Exercise 1

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

1. CPUs are latency-oriented, aiming at minimizing latency, small number of ALUs; GPUs are throughput-oriented, aiming at maximizing the throughput, large number of ALUs.
2. CPUs have many control logic; GPUs have a lot less.
3. CPUs have virtual multithreading; GPUs have hardware multithreading.

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

From Green500 November 2022 list,

1. Henri, NVIDIA H100
2. Frontier TDS, AMD Instinct MI250X
3. Adastra, AMD Instinct MI250X
4. Setonix, AMD Instinct MI250X
5. Dardel, AMD Instinct MI250X
6. Frontier, AMD Instinct MI250X
7. LUMI,AMD Instinct MI250X
8. ATOS THX.A.B, NVIDIA A100
9. MN-3, Preferred Networks MN-Core
10. Champollion, NVIDIA A100

3. Calculate the power efficiency for the top 10 supercomputers

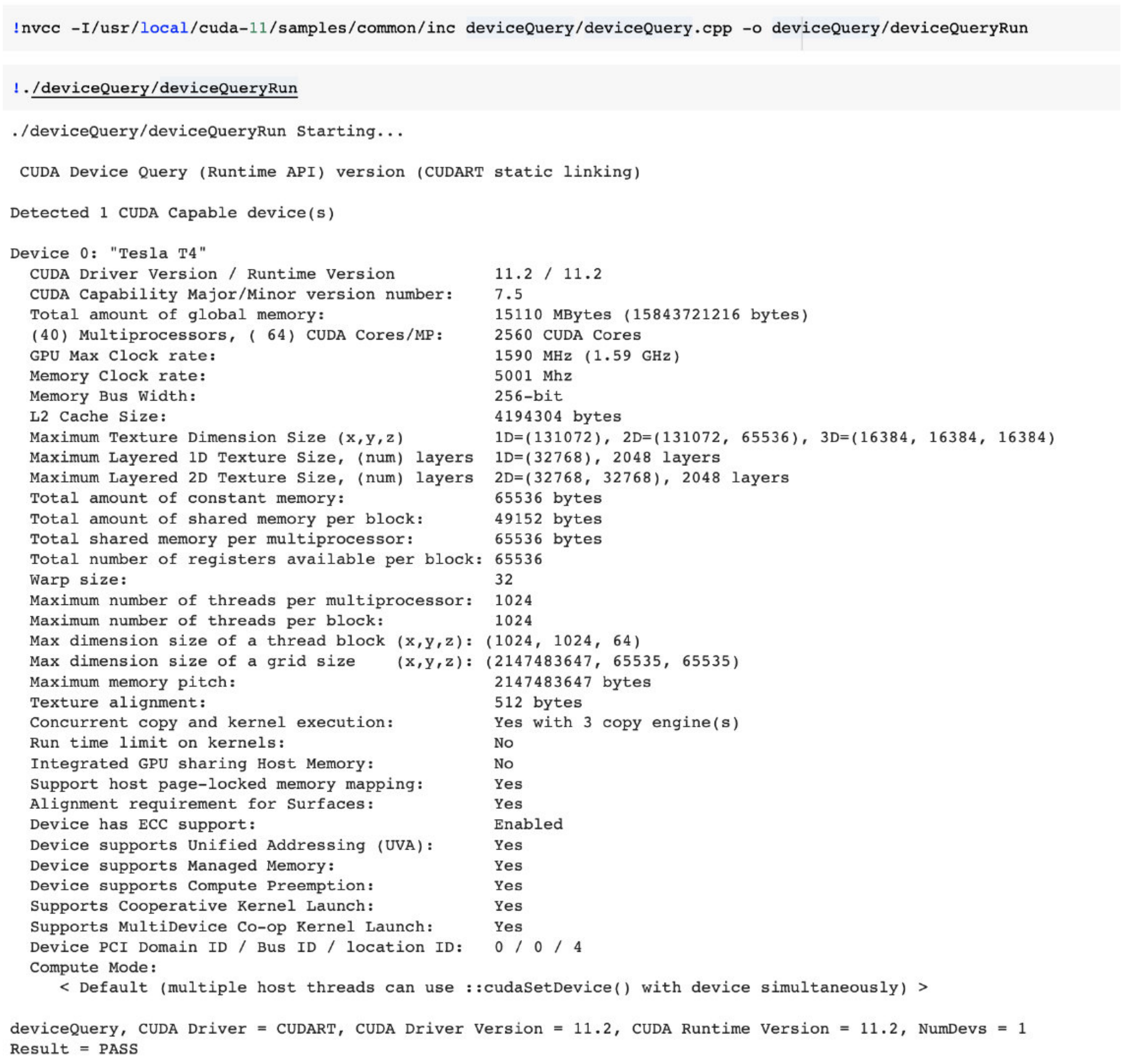
From Green500 November 2022 list, energy efficiency (GFlops/watts)

1. Henri, 65.091
2. Frontier TDS, 62.684
3. Adastra, 58.021
4. Setonix, 56.983

5. Dardel, 56.491
6. Frontier, 52.227
7. LUMI,51.382
8. ATOS THX.A.B, 41.411
9. MN-3, 40.901
10. Champollion, 38.555

Exercise 2

1. The screenshot of the output from you running deviceQuery test



```
!nvcc -I/usr/local/cuda-11/samples/common/inc deviceQuery/deviceQuery.cpp -o deviceQuery/deviceQueryRun
```

```
!./deviceQuery/deviceQueryRun
```

```
./deviceQuery/deviceQueryRun 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          11.2 / 11.2
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 15110 MBytes (15843721216 bytes)
  (40) Multiprocessors, ( 64) 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 = 11.2, CUDA Runtime Version = 11.2, NumDevs = 1
Result = PASS
```

2. What architectural specifications do you find interesting and critical for performance?

- CUDA Cores/MP: More cores can lead to better performance.

