SLATE MIXED PRECISION PERFORMANCE REPORT[1]

Ali Charara, Jack Dongarra, Mark Gates, Jakub Kurzak, Asim YarKhan

INTRODUCTION

Software for Linear Algebra Targeting Exascale (SLATE) is being developed as part of the Exascale Computing Project (ECP), which is a collaborative effort between two US Department of Energy (DOE) organizations, the Office of Science and the National Nuclear Security Administration (NNSA). The purpose of SLATE is to serve as a replacement for ScaLAPACK for the upcoming exascale DOE machines. SLATE will accomplish this objective by leveraging recent progress in parallel programming models and by strongly focusing on supporting hardware accelerators.

This report focuses on the set of SLATE routines for solving linear systems of equations using the technique of mixed-precision iterative refinement. Initial performance numbers are reported using the SummitDev system at the Oak Ridge Leadership Computing Facility (OLCF).

MOTIVATION

The goal of mixed precision algorithms is to benefit from the fact that single precision calculations are usually 2× faster than double precision calculations. The techniques presented here were pioneered in 2006-2007 for the Cell processor [2], which offered 14× more single precision performance than double precision performance, and later extended to other architectures [1] [3]. These days, the ratio is almost universally 2× for CPUs, while for GPUs it depends on the targeted market. Devices meant for scientific and engineering computing usually offer the same 2× ratio, although exceptions happen, while the devices meant mostly for graphics applications are typically slower in double precision by on order of magnitude.

Not without significance is the adoption of half precision in GPUs from NVIDIA and ADM, mostly targeting applications in deep learning. Due to the use of specialized cores, half precision offers an order of magnitude higher performance than single precision. At the same time, half precision suffers not only from much lower precision, but also severely limited range, and requires much more complex numerical approaches [4] [5]. At this time, we are focusing on single/double refinement.

ALGORITHM

Iterative refinement is a well-known method for improving the solution of linear systems of equations of the form $Ax = b$. Canonical approach is the Richardson iteration $x^{(k+1)} = x^{(k)} + A^{-1}\left(b - Ax^{(k)}\right)$, which is equivalent to gradient descent. In the case of Gaussian elimination, the coefficient matrix A is factorized using LU decomposition into the product of a lower triangular matrix L and an upper triangular matrix U.

[1] This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, in support of the nation's exascale computing imperative.

Commonly, partial row pivoting is used to improve numerical stability resulting in the factorization $PA = LU$, where P is the row permutation matrix. The system is solved by solving $Ly = Pb$ (forward substitution) and then solving $Ux = y$ (backward substitution). Owing to the roundoff error, the solution carries an error related to the condition number of the coefficient matrix A. In order to improve the computed solution an iterative refinement process is applied, which produces a correction to the computed solution, x, at each iteration and yields the basic iterative refinement algorithm.

In the case of mixed-precision iterative refinement the factorization $PA = LU$ and the solution of the triangular systems $Ly = Pb$ and $Ux = y$ are computed using single-precision arithmetic. The residual calculation and the update of the solution are computed using double-precision arithmetic and the original double-precision coefficients. The most computationally expensive operations, including the factorization of the coefficient matrix A and the forward and backward substitution, are performed using single-precision arithmetic and take advantage of the single-precision speed. The only operations executed in double precision are the residual calculation and the update of the solution. It can be observed that all operations of $O(n^3)$ computational complexity are handled in single precision and all operations performed in double precision are of at most $O(n^2)$ complexity. The coefficient matrix A is converted to single precision for the LU factorization. At the same time, the original matrix in double precision has to be retained for the residual calculation. Therefore, the method requires 1.5× the storage of the strictly double-precision method.

