Matrix Algebra for GPU and Multicore Architectures (MAGMA) for Large Petascale Systems

DOE Final Report
(covering period: 09/01/2010 – 08/31/2013)

Jack Dongarra

University of Tennessee

DE-FG02-10ER25987 / DE-SC0004983

03/24/2014

1. Overview

To develop software that will perform well on Petascale systems, with thousands of nodes and millions of cores, the list of major challenges is formidable:

- Dramatic escalation in the costs of intra-system communication between processors and/or levels of memory hierarchy;
- Increased hybridization of processor architectures (mixing CPUs, GPUs, etc.), in varying and unexpected design combinations;
- High levels of parallelism and more complex constraints mean that cooperating processes must be dynamically and unpredictably scheduled for asynchronous execution;
- Software will not run at scale without much better resilience to faults and far more robustness;
- New levels of self-adaptivity will be required to enable software to modulate process speed in order to satisfy limited energy budgets.

The MAGMA project represents how this historic set of challenges can be successfully attacked, in a coordinated way, for the critical area of linear algebra libraries. The goal is to create a new generation of linear algebra libraries that achieve the fastest possible time to an accurate solution on hybrid Multicore+GPU-based systems, using all the processing power that future high-end systems can make available within given energy constraints.

Achieving MAGMA's goals requires innovations in each of the five problem areas outlined above. Our efforts at the University of Tennessee achieved the goals set in all of the five areas identified in the proposal:

1. Communication optimal algorithms;
2. Autotuning for GPU and hybrid processors;
3. Scheduling and memory management techniques for heterogeneity and scale;

4. Fault tolerance and robustness for large scale systems;
5. Building energy efficiency into software foundations.

2. Activities and Findings

The University of Tennessee's main contributions, as proposed, were the research and software development of new algorithms for hybrid multi/many-core CPUs and GPUs, as related to two-sided factorizations and complete eigenproblem solvers, hybrid BLAS, and energy efficiency for dense, as well as sparse, operations. Furthermore, as proposed, we investigated and experimented with various techniques targeting the five main areas outlined in the *Overview* section above. The specific activities and findings for our main contributions are described as follows.

2.1. Communication Optimal Algorithms

The goals for our algorithmic work were 1) to experiment with established algorithms for two-sided factorizations, and 2) to research communication avoiding Krylov subspace methods. We developed a number of linear algebra algorithms, including the two-sided matrix factorizations (to upper Hessenberg, bidiagonal, and tridiagonal forms) and eigenproblem solvers related to them, and released them through the MAGMA website (see http://icl.cs.utk.edu/magma/). Similar in functionality to LAPACK but for GPU-accelerated hybrid architectures, these algorithms provide basic building blocks for higher level, heterogeneous multicore libraries. Below we list more specifics on the activities regarding these efforts.

MAGMA: We had four major MAGMA releases, providing highly optimized BLAS for NVIDIA GPUs and hybrid dense linear algebra algorithms for multicore CPUs and GPUs:

MAGMA 1.1 (released on 11/14/2011):

