

Performance Portability of Shallow Water Model with DPC++

Leila Ghaffari and Zephaniah Connell

Mentors: Supreeth Suresh, Cena Miller, Jian Sun, and John Dennis



SIParCS 2021
July 27, 2021



Motivation

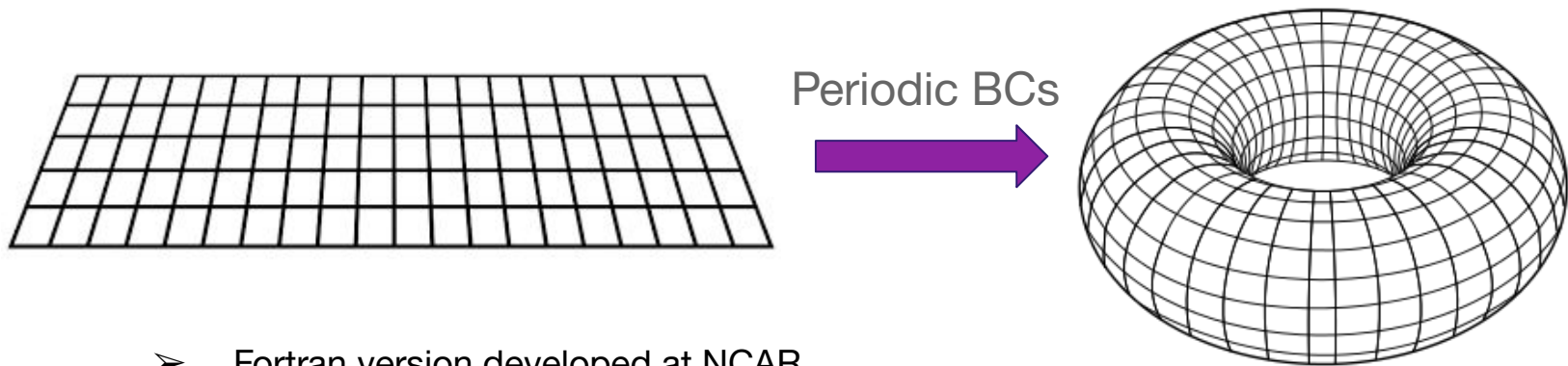
- ❑ Increase in the computational capacity of high-performance computing platforms
- ❑ GPUs could save energy to get the same amount of work done (higher performance)
- ❑ Weather and Climate models are usually computationally expensive and suited for parallelization.
- ❑ Execute single source code on different CPU and GPU platforms

Goals

- ❑ Port a Weather and Climate mini-app (SWM) to DPC++ with limited modifications
- ❑ Optimize the performance of the ported code on different CPU and GPU platforms

Shallow Water Model (SWM) mini-app

A venerable 2D shallow water model benchmark on staggered finite difference equations on a torus



- Fortran version developed at NCAR
- C version used by the UK Met Office as a mini-app
- C++ version developed at NCAR in 2021
- Part of SPEC-FP benchmark suite for many years.
- Written when peak flops and bandwidth were comparable

Reference: The Dynamics of Finite-Difference Models of the Shallow-Water Equations, by Robert Sadourny, J. Atm. Sciences, Vol 32, No 4, April 1975, p.680-688.

Programming Model - oneAPI?

