EXTENDING MAGMA PORTABILITY

Final Presentation

Anna Fortenberry, UNT
UTK RECSEM REU 2022

Mentors: Dr. Stan Tomov, UTK and Dr. Kwai Wong, UTK
CONTENTS

- Problem Overview
- Software and Hardware
- Methodology
- CUDA to DPC++ Translation
- Porting MAGMA SGEMM
- Hardware Usage
- Performance
- Conclusion
PROBLEM OVERVIEW
Supercomputers provide the computational power necessary to resolve problems in a vast number of important domains:

- Data science
- Quantum information science
- Applied mathematics
- High performance computing
- Cybersecurity
- Artificial intelligence research

[1], [2], [3]
EVOLUTION OF SUPERCOMPUTER SYSTEM DESIGN

- NVIDIA opened a new door for supercomputing (SC) capabilities with the invention of the GPU in 1999.
- NVIDIA Tesla K20X GPU powered the first successful hybrid SC system in 2012.
- SC Systems are continually increasing in diversity.

[4], [5]
<table>
<thead>
<tr>
<th>JUNE 2022</th>
<th>CPU/ Accelerator</th>
<th>JUNE 2019</th>
<th>CPU/ Accelerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frontier</td>
<td>AMD, AMD</td>
<td>Summit</td>
<td>IBM, NVIDIA</td>
</tr>
<tr>
<td>S.C. Fugaku</td>
<td>Fugaku</td>
<td>Sierra</td>
<td>IBM, NVIDIA</td>
</tr>
<tr>
<td>LUMI</td>
<td>AMD, AMD</td>
<td>Sunway TaihuLight</td>
<td>Sunway</td>
</tr>
<tr>
<td>Summit</td>
<td>IBM, NVIDIA</td>
<td>Tianhe-2A</td>
<td>Intel</td>
</tr>
<tr>
<td>Sierra</td>
<td>IBM, NVIDIA</td>
<td>Frontera</td>
<td>Intel</td>
</tr>
<tr>
<td>Sunway TaihuLight</td>
<td>Sunway</td>
<td>Piz Daint</td>
<td>Intel, NVIDIA</td>
</tr>
<tr>
<td>Perlmutter</td>
<td>AMD, NVIDIA</td>
<td>Trinity</td>
<td>Intel</td>
</tr>
<tr>
<td>Selene</td>
<td>AMD, NVIDIA</td>
<td>ABCI</td>
<td>Intel, NVIDIA</td>
</tr>
<tr>
<td>Tianhe-2A</td>
<td>Intel, NUDT</td>
<td>SuperMUC-NG</td>
<td>Intel</td>
</tr>
<tr>
<td>Adastra</td>
<td>AMD, AMD</td>
<td>Lassen</td>
<td>IBM, NVIDIA</td>
</tr>
</tbody>
</table>

[6], [7]
Anticipated for release in late 2022, Intel hopes to enter the supercomputer GPU vendor domain by powering the Aurora supercomputer at Argonne National Laboratory.
INTEL ONEAPI

- Intel recently released a new programming model called **oneAPI**
- Applications that take advantage of oneAPI gain portability to all supported hardware platforms
  - CPUs (Scalar Architecture)
  - GPUs (Vector Architecture)
  - FPGAs (Spatial Architecture)
  - Other Accelerators (Matrix Architecture)
Designed originally to run on NVIDIA GPUs
Extended to support AMD GPUs
oneAPI includes tools for adopting the model
- Data Parallel C++ (DPC++) Translation Tool (DPCT)
- oneAPI Math Kernel Library (oneMKL)
RESEARCH QUESTIONS

- How well does the DPCT tool translate CUDA code to DPC++ code?
- What are the common translation errors?
- Can this tool be used to translate MAGMA?
- Is DPC++ portable to Nvidia and AMD GPUs, and multicore CPUs?
- What is the performance of DPC++ on each of these accelerators comparative to CUDA?
SOFTWARE AND HARDWARE
OPTIMIZED APPLICATIONS

OPTIMIZED MIDDLEWARE & FRAMEWORKS

DIRECT PROGRAMMING
Data Parallel C++ (DPC++)

API-BASED PROGRAMMING
oneAPI Libraries

Analysis & Debug Tools

DPC++ is a oneAPI implementation of the Khronos standard SYCL
SYCL is an accelerator language that allows code reuse across hardware targets
SYCL adds data parallelism and heterogeneous programming to standard ISO C++

