Hands-On — CUDA SDK - Basic Concepts

Siegfried Höfinger

VSC Research Center, TU Wien

October 2, 2021

→ https://tinyurl.com/cuda4dummies/ii/ho3/notes-ho3.pdf

Outline

Hands-On — CUDA SDK - Basic Concepts

Exercise

Q1) *A simple example to probe tensor core operation is given in the SDK — 0_Simple/cudaTensorCoreGemm. Create your own space (maybe a separate dir myplayground), go there and copy over the sample from the SDK, examine the *.cu, compile it and run it. Can we check whether the GPU is really doing something ? Could we make the compute process on the GPU a bit "heavier" so that we can monitor it better ?*

15 min

A1) *Provided we currently reside in some private directory and have realized that the environmental variable $CUDA_PATH points us to the SDK, we can get the sample here,*

```
cp -r $CUDA_PATH/NVIDIA_CUDA-11.0_Samples/0_Simple/cudaTensorCoreGemm ./
cd cudaTensorCoreGemm
vi Makefile ( get rid of the ../../ in the end to not touch relative sites )
make
./cudaTensorCoreGemm
```

In a second xterm run

```
watch -n 0.1 nvidia-smi
```

and observe GPU 0/1 activity. If we increase M/N/K_TILES to 1024 GPU-Util in nvidia-smi will be monitoring >0%.

→ https://tinyurl.com/cuda4dummies/ii/t/cudaTensorCoreGemm_v2.cu

Exercise

Q2) *Consider the SDK sample 0_Simple/simplePrintf. Again, copy/compile/run it in some private space. Think about 2 modifications inserting an assert() call in the kernel code. One causing no termination, the other triggering exit/abortion, ideally dependent on some value of threadIdx/blockIdx.*

10 min

A2) *Repeat the copying/Makefile-editing/compiling for 0_Simple/simplePrintf then examine the kernel code. A simple non-harmful assert() could be*
assert(val < 100);
while another one causing exit could be
assert(k < 3);
with
*k = (blockIdx.y * gridDim.x) + blockIdx.x;*

→ https://tinyurl.com/cuda4dummies/ii/t/simplePrintf_v2.cu
→ https://tinyurl.com/cuda4dummies/ii/t/simplePrintf_v3.cu

Exercise

Q3) *Using again assert() in the kernel code of the SDK sample 0_Simple/simplePrintf, how could we quickly identify whether the shape of the used threadblock is 1D/2D/3D ?*

10 min

A3) *A useful threadblock must at least consist of a single thread, hence have threadIdx.x/y/z = 1/0/0. If the threadblock is 1D, then for all other threads threadIdx.y/z will remain 0, which can be probed from assert(threadIdx.z == 0) and assert(threadIdx.y == 0) (see below variant for download and own experiments).*

→ https://tinyurl.com/cuda4dummies/ii/t/stream_test_v4.cu

Exercise

Q4) *Verify the maximum available bandwidth of ≈ 25 GB/s for host-to-device transfers on the A40 architecture using the SDK sample 1_Utilities/bandwidthTest. Examine what could be optionally tested (calling ./bandwidthTest –help). Is the confirmed limit value a reasonable estimate ?*

10 min

A4) *Repeat the copying/Makefile-editing/compiling for 1_Utilities/bandwidthTest and run the sample program. Playing around with various command line arguments does actually confirm the limit of 25 GB/s and that is probably to be expected from Gen4 PCIe.*

Exercise

Q5) *Use the low-level profiling toolchain,*
nsys nvprof ./bandwidthTest -htod
and examine the output. Based on this, can we get some confirmation of the bandwidth computed by the SDK sample ?

15 min

A5) *From the profile we get confirmation of the amount of transferred data per copy to be 31250 kiB, i.e. 0.032 GB. The corresponding time for a single transfer read from the profile is ≈ 1225000 ns, i.e. 0.001225 s, so the resulting bandwidth then is, 0.032 / 0.0012249057 = 26.1 GB/s which is pretty close to what the SDK sample had told us.*