Assignment 1

Exercise 1 - Reflections on GPU-accelerated Computing

1

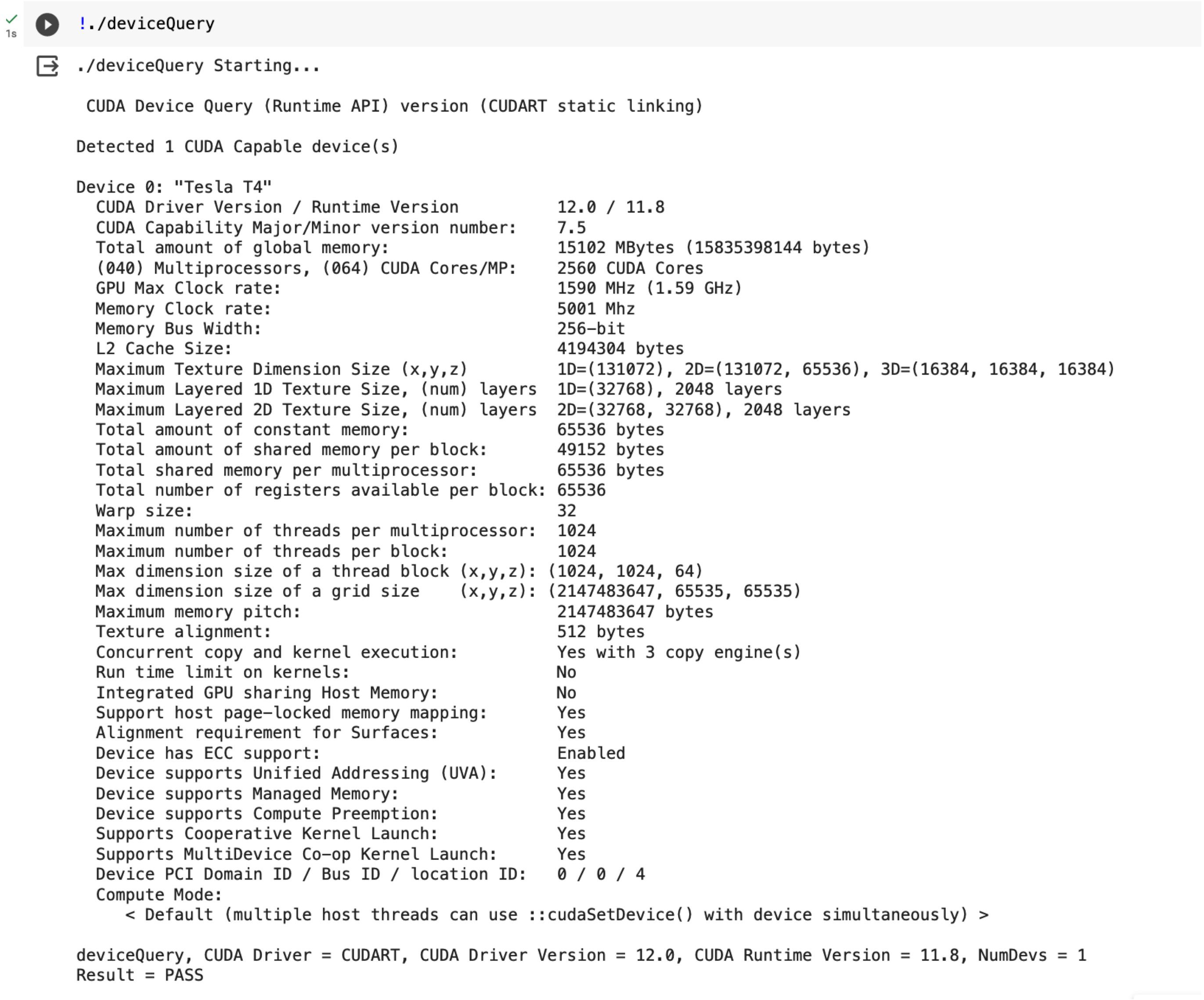
- CPUs are faster and better at handling complex instructions
- GPUs have a lot more cores which means they can work on a lot of tasks in parallel
-