Optimized Applications

Optimized Middleware & Frameworks

oneAPI Product

Direct Programming

Data Parallel C++

API-Based Programming

Libraries*

Analysis & Debug
Tools**

CPU

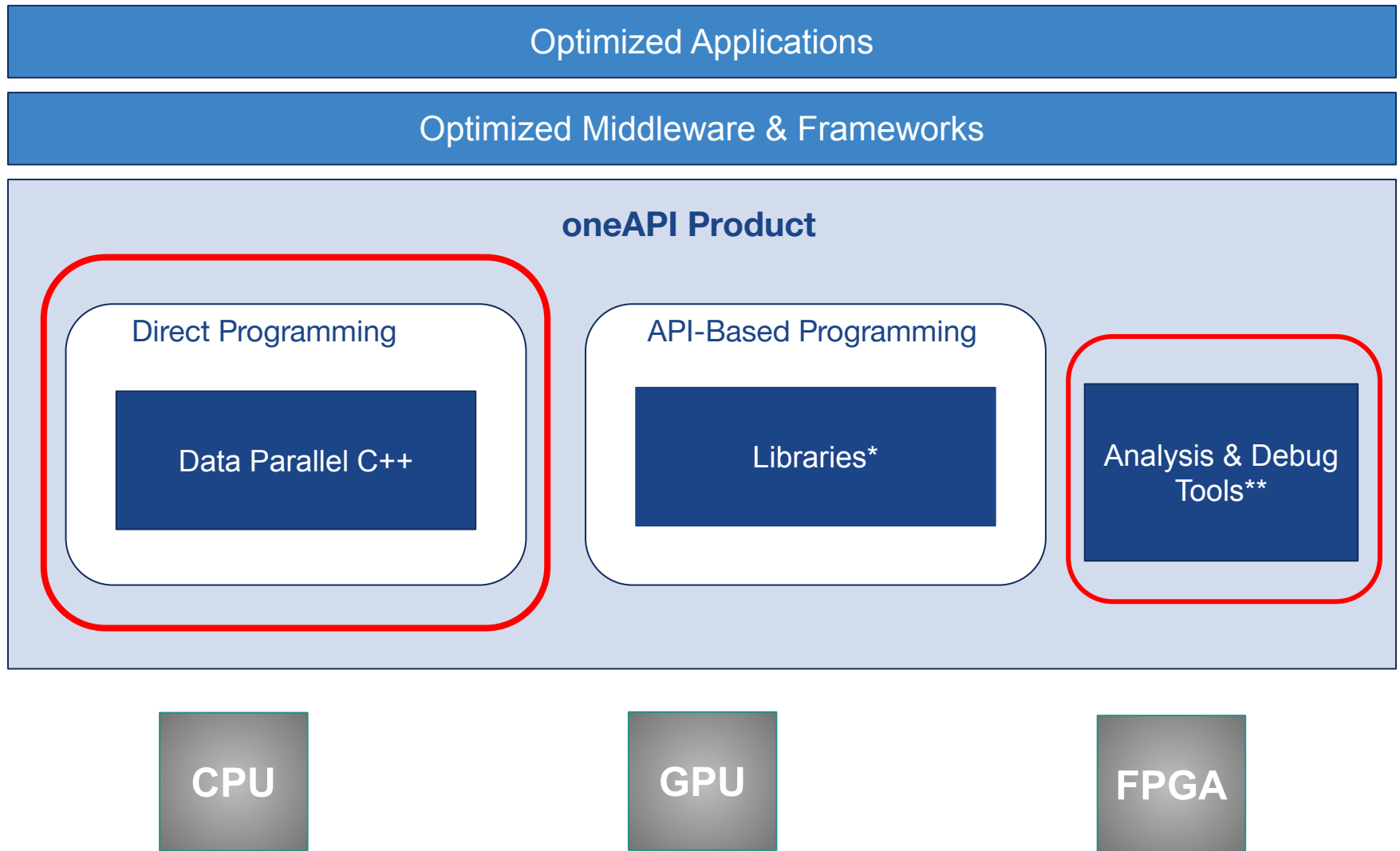
GPU

FPGA

*Libraries: oneCCL, oneDAL, oneDNN, oneDPL, oneMKL, oneTBB, oneVPL

