Heidelberg University
Institute of Computer Engineering
Hardware and Artificial Intelligence (HAWAII) Lab

Holger Fröning
Vimala Bauer
Elizaveta Mironovich

GPU Computing - Architecture and Programming

Exercise 3

- **Hand in via Moodle until Wednesday, Nov 12, 2025, 09:00.**
- **Include all names on the top sheet. Hand in a *single* PDF.**
- **Please use naming convention:** `gpu<XX>_ex<N>.{zip,tar.gz}`**.**
- **Maximum of 4 students may collaborate.**
- **For programming exercises:**
  - **include clean and documented code**
  - **include a Makefile for compiling**

General notes

Vary sizes between 1KB and 1GB with at least 10 measurement points. Graphs for 3.1/3.2 should first report the bandwidth [GB/s] on the y-axis and the size on the x-axis. If another parameter is required (threads per block, block count, etc), use different lines and a legend. For 3.3/3.4, we will change the use of the x-axis.

3.1 Global Memory - cudaMemcpy

First, test the performance of device (global) memory accesses by using cudaMemcpy: start with the program for PCIe data movements you developed last exercise. Modify the program to measure the performance of device-to-device cudaMemcpy too. Make sure to only copy between two memory locations on one GPU, not between multiple GPUs. Please note that you may need to synchronize the device after each cudaMemcpy call, see[1].

- Report throughput graphically (D2H/H2D pageable, pinned and D2D) in achieved GB/s!

(6 points)

3.2 Global Memory – coalesced thread access

Now, measure the performance of device memory accesses from thread level. Use the code skeleton we provide with this exercise. Implement a kernel that copies device memory using threads. Note that it might be necessary that one thread copies more than one data element (depending on the configuration). Ensure that the access pattern is coalesced, i.e. thread i accesses data[i] while thread i+1 accesses data[i+1], etc.

- Use one thread block and vary the number of threads per block between 1 and 1024. Report graphically (size vs throughput) and interpret your results!
- Choose the thread count that was optimal according to the previous experiment. Now vary the number of thread blocks from 1 to 32. Report graphically (size vs throughput) and interpret your results!
- Try to optimize the threads per block and block count to yield optimal performance. Report graphically (choose x- and y-axis as necessary) and interpret!

(18 points)

[1] `https://docs.nvidia.com/cuda/cuda-runtime-api/api-sync-behavior.html#api-sync-behavior__memcpy-sync`

3.3 Global Memory – thread access with stride

Instead of accessing contiguous data, now introduce a stride in the access pattern to unveil coalescing effects. Modify the previous program to include a stride in the access pattern, i.e. out[i] = in[i*stride]. Choose the parameters threads per block and block count in a way that together with the size per element they make up the data size, i.e. each thread transfers exactly one element.

- Report graphically (stride vs throughput for a given data size), compare against coalesced access and interpret your results!
- Give a possible explanation if your results do not match your expectations!

(8 points)

3.4 Global Memory – thread access with offset

Now introduce an offset in the access pattern to unveil coalescing effects. Modify the previous program to include an offset in the access pattern, i.e. out[i] = in[i+offset]. Choose the parameters threads per block and block count in a way that together with the size per element they make up the data size, i.e. each thread transfers exactly one element.

- Report graphically (offset vs throughput for a given data size), compare against coalesced access and interpret your results!
- Give a possible explanation if your results do not match your expectations!

(8 points)

3.5 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.

- Global Memory - cudaMemcpy (Section: 3.1)
- Global Memory – coalesced thread access (Section: 3.2)
- Global Memory – thread access with stride (Section: 3.3)
- Global Memory – thread access with offset (Section: 3.4)

(20 points)

Total: 60 points