

Accelerating the ANSYS Direct Sparse Solver with GPUs

Géraud P. Krawezik
Acceleware Corp.
Calgary, AB, Canada

Gene Poole
ANSYS Inc.
Canonsburg, PA, USA

Abstract—As hardware accelerators and especially GPUs become more and more popular to accelerate the compute intensive parts of an algorithm, standard high performance computing packages are starting to benefit from this trend.

We present the first GPU acceleration of the ANSYS direct sparse solver. We explain how such a multifrontal solver may be accelerated using an optimized dense matrix factorization, and show that with the current generation of hardware, speed-ups of up to $4\times$ can be obtained by utilizing mixed precision, and $2.9\times$ in double precision.

We discuss the impact of the model being studied on the overall performance, as well as the influence of using single precision factorizations on the accuracy of the solution.

I. INTRODUCTION

GPUs are now seen as efficient accelerators to process the compute-intensive parts of a program. Their large number of cores, fast memory, and improved programming models [1]–[3] allow researchers to investigate their use for linear algebra. Recent results have proven their efficiency in the case of BLAS3 matrix-matrix multiplication and its direct application: factorization of dense matrices [4]–[6]. However only few experiments have been done with higher level numerical methods.

Multifrontal solvers [7] are common direct sparse solvers due to their robustness and speed, and for the level of parallelism that they induce. They are based on the decomposition of a large sparse matrix through the factorization of dense matrices (fronts) and their assembly into supernodes. Each front decomposition relies on BLAS3 operations, leading to near peak performance on most processors.

We will present the first implementation of the GPU accelerated version of the ANSYS direct sparse solver. Sections II and III discuss which portions of the program were accelerated and the methods used to obtain the performance. Section IV presents, analyzes and discusses the results followed by some concluding remarks in section V.

II. ACCELERATING THE ANSYS DIRECT SPARSE SOLVER

Most of the runtime of a direct sparse solver is spent in the factorization of dense matrices (the fronts) and their assembly. For this reason, this decomposition was ported to GPUs. In the case of ANSYS, the matrices are symmetric, and factorized using a generalized Cholesky (LDL^T) decomposition.

In a typical run, there are many fronts of different sizes, and some of them may not be large enough to be efficiently

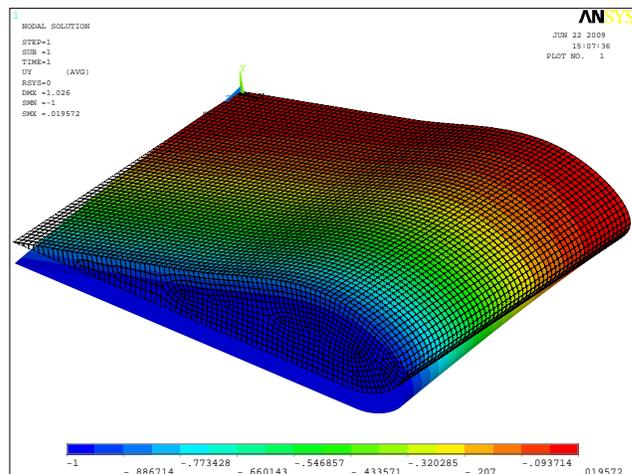


Fig. 1. The BM-7 wing model (dofs=500,000) in the ANSYS GUI: the wireframe shows the original structure; the colored part is the displacement

computed on GPUs. We will focus our tests on the BM-7 Wing model (see figure 1) from the standard ANSYS benchmarks as it enables us to arbitrarily set the number of degrees of freedom in the system. For 1.25 million degrees of freedom, figure 2 shows the distribution of fronts according to their size. As seen in the histogram in figure 2, most of the fronts are small in size. However, it is worth noting that the time spent factorizing depends on their size: this is shown on figure 3 for different numbers of degrees of freedom.

Although there are numerous small fronts, the total computation time is mainly spent in the factorization of the larger ones. As a result, our acceleration target will be the factorization of the large fronts, that can fit on the GPU and be efficiently computed there.

III. EFFICIENT LDL^T FACTORIZATION ON GPUS

A. Implementation Details

The different dense matrix factorizations have been described on standard general purpose processors as well as parallel machines and GPUs [4], [8]. One of the best performing implementations is the right-looking algorithm due to its ability to exploit parallelism and computational intensity.

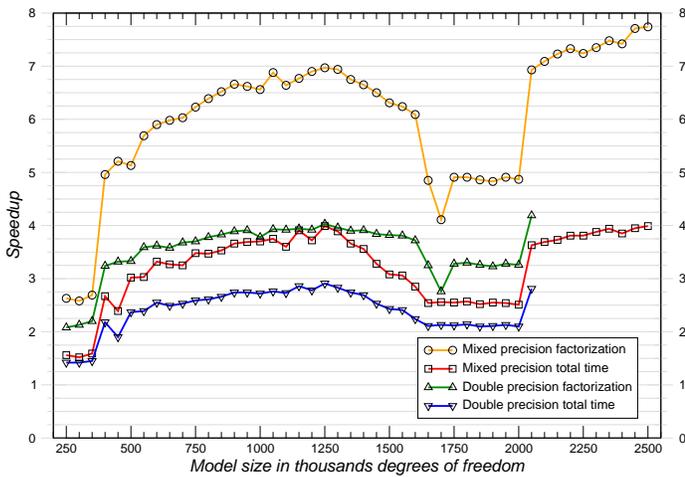


Fig. 6. Speedup versus two CPU cores

500 and 1250 thousands degrees of freedom, then after 2000. Figure 3 explains this behavior: before 500 thousand, the number of large fronts is too small to use the GPU efficiently, while after 1250, the overhead incurred from out-of-core computation becomes noticeable: the forward/backward solve part is increasing. After 2 million, very large fronts start to appear, which (the factorization is N^3) occupy most of the runtime. Since these fronts run at peak performance on the GPU, the overall speedup becomes optimal.

