess and J. Dongarra.** *Leading edge multi-GPU algorithms for generalized eigenproblems for electronic structure calculations*. International Supercomputing Conference IEEE-ISC 2013.

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
Department of Electrical Engineering and Computer Science

Conclusions

- **Presented methodology and algorithms for efficient eigensolvers on accelerator based systems**
  - **Parallelism** (task based + runtime system)
  - **Heterogeneity** (unified support for various accelerators + scheduling)
  - **Communication** (both kernel and algorithmic level; one and two-stage algorithms)
- **Significant acceleration and energy efficiency compared to standard algorithms on CPUs**

Future Directions

- **GPU-only algorithms and implementations**
- **Further improvements regarding main challenges to meet emerging application demands**
- **Optimizations for small sizes, including batched versions**
- **Embedded systems**

Collaborators and Support

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