DD2360
Assignment II: GPU architecture

Rui Shi

November 30, 2022

Link to github: `https://github.com/RuiShi324/DD2360`

1 Exercise 1 - Reflection on GPU-accelerated Computing

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

   - CPU is Central processing unit while GPU is graphics processing unit.
   - Usually GPU has more cores than CPU
   - CPU is composed of just a few cores with lots of cache memory that can handle a few software threads at a time. In contrast, a GPU is composed of hundreds of cores that can handle thousands of threads simultaneously.
   - CPU has lager size of cache than GPU, while GPU has much more ALUs.

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.
   Answer:

Rank	Name	GPU vendor	model	If use GPU
1	Frontier	HPE	HPE Cray EX235a	Yes
2	Fugaku		Supercomputer Fugaku	No
3	LUMI	AMD	HPE Cray EX235a	Yes
4	Leonardo	Atos	BullSequana XH2000	Yes
5	Summit	IBM	IBM Power System AC922	Yes
6	Sierra	IBM	IBM Power System S922LC	Yes
7	Sunway TaihuLight		Sunway MPP	No
8	Perlmutter	HPE	HPE Cray EX235n	Yes
9	Selene	Nvidia	Nvidia	Yes
10	Tianhe-2A	NUDT	TH-IVB-FEP	Yes

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.

Rank	Name	Rmax (PFlop/s)	model	Power (kW)
1	Frontier	1,102.00	HPE Cray EX235a	21,100
2	Fugaku	442.01	Supercomputer Fugaku	29,899
3	LUMI	309.10	HPE Cray EX235a	6,016
4	Leonardo	174.70	BullSequana XH2000	5,610
5	Summit	148.60	IBM Power System AC922	10,096
6	Sierra	94.64	IBM Power System S922LC	7,438
7	Sunway TaihuLight	93.01	Sunway MPP	15,371
8	Perlmutter	70.87	HPE Cray EX235n	2,589
9	Selene	63.46	Nvidia	2,646
10	Tianhe-2A	61.44	TH-IVB-FEP	18,482