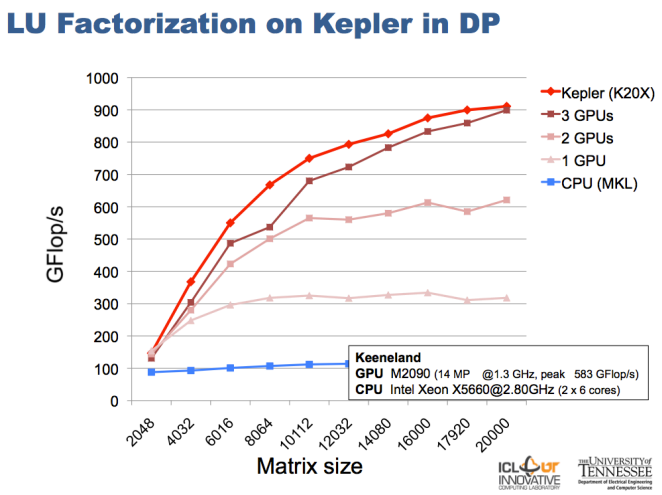
- The main new contribution of this release was added multi-GPU support. We developed two types of one-sided matrix factorizations with support for multicore and multiGPUs. The first set is hybrid LAPACK-compliant LU, QR, and Cholesky factorizations [1], and the second is tile LU, QR, and Cholesky factorizations and solvers with StarPU dynamic scheduling [2] (available at http://icl.cs.utk.edu/magma/software/);
- A new feature added is the "Non-GPU-resident" one-sided factorizations [1]. These are factorizations for large problems that do not fit entirely in the GPU memory. The algorithms developed are designed to optimize CPU-GPU communications. As a result, one can now solve large problems, through MAGMA, while retaining the high asymptotic performance for problems that fit entirely into the GPU's memory;
- More new functions were added, including matrix inversion, eigenvalue solver driver routines for the standard [3] and generalized symmetric/Hermitian eigenvalue problems [4];
- Routines were added in different CPU/GPU interfaces (e.g., used to minimize CPU-GPU data transfers when routines are called in sequence), Fortran interfaces, and more testing routines (integration with the LAPACK testing);
- An optimized GEMM for Fermi GPUs, based on autotuning, was developed and released. Although the MAGMA GEMM is used as a basis for the current GEMM implementation in CUBLAS, the autotuning techniques employed managed to generate new kernels, outperforming the previous best [5].

MAGMA 1.2 (released on 05/10/2012):

- The main contribution of this release was the added multi-GPU support for the two-sided factorizations—reduction to upper Hessenberg and tridiagonal forms. These routines are the current state-of-the-art in the area, significantly outperforming other solutions (including vendor optimized libraries). The reduction to tridiagonal, used for eigenproblem solvers for symmetric/Hermitian matrices, has two implementations. One is following a hybrid LAPACK-based approach [4], and second, following a two stage approach, incorporating and extending the latest developments on communication avoiding algorithms, for two-sided factorizations, to hybrid systems [6];
- Routines were added (both BLAS and higher level) for the multiGPU solution of generalized Hermitian-definite eigenvalue problems [4,6];
- Different and extended interfaces were added—requested by users to accommodate some of the newest features of CUDA, and to facilitate the exploration of some optimization opportunities (such as the use of multiple streams in solving many small problems).

MAGMA 1.3 (released on 11/15/2012)

- This release included a number of bug fixes, performance improvements, and support for Kepler GPUs. MultiGPU BLAS routines as needed for the symmetric eigenvalue problem solvers on multiGPUs were also released. The Figure on the right illustrates the performance achieved on the new Kepler GPUs (K20X) vs. the previous generation M2090 GPUs.



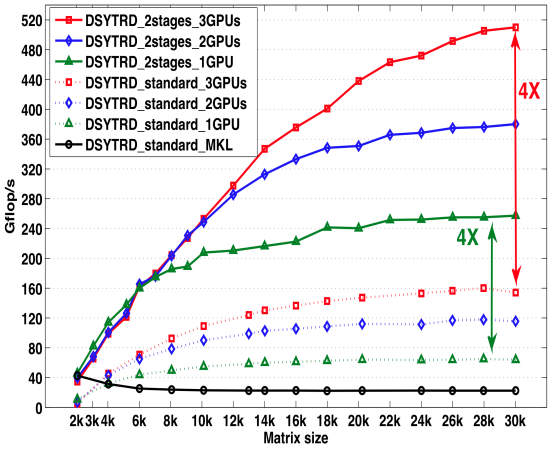
- We developed a hybrid QR with column pivoting (GEQP3). This routine is very slow on multicore architectures because it is memory bound. The algorithm that we developed is similar to the two-sided factorizations because at every iteration (every column) a Householder vector, used in the reduction of the current column, must be multiplied by the trailing matrix. This multiplication is GPU accelerated to benefit from the GPU's high bandwidth. We see accelerations compared to a multicore run similar to the accelerations for the Hessenberg reduction algorithm. The QR with pivoting was requested by users and is an important building block for certain preconditioners in sparse iterative solvers [25].

