CSCI-GA.3033-004
Graphics Processing Units (GPUs): Architecture and Programming
Homework Assignment 1

(total: 25 points)

***1.** [11] To design the next generation GPU, a company has several choices to make:*

1. *Increasing number of SM*
2. *Increasing number of SPs (or cuda cores) per SM*
3. *Increasing memory bandwidth*
4. *Increasing shared memory per SM*
5. *Increasing L2 cache size*

Discuss the pros and cons for each one of the following scenarios (at least one positive and one negative issue for every scenario to get full credit):

a) Increasing number of SM

Pros: higher computing performance

Cons: performance goes down for small workloads and programs that run sequentially

b) Increasing number of SPs (or cuda cores) per SM

Pros: Higher level of parallelism is supported which means increase in speed for applications, increased performance of GPU.

Cons: If there aren't enough warps being scheduled, SPs will be idle which would be inefficient.

c) Increasing memory bandwidth

Pro: Higher quality images, improved high-resolution displays

Con: less efficient with power consumption, performance depends on processing power of GPU

d) Increasing shared memory per SM

Pros: faster data interchange between threads speeds up applications

Cons: thread block may not be able to utilize all the shared memory available per SM which would be inefficient

e) Increasing L2 cache size
Pros: Would reduce the instruction and data access latencies of large complex applications

Cons: increased latency, far distance for data to travel when used

f) Increasing number of SM and Increasing number of SPs per SM

Pros: With more computing horsepower and a higher level of parallelism, GPU would be more powerful, faster and have high performance.

Cons: You would be using a lot of power.

g) Increasing number of SPs per SM and Increasing memory bandwidth

Pros: Both increases would complement each other and result in faster, higher performance where better core performance allows the GPU to process more data faster and the increased memory bandwidth allows for more data to fit through the bus.

Cons: The performance would be limited by the SMs. There could be idle SPs if there aren't many warps resident on an SM, which would decrease efficiency.

h) Increasing number of SM and Increasing shared memory per SM

Pros: You would have more processing power and more memory per SM for inter-thread communication, so faster processing of threads.

Cons: The threads might not utilize all that extra shared memory, performance is limited by the number of cuda cores per SM.

i) Increasing number of SPs per SM and Increasing shared memory per SM

Pros: Faster data interchange between threads and increased parallelism would give faster overall performance. The increased numbers of SPs could hide poor latency of memory access outside of the SMs.

Cons: This performance depends on the SMs and how many warps are being executed. If there isn't a lot of data, this setup would be inefficient.

j) Increasing shared memory per SM and Increasing L2 cache size

Pros: Faster data transfer between threads and SM's (separately), leading to overall faster execution time and faster performance.

Cons: Limit is the number of cuda cores, if the GPU isn't able to process a lot of data quickly then it is inefficient to implement larger shared memory and L2 cache sizes.

k) Increasing memory bandwidth and Increasing L2 cache size

Pros: You would get higher performance through higher quality images and lower latency of data access of large complex applications. You would have a wide bus and large cache for lots of data to travel to and from.

Cons: If the memory speed is slow, or if there isn't a lot of data traveling through the L2 bus then these increases would be inefficient.

While thinking about this problem, assume the GPU has L1 cache and shared memory in each SM and there is also L2 cache shared among all SMs.

***2.** [2 points] Let's assume an application has a lot of independent and similar operations to be performed on data. Does the amount of data has anything to do with the expected performance of the GPU? Justify.*

Yes, if there were a lot of independent and similar operations to be done on a small set of data, the CPU would outperform the GPU. However, if there is a large amount of data, the GPU would eventually outperform the CPU due to the amount of data, but would be slow because multiple operations would have to be performed. GPU's are closest to SIMD in architecture, where it works best performing one operation on a large amount of data.

***3.** [8 points] For each of the following applications state whether it is beneficial to implement them on a GPU, and justify your answer.*

a) Finding whether a number exist in an array of 10M integers

Yes because this is a searching job and there is a lot of data involved, the GPU is focusing all of its computing abilities to find a specific number. GPUs are excellent at performing one operation on a lot of data.

b) Calculating the first 1M Fibonacci numbers

No because this is a sequential program and has dependencies in the code, where past values are added together and recursion is involved, best for a CPU. This would require too many operations and storing of data for GPU.

c) Multiplying two 100x100 matrices

No, a CPU would be able to handle this because the matrices are small and to multiply matrices requires addition as well. Using a GPU on such a small amount of data with more than one operation would not be more beneficial than using CPU.

d) Adding 1Mx1M matrices

Yes, GPUs work excellently at performing a single task on massive amounts of data. It is focused only on the task of matrix addition and has a lot of data to go through, it would perform better than a CPU.

4. *[4 points] We said in class that communication between system memory and CPU memory is very expensive in terms of latency, which negatively affects performance. Why not having one main memory for both the CPU and GPU? State two reasons at least.*

1. Large thread management overheads would degrade performance.

2. Complicated cache coherence hardware requirements would degrade scalability.