2 Exercise 2 - Device Query

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

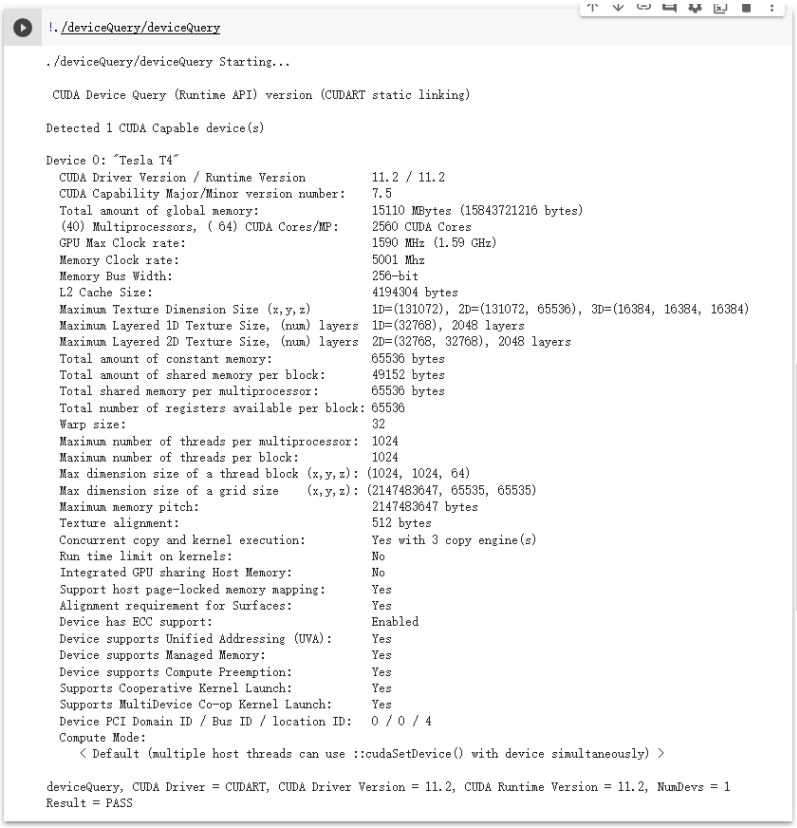


Figure 1: Screenshot of deviceQuery test

2. What architectural specifications do you find interesting and critical for performance? Please provide a brief description.
   Answer: I think the number of multiprocessors, number of MPs, memory bus width, L2 cache size, number of registers available per block, number of threads and GPU Max Clock rate would be critical for performance. When increasing the bus width, more data could be loaded in a second. When increasing L2 cache size, more data could be stored in the cache and it could be faster to reach. For larger number of registers, data stored in registers are faster to reach. For larger number of threads, more tasks could be executed at the same time. For large GPU max Clock rate, more operations could be made in a second.

3. How do 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)

$$bandwidth = 2 \times buswidth \times clockrate = 2 \times 256bit \times 5001Mhz = \frac{256 \times 5001 \times 10^6}{8 \times 10^9 GB/s} = 320.1GB/s \quad (1)$$

4. Compare your calculated GPU memory bandwidth with Nvidia published specification on that architecture. Are they consistent?
   Answer: From the website **https://www.nvidia.com/en-us/data-center/tesla-t4/**, I found that they were consistent.

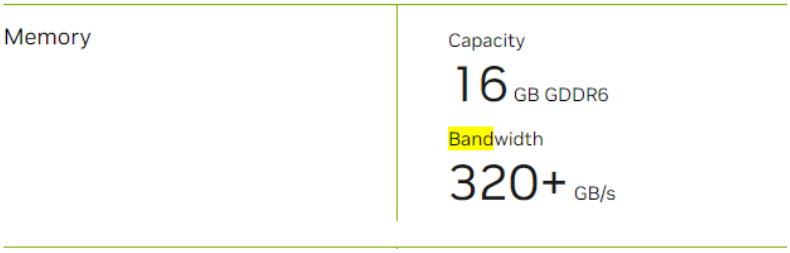


Figure 2: Bandwidth from official website

3 Exercise 3 - Compare GPU Architecture

Architecture	Model	Cores	L2 cache size	Memory size(GB)	SMs	Cores /SM	Base Clock (Mhz)	Throughput (TFLOPS)
Ada Lovelace	RTX 4090	16384	72MB	24	128	128	2235 Mhz	82.58
Ampere	RTX 3090 Ti	10752	6MB	24	84	128	1560 Mhz	40.00
Turing	RTX 2080 Ti	4352	6MB	11	68	64	1350 MHz	13.45
Turing	Tesla T4	2560	4MB	16	40	64	585 MHz	8.141

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

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

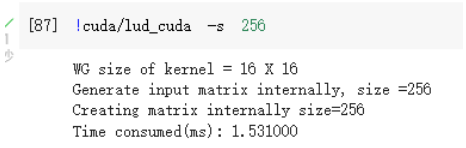
3. Compare (1) and (2) with the NVIDIA GPU that you are using for the course.

   All the information are shown in the table above. From the table, we can see that usually latest architecture has larger L2 cache size and memory size. And the GPU I use on Colab, Tesla T4 is also compared in this table.

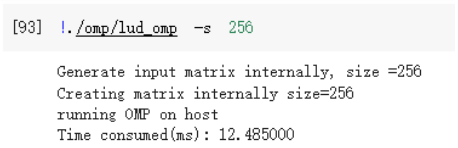
4 Exercise 4 - Rodinia CUDA benchmarks and Profiling

1. Compile both OMP and CUDA versions of your selected benchmarks. Do you need to make any changes in Makefile?
   Answer: If Makefile specified the architecture of cuda flag, then we need to update it manually. For colab Tesla T4 that I used, update sm_75 instead.

2. Ensure the same input problem is used for OMP and CUDA versions. Report and compare their execution time.
   Answer: I selected LUD, myocyte, and lavaMD. Here are the results comparing cuda and openmd.

