Extending MAGMA Portability with OneAPI

Anna Fortenberry
University of North Texas
Denton, TX, USA
AnnaFortenberry@my.unt.edu

Dr. Stan Tomov (advisor)
University of Tennessee
Knoxville, TN, USA
tomov@icl.utk.edu

Dr. Kwai Wong (advisor)
University of Tennessee
Knoxville, TN, USA
kwong@utk.edu

Supercomputers (SCs) solve important problems in a vast number of domains, notably High Performance Computing (HPC). The systems of supercomputers are becoming increasingly diverse in hardware. Specifically, GPUs and other accelerators are increasingly integrated with CPUs to boost performance. The Top500 list for June 2022 shows that three different vendors supply accelerators to eight of the top ten SCs, including Nvidia, AMD, and NUDT [1]. Further, with the announcement of Aurora, Intel plans to join as a supercomputer GPU vendor, as well. This illustrates how supercomputers are becoming increasingly diverse in both architecture types and designs. The effect of this is a decrease in interoperability between code and hardware. To take advantage of heterogeneous architectures without requiring significant code modifications, architecture-independent programming models are needed. Intel adopts such a programming model interface called oneAPI with the claim that it is the solution to this problem. Applications which adopt this model gain portability to all supported hardware, including scalar, vector, spatial, and matrix architectures. This research tests this programming model by migrating the dense linear algebra library known as Matrix Algebra on GPU and Multicore Architectures (MAGMA) to the direct programming language of oneAPI, Data Parallel C++ (DPC++). Specifically, the performance of the migrated code is tested with MAGMA's Single-Precision General Matrix Multiplication (SGEMM) algorithm. Although double-precision floating-point arithmetic is a standard in HPC, the MAGMA codes have a single implementation, regarding precision, and the rest are generated. Thus, the precision choice is not important for the results in this paper. One software tool to note is the DPC++ Compatibility Tool (DPCT). It is meant to encourage the use of oneAPI, as it aids in the migration of CUDA, the language for porting to NVIDIA GPUs, to DPC++. The hardware tested includes two CPUs, the AMD EPYC 7742 64-Core Processor @ 2.25GHz and the Intel® Xeon® CPU E5-2698 V4 20-Core Processor @ 2.20GHZ, and two GPUs, the NVIDIA GeForce RTX 3060, which is a discrete GPU, and the Intel UHD Graphics P630 [0x3e96], which is integrated. Running DPC++ on Nvidia GPUs required an additional software installation: DPC++-LLVM (Clang-LLVM) compiler [2]. The Intel GPU of choice was the only publicly available GPU at the time this research was conducted. It has since been discontinued. It is important to note that Intel's GPU is a mobile device and thus has significantly less computational power than the Nvidia GPU. Tests were run on high-end Intel GPUs in the early-access precursors of the Aurora system, but this results cannot be publicly shared. It can be said that the conclusions and main message of this research remained the same. The choices of accelerators tested were heavily influenced by the time constraint of this research. The DPCT tool proved to be successful for an initial migration of CUDA code to DPC++. For testing its performance on GPUs, MAGMA, which is tuned for Nvidia GPUs, was selected as an upper bound. A generic CUDA SGEMM algorithm provided by Nvidia [3] was migrated with DPCT to serve as a baseline for performance. Thus, the generic CUDA SGEMM algorithm, the generic DPC++ SGEMM algorithm, CUDA MAGMA SGEMM, and DPC++ MAGMA SGEMM were all four compared. For testing on CPUs, MKL was selected as an upper bound and OpenMP as a lower bound. The generic DCP++ SGEMM algorithm was included as well, all contrasted with the migrated DPC++ MAGMA SGEMM. The MAGMA GEMM algorithm is tuned with a set of nine constants; these constants were previously tuned to be optimal on Nvidia GPUs [4]. In the migrated code, these constants remained optimal for the Nvidia GPU and were optimal for the multicore CPUs, as well. This was confirmed by checking the usage of the accelerators. The constants were not optimal on the Intel GPU, so new constants were tested. Most optimal constants were not discovered, as further detail about the hardware is needed to find these without trial and error. The performance on CPUs was impressive; on the AMD CPU, DPC++ MAGMA SGEMM was able to outperform even MKL. On the Intel CPU, it outperformed OpenMP and the generic SGEMM algorithm. On the Nvidia GPU, DPC++ MAGMA SGEMM retained the performance of MAGMA. On the Intel GPU, performance was increased more than 10x by tuning the GEMM constants. The oneAPI programming model proved to be a promising approach to portable, parallel computing on heterogeneous systems. It can be concluded that large numerical libraries like MAGMA that were originally written in CUDA for Nvidia GPUs can be migrated to DPC++ relatively easily. In doing so, it gains functional portability to different vendor GPUs and multicore CPUs. Migrated code performs well on multicore CPUs and Nvidia GPUs without tuning. For the Intel GPUs, autotuning techniques must be used to discover the best performing versions of GEMM for taking full advantage of the computational capabilities of the Intel GPU.

ACKNOWLEDGMENTS

This project was sponsored by the National Science Foundation through the Research Experience for Undergraduates (REU) award no. 2020534 with additional support from the National Institute of Computational Sciences and Innovative Computing Laboratory at the University of Tennessee, Knoxville.

REFERENCES

[1] June 2022 | TOP500. Top500 The List. Retrieved August 5, 2022 from https://www.top500.org/lists/top500/2022/06/

[2] 2022. Compiling SYCL* for Different GPUs. Intel. Retrieved August 5, 2022 from https://www.intel.com/content/www/us/en/developer/articles/tool/opencl-drivers.html

[3] *CUDA Samples*. NVIDIA. https://github.com/NVIDIA/cuda-samples

[4] Rajib Nath, Stanimire Tomov, and Jack Dongarra. 2010. An Improved Magma Gemm For Fermi Graphics Processing Units. The International Journal of High Performance Computing Applications 24, 4 (2010), 511-515. DOI:https://doi.org/10.1177/1094342010385729