Extending MAGMA Portability with oneAPI

Anna Fortenberry[1], Dr. Kwai Wong[2], Dr. Stan Tomov[2]
[1]University of North Texas [2]University of Tennessee at Knoxville

Project Overview

Matrix Algebra on GPU and Multicore Architectures (MAGMA) is a dense linear algebra library for heterogeneous architectures [1]. It was originally designed to run with Nvidia GPUs. Portability was later extended to the AMD GPU. This means significant portions of MAGMA are written in CUDA, which was later translated to HIP. Intel has released a multi-architecture programming model called oneAPI, which claims portability to GPUs, CPUs, FPGAs, and more [2]. We want to translate MAGMA to be compatible with oneAPI to extend its portability.

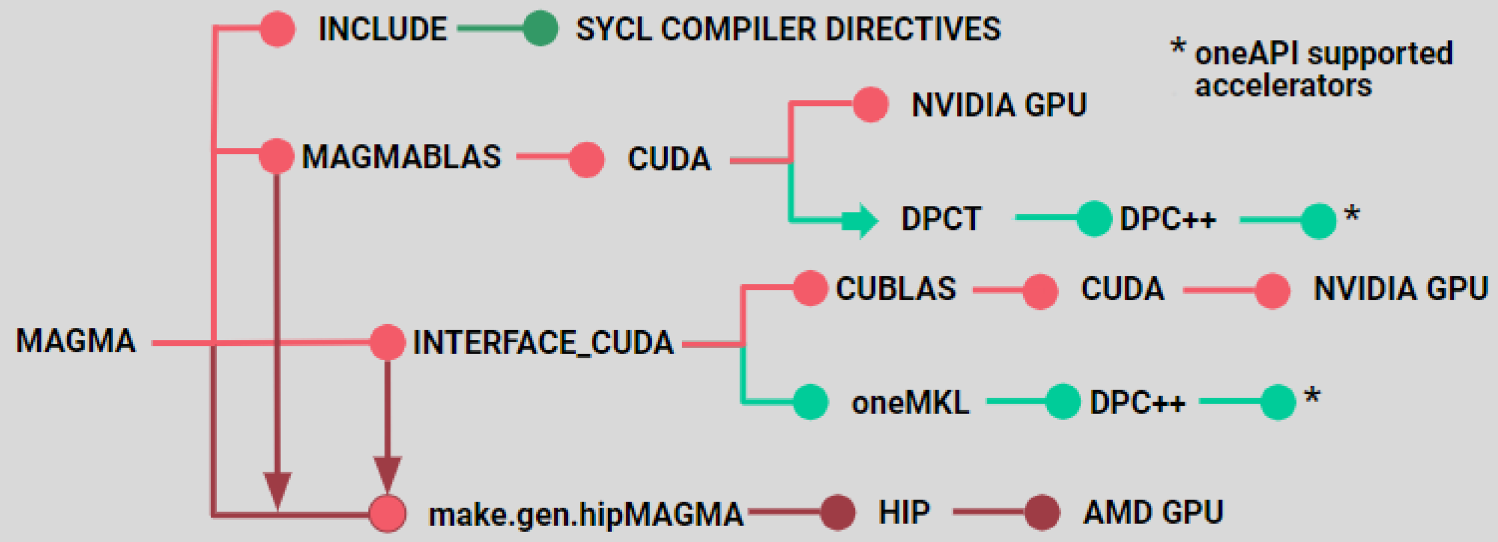

Figure 1: Structure of MAGMA with Intel GPU support

Data Parallel C++ (DPC++), the DPC++ Compatibility Tool (DPCT), and the oneAPI Math Kernel (oneMKL) are software tools within oneAPI. Figure 1 depicts where these tools fit into the translation process of MAGMA. Compiler directives, shown in green, must be implemented to complete the translation process.

Research Questions

- How well does the DPCT translate CUDA code to SYCL code?
- What are the common translation errors?
- What are the system requirements?
- Is SYCL portable to Nvidia and AMD GPUs?
- Can this tool be used to translate MAGMA?
- What is the performance of SYCL on multicore CPUs?
- How does the performance of SYCL on Nvidia and AMD GPUs compare to CUDA and HIP on their respective GPUs?
- How does the performance of SYCL on the Intel GPU compare to CUDA and HIP on their respective GPUs?