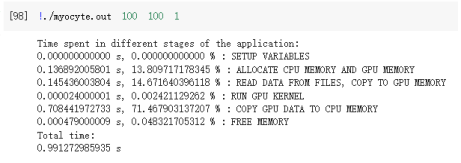


(a) cuda

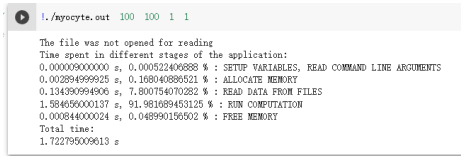


(b) openmd

Figure 3: LUD

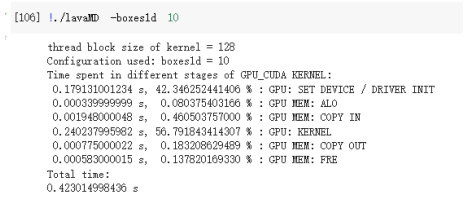


(a) cuda

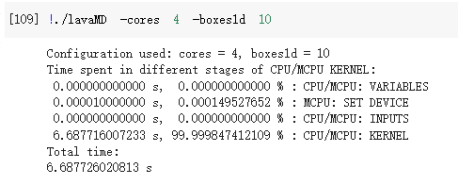


(b) openmd

Figure 4: myocyte



(a) cuda



(b) openmd

Figure 5: lavaMD

The experiments showed that the execution time of CUDA versions is less than the

OMP versions. However, depending on different benchmarks, the improvement of CUDA versions is different. For example, in LUD and lavaMD, the execution time of OMP versions would be times of CUDA versions'. But in myocyte, the improvement was not so significant.

3. Do you observe expected speedup on GPU compared to CPU? Why or Why not?
   Answer: Yes, the speedup on GPU is significant, as GPU could calculate in parallel and the structure.

5 Exercise 5 - GPU Architecture Limitations and New Development

In this exercise, I read the first paper: GPU Subwarp Interleaving. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 1184-1197). IEEE.

1. What limitations this paper proposes to address
   Answer: GPUs lose efficiency when threads in a warp diverge and do not share the same PC. And GPUs loss efficiency when the scheduler does not have enough activate warps to hide stalls.

2. What workloads/applications does it target?
   Answer: This paper targets to exploit thread divergence to hide pipeline stalls in divergent sections of low warp occupancy workloads.

3. What new architectural changes does it propose? Why it can address the targeted limitation?
   Answer: Subwarp Interleaving. When a long latency operation stalls a warp and the GPU's warp scheduler cannot find an active warp to switch to, a subwarp scheduler can instead switch execution to another divergent subwarp of the current warp.

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

Table I: Architecture simulation parameters.

# Streaming Multiprocessors	2
Processing blocks per SM	4
Warp slots per processing block	$\{2, 4, 8\}$
Warp slots per SM	$\{8, 16, 32\}$
Warp size	32
L1 data cache size	128KB
L1 instruction cache size	$\{64KB\}$
L0 instruction cache size	$\{16KB\}$
L1 miss latency	$\{300, 600, 900\}$ cycles
Subwarp switch latency	6 cycles

Figure 6: Architecture simulation parameters

Application	Trace Name	RT effect	Description
ArchViz Interior	AV1	GI-D	Architectural rendering [7]
ArchViz Interior	AV2	AO	Architectural rendering
Battlefield V scene 1	BFV1	R	Game [13]
Battlefield V scene 2	BFV2	R	Game
Control	Ctrl	M	Game
RTX Collage	Coll1	AO	Internal demo
RTX Collage	Coll2	R	Internal demo
DDGI Villa	DDGI	GI-D	Greek Villa demo for [20]
Mechwarrior 5	MW	R	Game
Minecraft	MC	M	Game

Figure 7: Real-time graphics applications

5. Do you have any doubts or comments on their proposed solutions?
   I think the proposed solution is very interesting and inspiring. But I wonder if this solution can improve performance in all cases, or only some specific benchmarks. Also, if this solution is sensitive to number of cores, size of memory, number of SMs, etc.