MAGMA: A Breakthrough in Solvers for Eigenvalue Problems

A. Haidar[1], S. Tomov[1], I. Yamazaki[1] R. Solca[3] and T. Schulthess[3,4]
T. Dong[1], and J. Dongarra[1,2]

[1] CCOE at University of Tennessee, Knoxville, TN
[2] Oak Ridge National Laboratory, Oak Ridge, TN
[3] ETH Zurich, Switzerland
[4] Swiss National Supercomputing Centre, Manno

Abstract:
The Matrix Algebra on GPU and Multicore Architectures (MAGMA) project aims to develop the next generation of LAPACK/ScaLAPACK-compliant linear algebra libraries for heterogeneous multicore-GPU architectures. The functionality covered in LAPACK and ScaLAPACK is a fundamental building block for many scientific computing applications, underlining the importance and potential for broad impact in developing these libraries for heterogeneous architectures. Currently, MAGMA's broad impact in the general area of technical computing is established through its incorporation in Matlab, contributions to CUBLAS, and use (starting) in a number of community codes, such as ABINIT, Quantum-Espresso, and the R project for statistical computing, just to name a few. Here, we describe a major breakthrough in the development of efficient solvers for eigenvalue problems on heterogeneous systems. The available implementations in LAPACK are bandwidth limited and therefore do not scale on multicore CPU systems. Indeed, performance improvements using state-of-the-art multicore hardware and vendor LAPACK implementations have been small. The new algorithms on the same multicore system, but enhanced with NVIDIA GPUs, improved the performance and hardware use more than one order of magnitude.

Background:
Scientific computing applications, ranging from computing frequencies that will propagate through a medium, to earthquake response of a bridge, or energy levels of electrons in nanostructure materials, require the solution of eigenvalue problems. There are many ways to formulate mathematically and solve these problems numerically [1]. In this work, we are interested in dense eigen-solvers, and in particular, generalized Hermitian-definite problems of the form

$$\mathbf{A\,x = \lambda\, B\,x} \qquad \textbf{(1)}$$

where A is a Hermitian dense matrix and B is Hermitian positive definite. These solvers are needed in practically all electronic structure methods [2, 3, 4].

Problem scope:
Solving (1) requires the development of a number of routines, e.g., first matrix **B** is factored using a Cholesky factorization $\mathbf{B = L\,L^H}$, next the **L** factors are used to transform (1) to a standard eigenvalue problem ($\mathbf{\tilde{A} = L^{-1}AL^{-H}}$, $\mathbf{\tilde{A}y = \lambda y}$). Finally, the eigenvectors of $\mathbf{\tilde{A}}$ must be transformed back to $\mathbf{x = L^{-H}\,y}$. The standard eigenvalue problem by itself can also be divided into sub problems; first tridiagonalize $\mathbf{\tilde{A}}$, next solve the tridiagonal eigenproblem, and finally, transform the eigenvectors of the tridiagonal problem back to the eigenvectors of $\mathbf{\tilde{A}}$. All of these steps are now developed in MAGMA. Asymptotically,

for large enough matrices, all of these algorithms, except the tridiagonalization, can be represented in terms of Level 3 BLAS, and hence can be efficiently implemented on current heterogeneous architectures. The tridiagonalization, though, has 50% of its flops in Level 2 BLAS, i.e., it is memory bound and would not scale on multicore platforms, and therefore it is the main target of this work.

Fast BLAS:
As 50% of the flops are in SYMV, we developed optimized versions in MAGMA. These are extensions of the MAGMA kernels presented at SC'11 [5], designed and optimized for Fermi GPUs. We used various blocking techniques, data reuse for the symmetry, and autotuning to optimize performance [6]. Performance is illustrated on Figure 1 below for single GPU (left) and multiGPUs (right). The performance improvement over CUBLAS 4.1 on Fermi is more than 2x, and scales properly when adding GPUs. The speedups for the other precisions (single real and complex single and double) are similar.

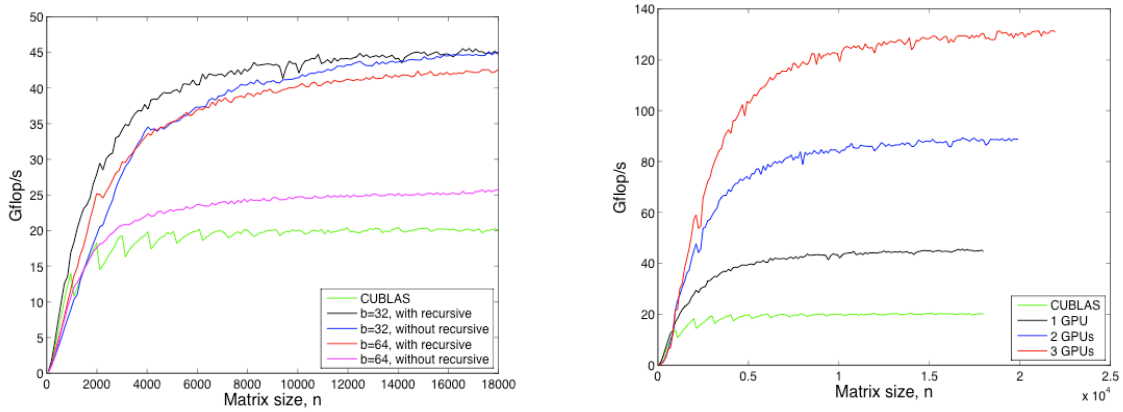

Figure 1. Performance of MAGMA DSYMV versions compared to CUBLAS 4.1 on one Fermi M2090 GPU (Left) and three Fermi M2090 GPUs (Right).