2

Rank	Name	GPU model	Rpeak / Power (TFlops/kW)
1	Frontier	AMD Instinct MI250X	73.95
2	Fugaku	Fujitsu A64FX	14.78
3	LUMI	AMD Instinct MI250X	51.36
4	Leonardo	Nvidia Ampere A100	32.14
5	Summit	Nvidia Tesla V100 / Volta GV100	14.66
6	Sierra	Nvidia Tesla V100 / Volta GV100	12.64
7	Sunway TaihuLight	-	6.05
8	Perlmutter	Nvidia Ampere A100	27.04
9	Selene	Nvidia Ampere A100	23.81
10	Tianhe-2A	Matrix-2000	3.30

- 9 out of 10 have a GPU
- Out of those 5 are by Nvidia 2 by AMD and one each from Fujitsu and Matrix
- source: https://www.top500.org/lists/top500/2023/06/

Exercise 2 - Query Nvidia GPU Compute Capability

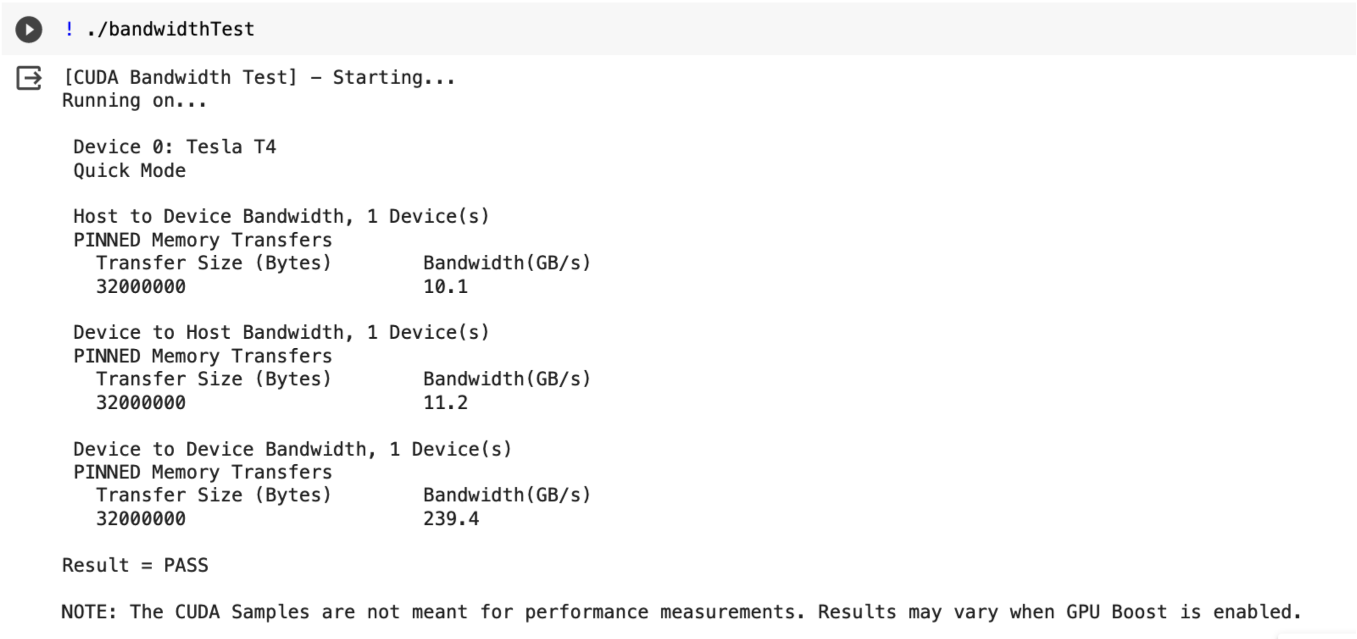


```
!./deviceQuery
```

```
./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
```

The compute capability is 7.5.

- Memory Clock rate: 5001 Mhz
- Bus width: 256-bit
- DDR: 2
- Memory bandwith: 5001 * 256 / 8 / 1024 * 2 = 312 GB/s