**Tools: Intel Advisor, Intel VTune Profiler, Intel-enhanced GDB

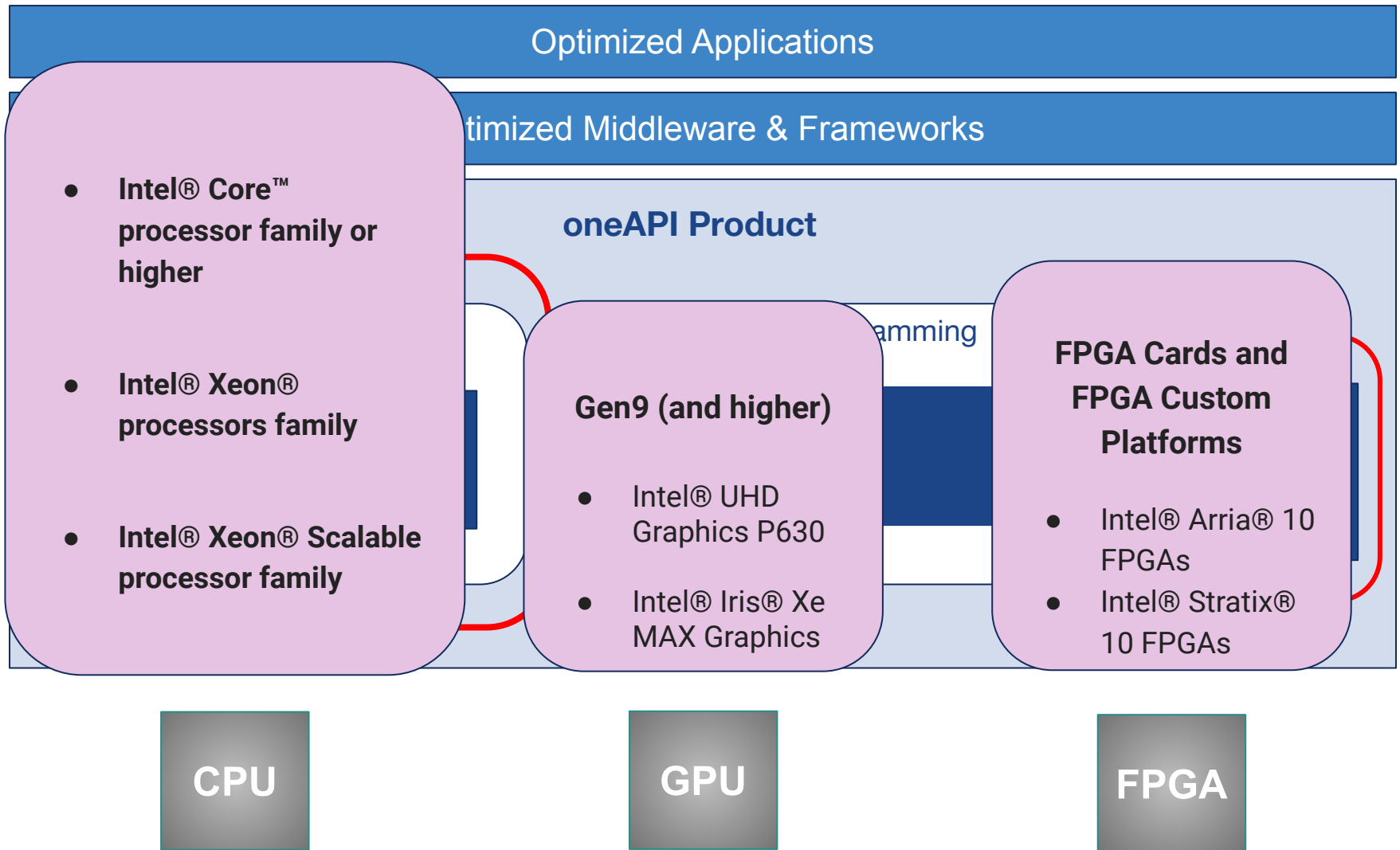
Programming Model - oneAPI?



*Libraries: oneCCL, oneDAL, oneDNN, oneDPL, oneMKL, oneTBB, oneVPL

**Tools: Intel Advisor, Intel VTune Profiler, Intel-enhanced GDB

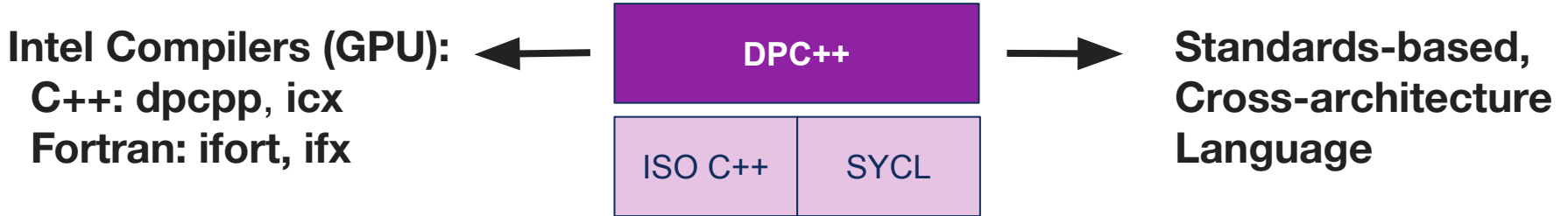
Programming Model - oneAPI?



