Cover Page

U.S. Department of Energy Office of Science

Architecture-Aware Algorithms for Scalable Performance and Resilience on Heterogeneous Architectures

DE-SC0003852

Jack Dongarra

University of Tennessee

FINAL REPORT

Reporting Period: 3/15/12 to 3/14/13

1 – Overview

There is a widening gap between the peak performance of high performance computers and the performance realized by full applications. Over the next decade, extreme-scale systems will present major new challenges to software development that could widen the gap so much that it prevents the productive use of future DOE Leadership computers due to:

- Heterogeneous processor architectures (mixing CPUs, GPUs, etc.) in varying and unexpected design combinations;
- High levels of parallelism due to multi-core processors and more complex constraints means that cooperating processes must be dynamically and unpredictably scheduled for asynchronous execution;
- Increase in system fault rates requiring algorithms to be resilient beyond just checkpoint/restart;
- Decreasing communication bandwidth and increasing latency between processors and between levels of the memory hierarchy;
- Performance, resilience, and power needs that require new levels of algorithm dynamic adaptability.

Some extreme-scale challenges can only be addressed by a cooperative math and computer science effort. The goal of the Extreme-scale Algorithms & Software Institute (EASI) is to close the "application-architecture performance gap" by exploring algorithms and runtime improvements that will enable key science applications to better exploit the architectural features of DOE extreme-scale systems.

For the past year of the project, our efforts at the University of Tennessee have concentrated on, and made significant progress related to, the following high-level EASI goals:

- Develop multi-precision and architecture-aware implementations of Krylov, Poisson, Helmholtz solvers, and dense factorizations for heterogeneous multi-core systems;
- Explore new methods of algorithm resilience, and develop new algorithms with these capabilities;
- Develop runtime support for adaptable algorithms that are dealing with resilience, scalability;
- Distribute the new algorithms and runtime support through widely used software packages;
- Establish a strong outreach program to disseminate results, interact with colleagues and train students and junior members of our community.

2 – Activities and findings

The University of Tennessee's main contribution to EASI for this period, as proposed, was:

- The extension of optimal communication algorithms to include dense operations needed for iterative methods and support preconditioning;
- The development of production capabilities in new distributions of ScaLAPACK through the PLASMA and MAGMA projects;
- Development of Heterogeneous Programming APIs and their dissemination in the DOE scientific computing community through workshops, tutorials, and mini-symposia.

2.1 – Extension of optimal communication algorithms to include dense operations needed for iterative methods and support preconditioning

The hybridization methodology that we have developed for the dense linear algebra was used to develop Krylov-type iterative linear system solvers as well as eigensolvers based on subspace diagonalization techniques. The hybrid iterative solvers support preconditioners and various CPU-GPU communication optimization techniques through a combination of appropriate heterogeneous scheduling and algorithm selection.

2.1.1 – Krylov-subspace iterative solvers

We developed hybrid CPU-GPU implementations for CG, BiCGStab, GMRES, and LOBPCG. Different versions, implementing various optimizations, are available. Support is provided for arithmetic in the four main precisions – double complex, single complex, double real, and single real. Routines in mixed-precision arithmetic, using inner-outer iterations are also available. The inner loop is in lower precision and is used as a preconditioner.

2.1.2 – Dense GPU operations for iterative methods

GPUs represent a formidable challenge to the developers of performance-optimized numerical kernels because GPUs are far more complex in terms of concurrency, and far more inscrutable in terms of resource scheduling. Hand tuning numerical kernels for such devices is unfeasible, and autotuning approaches that seek to use them must be dramatically improved in order to become both generally applicable and sustainable.

We built in our software mechanisms to automatically tune for application-specific use. We developed a framework for tuning GEMM that was successfully used on Fermi GPUs to discover faster versions, i.e., than available in CUBLAS [3]. The framework was used on Kepler to automatically discover new implementations that outperform those currently available. Of interest were various kernels for sparse direct solvers and FEMs where many, but small, basic linear algebra operations must be optimized. We achieved in various cases two to three times better results than corresponding kernels in CUBLAS.