IMPLEMENTATION

Two mixed precision solvers were implemented in SLATE – the `gesvMixed()` routine, based on LU factorization and the `posvMixed()` routine, based on the Cholesky factorization. The routines take the lower precision and the higher precision as template parameters. The factorizations are performed in the lower precision and the iterative refinement is performed in the higher precision. The refinement stops when the maximum number of iterations is reached (30) or when all right-hand side (RHS) columns satisfy the condition $\|r\|_\infty < \sqrt{n} * \|x\|_\infty * \|A\|_\infty * \varepsilon$, where r is the residual, n is the problem size, x is the solution vector, A is the matrix, and ε is the machine epsilon. The algorithm is identical to the one in LAPACK. It has never been implemented in ScaLAPACK.

The mixed precision solvers also required implementation of three additional auxiliary routines – elementwise matrix addition, matrix copy/conversion, and a routine for computing the max norm for all columns of the RHS matrix individually. Corresponding GPU kernels were developed to support GPU execution.

EXPERIMENTS

Environment

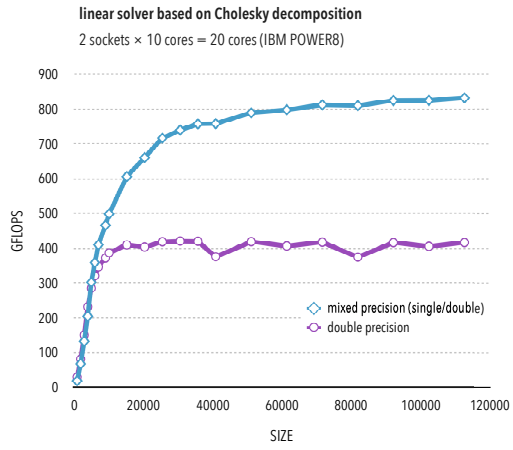
Performance numbers were collected using the SummitDev system at the OLCF, which is intended to mimic the OLCF's supercomputer Summit. SummitDev is based on IBM POWER8 processors and NVIDIA P100 (Pascal) accelerators, and is one generation behind Summit, which is based on IBM POWER9 processors and NVIDIA V100 (Volta) accelerators.

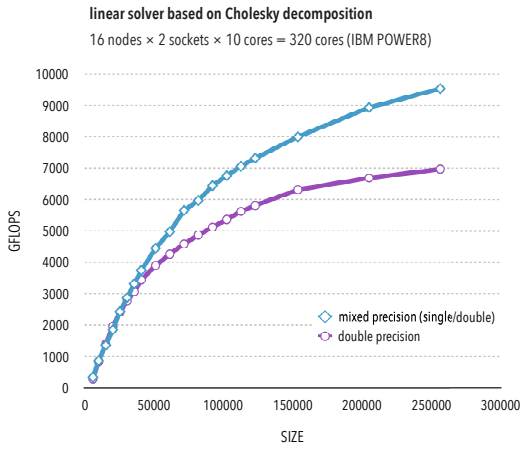
The SummitDev system contains three racks, each with eighteen IBM POWER8 S822LC nodes, for a total of fifty-four nodes. Each node contains two POWER8 CPUs, ten cores each, and four P100 GPUs. Each node has 256 GB of DDR4 memory. Each GPU has 16 GB of HBM2 memory. The GPUs are connected by NVLink 1.0 at 80 GB/s. The nodes are connected with a fat-tree enhanced data rate (EDR) InfiniBand.

The software environment used for the experiments included GNU Compiler Collection (GCC) 6.4.0, CUDA 9.0.69, Engineering Scientific Subroutine Library (ESSL) 6.1.0-1, Spectrum MPI 10.2.0.7, Netlib LAPACK 3.8.0, and Netlib ScaLAPACK 2.0.2.

