Amirkabir University of Technology

(Tehran Polytechnic)

Question 1:

The following questions are Based on chapters 3 and 5 of “CUDA Programming, A Developer’s Guide to Parallel Computing with GPUs” by Shane Cook, but for answering some questions you will need to do additional research. (Please keep you answers short)

a) How does the PCIe protocol differ from Its predecessor? What does a PCIe switch do?
b) Compare 4 system architectures depicted in chapter 3 (pages 38 to 43) on merits of bandwidth and latency of each component (RAM, GPU, Other Peripherals). How are modern consumer-grade system architected?
c) Explain the concept of GPU-Direct (used for both Storage and RDMA today) technology introduced in CUDA 4.0. What benefits does it provide in terms of data transfer efficiency? Explain how this technology works in current consumer-grade systems Where GPUs could be directly connected to the CPU?
d) In regards to Nvidia GPUs, describe each memory type, its position in the hierarchy and use cases (with an example for each one).
e) Discuss the CUDA concepts of thread, warp, thread block, SM, and grid. How does each level of this hierarchy contribute to the scalability of GPU programming?
f) Explain the role of large register banks in GPU compute units.
g) Based on the GPU available to you (or your group) figure out what CUDA compute level do you have access to and which debugging/profiling tools will you be able to use and what’s the latest version available for your device? (4 tools will suffice)
h) From a hardware point of view, which CUDA hierarchy level, is most similar to a single core in conventional CPUs. What about from a software point of view?

Question 2:

Using CUDA interface and a GPU with (10 SMs, each with 128 cores and a maximum of 1024 Threads/Block, 16 Blocks/SM, and 2048 Threads/SM) we want to add two vectors of length 20,000 and store the results in a third vector.

a) If we wanted each thread to do the calculations for a single index, what hierarchical dimensions (how many blocks and how many threads per block) would you suggest?
b) Is the answer from section (a) optimal without the given condition of section (a)? How many indices should each thread process?
c) For sections (a) (and (b) if different) write the array index calculation formula. (Hint: this formula might be inside of a for loop)

d) In section (a) if we were to choose blocks with *x* threads, how many threads will be generated for each SM and what percentage of these threads will be actually active? (Masked threads are counted as inactive) *x* = {16, 20, 32, 200, 256, 1000, 1024}
In which configurations will there be blocks waiting for other blocks to finish in order to get scheduled?

Question 3:

Write a kernel to calculate the weighted moving average (according to formula 1) of an input array of length *len* = {256, 512, 1024, 20k, 4M}. The number of Threads, and blocks scheduled should be optimized for GPU described in question 1 section (g). Report the number of SMs in the considered GPU and also, write the block size and block count as conditional functions of input array size. Prove that your answer is (roughly) optimal for the given GPU. Consider the maximum limitations of 1024 Threads/Block, 16 Blocks/SM, and 2048 Threads/SM (if your GPU has lower limits, report and use those instead).

$$out[i] = 0.14 \times in[i] + 0.29 \times in[i+1] + 0.57 \times in[i+2] \qquad \text{(Formula 1)}$$