- GPU Max Clock rate: The higher the clock rate is, the more calculations per second a core can perform, thus better.
- Total amount of global memory: A bigger memory means storing more data inside, useful for example running 4k games or having big batch size for machine learning.

3. How do you calculate the GPU memory bandwidth (in GB/s) using the output from deviceQuery?

2 * memory bus width * memory clock rate = 2 * 256 bit * 5001 MHz = 320.1 GB/s

4. Compare your calculated GPU memory bandwidth with Nvidia published specification on that architecture. Are they consistent?

The website(https://www.nvidia.com/en-us/data-center/tesla-t4/) states that it has "320+ GB/s" bandwidth, hence pretty consistent.

Exercise 3

The three recent NVIDIA GPU architectures, from oldest to newest, are: Volta, Ampere, Hopper

https://www.nvidia.com/en-us/technologies/

1. List 3 main changes in architecture (e.g., L2 cache size, cores, memory, notable new hardware features, etc.)

From Volta (V100 SXM2) -> Ampere (A100) -> Hopper (H100 SXM)

1. GPU Memory Bandwidth: 900 GB/s -> 1935 GB/s -> 3.35 TB/s
2. Interconnect: 300 GB/s -> 600 GB/s -> 900 GB/s
3. FP32: 15.7 TFLOPS -> 19.5 TFLOPS -> 67 TFLOPS

2. List the number of SMs, the number of cores per SM, the clock frequency and calculate their theoretical peak throughput.

For FP32,

- V100: 80 SMs, 64 cores per SM, with 1530 MHz clock frequency, 7.83 TFLOPS throughput.
- A100: 108 SMs, 64 cores per SM, with 1410 MHz clock frequency, 7.83 TFLOPS throughput.
- H100: 132 SMs, 128 cores per SM, with 1833 MHz clock frequency, 30.97 TFLOPS

throughput.

3. Compare (1) and (2) with the NVIDIA GPU that you are using for the course.

For the Tesla T4 I'm having on Colab, it has 32 GB/s interconnect bandwidth and 16 GB GPU memory, with 8.1 TFLOPS single-precision.

Ex4

See the result in the .ipynb file

Ex5 Bonus

1. What limitations this paper proposes to address

Right now GPUs are widely used for High Performance Computing (HPC) and Deep Learning (DL). However, these two areas are requiring more and more memory because of bigger models. 2 reasons why GPU can't just have very big memory: 1. GPUs prioritize memory speed over capacity in order to keep their many parallel cores busy; 2. Hign Bandwidth Memory (HBM) GPU has a maximum capacity of 32GB and for non-HBM ones, the number of channels is already near the practical pin count limit.

So, this paper wants to use a buddy compression method to increase the memory capacity of GPUs while sacrifycing just a little performance.

2. What workloads/applications does it target?

It targets HPC and DL workloads.

3. What new architectural changes does it propose? Why it can address the targeted limitation?

They proposed a system that is composed of multiple GPU nodes connected with a high-bandwidth NVLink2 interconnect, to a larger source of remote memory.

1. A fast and low-cost hardware compression algorithm. Buddy Compression shares this design decision and uses a 128B compression granularity to match the GPU cache block size.
2. At boot time, the driver carves out a contiguous chunk of pinned buddy-memory for each GPU. These regions are never directly accessed by the host CPU, eliminating any

coherence issues, and making the address translation for buddy-memory simple and fast.

3. A global base physical address for the buddy-memory carve-out region is stored in a Global Buddy Base-address Register (GBBR).

The current high-bandwidth interconnects and unified memory make sure this is possible. Also, the compressibility of GPU workloads make this new architecture doable.

4. What applications are evaluated and how did they setup the evolution environment (e.g., simulators, real hardware, etc)?

For HPC, it used a subset of SpecACCEL OpenACC v1.2 and CUDA versions of the DOE benchmarks HPGMG and LULESH as accessible and simulatable representatives for HPC applications.

For DL, it trained a set of 5 convolutional neural networks to represent DL workloads: AlexNet, Inception v2, SqueezeNetv1.1, VGG16, and ResNet50, all running on the Caffe framework with the ImageNet dataset. Additionally it considered a long short-term memory network, BigLSTM, using the English language model.

They used a dependency-driven GPU performance simulator, similar to the one used by Arunkumar et al. and others. They also compared the performance of Buddy Compression to an unrealistic GPU with no memory limits. The experiment was done on simulated environment (GPGPUSim).

5. Do you have any doubts or comments on their proposed solutions?

The Buddy Compression system offers 1.5-1.9X higher effective memory capacity while with only a 1–2% performance penalty relative to an unconstrainedcapacity GPU, due to its unique design where compressibility changes require no additional data movement. It can help to load bigger DL models which would help with both increasing performance and speeding up training process. I think this paper is both clear and informative.