To provide support for various applications, we teamed up with a number of application developers to map their sparse computations to CUDA-based architectures.

One application area targeted was in electronic structure calculations where we developed a generalized eigensolver [2,10]. Packages in this application area, like ABINIT and Quantum-Espresso, now support GPU acceleration through MAGMA.

Further, we developed a new class of eigensolvers and SVD that significantly outperform existing solutions. These solvers are the basis of many computational applications in mechanics, chemistry, big data, etc., and have spurred high interest and collaborations to include our solvers in packages like MATLAB (with MathWorks), Abaqus (with SIMULIA), and MSC Nastran and/or Marc (MSC Software).

Collaboration with CASC, LLNL led to the GPU development and acceleration of compressible hydrodynamics simulation codes using high order Finite Element Methods (in a moving Lagrangian frame) [11]. A major accomplishment is that we further established the appeal of high order FEMs, which despite their better approximation properties, are often avoided due to their high computational cost. GPUs, as we have shown, have the potential to make them the method of choice, as the increased computational cost is also localized, e.g., cast as Level 3 BLAS, and thus can be done very efficiently (close to “free” relative to the usual overheads inherent in sparse computations).

For computational electromagnetics applications, we worked on the development of "HP Numerical Libraries with Support for Low-Rank Matrix Computations." We developed the main GPU-accelerated kernels for sparse direct multi-frontal solvers.

2.1.3 – Support for preconditioners and mixed precision arithmetic

The iterative solvers developed support preconditioning. As already mentioned, inner-outer loop solvers were also developed where the inner loop is considered as a preconditioner, and in general can be used in lower precision. Thus, the overall solver uses mixed precision arithmetic.

BLAS kernels to support low-rank matrix approximations were also developed and can be used as preconditioners. Proof-of-concept developments were demonstrated on computational electromagnetics applications utilizing frequency-domain, hybrid finite element/boundary integral (FE-BI) techniques.

Further, we developed a set of asynchronous methods that perform more FLOP/s in order to reduce synchronization, and hence data movement. In particular, we developed asynchronous iteration algorithms as smoothers in multigrid methods, and have shown that the extra FLOP/s are done for "free," while synchronization is reduced and the convergence properties of multigrid with classical smoothers like Gauss-Seidel can be preserved. Since the parallelization potential of the classical smoothers like SOR and Gauss-Seidel is usually very limited, replacing them, for example, by our weighted block- asynchronous smoothers (see the award winning [9], as well as [4,5]) may have a considerable impact on the overall multigrid performance. Due to the increase of heterogeneity in today's architecture designs, the significance and the need for highly parallel asynchronous smoothers is expected to grow.

2.1.4 – Optimal communication through scheduling and new algorithms

We developed several techniques and the high-performance linear system solvers and eigenproblem solvers based on them. The new developments are based on the idea of trade-offs between communication and extra computation [4,5,6,9,10,13]. The development of new algorithms around this concept of trade-offs is motivated by an exponentially increasing gap over the years of processor speed vs. memory and interconnect speeds that has even established the notion that "FLOP/s are free" compared to the cost of fetching data. This concept, and the new algorithms based on it, will become increasingly important for current and up-coming hardware. We have demonstrated it in a number of application areas, ranging from dense linear and eigenproblem solvers to FEM-based simulations and sparse solvers.

To further optimize CPU-GPU communications, while retaining ease of programming, we coupled the new algorithms with runtime systems that properly schedule work and communications between CPUs and GPUs. Static schedulers were originally developed and specifically designed to optimize communications. Their optimal scheduling mechanisms were then incorporated into dynamic schedulers. In a particular example, we developed a number of Level 3 BLAS algorithms for multiGPUs that were used in higher-level algorithms—linear solvers and eigenproblem solvers. We obtain a perfect scalability due to an innovative hierarchical multi-GPU communication model to optimize communication for multi-GPUs, which can be applied in general, beyond the scope of the algorithms developed [2].