[10], [11]
SOFTWARE OVERVIEW

DPC++ Compatibility Tool (DPCT)
oneAPI tool to assist with migrating CUDA code to DPC++ code; translates with high accuracy

DPC++-LLVM (CLang-LLVM)
LLVM-based compiler project that supports SYCL language

oneAPI Math Kernel (oneMKL)
set of math routines for use in high performance computing on a variety of computational devices

DPC++ LLVM NVIDIA*
CLANG-LLVM build on Linux with CUDA NVIDIA support; allows DPC++ to port to NVIDIA GPUs

Compute Unified Device Architecture (CUDA)
NVIDIA parallel computing platform for harnessing power of GPUs

Intel DevCloud
Remote development environments that grant access to Intel hardware for testing oneAPI projects*

[12], [13], [14], [15], [20]
<table>
<thead>
<tr>
<th>CENTRAL PROCESSING UNITS</th>
</tr>
</thead>
</table>
| **AMD EPYC 7742**  
**PROCESSOR** |
| **INTEL® XEON®**  
**PROCESSOR E5-2698 V4** |
| **Cores:** | 64 | **Cores:** | 20 |
| **Base Clock:** | 2.25 Ghz | **Base Clock:** | 2.20 Ghz |
| **# of Threads:** | 128 | **# of Threads:** | 40 |
| **Cache:** | 256 MB | **Cache:** | 50 MB |

[16], [17]
<table>
<thead>
<tr>
<th>Graphics Processing Unit</th>
<th>GPU Cores</th>
<th>Base Clock</th>
<th>Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVIDIA GeForce RTX 3060 (Discrete)</td>
<td>3584</td>
<td>1320 MHz</td>
<td>12 GB</td>
</tr>
<tr>
<td>Intel UHD Graphics P630 [0x3e96] (Integrated)</td>
<td>192</td>
<td>350 MHz</td>
<td>Shared System</td>
</tr>
</tbody>
</table>

[18], [19]
3 METHODOLOGY
Configure system to run DPC++ code and document the process

Document translation process and common errors

Repeat steps 1-4 on Innovative Computing Lab (ICL) Account

Migrate CUDA code in directory to DPC++ with DPCT and correct errors

Translate different structures of CUDA files to DPC++ with DPCT for correctness

Configure system to run DPC++ code on Nvidia GPU

Set up directory with MAGMA CUDA sgemm and dependencies

Test and compare performance of sgemm on available hardware
CUDA TO DPC++ TRANSLATION
SIMPLE KERNEL TRANSLATION

```c
__global__ void VectorAddKernel(float* A, float* B, float* C)
{
    A[threadIdx.x] = threadIdx.x + 1.0f;
    B[threadIdx.x] = threadIdx.x + 1.0f;
    C[threadIdx.x] = A[threadIdx.x] + B[threadIdx.x];
}

void VectorAddKernel(float* A, float* B, float* C, sycl::nd_item<3> item_ct1)
{
    A[item_ct1.get_local_id(2)] = item_ct1.get_local_id(2) + 1.0f;
    B[item_ct1.get_local_id(2)] = item_ct1.get_local_id(2) + 1.0f;
    C[item_ct1.get_local_id(2)] =
        A[item_ct1.get_local_id(2)] + B[item_ct1.get_local_id(2)];
}
```
TEST 1: ISOLATED FILE

- Translated files for CUDA vector addition and vector-matrix multiplication
- 100% compilation and execution accuracy
- CUDA error handling dead code clean up for file readability

/* DPCT1003:30: Migrated API does not return error code. (*, 0) is inserted. You may need to rewrite this code. */

checkCudaErrors((h_C = (float*)sycl::malloc_host(mem_size_C, dpct::get_default_queue()), 0));

-> h_C = (float*)sycl::malloc_host(mem_size_C, dpct::get_default_queue());
TEST 2: FILE WITH HEADERS