Methodology

1. Configure system for running SYCL code and create documentation of the installment process
2. Translate different structures of CUDA files to SYCL with DPCT for correctness
3. Document translation process and errors
4. Configure system to run SYCL code on Nvidia GPU
5. Test performance of sgemm code
6. Repeat steps 1-5 on Innovative Computing Lab (ICL) account
7. Begin MAGMA translation process of CUDA to SYCL

Software Tools

oneAPI	DPC++	SYCL
Open, multi-vendor programming model that delivers a common developer experience across accelerator architectures [2]	Direct programming language of oneAPI. Comprised of C++, SYCL, and DPC++ language extensions; compiler implementation of SYCL [3], [4]	Cross-architecture language to allow code reuse across hardware targets; enables definitions of data parallel functions [3], [4]

DPCT	CUDA	oneMKL
Tool to translate CUDA code to DPC++ code with high accuracy [5]	Parallel computing platform and programming model for the Nvidia GPU [6]	Computing math library of highly optimized parallel routines [7]

DPC++ LLVM Nvidia	DPC++ LLVM AMD	ICL Account
Builds DPC++ (LLVM-based) compiler with CUDA support [8]	Builds DPC++ (LLVM-based) compiler with HIP AMD support [8]	Grants access to powerful Nvidia and AMD GPUs

CPU Performance

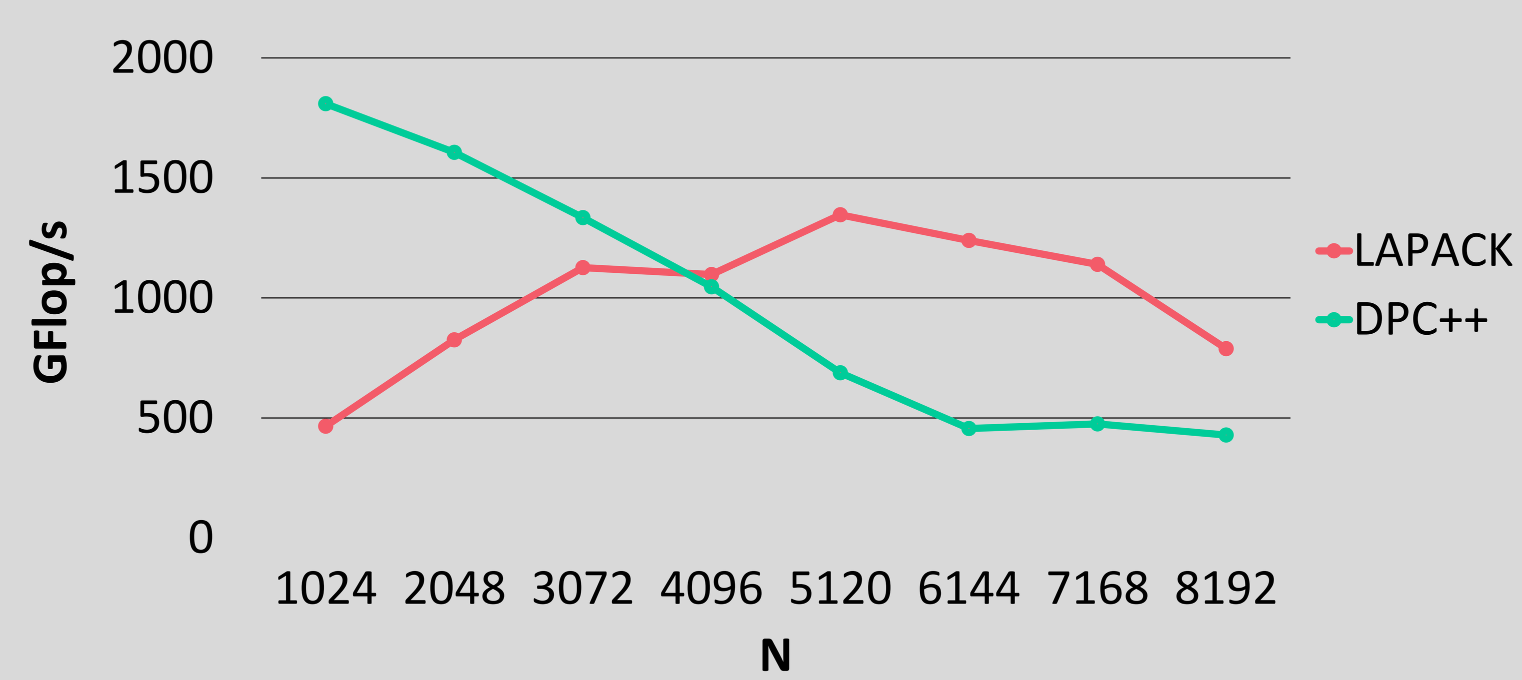

Figure 2: SGEMM Performance on AMD EPYC 7742 64-Core Processor @ 2.25GHz

Figures 2 and 3 compare a SGEMM code translated from CUDA to DPC++ and SGEMM from MKL.