*Libraries: oneCCL, oneDAL, oneDNN, oneDPL, oneMKL, oneTBB, oneVPL

**Tools: Intel Advisor, Intel VTune Profiler, Intel-enhanced GDB

What is Data Parallel C+ (DPC++)?



Queue

Submits command groups to be executed by the SYCL runtime

Device Selector

Queue is submitted to the device through device selectors.

- `gpu_selector`
- `cpu_selector`
- `default_selector`
- `host_selector`
- `intel::fpga_selector`

Memory Model

- **Unified Shared Memory:** pointer-based approach
- **Buffers:** Encapsulate data in a SYCL application
 - **Accessors:** Mechanism to access buffer data

Kernels

Encapsulates methods and data for executing code on the device

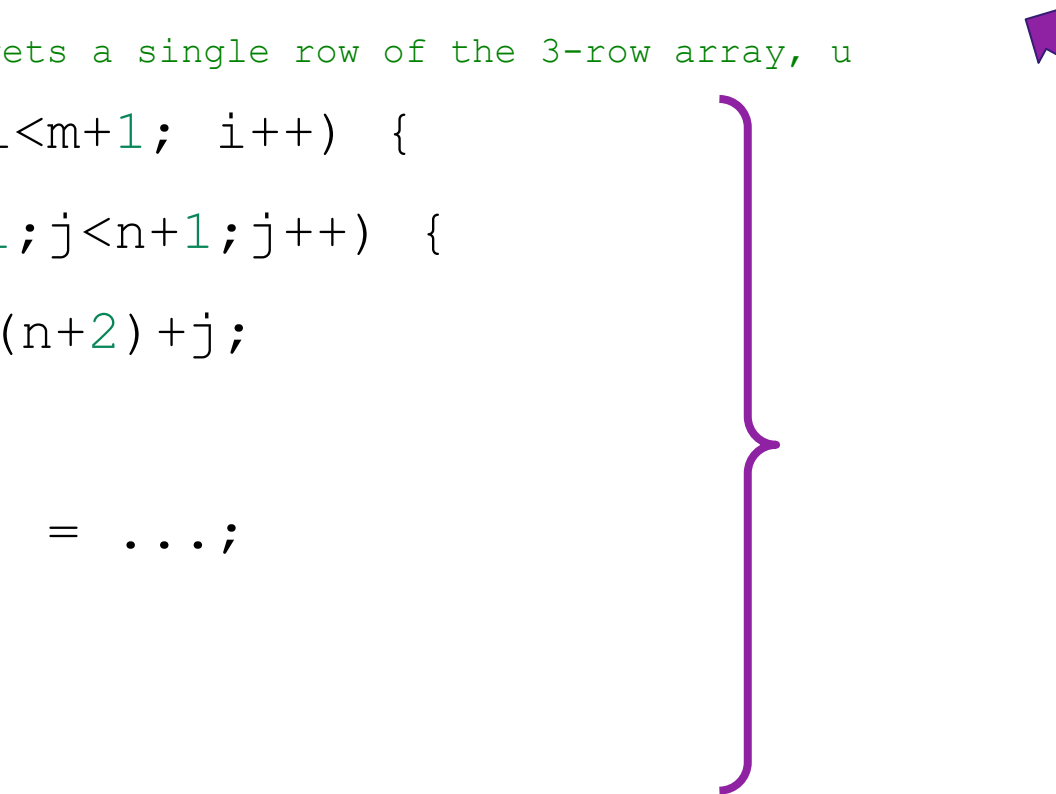
DPC++	Kokkos	CUDA	HIP
queue	Policy Instance	stream	hipStream_t
ND_range	league	grid	grid
work_group	team	block	workgroup
sub_group	vector	warp	wavefront
work_item	thread/ rank	thread	thread/work item

More info: <https://wiki.ucar.edu/x/XgXUGg>

Reference: J. Reinders, B. Ashbaugh, J. Brodman, M. Kinsner, J. Pennycook, X. Tian, vectors. In: Data Parallel C++, p. 259–276. Apress (2020).

