GPU Parameter Tuning for Dense Linear Least Squares Problems

Benjamin Sauk[1], Nikolaos Ploskas[1] and Nick Sahinidis[2], (1)Carnegie Mellon University, Pittsburgh, PA, (2)Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA

Abstract Text:

Linear least squares problems (LLSPs) routinely arise in many types of scientific and engineering problems. As the size of datasets used to generate linear models increases, there exists a need to efficiently solve large dense linear least squares problems. One of the fastest ways to solve large LLSPs is to utilize the parallel processing power of graphics processing units (GPUs), including cuSolverDN [5] and communication avoiding QR [3]. There has also been work to develop hybrid algorithms that take advantage of the processing power of both CPUs and GPUs through the MAGMA project [7]. However, these parallel algorithms are typically designed and optimized for one GPU architecture and may be unusable or suboptimal on another GPU. To design algorithms that are portable between devices, most algorithms are designed with tunable parameters that can be adjusted to avoid having to rewrite most of an algorithm to remain optimal on any GPU architecture. Traditionally, these parameters have been chosen through extensive experimentation or through heuristics that come from the knowledge of solver designers.

The primary contribution of this work is to propose systematic methods for tuning GPU or hybrid CPU/GPU algorithms through the use of derivative-free optimization (DFO) [6] and simulation optimization (SO) [1]. A second contribution of our work is to provide a comparison of thirty-one DFO and four SO solvers in the context of tuning GPU algorithms.

To determine a baseline of performance, we evaluated the performance of a few of the most well-known dense linear algebra libraries: LAPACK [2], a multicore solver PLASMA [4], a GPU only algorithm cuSolverDN [5], and a hybrid implementation MAGMA [7]. We evaluated each of these solvers over a wide range of different sized square and tall and skinny matrices. Tall and skinny matrices commonly arise when solving LLSPs, and are typically challenging to obtain high performance on because of the matrix structure. For square matrices, the solver MAGMA was able to outperform all of the other solvers. However, for tall and skinny matrices MAGMA was not able to perform as well as the other solvers.

Our computational results show that the best DFO solver is able to speed up the performance of MAGMA by 1.67x compared to default MAGMA parameters. After tuning MAGMA through the proposed approach, MAGMA was able to outperform all other solvers for tall and skinny matrices.

References cited

[1] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury. Simulation optimization: A review of algorithms and applications. Annals of Operations Research, 240:351–380, 2016.

[2] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen. LAPACK users' guide (third ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[3] M. Anderson, G. Ballard, J. Demmel, and K. Keutzer. Communication-avoiding QR decomposition for GPUs. In Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International, pages 48–58, 2011.

[4] B. Hadri, H. Ltaief, E. Agullo, and J. Dongarra. Tile QR factorization with parallel panel processing for multicore architectures. In Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on, pages 1–10, 2010.

[5] NVIDIA Corporation. cuSolver, Current as of 28 December, 2016. http://docs.nvidia.com/cuda/cusolver/#axzz4SZ3ssQJO.

[6] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. Journal of Global Optimization, 56:1247–1293, 2013.

[7] S. Tomov, J. Dongarra, and M. Baboulin. Towards dense linear algebra for hybrid GPU accelerated manycore systems. Parallel Computing, 36:232–240, 2010.