From fast BLAS to fast Eigen-Solvers:
To develop the tridiagonalization, we followed the *hybridization methodology* in MAGMA, where the algorithms are split into BLAS-based tasks and properly scheduled over the multicore host and the GPUs. Serial parts of the panels are scheduled on the multicore, and the large SYMV is scheduled on the GPUs. The trailing matrices are updated on the GPUs and reside on the GPUs to minimize communications. The results in Figure 2 demonstrate about an **8x performance improvement** over 12 Intel X5660 2.8 GHz cores. Details on the techniques used can be found in the ICL technical report [6] (submitted to SC'12).

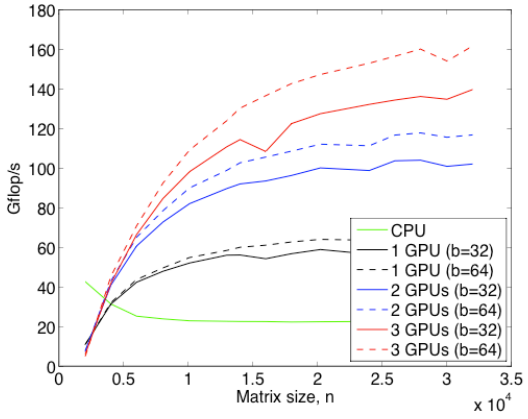

Figure 2. Performance of MAGMA DSYTRD using multiGPUs on a node of the Keeneland system (2 x 6 Intel X5660 2.8 GHz and 3 NVIDIA M2090 1.3 GHz GPUs) compared to MKL using all 12 cores.

An additional 4x speedup!

The top challenges in designing algorithms for current architectures revolve around the development of parallel algorithms that reduce synchronizations and maximize data-reuse. We developed a new tridiagonalization algorithm that addresses these challenges by considering the hybridization of a "two-stage" approach to tridiagonalization—reduction to band-diagonal, followed by reduction to tridiagonal. The first step of this approach can be represented in terms of Level 3 BLAS calls, and hence implemented very efficiently. The second involves "bulge chasing" operations that are challenging to develop [7]. We managed to extend this approach to heterogeneous systems to an **additional 4x performance improvement** over the previous results using a single GPU. This is illustrated on Figure 3. Details on the approach can be found in the ICL technical report [7] (submitted to SC'12).

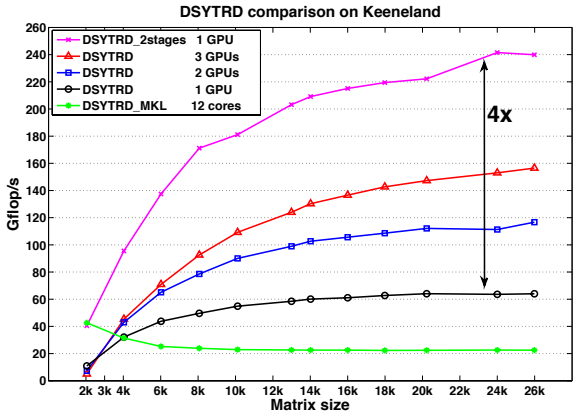

Figure 3. Performance of the two-stage approach in double precision using a single GPU on the Keeneland system (2x6 Intel X5660 2.8 GHz and 3 NVIDIA M2090 1.3 GHz GPUs).

References:

[1] James Demmel, Jack Dongarra, Axel Ruhe, Henk van der Vorst. 2000. ***Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide***. Zhaojun Bai (Ed.). Soc. for Industrial and Applied Math., Philadelphia, PA, USA.

[2] Kent, P. R. C., ***Computational Challenges of Large-Scale Long-Time First-Principles Molecular Dynamics***, J. Phys.: Conf. Series 125, 012058 (2008).

[3] Auckenthaler, T., Blum, V., Bungartz, H. J., Huckle, T., Jahanni, R., Krämer, L., Lang, B., Lederer, H., and Willems, P. R**.,** ***Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations***, preprint (2010).

[4] Singh, D. J. and Nordstrom, L., ***Plane waves, pseudo-potentials, and the LAPW method***, Springer (2006).

[5] R. Nath, S. Tomov, T. Dong, and J. Dongarra, ***Optimizing symmetric dense matrix-vector multiplication on GPUs***, Proc. of SC'11.

[6] T. Dong, J. Dongarra, T. Schulthess, R. Solca, S. Tomov, and I. Yamazaki, ***Symmetric dense matrix-vector multiplication on multiple GPUs and its application to symmetric dense and sparse eigenvalue problems***, UTK Technical report (submitted to SC'12).

[7] J. Dongarra, A. Haidar, T. Schulthess, R. Solca, and S. Tomov, ***A novel hybrid CPU-GPU generalized eigensolver for electronic structure calculations based on fine grained memory aware tasks***, UTK Technical report (submitted to SC'12).