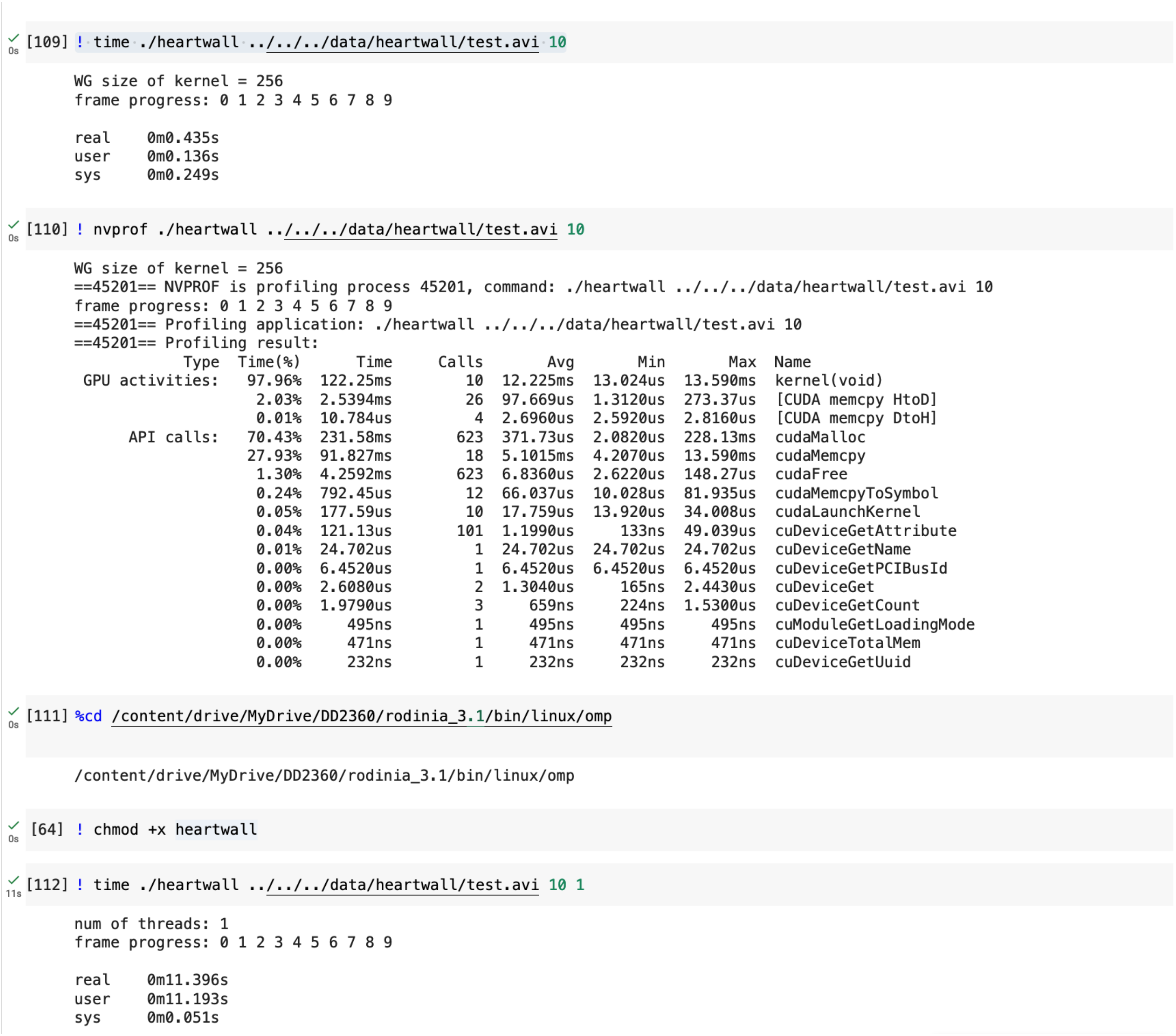
```
! ./bandwidthTest
```

```
[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                     10.1

 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                     239.4

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU Boost is enabled.
```

The bandiwth from the test is lower than the value calculated.