MAGMA 1.4 (released on 08/14/2013)

- We added multi-GPU Hessenberg and non-symmetric eigenvalue routines. Compared to the symmetric eigenvalue problem, where there have been recent algorithmic and software breakthroughs (see the third item below), the best performing Hessenberg reduction algorithm still has a computational component that is memory-bound. In MAGMA, this component is computed on the GPU, using its high-bandwidth. The

performance of the hybrid algorithm, using a single GPU, can outperform the latest multicore CPU systems by an order of magnitude. In addition, we developed a version for multiple GPUs and demonstrated that the computation scales.

- We developed and released panel factorizations for LU, QR, and Cholesky entirely on the GPU. The new kernels are written entirely in CUDA and show 2x to 3x performance improvement for tall and skinny panels, compared to the hybrid algorithms in the MAGMA library.
- The main new components of this release were the multi-GPU symmetric eigenvalue routines using the one-stage approach, and the single and multi-GPU symmetric eigenvalue routines for the two-stage approach. The one-stage approach follows the standard algorithm from LAPACK, and it has a memory bound component [3]. The figure on the right demonstrates the performance on up to 3 GPUs [4]. It scales, but, as shown, is about four times slower than the two-stage approach that eliminates the memory bound component [6, 16, 17].



In addition to the main MAGMA efforts, we released **clMAGMA 0.1 Beta** (on 04/04/2012) and clMAGMA 1.0 (on 10/24/2012; available at http://icl.cs.utk.edu/magma/software/index.html). clMAGMA is an OpenCL port of MAGMA, intended for a single GPU, and supports AMD GPUs. Included are routines for the LU, QR, and Cholesky factorizations, and the linear solvers based on them, eigen- and singular-value problem solvers, matrix inversion routines, and orthogonal transformations routines (in all four main precisions S/D/C/Z). A **MAGMA** port has also been developed for Intel MIC architectures. MAGMA MIC 1.0 was released on 05/03/2013 and included the main one- and two-sided factorizations and solvers for both single and multiple MICs.

We developed, within the MAGMA framework, a class of communication-avoiding algorithms for solving general dense linear systems on CPU/GPU parallel machines [14]. In particular, we studied several solvers for the solution of general linear systems where the main objective is to reduce the communication overhead due to pivoting. Considered were two algorithms for the LU factorization on hybrid CPU/GPU architectures. The first one is based on partial pivoting, and the second uses a random preconditioning of the original matrix to avoid pivoting. We introduced a solver where the panel factorization is performed using a communication-avoiding pivoting heuristic, while the update of the trailing submatrix is performed by the GPU. We provided performance comparisons for these solvers on current hybrid multicore-GPU parallel machines [14].

Further, we researched the feasibility of using hybrid systems for sparse matrix operations. In particular, we developed and analyzed the potential of **asynchronous relaxation methods** on GPUs [7], and moreover, their application as smoothers in multigrid methods [8]. To pave the road for the efficient use of large peta-scale systems, challenges related to the established notion that "*data movement, not FLOPS, is the bottleneck to performance*" must be resolved. Our work was in this direction—we designed block-asynchronous relaxation

methods and multigrid smoothers that perform more flops in order to reduce synchronization, and hence data movement. We showed that the extra flops are done for "free," while synchronization was reduced and the convergence properties of multigrid with classical smoothers like Gauss-Seidel could be preserved.