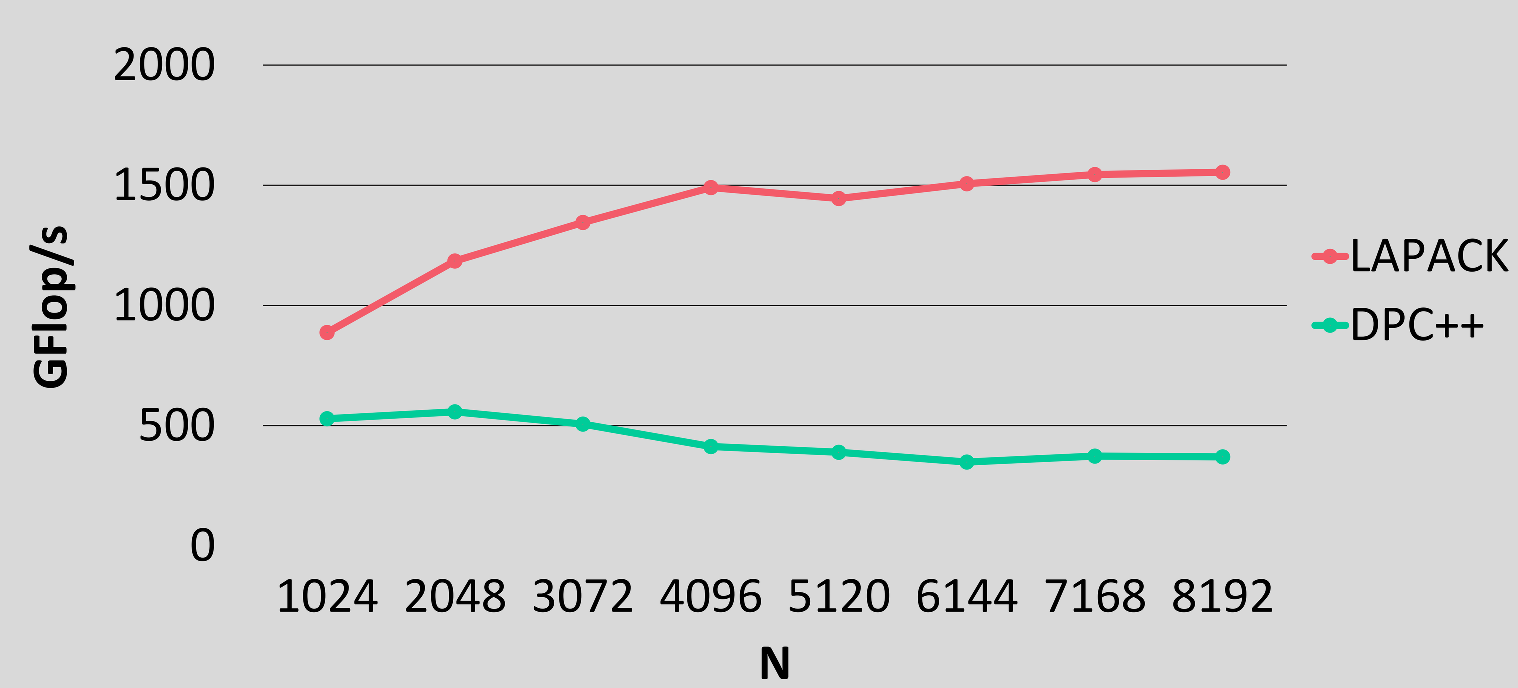

Figure 3: SGEMM Performance on Intel® Xeon® CPU E5-2698 V4 20-Core Processor @ 2.20GHz

