Assignment II: GPU architecture

New Attempt

Due 30 Nov 2022 by 23:59 **Points** 4 **Submitting** a file upload
File types pdf **Available** after 7 Nov 2022 at 0:00

To submit your assignment, prepare and upload a PDF file fulfilling the following requirements:

- Named as *DD2360HT22_HW2_Surname_Name.pdf*
- Include your answers to each exercise
- Create a Git repository named *DD2360HT22* and make it public so we can access it. Include the link in your report.
- Use the following folder structure in this Git repo : hw_2/ex_*ExerciseNumber* */your_source_code_files*

Important note: This assignment will need knowledge of GPU architecture. You will find lectures, including the introductory and video lectures, tutorials, and the reading materials useful for this assignment.

Exercise 1 - Reflection on GPU-accelerated Computing

In the first lecture, we covered the main architectural differences between CPU and CPU. Please refresh or read further and answer the following questions:

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

Exercise 2 - Device Query

In the first assignment, you are asked to measure GPU-CPU bandwidth using the *bandwidthTest* utility that is included in CUDA SDK. In the same utility folder, a *deviceQuery* test is provided (i.e., 1_Utilities/deviceQuery) to query some architectural specifications on the GPU that you are running on.

Revisit the instructions in assignment 1 on how to compile and run the utility tests either in the KTH computer room or the Google Colab platform.

Questions to answer in the report

1. The screenshot of the output from you running deviceQuery test.
2. What architectural specifications do you find interesting and critical for performance? Please provide a brief description.
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)
4. Compare your calculated GPU memory bandwidth with Nvidia published specification on that architecture. Are they consistent?

Exercise 3 - Compare GPU Architecture

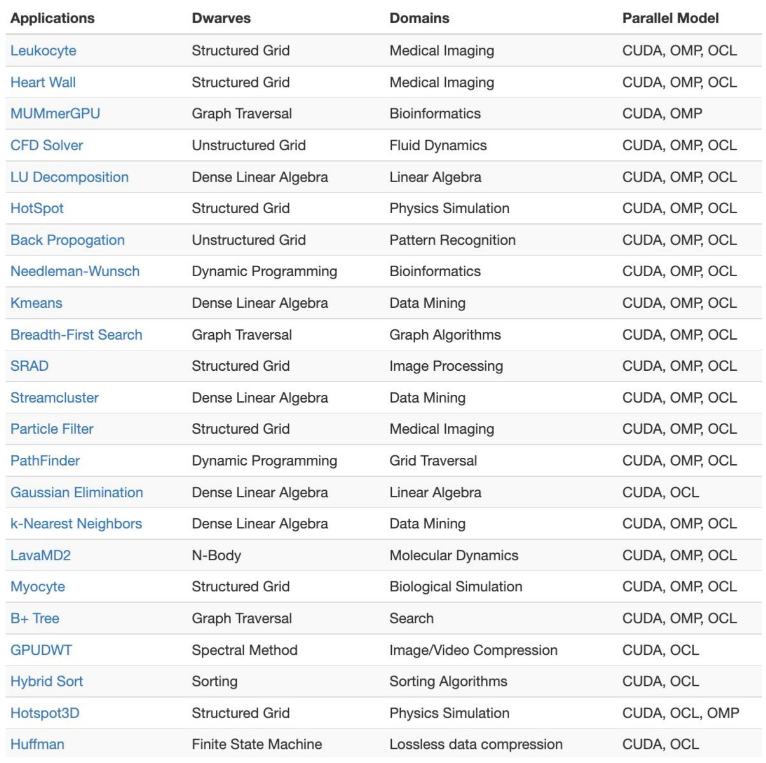
Use the Internet to search to find **3 latest Nvidia GPU architectures** in which you are interested. Pick a specific model for each selected architecture, and answer the following questions:

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

Exercise 4 - Rodinia CUDA benchmarks and Profiling

In the lectures, we emphasized that GPU is a throughput-oriented accelerator for compute-intensive workloads. We also briefly introduced a simple metric to quantify compute/memory intensity, e.g., Arithmetic Intensity, approximated by FLOP/Byte, and Roofline model. Another approach to characterization of parallel computation is through The **Dwarfs** of parallel computing. Reference: Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., ... & Yelick, K. A. (2006). The landscape of parallel computing research: A view from Berkeley.