2.2. Autotuning for GPU and hybrid processors

We developed an autotuning framework called Automatic Stencil TuneR for Accelerators (ASTRA) [5]. ASTRA's prototype was developed on the Fermi architecture and used to produce fast BLAS-compliant GEMM kernels, covering all precisions (single, double / real, complex) and all cases of transposed inputs (A: Trans, No- Trans, ConjTrans; B: Trans, NoTrans, ConjTrans). At the time of ASTRA's conception, the Fermi architecture had already been available for more than a year, and GEMM kernels in CUBLAS and MAGMA were already delivering very good performance. As a result, ASTRA produced only marginal performance improvements in most cases, with the most noticeable effect on the double-complex (ZGEMM) kernels, where the performance was increased from ~300 GigaFLOPS to ~340 GigaFLOPS (ca. 13%). However, the true motivation for ASTRA was the capability to quickly deploy fast GEMM kernels, as well as other BLAS kernels, when a new architecture becomes available. Therefore, the availability of the Kepler architecture creates the first true opportunity for validation of the ASTRA methodology. In particular, we automatically produced GEMM kernels for the GTX 680 Kepler GPU in the four main precisions that outperform or are competitive with their counterparts from CUBLAS [10]. The framework is currently used to produce optimized kernels on demand, when linear algebra kernels on problems of particular sizes are needed, but not well optimized in CUBLAS.

2.3. Scheduling and memory management techniques for heterogeneity and scale

We developed a parallel programming model and scheduling using the Directed Acyclic Graphs (DAGs) approach. In particular, this approach uses run-time systems to schedule tasks dynamically, supporting massive parallelism, and applying common optimization techniques to increase throughput. Examples are the QUARK scheduler from PLASMA, and StarPU (see http://runtime.bordeaux.inria.fr/StarPU/). The algorithms that we had developed for these run-time systems are now available in MAGMA 1.4 (starting from MAGMA 1.1).

Further, we explored a new approach to utilizing all CPU cores and all GPUs on heterogeneous multicore and multi-GPU systems to support dense matrix computations efficiently. The main idea is that we treat a heterogeneous system as a distributed- memory machine, and use a heterogeneous multi-level block cyclic distribution method to allocate data to the host and multiple GPUs to minimize communication. We designed heterogeneous algorithms with hybrid tiles to accommodate the processor heterogeneity, and introduced an auto-tuning method to determine the hybrid tile sizes to attain both high performance and load balancing. We have also implemented a new runtime system and applied it to the Cholesky and QR factorizations. Our approach is designed to achieve four objectives: a high degree of parallelism, minimized synchronization, minimized communication, and load balancing. Our experiments on a compute node (with two Intel Westmere hexa-core CPUs and three Nvidia Fermi GPUs), as well as on up to 100 compute nodes on the Keeneland system, demonstrate the great scalability, good load balancing, and efficiency of our approach [11].

Finally, we developed and deployed software tools for hybrid architectures to ensure productivity. Under the Matrices Over Runtime Systems @ Exascale (MORSE) project, we designed dense and sparse linear algebra methods for petascale and exascale systems. MORSE targets high level

of abstraction and strong collaboration between research groups in linear algebra and runtime systems. MORSE uses the StarPU runtime system for accelerator-based manycore architectures to express parallelism through sequential-like codes. The work is essential in setting up standards and a software development methodology that ensures productivity. We released MORSE 1.0 on November 19, 2013. In this release, we illustrated our methodology on three classes of problems – dense linear algebra, sparse direct methods and fast multipole methods [24].

2.4. Fault tolerance and robustness for large scale systems

We developed a soft error recovery approach to hybrid systems. This work is important because fault tolerance has become a more serious concern to GPUs compared to the period when the GPUs were used exclusively for graphics applications. Using GPUs and CPUs together in a hybrid computing system increases flexibility and performance, but also increases the possibility of the computations being affected by soft errors, for example, in the form of bit flips. We proposed a soft error resilient algorithm for the QR factorization on such hybrid systems. Our contributions include:

1) A checkpointing and recovery mechanism for the left-factor Q whose performance is scalable on hybrid systems;

2) Optimized Givens rotation utilities on GPGPUs to efficiently reduce an upper Hessenberg matrix to an upper triangular form for the protection of the right factor R; and

3) A recovery algorithm based on QR update on GPGPUs.

Experimental results show that our fault tolerant QR factorization can successfully detect and recover from soft errors in the entire matrix, with little overhead on hybrid systems with GPGPUs [12].

