Practical Course on Parallel Computing

Assignment Sheet 8

Assignment 1

a) What kind of performance challenges does recent CPU development face, i.e. why would one consider utilizing GPUs for general computing purposes in the first place?

The “pure” power of CPUs are not improved every year, because it is not possible anymore to place more transistors or provide them with more power, because this would result in a greater heat generation. As a follow up the improvement of CPUs are focusing more on parallelization. To effectively use the parallelization, the programs need to be rewritten. GPUs also support parallelization. In detail they are effective to perform the same instruction on a big chunk of data, especially for single and floating point instructions . Moreover, the GPU has a higher memory bandwidth then a CPU and is more scalable. To include the GPU in the program some special libraries need to be include and the code rewritten. If one wants to (re-) write a program with many of the same single or floating point instructions, it should be considered to do it on the GPU, because it could be more faster than on a CPU.

b) Describe the GPGPU approach to parallel computing. Which steps have to be taken before code is actually run on the graphics device?

The program needs to partitioned in into blocks of threads. Depending on the program, memory has to be allocated for the host as well as on the GPU. Moreover, the needed data has to be copied in the correct section of memory. In order to determine these and also to use later the correct pointers to access the data one needs to know the structure of the GPU. The code needs be compiled via a special CUDA compiler that understands the compiler directives used to offload computations to the GPU.

c) If a kernel is run on the GPU, how are its parallel threads organized? Pick a Compute Capability version of your choosing to explicitly discuss the constraints that are in place with respect to the number of threads and their memory requirements.

On a GPU threads are grouped in blocks. These blocks are grouped in one grid again. In the block threads are organized in warps. All threads in one warp perform the same instruction. In Compute Capability 5.2 there can be max. 32 Warps. The number of threads per block is limited to 1024. There are 32 shared memory banks. Each thread has 512KB local memory. Threads can share max 48KB with each other.

d) Under which circumstances can CUDA threads communicate without resorting to the global device memory? How can you ensure that in such a case reading and writing data happens in the correct order?

The shuffle operation allows threads inside a warp to exchange data with each other without resorting to shared or global memory. For inter block communication shared memory can be used. To synchronize access to this memory different synchronization routines are available, e.g. a simple barrier `__synchthreads()` or atomic operations.