Tutorial 2: Exploring CUDA

Siegfried Höfinger

VSC Research Center, Vienna University of Technology

October 28, 2024

→ https://tinyurl.com/cudafordummies/ii/t/notes-t2.pdf

VIENNA SCIENTIFIC CLUSTER

Outline

Second Steps with CUDA
CUDA SDK

Exercise

Q1) *Using again assert() in the kernel code of the SDK sample 0_Introduction/simplePrintf, how could we quickly identify whether the shape of the used threadblock is 1D/2D/3D ?*

10 min

Second Steps with CUDA cont.

CUDA SDK cont.

A1) *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).*
n.b. first change to private dir (or create it from scratch)

```
cd wherever_the_SDK_may_be/Samples/0_Introduction/my_simplePrintf
```

then edit simplePrintf.cu therein (or download the variant and move/copy it to simplePrintf.cu) then compile and run the modified version

```
make
./simplePrintf
```

→ https://tinyurl.com/cudafordummies/ii/t/simplePrintf_v4.cu

Second Steps with CUDA cont.

CUDA SDK cont.

Exercise

Q2) *Verify the maximum available bandwidth of $\approx$ 25 GB/s for host-to-device transfers on the A100 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

VIENNA SCIENTIFIC CLUSTER

Second Steps with CUDA cont.

CUDA SDK cont.

A2) *Repeat the copying/compiling for 1_Utilities/my_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.*

Second Steps with CUDA cont.

CUDA SDK cont.

Exercise

Q3) *Use the low-level profiling toolchain,*
nsys nvprof ./bandwidthTest --htod
and examine the output. Based on this, can we get confirmation of the bandwidth computed by the SDK sample ?

15 min

Second Steps with CUDA cont.

CUDA SDK cont.

A3) *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 $\approx$ 1576512 ns, i.e. 0.001577 s, so the resulting bandwidth then is, 0.032 / 0.001577 = 20.3 GB/s which is pretty close to what the SDK sample keeps telling us.*

Second Steps with CUDA cont.

CUDA Libraries: CUSOLVER

Exercise

Q4) *Solve the eigenvalue/eigenvector problem of the following matrix*

$$\mathbf{M} = \begin{pmatrix} 1.96 & -6.49 & -0.47 & -7.20 & -0.65 \\ -6.49 & 3.80 & -6.39 & 1.50 & -6.34 \\ -0.47 & -6.39 & 4.17 & -1.51 & 2.67 \\ -7.20 & 1.50 & -1.51 & 5.70 & 1.80 \\ -0.65 & -6.34 & 2.67 & 1.80 & -7.10 \end{pmatrix}$$

using cusolverDN

20 min

VIENNA SCIENTIFIC CLUSTER

Second Steps with CUDA cont.

CUDA Libraries: CUSOLVER cont.

A4) *Eigenvalues are,*
-11.07 -6.23 0.86 8.87 16.09
and eigenvectors those already mentioned in Q1 of the hands-on session 4
(see below version for download)

→ https://tinyurl.com/cudafordummies/ii/t/chck_cusolver_syevd.cu
→ https://tinyurl.com/cudafordummies/ii/ho4/notes-ho4.pdf

Second Steps with CUDA cont.

NVIDIA Nsight Compute CLI

Exercise

Q5) *Use the 3-step process for NVIDIA Nsight Compute CLI to determine causes of improvement when profiling the series mmm_example_[1-3].cu*

→ `https://docs.nvidia.com/nsight-compute/NsightComputeCli/index.html`
→ `https://tinyurl.com/cudafordummies/ii/t/mmm_example_1.cu`
→ `https://tinyurl.com/cudafordummies/ii/t/mmm_example_2.cu`
→ `https://tinyurl.com/cudafordummies/ii/t/mmm_example_3.cu`
15 min

Second Steps with CUDA cont.

NVIDIA Nsight Compute CLI cont.

A5)

i) *Adding a baseline (using the most simplistic approach) and doing relative comparisons to this baseline is a very straightforward way of identifying improvements in different implementations; for example, mmm_example_2.cu versus mmm_example_1.cu (baseline) immediately points out improvements in terms of memory access, but degradation in SM performance mainly because of fewer instructions per cycle, which is a consequence of a reduced number of warps (eligible as well as active) per cycle making warp cycles per issued instruction increase; also simply looking at Duration [msecond] reveals an almost doubling of the exe-time while memory throughput has dropped from 67 GB/s to 2 GB/s*

ii) *mmm_example_3.cu versus mmm_example_1.cu (baseline) demonstrates a significant reduction in exe-time (Duration) at better SM utilization and comparable memory access (both at 80% of theoretical max, aka SOL); here the inverse trend is observed w.r.2 instructions/cycle and warps dynamics; memory throughput however has also dropped from 67 GB/s to 17 GB/s*

Second Steps with CUDA cont.

CUDA Streams

Exercise

Q6) *Check whether we could make use of CUDA managed unified memory, i.e. cudaMallocManaged(), within applications using CUDA streams, for example stream_test.cu*
Evaluate concurrency with the help of nvvp.

→ https://tinyurl.com/cudafordummies/ii/t/stream_test.cu
15 min

Second Steps with CUDA cont.

CUDA Streams cont.

A6) *Yes, in principle (see below version for download).*
However, care must be taken to synchronize individual streams before managed unified memory can be accessed on the host in the usual manner.

→ `https://tinyurl.com/cudafordummies/ii/t/stream_test_v2.cu`