SWM code - Application of oneAPI

```
// Initialize velocities
DOMAIN_SIZE = (m+2) * (n+2); // Size of each array row
double u[3][DOMAIN_SIZE]; // 2D array with 3 rows (time levels)
// For-loop which targets a single row of the 3-row array, u
for (int i=1; i<m+1; i++) {
    for (int j=1; j<n+1; j++) {
        int ij = i*(n+2)+j;
        ...
        u[0][ij] = ...;
    }
}
```



Serial

Buffer

Unified Shared
Memory

SWM code - Application of oneAPI

```
double u[3][DOMAIN_SIZE];  
auto R = range<1>{DOMAIN_SIZE};  
buffer<double, 1> u0_buf(u[0], R);  
q.submit([&](handler &h) {  
    auto u0 = u0_buf.get_access(h, write_only);  
    ...  
    h.parallel_for(R, [=](auto ij) {  
        int j = ij%(n+2);  
        int i = (int) (ij - j)/(n+2);  
        ...  
        if (i==0 || j==0 || i == m+1 || j== n+1) {} else  
        { u0[ij] = ...; } }); });
```

Serial

Buffer

Unified Shared
Memory

SWM code - Application of oneAPI

```
double **u = malloc_shared<double *>(3*DOMAIN_SIZE, q);
for(int i=1; i<m+1; i++)
    u[i] = malloc_shared<double>(DOMAIN_SIZE, q);
auto R = range<1>{DOMAIN_SIZE};
q.parallel_for(R, [=](auto ij) {
    int j = ij%(n+2);
    int i = (int) (ij - j)/(n+2);
    ...
    if (i==0 || j==0 || i == m+1 || j== n+1) {}
    else {
        u[ij] = ...;
    } });
```


Serial

Buffer

Unified Shared
Memory

SWM code - Application of oneAPI

```
double **u = malloc_shared<double *>(3*DOMAIN_SIZE, q);  
for(int i=1; i<m+1; i++)  
    u[i] = malloc_shared<double>(DOMAIN_SIZE, q);  
auto R = range<1>{DOMAIN_SIZE};  
q.parallel_for(R, [=](auto ij) {  
    int j = ij%(n+2);  
    int i = (int) (ij - j)/(n+2);  
    ...  
    if (i==0 || j==0 || i == m+1 || j== n+1) {}  
    else {  
        u[ij] = ...;  
    } });
```



