Heidelberg University
Institute of Computer Engineering
Computing Systems Group

Kazem Shekofteh
Christian Alles

GPU Computing
Winter Term 2022/2023

Exercise 2

- **Return electronically until Monday, Nov 07, 2022, 09:00**
- **Include your names on the top sheet. Hand in only one PDF.**
- **A maximum of four students are allowed to work jointly on the exercises.**
- **When you include plots add a short explanation of what you expected and what you observed.**
- **Hand in source code if the exercise required programming. You can bundle the source code along with the PDF in a .zip file.**
- **Programming exercises can only be graded if they compile on the cluster head node. Make sure to document the commands which you used for the compilation.**

2.1 Raw Kernel Startup Time

- Start with the program ,,nullKernelAsync.cu", which you can find in the ,,template/src" folder. You can compile the program using *make* or manually by using *nvcc*.
- Measure asynchronous kernel startup time
- Add code to measure the synchronous kernel startup time. Compare and interpret!
- For both kernels, vary the grid size (numBlocks) from 1 to 16384 and the block size (threadsPerBlock) from 1 to 1024. Use at least 10 measurement points per dimension. Report the results graphically and interpret!

(12 points)

2.2 Break-even Kernel Startup Time

The GPU may idle if a kernel is finished before new work is ready. Thus, we have to start kernels that perform enough work to completely hide start-up latencies. In order to assess this, write a kernel that busy-waits a certain number of clock cycles before it completes.
Use the `clock_t clock();` or the `long long int clock64();` function to determine the number of clock cycles. For further information have a look at the cuda documentation from nvidia [1] or look at the samples of your cuda SDK [2]. To avoid compiler optimizations, conditionally (with regard to a kernel parameter) write to a `__device__` variable the clock cycle difference (stop - start).

- How many clock cycles do we have to wait to allow for another kernel launch call (an accuracy within 100 clock cycles is sufficient)? To measure this, busy-wait until you observe that the asynchronous kernel startup time doubles.

(14 points)

2.3 PCIe Data Movements

Compare:

- cudaMemcpy on pageable memory (malloc call)
- cudaMemcpy on pinned memory (cudaMallocHost call)
- Both directions (H2D, D2H), vary sizes between 1kB and 1GB with at least 10 measurement points.

[1] `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#time-function`
[2] located at: `$CUDA_PATH/samples/0_Simple/clock`

Kazem Shekofteh
Christian Alles

Report the results graphically and interpret your results! You are free to display either time or throughput in your plot. Note that the theoretical maximum throughput of the PCIe 3.0 x16 connection is about 15.8 GByte/s, so you can expect the large data transfers to be in the same ballpark.
(14 points)

2.4 Willingness to present

Please declare whether you are willing to present any of the previous exercises.
The declaration can be made on a per-exercise basis. Each declaration corresponds to 50% of the exercise points. You can only declare your willingness to present exercises for which you also hand in a solution. If no willingness to present is declared you may still be required to present an exercise for which your group has handed in a solution. This may happen if as example nobody else has declared their willingness to present.

- Raw Kernel Startup Time (Section: 2.1)
- Break-even Kernel Startup Time (Section: 2.2)
- PCIe Data Movements (Section: 2.3)

(20 points)

Total: 60 points