Hands-On — Introduction to GPU Computing with CUDA

Siegfried Höfinger

VSC Research Center, TU Wien

October 18, 2022

→ https://tinyurl.com/cuda4dummies/i/ho1/notes-ho1.pdf

Exercise

Q1) *Figure out what type of GPU is installed on the compute-node you had been given access to. Is it a device of type "enterprise grade" or of type "consumer grade" ? Is there a single GPU on-board, or are there multiple GPUs (if so how many and how are they inter-linked) ? What could be the most convincing architectural feature to acquire such a device for the purpose of scientific computing ?*

10 min

A1)

i) *The command to use for querying basic GPU information on a particular compute node is*
nvidia-smi
which reveals "NVIDIA A100-PCIE-40GB"

ii) *"A100" is of type "enterprise grade"*

iii) *There are two A100 GPUs on these nodes interlinked via slowest SYS connects;*

iv) *"A100" is the current flagship model in NVIDIA's portfolio with 40 GB on-board memory and very high memory bandwith of 1.6 TB/s. The greatest design advantage is its strong FP64 peformance of 10 TFLOPs/s or even 20 TFLOPs/s when operating tensor cores;*

Exercise

Q2) *Examine the discussed example,*
single_thread_block_matrix_addition.cu
compile and execute it and see whether it's creating the output expected;

10 min
→ https://tinyurl.com/cuda4dummies/i/l1/single_thread_block_matrix_addition.cu

A2)

i) *Look into the mentioned sample program*
vi ./single_thread_block_matrix_addition.cu

ii) *Once everything is clear, compile it directly on the GPU node using,*
nvcc ./single_thread_block_matrix_addition.cu

iii) *Run the resulting executable, a.out, directly on the GPU node,*
./a.out

iv) *Examine the output and see whether or not it can serve as a proof of correctness*

Q3) *For any *.cu code where the size of a given array, N, is not an integral multiple of the anticipated size of the threadblock, how can we improve the kernel (and related code sections) to properly work on such arbitrary sized arrays ?*

10 min

A3)

i) *Let's take the previous example and modify the dimension of the threadblock to* $(N+1) \times (N+1)$

```
cp ./single_thread_block_matrix_addition.cu \
./single_thread_block_matrix_addition_mod.cu
vi ./single_thread_block_matrix_addition_mod.cu
... threadsPerBlock.x = N + 1;
```

ii) *Since now there are threads that would refer to non-existing array elements, we need to exclude these cases in the kernel,*

```
... if ( (i < N) && (j < N)) { .... }
```

iii) *Compile and run it as previously,*

```
nvcc ./single_thread_block_matrix_addition_mod.cu
./a.out
```

→ https://tinyurl.com/cuda4dummies/i/l1/single_thread_block_matrix_addition.cu
→ https://tinyurl.com/cuda4dummies/i/ho1/single_thread_block_matrix_addition_v2.cu