Extending MAGMA Portability with OneAPI

Anna Fortenberry
University of North Texas

Stanimire Tomov (advisor)
Kwai Wong (advisor)
University of Tennessee, Knoxville

INTRODUCTION

Supercomputers (SC) provide computational power necessary to resolve problems in a vast number of important domains, such as High Performance Computing (HPC).

SC architectures are becoming increasingly diverse in architecture types and designs.

Eight of the top ten SCs ranked in the Top500 list use accelerators, coming from different vendors [1].

Researchers aim to join as a SC GPU vendor with their announcement of Aurora.

Irrespective of differences in code and hardware with this trend.

Intrinsically supports an architecture-independent programming model called onmex as a solution.

Similar, vector, scalar, and spatial code is supported.

Matrix Algebra on GPU and Multicore Architectures (MAGMA), a dense linear algebra library, can be used to test the capability and value of this programming model.

ONEAPI PROGRAMMING MODEL

OPTIMIZED APPLICATIONS

OMITTED MIDDLEWARE & FRAMEWORKS

OBSERVATION & ANALYSIS

STRUCTURE OF MAGMA

Listing 1 is a reference GEMM implementation of optimal block size which demonstrates that code never exceeded two G/Flops.

Parallel implementation rendered using OpenMP - i and j loops are collapsed to be performed in data-parallel fashion on different DPC++ threads.

Implementation is parameterized, allowing for tuning

Listing 1: Blocked OpenMP implementation of the SGEMM routine.

<table>
<thead>
<tr>
<th>DIM_N</th>
<th>DIM_K</th>
<th>DIM_X</th>
<th>DIM_Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>

Listing 2 shows a higher performance implementation of Listing 1, which looks for computation for higher memory reuse.

Listing 2: Blocked OpenMP implementation of the SGEMM algorithm.

<table>
<thead>
<tr>
<th>DIM_N</th>
<th>DIM_K</th>
<th>DIM_X</th>
<th>DIM_Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

In Table 1, function portability to different vendor GPUs performs well on multicore CPUs and retains optimal performance on NVIDIA GPUs. As well as multicore CPUs.

Initial migration code ran successfully on the Nvidia GPU with the DPC++-LLVM compiler. Achieved 100% usage, with watched nvidia-smi performance. CUDA denotes the generic SGEMM algorithm.

GEMM PARAMETERS

cuda set performed well on multicore CPUs.

DPC++ code ran successfully on the NVIDIA GeForce RTX 3060 GPU.

Able to outperform higher-bound MKL on the AMD CPU (MKL is tuned for Intel CPU).

Outperformed lower-bound on Intel CPU.

DPC++ (MAGMA) is a migrated generic CUDA SGEMM code.

REFERENCES

Applications 24, 4 (2010), 511-515. DOI:https:/ /doi.org/10.1177/1094342010385729

Rajib Nath, Stanimire Tomov, and Jack Dongarra. 2010. An Improved Magma Gemm For Fermi

August 5, 2022 from https:/ /github.com/NVIDIA/cuda-samples

CONCLUSION AND FUTURE DIRECTIONS

-oneAPI is a promising approach for portable, parallel programming on heterogeneous systems and various architectures.

-DPC++ useful for initial port of CUDA to DPC++

-Large numerical libraries written in CUDA can be easily translated to DPC++ to provide functional portability to different vendor GPUs, as well as multicore CPUs.

-Initial migration code tuned for NVIDIA GPUs performs well on multicore CPUs and retains performance on NVIDIA GPUs.

-Further sets of constants must be tested using autotuning techniques to discover the best performing versions for taking full advantage of the computational capabilities of the Intel GPU for GEMM algorithms.