We extended the approach to the area of solvers for eigenvalue problems. In particular, we designed and implemented a soft error resilient Hessenberg reduction algorithm on GPU based hybrid platforms [20]. Our design employs an algorithm based fault tolerance technique, diskless check-pointing and a reverse computation. We detect and correct soft errors on-line without delaying the detection and correction to the end of the factorization. By utilizing idle time of the CPUs and overlapping both host side and GPU side workloads, we minimize the observed overhead. Experiment results validated our design philosophy. Our algorithm introduces less than 2% performance overhead compared to the non-fault tolerant hybrid Hessenberg reduction algorithm.

2.5. Building energy efficiency into software foundations

We investigated the energy efficiency of different hardware systems. This includes setting up an environment and performing various experiments to measure power consumption for different dense linear algebra algorithms on GPUs. Energy and power density concerns in modern processors have led to significant computer architecture research efforts in power-aware and temperature-aware computing. With power dissipation becoming an increasingly vexing problem, power analysis of GPUs and its components has become crucial for hardware and software system design. We developed a technique for a coordinated measurement approach that combines real total power measurement and per-component power estimation. To identify power consumption accurately, we introduced the Activity-based Model for GPUs (AMG), from which we identify activity factors and power for microarchitectures on GPUs that will help in analyzing power tradeoffs of one component versus another using microbenchmarks. The key challenge

addressed is real-time power consumption, which can be accurately estimated using NVIDIA's Management Library (NVML) through pthreads. We validated our model using a Kill-A-Watt power meter and the results are accurate within 10%. The resulting Performance Application Programming Interface (PAPI) NVML component offers real-time total power measurements for GPUs. We analyzed MAGMA's high-level one and two-sided factorizations, as well as low Level 2 BLAS and Level 3 BLAS kernels [13].

2.6. Communication-avoiding algorithms

We developed several communication-avoiding techniques and algorithms. In [15], we developed a randomization technique for GPUs that enables us to avoid pivoting and then reduce the amount of communication in the LU factorization. We showed that a randomization transformation can be performed at a very affordable computational price while providing us with a satisfying accuracy when compared to partial pivoting. This random transformation, called Partial Random Butterfly Transformation (PRBT), is optimized in terms of data storage and flops count. We proposed a solver where PRBT and the LU factorization with no pivoting take advantage of the latest generation of hybrid multicore/GPU machines.

In the area of solvers for eigenvalue problems, we designed new algorithms that increase the computational intensity of the major compute kernels and reduce synchronization and data transfers between GPUs. This increases the number of flops, but the increase is offset by the more efficient execution and reduced data transfers [16, 17, 19, 23]. Our performance results are the best available, providing an enormous performance boost compared to current state-of-the-art solutions. In particular, our software scales up to 1070 Gflop/s using 16 Intel E5-2670 cores and eight M2090 GPUs, compared to 45 Gflop/s achieved by the optimized Intel Math Kernel Library (MKL) using only the 16 CPU cores.

3. Outreach activities

- ***Mini-symposiums MS 44 and MS 63 at SIAM CSE13***
  February 25—March 1, 2013, Boston, MA
  Title: Computational Challenges for Large Scale Heterogeneous Applications
  Presenters: Stan Tomov and Azzam Haidar

- ***Tutorial at ISC'13***
  June 16-20, 2013, Leipzig, Germany
  Title: Dense Linear Algebra with MAGMA & PLASMA
  Presenter: Jack Dongarra

- ***Tutorial at ICS'13***
  June 10-14, 2013, Eugene, Oregon
  Title: DLA on Multicore with Accelerators
  Presenter: Piotr Luszczek and Aurelien Butelier

- ***Tutorial at SC'12***
  November, 2012, Salt Lake City, Utah
  Title: Linear Algebra Libraries for High-Performance Computing: Scientific Computing with Multicore and Accelerators
  Presenter: Jack Dongarra

- ***DOD CREATE Developers' Review***
  February 28, 2012, Savannah, GA, USA

Title: The Future of Computing: Software Libraries
Presenter (Keynote presentation): S. Tomov