Serial

Buffer

Unified Shared
Memory

SWM code - Application of oneAPI

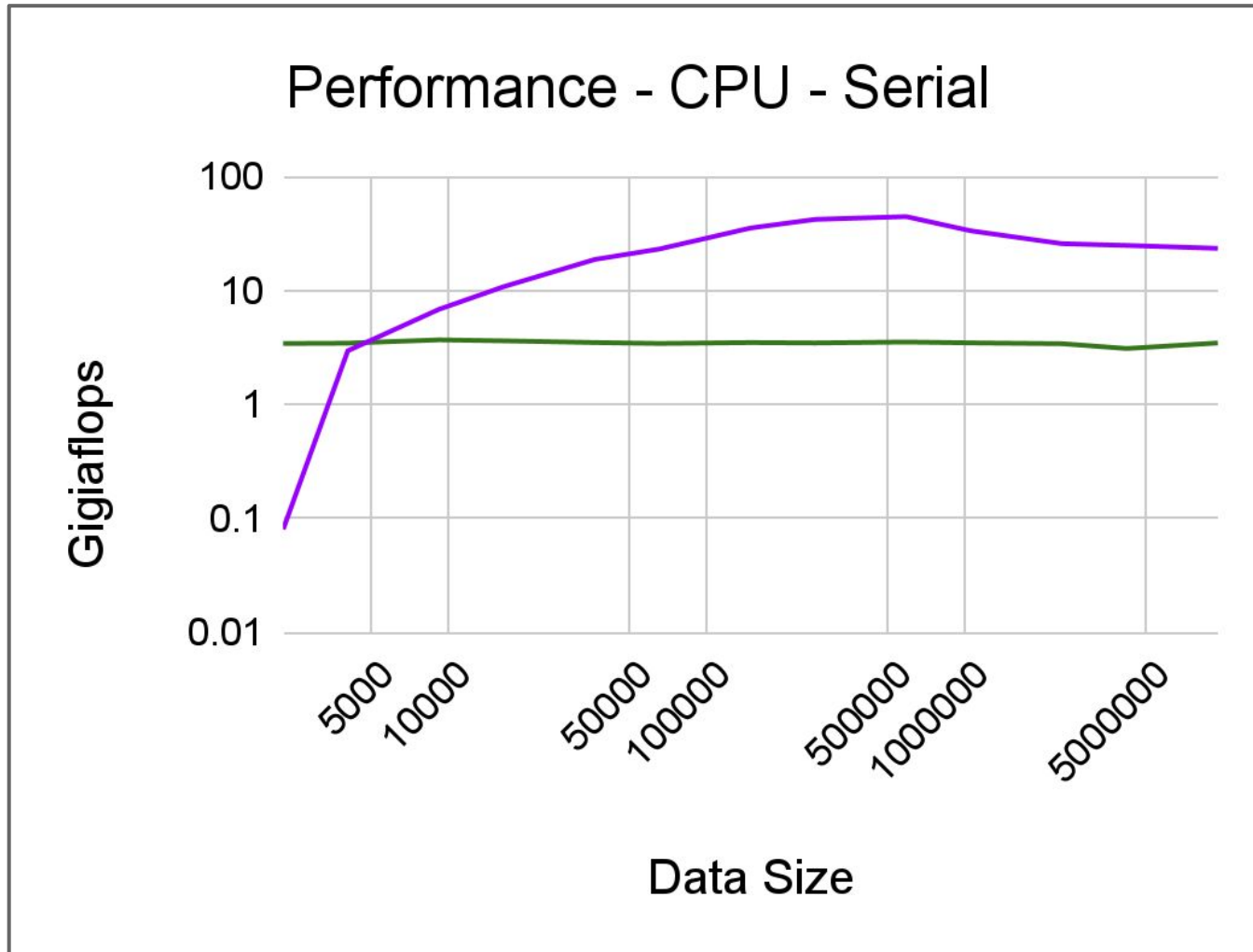
```
double **u = malloc_shared<double *>(3*DOMAIN_SIZE, q);  
for(int i=1; i<m+1; i++)  
    u[i] = malloc_shared<double>(DOMAIN_SIZE, q);  
auto R = range<1>{DOMAIN_SIZE};  
q.parallel_for(R, [=](auto ij) {  
    int j = ij%(n+2);  
    int i = (int) (ij - j)/(n+2);  
    ...  
    if (i==0 || j==0 || i == m+1 || j== n+1) {}  
    else {  
        u[ij] = ...;  
    } });
```

