Breakthroughs in Sparse Solvers for GPUs

CCOE at University of Tennessee, Knoxville

Abstract:
The CUDA Center of Excellence (CCOE) at UTK targets the development of innovative algorithms and technologies to tackle challenges in Heterogeneous High Performance Computing. Over the last year, the CCOE at UTK developed CUDA-based breakthrough technologies in sparse solvers for GPUs. Here, we describe the main ones – a sparse iterative solvers package, a communication-avoiding (CA) sparse iterative solver (CA-GMRES), preconditioners based on dense linear algebra (DLA) operations including batched GEMM, GEMV and batched LU, QR, and Cholesky for the parallel factorization of many small matrices, and a mixed-precision orthogonallization with application to sparse linear and eigenproblem solvers. Sparse linear algebra computations comprise a fundamental building block for many scientific computing applications, ranging from national security to medical advances, highlighting their importance and potential for broad impact. The new developments harness our expertise in DLA – namely, the MAGMA libraries, providing LAPACK for GPUs and auto-tuned BLAS – to develop high-performance sparse solvers, and building blocks for sparse computations in general.

MAGMA Sparse

Solving linear systems of equations is a fundamental problem in scientific computing. Numerical simulations involving complex systems represented in terms of unknown variables and relations between them often lead to linear systems of equations that must be solved as fast as possible. Recent hardware trends require the redesign of what were considered conventional solvers in order to make them efficient on modern architectures. In particular, the following two trends are most challenging to address:

- **The explosion of parallelism** where a single GPU can have thousands of cores (e.g., there are 2,880 CUDA cores in a K40), and algorithms must account for this level of parallelism in order to use the GPUs efficiently;
- **The growing gap of compute vs. data-movement capabilities** that has been increasing over the years (exponentially). To use modern architectures efficiently, new algorithms must be designed to reduce their data movements. Current discrepancies between the compute- vs. memory-bound computations can be orders of magnitude, e.g., from 1,200 GFlop/s on DGEMM to only 46 GFlop/s on DGEMV to even less on SpMV, all on a K40 NVIDIA GPU (see Figure 1).

To tackle these challenges in the area of sparse computations, we developed a number of innovative algorithms and technologies (described in the main sections below) and packaged them in the **MAGMA Sparse** library. MAGMA Sparse is currently under evaluation from collaborators, friendly users, and MathWorks (for inclusion in Matlab), in preparation for a Beta Release. Included are the following:

- Krylov subspace iterative solvers: CG, BiCGSTAB, GMRES, LOBPCG;
- Accelerated versions of BiCGSTAB and CG [1];

- Support for various matrix formats, including DENSE, CSR, Block-CSR, ELLPACK, ELLPACKT, ELLPACKRT, HYB, COO, CSC, SELLC/SELLC-σ;
- SpMV in CSR, Block-CSR, ELLPACK, ELLPACKT, and SELLC/SELLC-σ [2];
- Support for all four main arithmetic operations (single and double, real and complex), extended precision (double-double), mixed precision solvers based on Communication-avoiding GMRES [3];
- Jacobi, sparse direct LU, ILU [6], and building blocks for other algebraic preconditioners [8], including for low-rank approximations of HSS matrices [5].

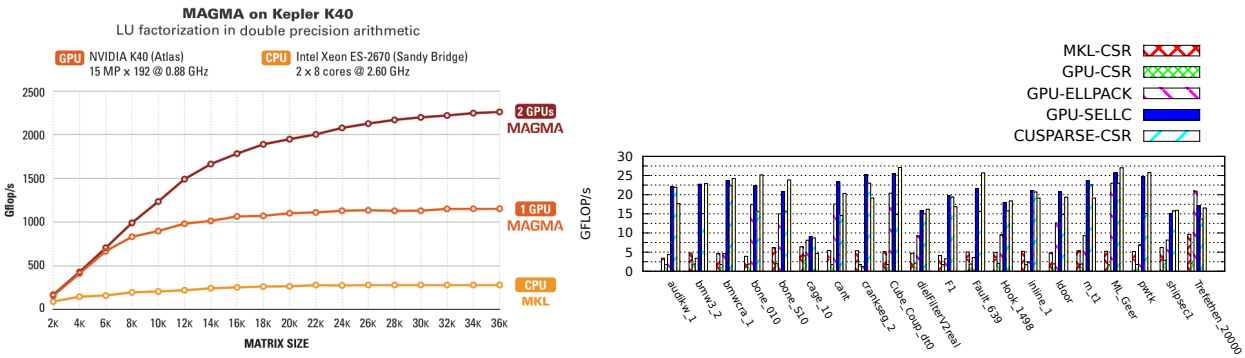

Figure 1. Performance and scalability of compute-intensive routines, e.g., current MAGMA LU for dense matrices on up to two K40 GPUs (Left), vs. memory- and latency-bound computations, e.g., SpMVs on K40 GPUs (Right).

MAGMA Sparse is a breakthrough as it provides highly optimized implementations of state-of-the-art LA developments to GPUs. On the other hand, sparse computations can not use GPUs as efficiently as DLA operations (illustrated on Figure 1), raising the need for new algorithms and techniques to address this challenge (next).