The Rodinia (https://www.cs.virginia.edu/rodinia/doku.php) benchmark suitę provides a set of mini-applications in different Dwarfs**.** It also provides different implementation in CUDA, OpenMP, etc. The table of benchmarks looks like the table below:



Applications	Dwarves	Domains	Parallel Model
Leukocyte	Structured Grid	Medical Imaging	CUDA, OMP, OCL
Heart Wall	Structured Grid	Medical Imaging	CUDA, OMP, OCL
MUMmerGPU	Graph Traversal	Bioinformatics	CUDA, OMP
CFD Solver	Unstructured Grid	Fluid Dynamics	CUDA, OMP, OCL
LU Decomposition	Dense Linear Algebra	Linear Algebra	CUDA, OMP, OCL
HotSpot	Structured Grid	Physics Simulation	CUDA, OMP, OCL
Back Propogation	Unstructured Grid	Pattern Recognition	CUDA, OMP, OCL
Needleman-Wunsch	Dynamic Programming	Bioinformatics	CUDA, OMP, OCL
Kmeans	Dense Linear Algebra	Data Mining	CUDA, OMP, OCL
Breadth-First Search	Graph Traversal	Graph Algorithms	CUDA, OMP, OCL
SRAD	Structured Grid	Image Processing	CUDA, OMP, OCL
Streamcluster	Dense Linear Algebra	Data Mining	CUDA, OMP, OCL
Particle Filter	Structured Grid	Medical Imaging	CUDA, OMP, OCL
PathFinder	Dynamic Programming	Grid Traversal	CUDA, OMP, OCL
Gaussian Elimination	Dense Linear Algebra	Linear Algebra	CUDA, OCL
k-Nearest Neighbors	Dense Linear Algebra	Data Mining	CUDA, OMP, OCL
LavaMD2	N-Body	Molecular Dynamics	CUDA, OMP, OCL
Myocyte	Structured Grid	Biological Simulation	CUDA, OMP, OCL
B+ Tree	Graph Traversal	Search	CUDA, OMP, OCL
GPUDWT	Spectral Method	Image/Video Compression	CUDA, OCL
Hybrid Sort	Sorting	Sorting Algorithms	CUDA, OCL
Hotspot3D	Structured Grid	Physics Simulation	CUDA, OCL, OMP
Huffman	Finite State Machine	Lossless data compression	CUDA, OCL

In this assignment, you will use the latest version **Rodinia 3.1 (downloads at** http://lava.cs.virginia.edu/Rodinia/download.htm**).** You **ONLY need to choose 3 benchmarks** from 3 different dwarfs for completing this assignment. Please complete the following questions and include answers in your report:

1. Compile both OMP and CUDA versions of your selected benchmarks. Do you need to make any changes in Makefile?

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

3. Do you observe expected speedup on GPU compared to CPU? Why or Why not?

Hint: If the execution times between the CPU and the GPU are very similar, try to use a larger input problem. If timing information is missing, you can use the **gettimeofday()** **(http://man7.org/linux/man-pages/man2/gettimeofday.2.html)** function to get timestamps in order to calculate execution time (**Tutorial: Timing your Kernel - CPU Timer & nvprof** **(https://canvas.kth.se/courses/36161/pages/tutorial-timing-your-kernel-cpu-timer-and-nvprof)**).

(Bonus) Exercise 5 - GPU Architecture Limitations and New Development

So far, we have been working a lot on understanding an existing GPU architecture. This bonus exercise will get you familiar with how cutting-edge research is conducted on designing and developing new GPU architecture.

When designing a new GPU architecture, most evaluation and experiments will be run on simulators instead of real hardware. **GPGPU-Sim** is one of the most popular open-source simulators. **http://www.gpgpu-sim.org/**

A typical research work on GPU architecture will identify some limitations of a current architecture, propose some architectural changes, and evaluated in a simulated environment.

To give you a flavor of GPU architecture research, here is a list of 3 recent publications.

1. Damani, S., Stephenson, M., Rangan, R., Johnson, D., Kulkami, R., & Keckler, S. W. (2022, April). GPU Subwarp Interleaving. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA) (pp. 1184-1197). IEEE.
2. Choukse, E., Sullivan, M. B., O'Connor, M., Erez, M., Pool, J., Nellans, D., & Keckler, S. W. (2020, May). Buddy compression: Enabling larger memory for deep learning and HPC workloads on GPUs. In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA) (pp. 926-939). IEEE.
3. Ganguly, D., Zhang, Z., Yang, J., & Melhem, R. (2019, June). Interplay between hardware prefetcher and page eviction policy in cpu-gpu unified virtual memory. In Proceedings of the 46th International Symposium on Computer Architecture (pp. 224-235).

Pick one or more that you find interesting, and answer the following questions in the report.

1. What limitations this paper proposes to address
2. What workloads/applications does it target?
3. What new architectural changes does it propose? Why it can address the targeted limitation?
4. What applications are evaluated and how did they setup the evolution environment (e.g., simulators, real hardware, etc)?
5. Do you have any doubts or comments on their proposed solutions?