GPU Performance

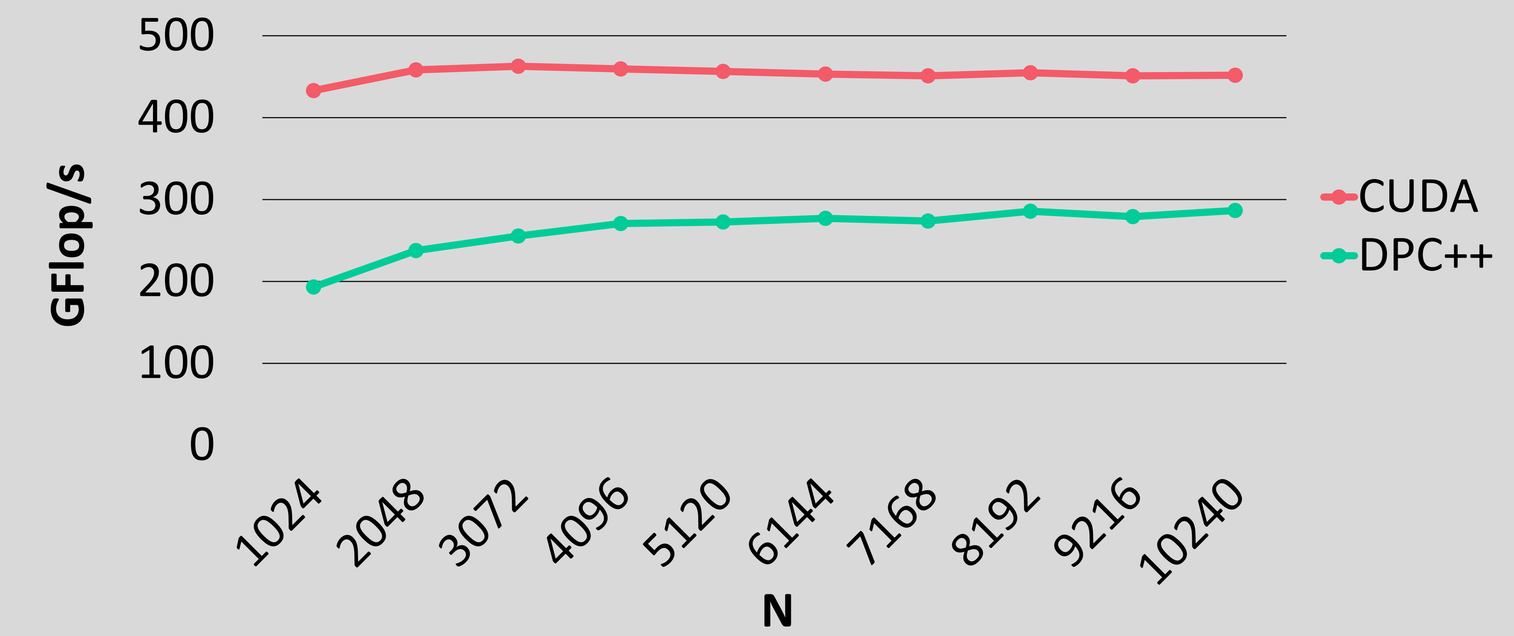

Figure 4: SGEMM Performance on Nvidia GeForce GTX 1650

To perform this test, an optimized single-precision general matrix multiplication (SGEMM) CUDA file was translated to DPC++ with the DPCT tool. It performs reasonably well on the Nvidia GPU. DPC++ achieves 45% of the original CUDA code performance when N equals 1024; this percentage increases to 63% when N equals 10240.

Conclusion and Future Directions

Intel's oneAPI proves to be a promising approach for parallel programming on heterogeneous systems. The DPCT tool can be used successfully for an initial port of CUDA code to DPC++. DPC++ code was successfully compiled and tested on Nvidia GPUs and multicore CPUs. Thus, the MAGMA port to DPC++ can be used to provide support for Intel GPUs, Nvidia GPUs, AMD GPUs, and multicore CPUs. DPC++ shows that large numerical libraries like MAGMA, originally written in CUDA to support Nvidia GPUs, can be easily translated to DPC++ to provide functional portability to different vendor GPUs, as well as multicore CPUs. Initial performance results show reasonable performance that can be further improved through hardware-specific tuning.

Acknowledgements

We would like to extend a special thanks to the National Science Foundation for funding this project through the RECSEM REU program hosted at the University of Tennessee at Knoxville.

References

1 Tomov, S., Nath, R., Ltaief, H. and Dongarra, J., 2010. Dense Linear Algebra Solvers for Multicore with {GPU} Accelerators In: *Proceedings of the IEEE IPDPS'10*. Atlanta: IEEE Computer Society, pp.1-8.
2 *oneAPI*. Intel. oneAPI Programming Model | oneAPI
3 *DPC++*. Intel. DPC++ — oneAPI Specification 1.1-rev-1 documentation
4 *Data Paralell C++: the oneAPI Implementation of SYCL*. Intel. DPC++
5 *DPCT*. Intel. Migrate CUDA* to DPC++ Code: Intel® DPC++ Compatibility Tool
6 *CUDA*. Nvidia. What Is CUDA | NVIDIA Official Blog
7 *oneMKL*. Intel. Intel oneAPI Math Kernel Library (oneMKL)
8 Compiling SYCL* for Different GPUs. Intel. Compiling SYCL with Different GPUs