- Matrix-matrix multiplication file with six included headers
- 98.7% compilation accuracy and 98.0% execution accuracy in the main file
- 10% of the code needed dead code touchups
- Header files had 100% compilation accuracy and execution accuracies ranging from 75%-100%
cudaGetDeviceCount(&device_count);

while (current_device < device_count) {
    cudaGetDeviceProperties(&deviceProp, current_device);
    if (deviceProp.computeMode != cudaComputeModeProhibited) {
        . . . }
    else {
        devices_prohibited++;
    }
    current_device++;
}

device_count = dpct::dev_mgr::instance().device_count();

while (current_device < device_count) {
    dpct::dev_mgr::instance().get_device(current_device).
        get_device_info(deviceProp);
    if (true) {
        . . . }
    else {
        devices_prohibited++;
    }
    current_device++;
}
PORTING MAGMA SGEMM
Implementation is templated with 9 parameters

Computation is done with thread blocks of size $[\text{DIM}_X, \text{DIM}_Y]$

Thread $t_{ij}$ computes $[\text{DIM}_M / \text{DIM}_X, \text{DIM}_N / \text{DIM}_Y]$ elements of $C_{ij}$

$A_{ik}$ gets loaded in shared memory by $[\text{DIM}_{XA}, \text{DIM}_{YA}]$ threads

$B_{kj}$ gets loaded in shared memory by $[\text{DIM}_{XB}, \text{DIM}_{YB}]$ threads

$C_{ij}$ is held and computed in registers
MAGMA SGEMM TRANSLATION PROCESS

- Collected MAGMA SGEMM CUDA code and dependencies in one directory
- Used DPCT to recursively migrate CUDA code to DPC++
- Translated header files that did not migrate independently in a separate directory and then copied them into the MAGMA SGEMM directory
- Implemented compiler directives as needed
HARDWARE USAGE
## MULTICORE CPUs

### System Information

- **CPU Model**: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz
- **Load Average**: 15.02 13.19 12.02
- **Uptime**: 23 days, 05:45:41

### System Memory

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Usage</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>7.24G</td>
<td>47.1G</td>
</tr>
</tbody>
</table>

### Top Processes

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PRI</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHRM</th>
<th>SRM</th>
<th>CPU%</th>
<th>MEM%</th>
<th>TIME+</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>3602506</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>15h41:40</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602527</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:18:57</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602528</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:18:57</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602523</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:19:05</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602526</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:18:44</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602515</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:16:11</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602521</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:18:58</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602519</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:19:47</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602534</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:19:03</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602518</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:18:50</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602516</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:18:09</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3602525</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>13.1G</td>
<td>1628M</td>
<td>267M</td>
<td>1178</td>
<td>3.4</td>
<td>11:18:23</td>
<td>/IntelCpuExec -wA=8192 -wB=8192 -hA=8192 -hB=8192</td>
<td></td>
</tr>
<tr>
<td>3603925</td>
<td>user1</td>
<td>20</td>
<td>0</td>
<td>11708</td>
<td>5128</td>
<td>3228</td>
<td>2.6</td>
<td>0.6</td>
<td>0:16:55</td>
<td>htop</td>
<td></td>
</tr>
</tbody>
</table>

**Tasks:** 207, 1038 thr; 12 running
### MULTICORE CPUs

<table>
<thead>
<tr>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>37</th>
<th>38</th>
<th>39</th>
<th>40</th>
<th>41</th>
<th>42</th>
<th>43</th>
<th>44</th>
<th>45</th>
<th>46</th>
<th>47</th>
<th>48</th>
<th>49</th>
</tr>
</thead>
<tbody>
<tr>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
<td>105</td>
<td>106</td>
<td>107</td>
<td>108</td>
<td>109</td>
<td>110</td>
<td>111</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td></td>
</tr>
</tbody>
</table>