Exercise 3 - Rodinia CUDA benchmarks and Comparison with CPU

In both images provided the results on the top are on a GPU and on the bottom on one thread using OpenMP. The programs executed on CUDA is faster ann both of the programs - heartwall and k-means. These tasks are imaging and data mining tasks which can be done in parallel. For this reason GPU performs it muchs faster.



```
[109] ! time ./heartwall ../../../data/heartwall/test.avi 10
```

```
WG size of kernel = 256
frame progress: 0 1 2 3 4 5 6 7 8 9

real    0m0.435s
user    0m0.136s
sys     0m0.249s
```

```
[110] ! nvprof ./heartwall ../../../data/heartwall/test.avi 10
```

```
WG size of kernel = 256
==45201== NVPROF is profiling process 45201, command: ./heartwall ../../../data/heartwall/test.avi 10
frame progress: 0 1 2 3 4 5 6 7 8 9
==45201== Profiling application: ./heartwall ../../../data/heartwall/test.avi 10
==45201== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   97.96%  122.25ms        10  12.225ms  13.024us  13.590ms  kernel(void)
                    2.03%  2.5394ms        26  97.669us  1.3120us  273.37us  [CUDA memcpy HtoD]
                    0.01%  10.784us         4  2.6960us  2.5920us  2.8160us  [CUDA memcpy DtoH]
      API calls:   70.43%  231.58ms       623  371.73us  2.0820us  228.13ms  cudaMalloc
                   27.93%  91.827ms        18  5.1015ms  4.2070us  13.590ms  cudaMemcpy
                    1.30%  4.2592ms       623  6.8360us  2.6220us  148.27us  cudaFree
                    0.24%  792.45us        12  66.037us  10.028us  81.935us  cudaMemcpyToSymbol
                    0.05%  177.59us        10  17.759us  13.920us  34.008us  cudaLaunchKernel
                    0.04%  121.13us       101  1.1990us     133ns  49.039us  cuDeviceGetAttribute
                    0.01%  24.702us         1  24.702us  24.702us  24.702us  cuDeviceGetName
                    0.00%  6.4520us         1  6.4520us  6.4520us  6.4520us  cuDeviceGetPCIBusId
                    0.00%  2.6080us         2  1.3040us     165ns  2.4430us  cuDeviceGet
                    0.00%  1.9790us         3     659ns     224ns  1.5300us  cuDeviceGetCount
                    0.00%     495ns         1     495ns     495ns     495ns  cuModuleGetLoadingMode
                    0.00%     471ns         1     471ns     471ns     471ns  cuDeviceTotalMem
                    0.00%     232ns         1     232ns     232ns     232ns  cuDeviceGetUuid
```