Increasing the model size leads to the creation of fronts that cannot fit on the GPU (> 46000 for single precision, > 32000 for double precision) while greatly increasing the factorization time. In practice, models of this size would use more than one host and were not represented in this study.

C. Discussion

1) *Mixed Precision Computation*: Two issues can arise while using single precision computations in a scientific code. The first one involves the overflow of the single precision units, which can happen when dealing with very large or very small values. In the case of physical simulations, this is highly unlikely.

The second point is more crucial: due to the loss in precision, an overall loss in accuracy for the whole solver is observed. Note that in the case of iterative solvers, this is a non-issue as the problem can be easily circumvented by adding more iterations to reach the desired accuracy. This idea can be applied to direct solvers as well by using iterative refinement, such as the approach presented in [10].

2) *Model Influence*: An aspect which has not been explored in this paper is the influence of the model on the global performance. The BM-7 wing is, as shown in figure 1, a very blocky model: it does not contain holes or thin shapes that tend to result in many small matrices and few large supernodes.

In the real world, such models can be found, for instance while analyzing a block engine, or a car structure. In order to get better acceleration for these models, where large matrices

might be fewer, if not absent, it becomes necessary to accelerate the computation of small independent fronts, by either factorizing them in parallel on the GPU, or on both the host processor and the accelerator. Such approaches are part of our future work.

V. CONCLUSION

Despite the increasing number of studies regarding GPU acceleration of dense linear algebra problems, a single experiment has been done so far with multifrontal solvers [11], while the others have limited their scope to BLAS3 functions and dense matrices factorizations [4]–[6].

We have described how to accelerate the ANSYS direct sparse solver by moving the decomposition of large dense matrices from the CPU to a GPU. This leads to a speedup of $4\times$ with mixed factorization (single precision factorizations) and $2.9\times$ with double precision.

Future work includes exploiting the inherent parallelism of multifrontal solvers by computing concurrently factorizations on both CPU and GPU, as well as better understanding the influence of single precision factorization on the global error of the solver and testing iterative refinement schemes.

ACKNOWLEDGMENT

The authors would like to thank Chris Mason for his precious feedback, and the reviewers for their useful comments.

REFERENCES

- [1] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable Parallel Programming with CUDA," in *ACM Queue*, ACM, Ed., March/April 2008, vol. 6, Issue 2, pp. 40–53.
- [2] A. M. D. Inc., "Brook+ SC07 BOF Session," Advanced Micro Devices, Inc., November 2007.
- [3] *The OpenCL Specification*, 1st ed., Khronos OpenCL Working Group, December 2008.
- [4] V. Volkov and J. W. Demmel, "Benchmarking GPUs to tune dense linear algebra," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*.
- [5] M. Baboulin, J. Dongarra, and S. Tomov, "Some issues in dense linear algebra for multicore and special purpose architectures," University of Tennessee, Tech. Rep. UT-CS-08-200, May 2008.
- [6] S. Barrachina, M. Castillo, F. D. Igual, R. Mayo, and E. S. Quintana-Ortí, "Solving dense linear systems on graphics processors," Universidad Jaime I, Tech. Rep. ICC 02-02-2008, February 2008.
- [7] I. S. Duff and J. K. Reid, "The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations," in *ACM Transactions on Mathematical Software (TOMS)*, September 1983, vol. 9, Issue 3, pp. 302–325.
- [8] J. Choi, J. J. Dongarra, L. S. Ostrouchov, A. P. Petitet, D. W. Walker, and R. C. Waley, "Design and Implementation of the ScaLAPACK LU, QR, and Cholesky Factorization Routines," in *Scientific Programming*, I. Press, Ed., August 1996, vol. 5, Issue 3, pp. 173–184.
- [9] F. G. Gustavson, J. Wasniewski, J. J. Dongarra, and J. Langou, "The Relevance of New Data Structure Approaches for Dense Linear Algebra in the New Multi-Core / Many Core Environments," January 2009, IBM Technical Report RC24728.
- [10] A. Buttari, J. J. Dongarra, J. Langou, J. Langou, P. Luszczek, and J. Kurzak, "Mixed Precision Iterative Refinement Techniques for the Solution of Dense Linear Systems," in *International Journal of High Performance Computing Applications*, 2007, vol. 21, Issue 4, pp. 457–466.
- [11] R. F. Lucas, G. Wagenbreth, and D. M. Davis, "Implementing a GPU-enhanced cluster for large-scale simulations," in *Proceedings of the Interservice/Industry Training, Simulation & Education Conference (I/IT SEC)*, 2007.