Serial

Buffer

Unified Shared
Memory

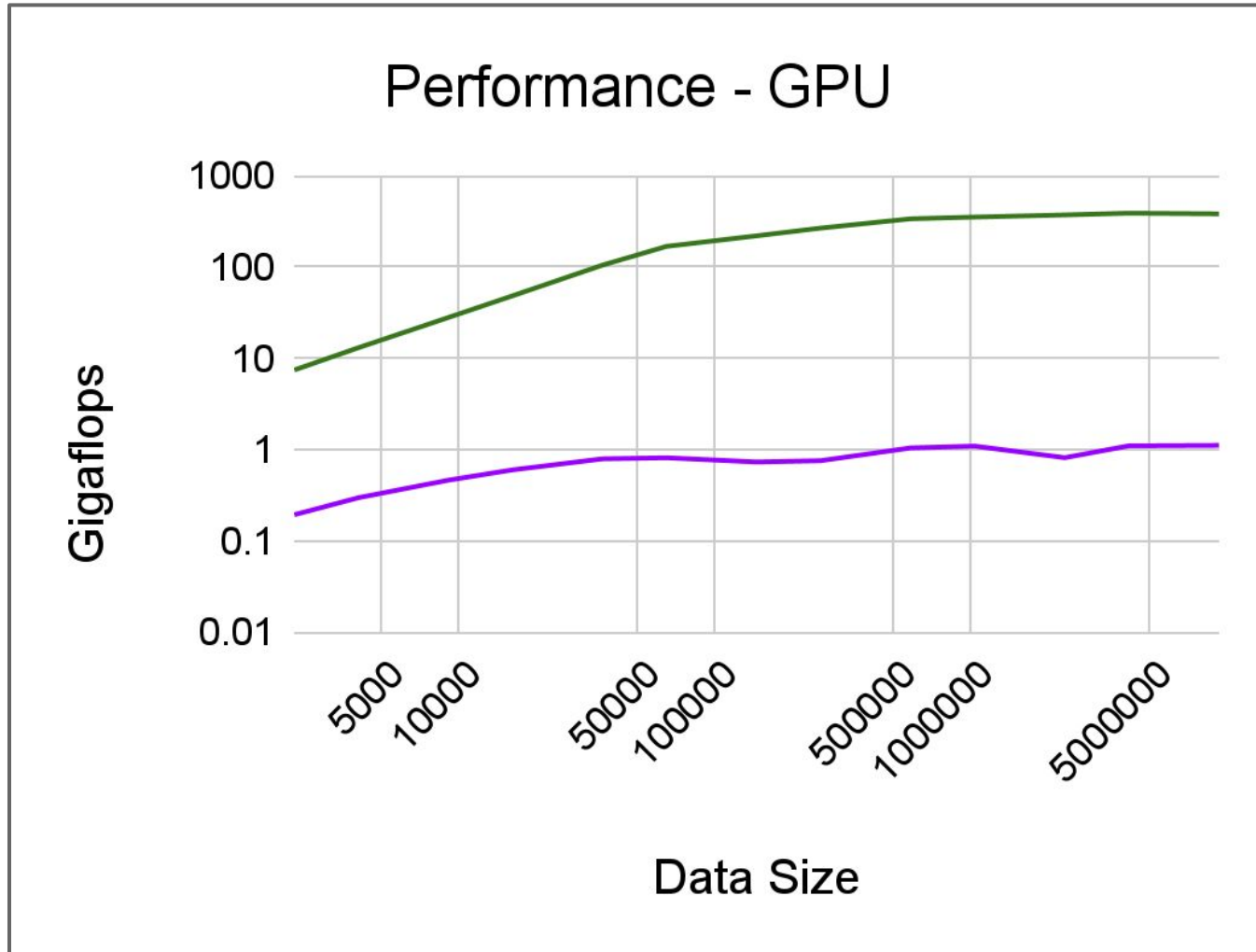
Results



— Skylake - Single Core
C++ (gnu/8.3.0 -O2)

— Skylake - Single Core
DPC++ (dpcpp -O2)