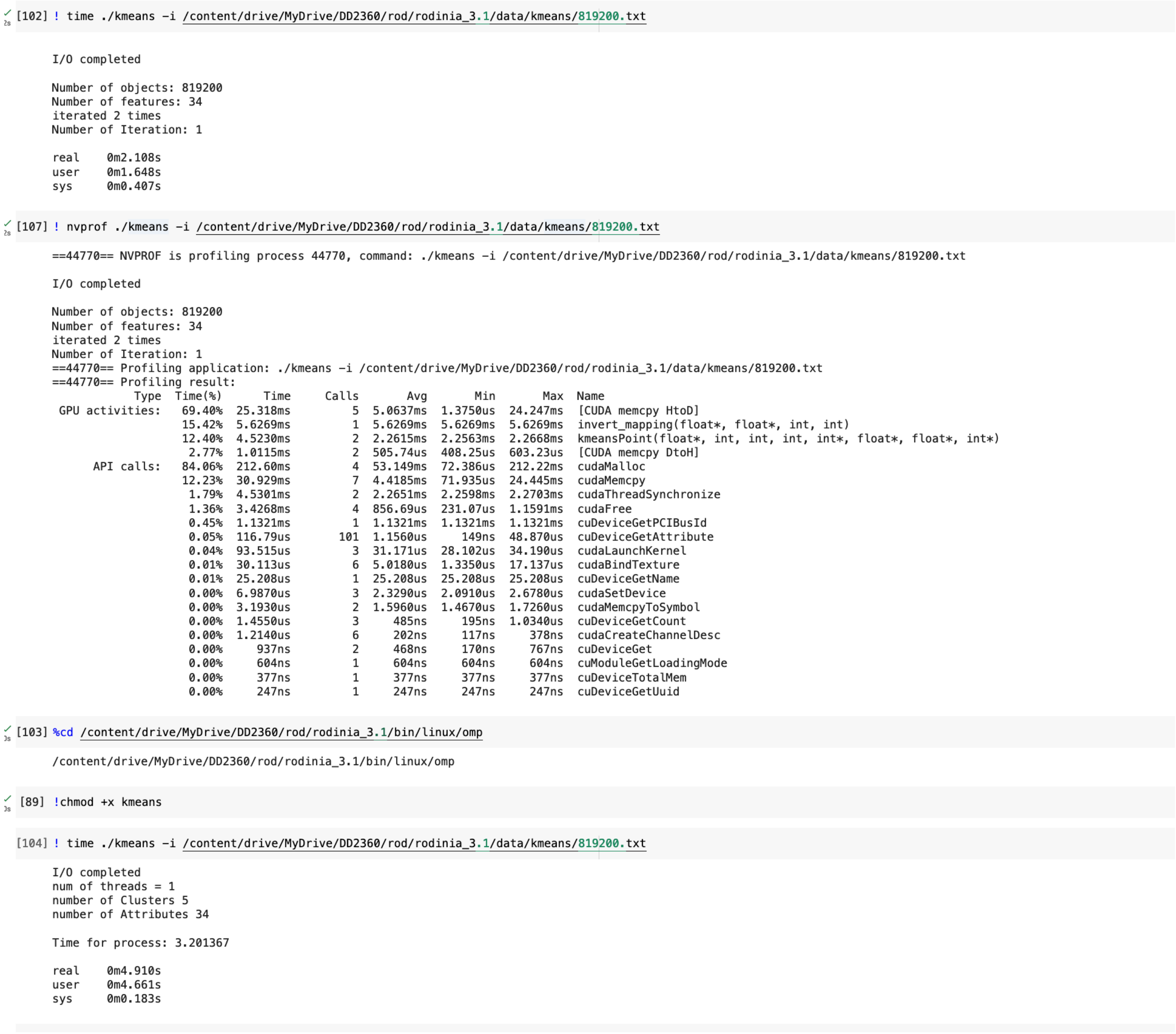
```
[111] %cd /content/drive/MyDrive/DD2360/rodinia_3.1/bin/linux/omp
```

```
/content/drive/MyDrive/DD2360/rodinia_3.1/bin/linux/omp
```

```
[64] ! chmod +x heartwall
```

```
[112] ! time ./heartwall ../../../data/heartwall/test.avi 10 1
```

```
num of threads: 1
frame progress: 0 1 2 3 4 5 6 7 8 9

real    0m11.396s
user    0m11.193s
sys     0m0.051s
```



```
[102] ! time ./kmeans -i /content/drive/MyDrive/DD2360/rod/rodinia_3.1/data/kmeans/819200.txt


      I/O completed

      Number of objects: 819200
      Number of features: 34
      iterated 2 times
      Number of Iteration: 1

      real    0m2.108s
      user    0m1.648s
      sys     0m0.407s
```

```
[107] ! nvprof ./kmeans -i /content/drive/MyDrive/DD2360/rod/rodinia_3.1/data/kmeans/819200.txt

      ==44770== NVPROF is profiling process 44770, command: ./kmeans -i /content/drive/MyDrive/DD2360/rod/rodinia_3.1/data/kmeans/819200.txt

      I/O completed

      Number of objects: 819200
      Number of features: 34
      iterated 2 times
      Number of Iteration: 1
      ==44770== Profiling application: ./kmeans -i /content/drive/MyDrive/DD2360/rod/rodinia_3.1/data/kmeans/819200.txt
      ==44770== Profiling result:
                  Type  Time(%)      Time     Calls       Avg       Min       Max  Name
       GPU activities:   69.40%  25.318ms         5  5.0637ms  1.3750us  24.247ms  [CUDA memcpy HtoD]
                         15.42%  5.6269ms         1  5.6269ms  5.6269ms  5.6269ms  invert_mapping(float*, float*, int, int)
                         12.40%  4.5230ms         2  2.2615ms  2.2563ms  2.2668ms  kmeansPoint(float*, int, int, int, int*, float*, float*, int*)
                          2.77%  1.0115ms         2  505.74us  408.25us  603.23us  [CUDA memcpy DtoH]
            API calls:   84.06%  212.60ms         4  53.149ms  72.386us  212.22ms  cudaMalloc
                         12.23%  30.929ms         7  4.4185ms  71.935us  24.445ms  cudaMemcpy
                          1.79%  4.5301ms         2  2.2651ms  2.2598ms  2.2703ms  cudaThreadSynchronize
                          1.36%  3.4268ms         4  856.69us  231.07us  1.1591ms  cudaFree
                          0.45%  1.1321ms         1  1.1321ms  1.1321ms  1.1321ms  cuDeviceGetPCIBusId
                          0.05%  116.79us       101  1.1560us     149ns  48.870us  cuDeviceGetAttribute
                          0.04%  93.515us         3  31.171us  28.102us  34.190us  cudaLaunchKernel
                          0.01%  30.113us         6  5.0180us  1.3350us  17.137us  cudaBindTexture
                          0.01%  25.208us         1  25.208us  25.208us  25.208us  cuDeviceGetName
                          0.00%  6.9870us         3  2.3290us  2.0910us  2.6780us  cudaSetDevice
                          0.00%  3.1930us         2  1.5960us  1.4670us  1.7260us  cudaMemcpyToSymbol
                          0.00%  1.4550us         3     485ns     195ns  1.0340us  cuDeviceGetCount
                          0.00%  1.2140us         6     202ns     117ns     378ns  cudaCreateChannelDesc
                          0.00%     937ns         2     468ns     170ns     767ns  cuDeviceGet
                          0.00%     604ns         1     604ns     604ns     604ns  cuModuleGetLoadingMode
                          0.00%     377ns         1     377ns     377ns     377ns  cuDeviceTotalMem
                          0.00%     247ns         1     247ns     247ns     247ns  cuDeviceGetUuid
```