Building Blocks – SpMV kernels, batched LU, QR, and Cholesky factorizations

A common technique is to redesign the sparse solvers to use more DLA operations, Examples are block eigensolvers, higher order finite elements, properly designed direct multifrontal solvers and preconditioners, etc. (next). We developed a number of building blocks for many of these algorithms, including optimize SpMV kernels (and SpMM, e.g. for blocked solvers) with support for various matrix-formats [2], and batched LU, QR, and Cholesky for the simultaneous factorization of many very small dense matrices [8]. This is a breakthrough because it enables sparse solvers to become very efficient on GPUs. The new developments leveraged our extensive expertise in developing highly efficient DLA libraries like LAPACK, MAGMA, and MAGMA BLAS. Auto-tuning frameworks for BLAS on GPUs are in place to develop on demand application-specific kernels for matrices of particular sizes.

Preconditioners

The effectiveness of iterative solvers for sparse linear systems strongly depends on the use of properly designed preconditioners. We developed preconditioners that rely on DLA computations, and thus are efficient on GPUs. In particular, we develop a multi-elimination preconditioner for GPUs that is based on a multi-level incomplete LU factorization and uses a direct dense solver for the bottom-level system; numerous tests show its competiveness against other popular preconditioners (see [6] for more detail). Further, we developed dense QR factorizations with column pivoting and rank-revealing QR factorizations to compute the low-rank approximations for constructing preconditioners within sparse solvers (StruMF) that exploit Hierarchically Semi-Separable (HSS) matrix structures [5].

Orthogonalization prcedures

We developed a number of GPU-accelerated orthogonalizations – a key component for many sparse linear system and eigenproblem iterative solvers. The Cholesky QR showed superior performance as it is based on DLA operations, but unfortunately can be numerically unstable. We designed a mixed-precision extension that resolved the stability issues by applying extended precision calculations at critical places, and studied its effect in CA-GMRES [4].

CA-GMRES

GMRES is one of the most widely used iterative methods. In recent years, techniques to avoid communication in GMRES have gained attention, leading to the development of the CA-GMRES algorithms. We developed it for heterogeneous systems with multiple GPUs [3].

High-order FEMs

We worked on complete applications to further establish the appeal of high-order FEMs on new architectures. Despite their better approximation properties, higher-order FEMs are often avoided due to their high computational cost. GPUs have the potential to make them the method of choice, as the increased computational cost is also localized, e.g., cast as DLA operations, and thus can be done very efficiently. To this end we collaborated on a Hydrodynamic simulation that needed to compute thousands of 81*64 GEMMs or GEMVs, for which we developed high-performance batched versions [7].

References

[1] Hartwig Anzt, Stanimire Tomov, Piotr Luszczek, Ichitaro Yamazaki, Jack Dongarra, and William Sawyer, ***"Optimizing Krylov Subspace Solvers on Graphics Processing Units",*** UTK Technical Report UT-EECS-14-725, February, 2014.

[2] Hartwig Anzt, Stanimire Tomov, and Jack Dongarra, ***"Implementing a Sparse Matrix Vector Product for SELLC/SELLC-σ format on NVIDIA GPUs",*** Euro-Par 2014 Parallel Processing (submitted), Porto, Portugal, 25—29 August, 2014.

[3] Ichitaro Yamazki, Hartwig Anzt, Stanimire Tomov, Mark Hoemmen, and Jack Dongarra, "***Improving the Performance of CA-GMRES on Multicores with Multiple GPUs",*** Proc. of IPDPS 2014 (accepted), Phoenix, Arizona, May 19—23, 2014.

[4] Ichitaro Yamazaki, Stanimire Tomov, Tingxing Dong, and Jack Dongarra, ***"Mixed-precision orthogonalization scheme and adaptive step size for CA-GMRES on GPUs",*** VECPAR 2014 (submitted), Eugene, Oregon, USA, June 30—July 3, 2014.

[5] Ichitaro Yamazaki, Artem Napov, Stanimire 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.

[6] Dimitar Lukarski, Hartwig Anzt, Stanimire Tomov, and Jack Dongarra, ***"Hybrid Multi-Elimination ILU Preconditioners on GPUs"***, 23rd Heterogeneity in Computing Workshop (HCW 2014), in Proc. of IPDPS 2014 (accepted), Phoenix, Arizona, May 19—23, 2014.

[7] Tingxing Dong, Veselin Dobrev, Tzanio Kolev, Robert Rieben, Stanimire Tomov, and Jack Dongarra, ***"A Step towards Energy Efficient Computing: Redesigning a Hydrodynamic Application on CPU-GPU",*** Proc. of IPDPS 2014 (accepted), Phoenix, Arizona, May 19—23, 2014.

[8] Tingxing Dong, Azzam Haidar, Stanimire Tomov, and Jack Dongarra, ***"Batched Cholesky Factorization on GPU",*** VECPAR 2014 (submitted), Eugene, Oregon, USA, June 30—July 3, 2014.