Related to schedulers, we developed several runtime systems, including QUARK and QUARK-D. Tile dense linear algebra algorithms that use them were also developed. Independent of the scheduler selected, the execution of a DAG is dynamic in the sense that tasks are scheduled (dynamically by the runtime system) on different computational units (properly selected; when ready to compute, or pipelined, etc). Although the work is scheduled dynamically, if we concentrate on a particular data-tile, our DAG scheduling mechanisms guarantee that the tasks modifying this particular tile will be applied in a particular order, and thus guaranteeing

consistency of the computation from run to run. The scheduling mechanisms in QUARK and QUARK-D use ideas from task superscalar execution in order to take a serial code as input, determine inter-task dependencies dynamically, and execute the tasks using parallelism.

Another scheduler that we have used, and developed algorithms using it, is StarPU. StarPU can dynamically decide if certain tasks must be executed on a CPU or GPU. Currently, there are no mechanisms incorporated in StarPU that will guarantee consistent scheduling from one run to another. Because of differences in the arithmetic of CPUs and GPUs, the results may not to be bit-wise consistent from run-to-run (e.g., the result of a dot product, done sequentially on a CPU core, will be bit-wise different than the parallel dot product on a GPU). Enforcing specific scheduling in this case would be against the design principles of StarPU. Nevertheless, consistency can be verified from run to run by forcing the use of only CPUs or only GPUs.

2.2 – Software development – heterogeneous programming APIs

We continue our work developing runtime environments that focus on supporting linear algebra applications. QUARK (QUeuing and Runtime for Kernels) is our runtime environment for the dynamic scheduling and execution of applications that consist of precedence-constrained kernels on multicore, multi-socket shared memory systems. During the last year we focused on analyzing the performance of QUARK on large-scale platforms and while using smaller tasks [14]. We also experimented with extending QUARK to distributed memory platforms under the QUARK-D project.

- **QUARK**
  QUARK is a dynamic scheduler that takes serial code as input, with calls to kernel routines. These kernels are related by data dependencies, which are used to create precedence constrained DAG of tasks. These tasks are then executed by the QUARK runtime using ideas from superscalar scheduling, in order to enable parallel execution of the tasks. During this last year, we have studied various bottlenecks in the QUARK implementation to improve performance at smaller task sizes [14]. QUARK is included within the PLASMA distribution, but is also distributed as an independent software package.
- **QUARK-D**
  The QUARK-D project extends QUARK to distributed memory environments by adding components to enable distributed scheduling, a decentralized data coherency protocol, and an engine for asynchronous data transfers [15]. The end result is a runtime engine that can be programmed using loop-structured serial code so that it executes on distributed memory resources transparently. The engine manages all the complications of data transfers, managing copies of data and making scheduling decisions without requiring the users involvement. Currently, QUARK-D remains an experimental project, but the ideas here will be wrapped into our distributed code in the future.

2.3 – Software development – new distributions of PLASMA and MAGMA

We developed a number of linear algebra algorithms and released them through the PLASMA (see http://icl.cs.utk.edu/plasma/) and MAGMA (see http://icl.cs.utk.edu/magma/) websites. Similar in functionality to LAPACK, these libraries provide basic building blocks for higher-level, heterogeneous multicore libraries. Below we list more specifics on the activities regarding these libraries.

2.3.1 – PLASMA

This year we had two major PLASMA releases that provide a number of tiled and communication-avoiding dense linear algebra algorithms for multicore architectures. The most important new features for each release are as follows:

Release 2.4.6 – August 21, 2012

This release contains new algorithms to compute eigenvectors in symmetric/hermitian cases as well as in generalized problems. In particular, the package contains the following updates:

- Eigenvectors support in eigensolvers for symmetric/hermitian problems and generalized problems;
- Add support of Frobenius norm;
- Release the precision generation script used to generate the precision s, d, and c from z, as well as ds from zc;
- Add all Fortran90 for mixed precision routines;
- Add all Fortran90 wrappers to tile interface and asynchronous interface. Thanks to NAG for providing those wrappers;
- Add 4 examples with Fortran90 interface provided by NAG;
- Add support for all computational functions in F77 wrappers;
- Fix memory leaks related to fake dependencies in dynamically scheduled algorithms;
- Fix interface issues in eigensolvers routines;
- Fixed returned info in PLASMA_zgetrf function;
- Fixed bug with matrices of size 0.