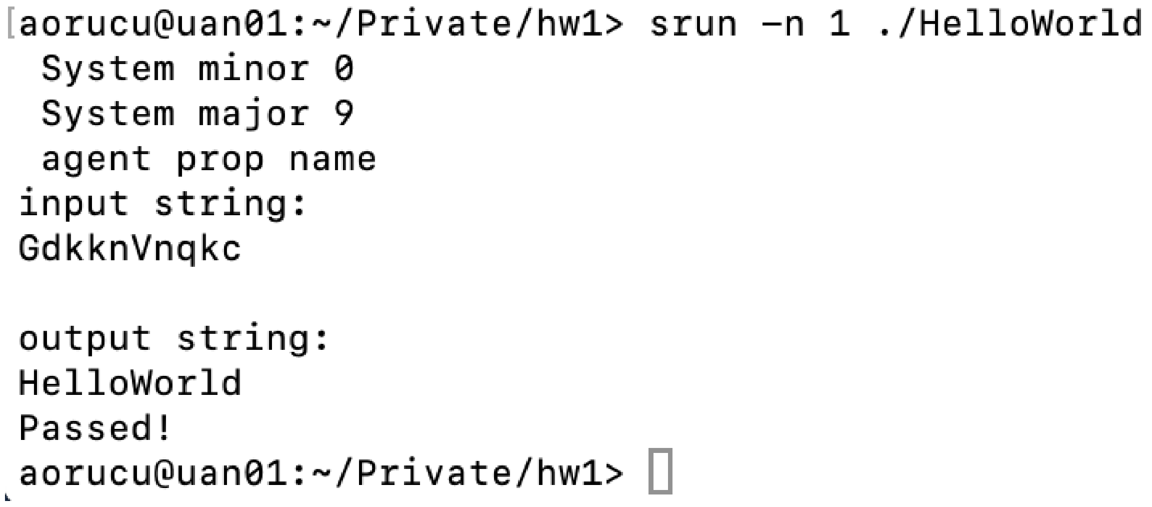
```
[103] %cd /content/drive/MyDrive/DD2360/rod/rodinia_3.1/bin/linux/omp

      /content/drive/MyDrive/DD2360/rod/rodinia_3.1/bin/linux/omp
```

```
[89] !chmod +x kmeans
```

```
[104] ! time ./kmeans -i /content/drive/MyDrive/DD2360/rod/rodinia_3.1/data/kmeans/819200.txt

      I/O completed
      num of threads = 1
      number of Clusters 5
      number of Attributes 34

      Time for process: 3.201367

      real    0m4.910s
      user    0m4.661s
      sys     0m0.183s
```

Exercise 4 - Run a HelloWorld on AMD GPU

To launch the code on the AMD GPUs in Dardel. First, one needs to get a GPU allocation. Then the executable created can be launched on the GPU using the `srun` command and specifying which node the program should run on with the `-n` flag.



```
aorucu@uan01:~/Private/hw1> srun -n 1 ./HelloWorld
  System minor 0
  System major 9
  agent prop name
 input string:
 GdkknVnqkc

 output string:
 HelloWorld
 Passed!
aorucu@uan01:~/Private/hw1> 
```