# Assignment I:

## Exercise 1 - Reflection on GPU-accelerated Computing

**1. List the main differences between GPUs and CPUs in terms of architecture.**

**1. Parallel:** GPU is highly parallel processors which can handle multiple tasks simultaneously. GPU has large number of cores, each capable of executing its own set of instructions independently.

CPU is optimized for single-threaded performance and tasks that require more sequential processing. CPU has higher clock speeds and more complex cores.

**2. Number of Cores:** GPU has larger number of simple cores. This allows them to process many simple tasks simultaneously, making them suitable for parallelizable tasks.

CPUs has fewer but more powerful cores that are optimized for handling complex tasks and executing a variety of instructions.

**2.Check the latest Top500 list that ranks the top 500 most powerful supercomputers in the world. In the top 10, how many supercomputers use GPUs? Report the name of the supercomputers and their GPU vendor (Nvidia, AMD, . . . ) and model.** 8 supercomputers ues GPUs in top10.

| name | GPU vendor | model |
|---|---|---|
| FRONTIER | AMD | HPE Cray EX235a |
| LUMI | AMD | HPE Cray EX235a |
| Leonardo | Nvidia | BullSequana XH2000 |
| Summit | Nvidia | IBM Power SystemAC922 |
| Sierra | Nvidia | IBM Power SystemS922LC |
| Perlmutter | Nvidia | HPE Cray EX235a |
| Selene | Nvidia | Nvidia |

**3. One main advantage of GPU is its power efficiency, which can be quantified by Performance/Power, e.g., throughput as in FLOPS per watt power consumption. Calculate the power efficiency for the top 10 supercomputers. (Hint: use the table in the first lecture)**

| name | Rpeak (PFlop/s) | Power (kW) | power efficiency |
|---|---|---|---|
| FRONTIER | 1,679.82 | 22,703 | 0.074 |
| Supercomputer Fugaku | 537.21 | 29,899 | 0.018 |
| LUMI | 428.70 | 6,016 | 0.071 |
| Leonardo | 304.47 | 7,404 | 0.041 |
| Summit | 200.79 | 10,096 | 0.020 |
| Sierra | 125.71 | 7,438 | 0.017 |
| Sunway TaihuLight | 125.44 | 15,371 | 0.008 |
| Perlmutter | 93.75 | 2,589 | 0.036 |
| Selene | 79.22 | 2,646 | 0.030 |
| Tianhe-2A | 100.68 | 18,482 | 0.005 |

## Exercise 2 - Query Nvidia GPU Compute Capability

**1. The screenshot of the output from running deviceQuery test in /1_Utilities.**

```
./deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Tesla T4"
  CUDA Driver Version / Runtime Version          12.0 / 11.8
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 15102 MBytes (15835398144 bytes)
  (040) Multiprocessors, (064) CUDA Cores/MP:    2560 CUDA Cores
  GPU Max Clock rate:                            1590 MHz (1.59 GHz)
  Memory Clock rate:                             5001 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 4194304 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        65536 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1024
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 3 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 4
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.0, CUDA Runtime Version = 11.8, NumDevs = 1
Result = PASS
```

Figure 1: The screenshot of running deviceQuery test

**2. What is the Compute Capability of your GPU device?** As shown in the output from deviceQuery, the compute capability of the GPU device is "7.5."

**3. The screenshot of the output from running bandwidthTest test in /1_Utilities.**

**4. How will you calculate the GPU memory bandwidth (in GB/s) using the output from deviceQuery? (Hint: memory bandwidth is typically determined by clock rate and bus width, and check what double date rate (DDR) may impact the bandwidth). Are they consistent with your results from bandwidthTest?** To calculate the GPU memory bandwidth (in GB/s) using the output from deviceQuery, the memory bandwidth can be calculated using the following formula:

Memory Bandwidth (GB/s) = Memory Clock Rate (GHz) x Memory Bus Width (bits) / 8

In the provided output, the memory clock rate is 5.001 MHz, and the memory bus width is 256 bits. The memory bandwidth should be:

Memory Bandwidth (GB/s) = 5.001 GHz x 256 bits / 8 x 2 = 320.064 GB/s

The calculated memory bandwidth is approximately 320.064 GB/s based on the output of deviceQuery.

## Exercise 3 - Rodinia CUDA benchmarks and Comparison with CPU

**1. Compile both OMP and CUDA versions of your selected benchmarks. Do you need to make any changes in Makefile?** The modifications we need to make to the makefile include paths and compute capabilities.

From exercise we found that the computing capability of Google Colab is "7.5". We specify the target architecture in the Makefile as "-arch sm_75" to support the code running on GPUs with sm_75 computing

```
! ./bandwidthTest

[CUDA Bandwidth Test] - Starting...
Running on...

 Device 0: Tesla T4
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(GB/s)
   32000000                     10.2

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(GB/s)
   32000000                     11.2

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(GB/s)
   32000000                     239.5

Result = PASS
```

Figure 2: The screenshot of running bandwidthTest

capabilities.