**AMD EPYC 7742 64-Core Processor**
<table>
<thead>
<tr>
<th>GPU Name</th>
<th>Fan Temp</th>
<th>Perf</th>
<th>Pwr:Usage/Cap</th>
<th>Bus-Id Memory-Usage</th>
<th>Volatile Uncorr. ECC</th>
<th>GPU-Util</th>
<th>Compute M. MIG M.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVIDIA GeForce GTX 1650</td>
<td>35% 63C</td>
<td>P3</td>
<td>60W / 100W</td>
<td>1345MiB / 3909MiB</td>
<td>N/A</td>
<td>100%</td>
<td>Default N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processes:</th>
<th>GPU</th>
<th>GI</th>
<th>CI</th>
<th>PID</th>
<th>Type</th>
<th>Process name</th>
<th>GPU Memory Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 N/A N/A</td>
<td>1239</td>
<td>G</td>
<td>/usr/lib/xorg/Xorg</td>
<td>23MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 N/A N/A</td>
<td>240692</td>
<td>G</td>
<td>/usr/lib/xorg/Xorg</td>
<td>241MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 N/A N/A</td>
<td>240826</td>
<td>G</td>
<td>/usr/bin/gnome-shell</td>
<td>25MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 N/A N/A</td>
<td>258953</td>
<td>G</td>
<td>...RendererForSitePerProcess</td>
<td>13MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 N/A N/A</td>
<td>3368334</td>
<td>G</td>
<td>/usr/lib/firefox/firefox</td>
<td>111MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 N/A N/A</td>
<td>3634687</td>
<td>C</td>
<td>office/program/soffice bin</td>
<td>47MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 N/A N/A</td>
<td>3643058</td>
<td>C</td>
<td>./cudaGpuExec</td>
<td>862MiB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PERFORMANCE
CUDA =
-DMAGMA_TUNING
-DDIM_X=16
-DDIM_Y=16
-DBLK_M_nn=96
-DBLK_N_nn=96
-DBLK_K_nn=16
-DDIM_XA=32
-DDIM_YA=8
-DDIM_XB=8
-DDIM_YB=32

TEST PARAMETERS

C = A B

template < 16, 16, 96, 96, 16, 32, 8, 8, 32>

For I = 1 .. M step 16
For J = 1 .. N step 16
For K = 1 .. K step 16
C_{IJ} += A_{IK} B_{KJ}

- Thread t_j computes \([96 / 16, 96 / 16]\) elements of C_{IJ}
- A_{IK} gets loaded in shared memory by \([32, 8]\) threads
- B_{KJ} gets loaded in shared memory by \([8, 32]\) threads
- C_{IJ} is held and computed in registers
AMD EPYC 7742 64-CORE PROCESSOR @ 2.25GHZ

DPC++ (MAGMA)
MKL
DPC++ (CUDA)
C++ (OpenMP)
INTEL® XEON® CPU E5-2698 V4 20-CORE PROCESSOR @ 2.20GHZ
INTEL UHD GRAPHICS P630 [0x3e96]
<table>
<thead>
<tr>
<th></th>
<th>DIM_X</th>
<th>DIM_Y</th>
<th>DIM_M</th>
<th>DIM_N</th>
<th>DIM_K</th>
<th>DIM_XA</th>
<th>DIM_YA</th>
<th>DIM_XB</th>
<th>DIM_YB</th>
</tr>
</thead>
<tbody>
<tr>
<td>cuda</td>
<td>16</td>
<td>16</td>
<td>96</td>
<td>96</td>
<td>16</td>
<td>32</td>
<td>8</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>ker2</td>
<td>16</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td>8</td>
<td>32</td>
<td>8</td>
<td>8</td>
<td>32</td>
</tr>
<tr>
<td>ker11</td>
<td>12</td>
<td>4</td>
<td>48</td>
<td>48</td>
<td>2</td>
<td>24</td>
<td>2</td>
<td>24</td>
<td>2</td>
</tr>
</tbody>
</table>
CONCLUSION
SUMMARY

- oneAPI is a promising approach for parallel programming across various architectures
- DPCT tool can be used successfully for an initial port of CUDA code to DPC++
- 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
SUMMARY

◦ Initial migrated code tuned for Nvidia GPUs performs well on multicore CPUs
◦ Initial migrated code tuned for Nvidia GPUs retains performance on Nvidia GPUs
◦ Initial migrated code tuned for Nvidia GPUs performs poorly on the available Intel GPU
  ◦ Tuning is required, but optimal parameters are difficult to find without further knowledge on the hardware design
ONGOING AND FUTURE WORKS

- Full translation of MAGMA
- ICL account configuration
- Finding near optimal parameters for the Intel integrated GPU
- Testing migrated code on discrete Intel GPU upon release
ACKNOWLEDGEMENTS

This research was funded by the National Science Foundation.

This research was conducted at the University of Tennessee at Knoxville through the RECSEM REU.

Home University
University of North Texas
REFERENCES


REFERENCES


REFERENCES


[16] AMD EPYC™ 7742. AMD. 
REFERENCES


Presentation Template: