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

(total: 25 points)

1. [11 points (0.5 per table entry)] The following are not exhaustive answers. You can come up with other, but correct answers too.

• **Scenario**	• **Pros**	• **Cons**
• **Increasing number of SM**	• More parallelism	• More pressure on L2 cache
• **Increasing number of SPs (or cuda cores) per SM**	• More opportunities for thread interactions • More parallelism • More resources = more blocks assigned to that SM	• Pressure on shared memory • Pressure on L1 cache
• **Increasing memory bandwidth**	• More efficient data supply to threads	• Wasted resource if not enough parallelism
• **Increasing shared memory per SM**	• More sharing among threads leading to less bandwidth requirement to global memory	• Wasted resources if not much data are shared
• **Increasing L2 cache size**	• L2 by definition coalesces memory accesses	• Many L2 cache misses consumes a lot of global memory bandwidth.
• **Increasing number of SM** • **Increasing number of SPs (or cuda cores) per SM**	• More parallelism	• More power consumption • More pressure on L2 caches
• **Increasing number of SPs (or cuda cores) per SM** • **Increasing memory bandwidth**	• More parallelism • More opportunities for memory coalescing	• More pressure on L1 cache • More pressure on shared memory • Wasted resources if not enough parallelism
• **Increasing number of SM** • **Increasing shared memory per SM**	• More parallelism	• More pressure on L2 cache
• **Increasing number of SPs (or cuda cores) per SM** • **Increasing shared memory per SM**	• More parallelism	• More pressure on L1 cache
• **Increasing L2 cache size** • **Increasing shared memory per SM**	• Less trips to global memory = less bandwidth requirement	• Wasted resources if not much data are shared • Bad L2 performance leads to more global memory access
• **Increasing memory bandwidth** • **Increasing L2 cache size**	• Can provide data faster to threads	• Wasted resources if not much data are shared

2. [2 points] Yes, because there is a big slowdown caused by moving the data from system memory to GPU memory. If we do not have enough data to ensure a lot of parallelism in the GPU, to overcome this performance loss, we may see *worse* performance for GPU over CPU-only version.

3. [8 points]

a) Yes, each core can be assigned a subset of the numbers and check the existence of the number in them. This is done in parallel.

b) Cannot be done on GPU because the operations are dependent on each other.

c) The problem size is not big enough to ensure enough parallelism to overcome communication overhead.

d) Yes, it is a highly parallel operation and we have enough independent computations. Moreover, the problem size is big enough to ensure enough parallelism.

4. [4 points]

- This will put a lot of pressure on that memory as it cannot serve all the cores in parallel. This memory will be a bottleneck that may affect scalability if we want to add more CPUs and/or GPUs.
- GPU needs a memory optimized for bandwidth while CPU needs memory optimized for latency.