

HANDS-ON — INTRODUCTION TO GPU COMPUTING WITH CUDA

Siegfried Höfingier

VSC Research Center, TU Wien

March 8, 2020

→ <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 striking design feature to acquire such a device for the purpose of scientific computing ?*

10 min

- A1)**
- i) *Running (in some terminal) the command `nvidia-smi` reveals “Tesla V100-SXM2” or “TITAN V” or “GeForce GTX 1080”*
 - ii) *“GeForce GTX 1080” and “TITAN V” are “consumer grade”, “Tesla V100-SXM2” is “enterprise grade”*
 - iii) *Again, running in some terminal `nvidia-smi topo --matrix` tells us that the node featuring “Tesla V100-SXM2” GPUs has 4 of them connected via NV-LINK-2, the other nodes have either 4 or 7 GPUs connected via PCIe, or just a single GPU on-board;*
 - iv) *“Tesla V100-SXM2” GPUs are extremely powerful devices featuring up to 7 TFLOPs in FP64 precision; however, they are expensive ! “TITAN V” is “consumer grade” at approximately $\frac{1}{3}$ the price of “Tesla V100-SXM2” with pretty much comparable hardware specs, i.e. Tesla-V100-like, however lacking NV-LINK-2;*

Exercise

- Q2)** *Examine the discussed example, `single_thread_block_matrix_addition.cu` compile and execute it and see whether it's creating some "noise" on the GPU;*

10 min

→ https://tinyurl.com/cuda4dummies/i/11/single_thread_block_matrix_addition.cu

- A2)**
- i) *Download/transfer/copy-paste/somehow-get-the-thing-here, e.g.*
`wget https://tinyurl.com/cuda4dummies/i/11/single_thread_block_matrix_addition.cu`
 - ii) *Compile it like,*
`nvcc ./single_thread_block_matrix_addition.cu`
 - iii) *Run it like,*
`./a.out`
 - iv) *In a second terminal window launch a “monitoring” nvidia-smi session, in particular,*
`watch -n 0.1 nvidia-smi`
then run the executable again and check whether this can be observed in the 2nd monitoring terminal;

Q3) *For any *.cu code where the dimension of a given array, N , is not an integral multiple of the anticipated dimension 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 $N+1 \times N+1$*
vi ./single_thread_block_matrix_addition.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,*
vi ./single_thread_block_matrix_addition.cu
... if ((i < N) && (j < N)) {
C[i][j] = A[i][j] + B[i][j];
}
 - iii) *Compile and run it as previously*

→ 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