Performance

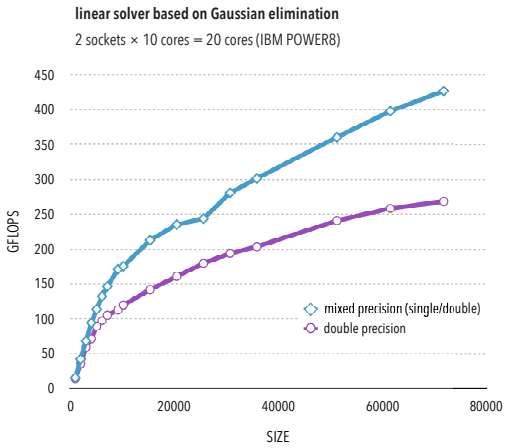
The figures blow show the performance of the posvMixed() routine. Multicore performance using 20 cores is shown on the left, and distributed memory performance using 16 nodes (320 cores) is shown on the right. In shared memory the posvMixed() routine basically reaches the theoretical limit of 2× performance boost (100%), while in distributed memory the benefit is only about 35%.

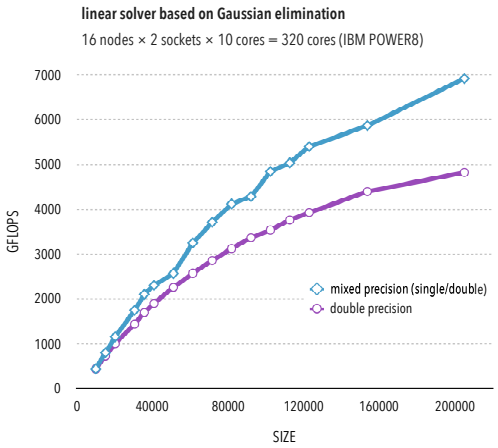




The figures blow show the performance of the gesvMixed() routine. Multicore performance using 20 cores is shown on the left, and distributed memory performance using 16 nodes (320 cores) is shown on the right. In general, the LU factorization performs much worse than the Cholesky factorization, mainly due to the overhead of pivoting (row swaps). Because pivoting is also part of the refinement process, this also diminished the performance gain of mixed precision. In shared memory the boost reaches about 60%, while in distributed memory it reaches about 40%.

Unfortunately, the GPU implementations of mixed-precision solvers are not currently outperforming the GPU implementations of fixed-precision solvers. This is most likely caused by excessive communication between the CPU memory and the GPU memories and will require further performance engineering efforts. Eventually, we expect close to the full 2× speedup across all mixed-precision solvers, single-node and multi-node, CPU and GPU.





SUMMARY

We delivered mixed precision linear solvers based on the Cholesky and LU factorizations, with support for distributed memory, multithreading, and multi-GPUs. We achieved significant performance improvements for multicore runs in shared memory and in distributed memory. At the same time, outperforming fixed-precision solvers proved difficult in distributed memory with multi-GPU acceleration, which is most likely due to suboptimal CPU to GPU communication. Further performance engineering efforts will be needed. Eventually full performance benefit is expected (close to 2× speedup).

ACKNOWLEDGMENTS

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

BIBLIOGRAPHY

[1] J. Langou, J. Langou, P. Luszczek, J. Kurzak and A. Buttari, "Exploiting the Performance of 32 bit Floating Point Arithmetic in Obtaining 64 bit Accuracy (Revisiting Iterative Refinement for Linear Systems)," in *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, Tampa, FL, 2006.

[2] J. Kurzak and J. Dongarra, "Implementation of mixed precision in solving systems of linear equations on the Cell processor," *Concurrency and Computation: Practice and Experience,* vol. 19, p. 1371–1385, 2007.

[3] A. Buttari, J. Dongarra, J. Langou, J. Langou, P. Luszczek and J. Kurzak, "Mixed Precision Iterative Refinement Techniques for the Solution of Dense Linear Systems," *The International Journal of High Performance Computing Applications,* vol. 21, no. 4, p. 457–466, 2007.

[4] E. Carson and N. J. Higham, "Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions," *Methods and Algorithms for Scientific Computing,* vol. 40, no. 2, p. A817–A847, 2018.

[5] A. Haidar, S. Tomov, J. Dongarra and N. J. Higham, "Harnessing GPU tensor cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers," in *SC '18: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, Dallas, TX, 2018.