Hands-On — CUDA SDK - Basic Concepts

Siegfried Höfinger

VSC Research Center, TU Wien

March 8, 2020

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

Outline

Hands-On — CUDA SDK - Basic Concepts

Exercise

Q1) *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/cuda4dummies/ii/t/stream_test.cu
10 min

A1) *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/cuda4dummies/ii/t/stream_test_v2.cu

Exercise

Q2) *What happens if we drop the kernel call to the default stream and is the mentioned compile flag --default-stream per-thread required under all circumstances ?*

10 min

A2) *The resulting multi-stream code achieves good concurrency as long as no unspecified kernel call (going into the default stream) is among the execution configurations. If the latter is true, there is no need for the compiler flag --default-stream per-thread*

→ https://tinyurl.com/cuda4dummies/ii/t/stream_test_v3.cu

Exercise

Q3) *What is the observed behaviour regarding stream concurrency if we introduce true grids per stream rather than dummy grids with just a single threadblock ?*

15 min

A3) *The resulting concurrency pattern becomes quasi-sequential with individual streams executed one after the other. So since the initial kernel launch already consumes all of the avaiable CUDA cores, all subsequent streams have to wait for resources to become vacant again.*

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

Exercise

Q4) *How could one introduce OpenMP here ? Imagine a scenario where instead of the loop over i we have an OpenMP code branching out into individual threads on the host, each of them operating its own stream on the GPU in concurrent manner; wouldn't that be the simplest way to make optimal use of a particular compute server ?*

15 min

A4) *In principle the default implementation was already operational (see below version for download). However, concurrency of streams becomes visible only within nvvp not within nvidia-smi for example ! There is also a certain delay for concurrent streams, hence going parallel with OpenMP is not without overhead !*

→ https://tinyurl.com/cuda4dummies/ii/t/stream_test_v5.cu

Exercise

Q5) *What could we do to make individual OMP threads visible within a standard 'top' in the xterm ?*

10 min

A5) *Introduce a dummy loop inside the OMP-thread section (see below version for download), or alternatively inside the kernel section itself; again, concurrent streams will not become visible inside nvidia-smi ! The non-perfect level of concurrency of individual OMP threads is hardly visible in nvvp due to the extended runtime !*

→ https://tinyurl.com/cuda4dummies/ii/t/stream_test_v6.cu