Release 2.5 – November 14, 2012

This release contains new algorithms to compute condition estimators for general and positive definite cases. In particular, the package contains the following updates:

- Introduce condition estimators for General and Positive Definite cases (xGECON, xPOCON);
- Fix recurring with LAPACK release number in plasma-installer;
- Fix out-of-order computation in QR/LQ factorization that was causing numerical issues with dynamic scheduling;
- Fix many comments in the Doxygen documentation;
- Correct some performance issues with in-place layout translation.

2.3.2 – MAGMA

This year we had three significant MAGMA releases, providing highly optimized BLAS for NVIDIA GPUs and hybrid dense linear algebra algorithms for multicore and GPUs:

Release 1.2 – May 15, 2012

This release provided the following new functionalities:

- Two-stage reduction to tridiagonal form;
- Updated eigensolvers for standard and generalized eigenproblems for symmetric/Hermitian matrices;
- Tracing functions;
- Improved error checking, interfaces, and documentation;
- Extended LAPACK testing.

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).

Release 1.2.1 – June 29, 2012

Included were performance improvements on a number of routines, bug fixes, and user suggested new functionalities (through MathWorks) such as QR with pivoting, added routine interfaces, support for 64-bit integers, etc. Further detail can be found in the Release Notes (see http://icl.cs.utk.edu/projectsfiles/magma/magma-1.2.1.tar.gz)

Release 1.3 – November 14, 2012

This release included the following new developments:

- Kepler GPU support;
- Bug fixes and performance improvements;
- MultiGPU BLAS routines, as needed for eigensolvers;
- MultiGPU eigensolvers;

All MAGMA software releases are available as open software (with modified BSD license) at http://icl.cs.utk.edu/magma/software/.

3 – Outreach activities

Tutorials/workshops:

1. M. Gates, "Numerical Libraries: MAGMA," Tutorial of using MAGMA for NICS users at the HPC workshop http://www.nics.tenessee.edu/nics-rdav-hpc-workshop, March 22, 2012.
2. A. Bouteiller, P. Luszczek, "T4: DLA on Multicore with Accelerators," Tutorial for Dense Linear Algebra Libraries for the The 27th International Conference on Supercomputing (ICS2013) http://www.ics-conference.org/, June 10, 2013
3. J. Dongarra, J. Kurzak, and H. Ltaief, "Dense Linear Algebra Libraries for High Performance Computing," ISC'12 Tutorial, Hamburg, Germany, June 17, 2012.
4. V. Alexandrov, J. Dongarra, Al Geist, Christian Engelmann, "3rd Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems – ScalA," Workshop at SC12, Salt Lake City, Utah, November, 2012.
5. "Linear Algebra Libraries for High-Performance Computing: Scientific Computing with Multicore and Accelerators," Tutorial at SC12, Salt Lake City, Utah, November, 2012.
6. "Computational Challenges for Large Scale Heterogeneous Applications," Mini-symposiums MS 44 and MS 63 at SIAM CSE13, Boston, MA, February 25—March 1, 2013.
7. "MAGMA: State-of-the-art developments in high-performance linear algebra for GPUs," Mini-symposium at the GPU Technology Conference 2013, San Jose, CA, March 2013.

4 – Participants partially funded from this effort

- 1 PI
- 2 Post-Doc
- 1 Graduate Student
- 5 Computational Scientists
- 2 Technicians/Programmers