**2. Ensure the same input problem is used for OMP and CUDA versions. Report and compare their execution time.** particlefilter[CUDA]

```
TIME TO CALC NEW ARRAY X AND Y TOOK: 0.000308
TIME TO RESET WEIGHTS TOOK: 0.000004
PARTICLE FILTER TOOK 0.162039
ENTIRE PROGRAM TOOK 0.176173
```

Figure 3: particlefilter_cuda

particlefilter[OpenMP]

```
TIME TO CALC CUM SUM TOOK: 0.000035
TIME TO CALC U TOOK: 0.000012
TIME TO CALC NEW ARRAY X AND Y TOOK: 0.043000
TIME TO RESET WEIGHTS TOOK: 0.000066
PARTICLE FILTER TOOK 0.417433
ENTIRE PROGRAM TOOK 0.431620
```

Figure 4: particlefilter_openmp

lavaMD(CUDA)

lavaMD(OpenMP)

```
! ./lavaMD -boxes1d 10

thread block size of kernel = 128
Configuration used: boxes1d = 10
Time spent in different stages of GPU_CUDA KERNEL:
 0.321029990911 s, 56.934444427490 % : GPU: SET DEVICE / DRIVER INIT
 0.000501999981 s,  0.089029349387 % : GPU MEM: ALO
 0.002190999920 s,  0.388572305441 % : GPU MEM: COPY IN
 0.238576993346 s, 42.311466217041 % : GPU: KERNEL
 0.000974000024 s,  0.172738224268 % : GPU MEM: COPY OUT
 0.000584999972 s,  0.103749349713 % : GPU MEM: FRE
Total time:
0.563858985901 s
```

Figure 5: labaMD_cuda

```
! ./lavaMD -boxes1d 10

Configuration used: cores = 1, boxes1d = 10
Time spent in different stages of CPU/MCPU KERNEL:
 0.000000000000 s,  0.000000000000 % : CPU/MCPU: VARIABLES
 0.000009000000 s,  0.000221947994 % : MCPU: SET DEVICE
 0.000000000000 s,  0.000000000000 % : CPU/MCPU: INPUTS
 4.054995059967 s, 99.999778747559 % : CPU/MCPU: KERNEL
Total time:
4.055004119873 s
```

#### 3. Do you observe expected speedup on GPU compared to CPU? Why or Why not? We can observe obvious speedup in the benchmark we used, whether particlefilter or lavaMD.

That's because GPU is throughput-oriented, which performs better in computing and memory-intensive applications. But CPU waste much time on control path.

## Exercise 4 - Run a HelloWorld on AMD GPU

**1. How do you launch the code on GPU on Dardel supercomputer?** Firstly, allocate for a time slot `salloc -A edu23.dd2360 -p gpu -N 1 -t 00:10:00`

Then, we compile the HelloWorld.cpp using `make`

Finally, run the Helloworld file using `srun -n 1 ./Helloworld`

**2. Include a screenshot of your output from Dardel**

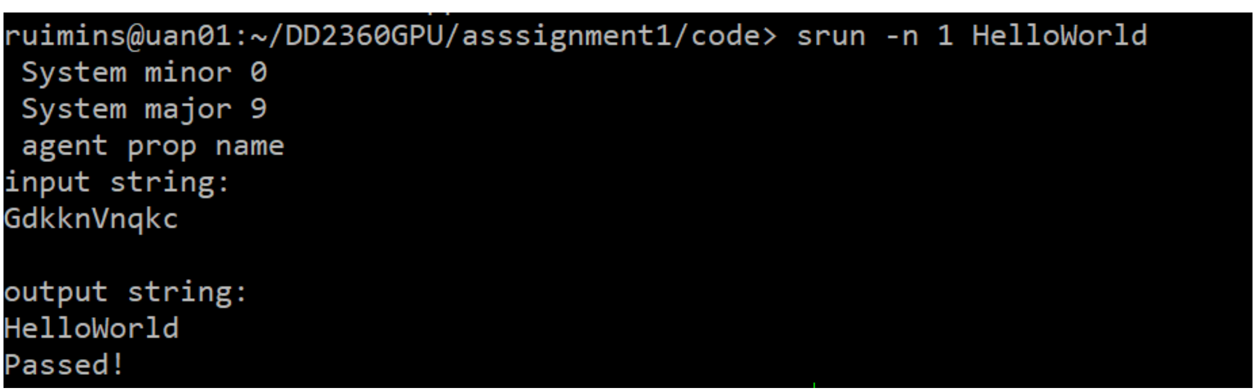


Figure 6: a screenshot of output from Dardel