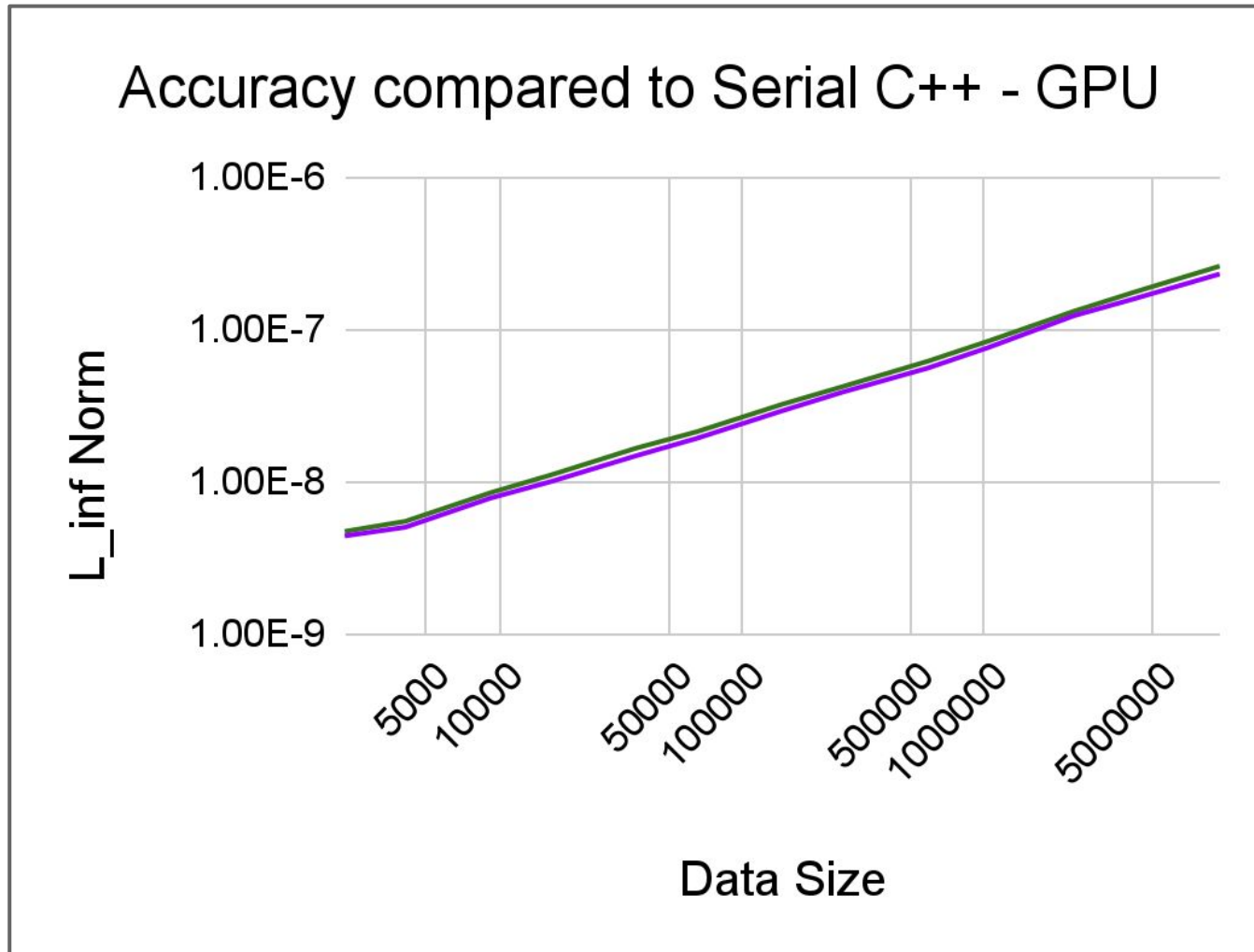
Results



— Casper V100
OpenACC (nvhpc/21.3 cuda/10.2 -O2)

— Intel® Iris® Xe MAX
DPC++ (dpcpp -O2)

Results



— Casper V100
OpenACC (nvhpc/21.3 cuda/10.2 -O2)

— Intel® Iris® Xe MAX
DPC++ (dpcpp -O2)

Conclusions

	USM	Buffer
Portability	●●●	●
Easiness	●●	●●
Syntax	●●●	●●
Compiler	●●●	●●
CPU Performance	●●●●●	?
GPU Performance	●	?
Documentation & Support	●●●	

Future Work

- More investigation on the buffer model
- More optimization on the USM model
- Change the data structure in the SWM mini-app
- Run the code on Nvidia and AMD GPUs
- Compile and run the ported code on FPGAs
- Add support for OpenMP



Acknowledgements

Mentors: Supreeth Suresh, Cena Miller, Jian Sun, and John Dennis

Research Support: Richard Loft and Thomas Hauser

SIParCS Admins and CODE Assistants: AJ Lauer, Virginia Do, Jerry Cyccone, Max Cordes Galbraith

Thank you!

Leila.Ghaffari@colorado.edu