- ***Workshop on "Electronic structure calculation Methods on Accelerators"***
  February 5—8, 2012, Oak Ridge National Laboratory, TN, USA
  Title: Matrix Algebra on GPU and Multicore Architectures
  Presenter: S. Tomov

- ***Keeneland Workshop***
  February 20—21, 2012, Atlanta, GA, USA
  Title: MAGMA Tutorial
  Presenter: M. Gates

- ***Workshop on "Electronic structure calculation Methods on Accelerators"***
  February 5—8, 2012, Oak Ridge National Laboratory, TN, USA
  Title: Matrix Algebra on GPU and Multicore Architectures
  Presenter: S. Tomov

- ***Third Uio-MSU-ORNL-UT School "Nuclear Physics: The computational quantum many-body problem"***
  January 23, 2012, Oak Ridge National Laboratory, TN, USA
  Title: MAGMA: Challenges and Approaches for Heterogeneous HPC
  Presenter: S. Tomov

- ***Titan Summit 2011***
  August 15–17, 2011, Oak Ridge National Laboratory, TN, USA
  Title: MAGMA – LAPACK for HPC on Heterogeneous Architectures
  Presenter: S. Tomov

We taught the use of CUDA and OpenCL for HPC on GPUs in the "Scientific Computing for Engineers" class (CS594), offered at UTK for the Spring 2012 semester.

Further, we participated and provided expertise related to linear algebra and GPU computing through the MAGMA forum (on the MAGMA homepage http://icl.cs.utk.edu/magma/).

4. Participants partially funded from this effort

2 Graduate Students

2 Post-docs

4 Research Scientists

5. Publications for this funding period

1. I. Yamazaki, S. Tomov, and J. Dongarra, ***"One-sided dense matrix factorizations on a multicore with multiple GPU accelerators",*** Proc. of ICCS 2012, Omaha, Nebraska, June 4-6, 2012.

2. E. Agullo, C. Augonnet, J. Dongarra, M. Faverge, J. Langou, H. Ltaief, and S. Tomov, ***"LU factorization for accelerator-based systems"***, Proc. of AICCSA 2011 (best paper award), Sharm El-Sheikh, Egypt, December 27-30, 2011.
3. C. Vomel. S. Tomov, and J. Dongarra, ***"Divide and Conquer on Hybrid GPU-Accelerated Multicore Systems"***, SIAM Journal on Scientific Computing, Vol.34, No.2, April 12, 2012.
4. T. Dong, J. Dongarra, S. Tomov, I. Yamazaki, T. Schulthess, and R. Solca, ***"Symmetric dense matrix-vector multiplication on multiple GPUs and its application to symmetric dense and sparse eigenvalue problems"***, ICL Technical report (submitted to SC'12).
5. J. Kurzak, S. Tomov, and J. Dongarra, ***Autotuning GEMMs for Fermi***. Technical Report UT-CS-11-671, Electrical Engineering and Computer Science Department, University of Tennessee, 2011. (accepted to IEEE TPDS).
6. A. Haidar, S. Tomov, J. Dongarra, R. Solca, and T. Schulthess, ***"A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks"***, ICL Technical report (submitted to SC'12).
7. Anzt, H., Tomov, S., Dongarra, J., Heuveline, V. **"A Block-Asynchronous Relaxation Method for Graphics Processing Units,"** *University of Tennessee Computer Science Technical Report*, UT-CS-11-687 / LAWN 258, November 30, 2011.
8. Anzt, H., Tomov, S., Gates, M., Dongarra, J., Heuveline, V. **"Block-asynchronous Multigrid Smoothers for GPU-accelerated Systems,"** UT-CS-11-689, December 6, 2011 (Proc. of ICCS 2012, Omaha, Nebraska, June 4-6, 2012).
9. Anzt, H., Tomov, S., Dongarra, J., Heuveline, V. **"A Block-Asynchronous Relaxation Method for Graphics Processing Units,"** *University of Tennessee Computer Science Technical Report*, UT-CS-11-687 / LAWN 258, November 30, 2011.
10. Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, and Jack Dongarra, ***Preliminary Results of Autotuning GEMM Kernels for the NVIDIA Kepler Architecture – GeForce GTX 680***, LAPACK Working Note 267, 04/2012.
11. F. Song, S. Tomov, and J. Dongarra, ***"Enabling and Scaling Matrix Computations on Heterogeneous Multi-Core and Multi-GPU Systems",*** Proc. of International Conference on Supercomputing (ICS 2012), San Servolo Island, Venice, Italy, June 25—29, 2012.
12. P. Du, P. Luszczek, S. Tomov, and J. Dongarra, ***"Soft Error Resilient QR Factorization for Hybrid System,"*** UT-CS-11-675 (also LAPACK Working Note #252), ICL-CS-11-675, July 1, 2011.
13. K. Kaisichayanula, D. Terpstra, P. Luszczek, S. Tomov, S. Moore, and G. Peterson, ***"Power Aware Computing on GPUs,"*** ICL Technical report, 05/2012.
14. M. Baboulin, S. Donfack, J. Dongarra, L. Grigori, A. Remy, S. Tomov, ***"A class of communication-avoiding algorithms for solving general dense linear systems on CPU/GPU parallel machines,"*** Proc. of ICCS 2012, Omaha, Nebraska, June 4-6, 2012.
15. M. Baboulin, J. Dongarra, J. Hermann, and S. Tomov, ***"Accelerating linear system solutions using randomization techniques,"*** ACM Transactions on Mathematical Software (TOMS), Vol. 39, No 2, February 2013.
16. A. Haidar, M. Gates, S. Tomov, and J. Dongarra, ***"Toward a scalable multi-GPU eigensolver via compute-intensive kernels and efficient communication,"*** ICS'2013, Eugene, Oregon, USA.
17. A. Haidar, R. Solca, M. Gates, S. Tomov, T. Schulthess, and J. Dongarra, ***"Leading edge hybrid multi-GPU algorithms for generalized eigenproblems in electronic structure calculations"***, ISC'13, Leipzig, Germany.