5 – Publications

1. C. Vomel, S. Tomov, and J. Dongarra, “Divide and conquer on hybrid GPU-accelerated multicore systems,” SIAM Journal on Scientific Computing, 34 (2), C70-C82, April 12, 2012.
2. A. Haidar, S. Tomov, I. Yamazaki, T. Dong, J. Dongarra, R. Solca, and T. Schulthess, “MAGMA: A Breakthrough in Solvers for Eigenvalue Problems,” ICL Technical report, GTC 2012 (CUDA Center of Excellence Achievement Finalist 2012 Award), San Jose, CA, May 16, 2012.
3. J. Kurzak, P. Luszczek, S. Tomov, and J. Dongarra, “Preliminary results on autotuning GEMM Kernels for the NVIDIA Kepler Architecture – GeForce GTX 680,” LAPACK Working Note #267, May 2012.
4. H. Anzt, S. Tomov, J. Dongarra, and V. Heuveline, “A block-asynchronous relaxation method for graphics processing units,” 21st International Heterogeneity Workshop (HCW’12), in conjunction with IPDPS’12 (LAWN 258), Shanghai, China, May 21, 2012.
5. H. Anzt, S. Tomov, M. Gates, J. Dongarra, and V. Heuveline, “Block-asynchronous Multigrid Smoothers for GPU-accelerated Systems,” Proc. of ICCS’12, Omaha, Nebraska, June 4—6, 2012.
6. M. Baboulin, S. Donfack, J. Dongarra, L. Grigori, A. Rémy, S. Tomov, “A class of communication-avoiding algorithms for solving general dense linear systems on CPU/GPU parallel machines,” Proc. of ICCS’12, Omaha, Nebraska, June 4—6, 2012.
7. F. Song and J. Dongarra, “A scalable framework for Heterogeneous GPU-Based Clusters,” The 24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2012), Pittsburgh, USA, June 25—27, 2012.
8. F. Song, S. Tomov, and J. Dongarra, “Enabling and scaling matrix computations on heterogeneous multi-core and multi-GPU systems,” Proc. of ICS’12, p.365-376, Venice, Italy, June 25-29, 2012.
9. H. Anzt, S. Tomov, J. Dongarra, and V. Heuveline, “Weights for Block-Asynchronous Iteration on GPU-Accelerated Systems,” 10th HeteroPar'2012 (Tenth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms), Best Paper Award, Rhodes Island, Greece, August 2012.
10. 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,” International Journal of High Performance Computing Applications (IJHPCA), September 2012.
11. T. Dong, T. Kolev, R. Rieben, V. Dobrev, S. Tomov, and J. Dongarra, “Acceleration of the BLAST hydro code on GPUs,” SC’12 Poster, Salt Lake City, Utah, November, 2012.
12. Peng Du, Stanimire Tomov, and Jack Dongarra, “Providing GPU capability to LU and QR within the ScaLAPACK framework,” LAPACK Working Note #272, September 2012.
13. 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.
14. V. Joshi, "A study of possible optimizations for the task scheduler ‘QUARK’ on the shared

memory architecture. " Master's Thesis, University of Tennessee, 2013.
15. A. YarKhan, "Dynamic Task Execution on Shared and Distributed Memory Architectures," Ph.D. dissertation, University Of Tennessee, Knoxville, Major advisor: Jack Dongarra, December 2012.
16. J. Dongarra, M. Gates, A. Haidar, Y. Jia, K. Kabir, P. Luszczek, S. Tomov, "Portable HPC Programming on Intel Many-Integrated-Core Hardware with MAGMA Port to Xeon Phi," in Proceedings of 10th International Conference on Parallel Processing and Applied Mathematics, PPAM 2013, Warsaw, Poland, September 8-11, 2013.
17. Y. Jia, G. Bosilca, P. Luszczek, J. Dongarra, "Parallel Reduction to Hessenberg Form with Algorithm-Based Fault Tolerance," accepted to SC13, Denver, Colorado, November 2013.
18. Y. Jia, G. Bosilca, P. Luszczek, J. Dongarra, "CPU-GPU Hybrid Bidiagonal Reduction With Soft Error Resilience," accepted to Scala 2013 Workshop, Denver, Colorado, November, 2013.
19. S. Tomov, C. Cao, J. Dongarra, P. Du, M. Gates, P. Luszczek, "clMAGMA: High Performance Dense Linear Algebra with OpenCL," in Proceedings of IWOCL 2013, Atlanta, Georgia, May 13-14, 2013.