18. H. Anzt, S. Tomov, J. Dongarra, and V. Heuveline, ***"A block-asynchronous relaxation method for graphics processing units"*** , J. of Parallel and Distributed Computing, May 21, 2013 (accepted).
19. T. Dong, J. Dongarra, S. Tomov, and I. Yamazaki, ***"Tridiagonalization of a symmetric dense matrix on a GPU cluster,"*** AsHES'13 IPDPS workshop, Boston, USA.
20. Y. Jia, P. Luszczek, J. Dongarra, ***"Transient Error Resilient Hessenberg Reduction on GPU-based Hybrid Architectures,"*** UTK Technical Report UT-CS-13-712 (LAWN #279), June 14, 2013.
21. S. Donfack, S. Tomov, and J. Dongarra, ***"Dynamically balanced synchronization-avoiding LU factorization with multicore and GPUs,"*** UTK Technical Report UT-CS-13-713, July 11, 2013.
22. T. Dong, V. Dobrev, T. Kolev, R. Rieben, S. Tomov, and J. Dongarra, ***"Hydrodynamic Computation with Hybrid Programming on CPU-GPU Clusters***," UTK Technical Report UT-CS-13-714, July 11, 2013.
23. T. Dong, J. Dongarra, T. Schulthess, R. Solca, S. Tomov, and I. Yamazaki, ***"Tridiagonalization of a dense symmetric matrix on multiple GPUs and its application to symmetric eigenvalue problems***," J. of Concurrency and Computation: Practice and Experience, July, 2013 (submitted).
24. ***"Matrices Over Runtime Systems @ Exascale,"*** SC'12 Poster, Salt Lake City, Utah, November 10-16, 2012.
25. I. Yamazaki, A. Napov, S. Tomov, and J. Dongarra, ***"Computing Low-Rank Approximation of Dense Submatrices in a Heirarchically Semiseparable Matrix and its GPU Acceleration***", UTK Technical Report, August, 2013.