Code Coffee: Introduction to GPU programming with CUDA

Korbinian Huber

Excellence cluster Universe

July 17, 2018

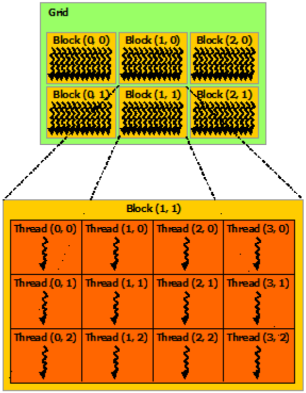
Outline

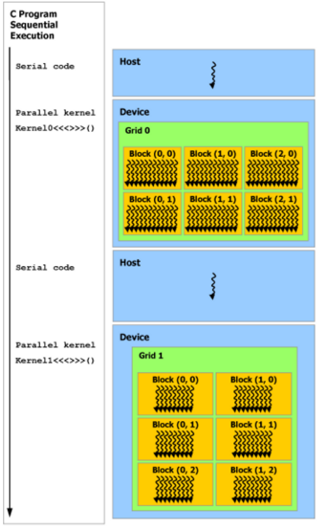
- Core concepts
- a most simple example
- CUDA and Python
- CUDA libraries

Core concepts

- CUDA C/C++: Nvidia's GPGPU language
- GPU vs. CPU: grid of many (albeit slower) cores; ideal for massively data-parallel tasks of a certain size
- Requirements: Nvidia GPU, Nvidia driver, CUDA toolkit

Core concepts





taken from `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`

Core concepts

When to use it?

- large data (but not too large)
- many data parallel tasks
- locality of data access
- limited interaction between host and device

Core concepts

dedicated memory types

- global: read/write, GBs, off-chip
- shared: read-only, kBs, on-chip, shared between threads of same block
- constant: read-only, kBs, on-chip
- texture: read-only, out of global but cached on chip, optimized access patterns

Simple example

adding vectors

CUDA and Python

no "official" package, but several frameworks

- numba
- PyCUDA
- Copperhead
- ...

Numba

known from last code coffee

- allows for simple device usage via `@vectorize(['float32(float32, float32)'], target='cuda')`
- … but also for elaborate device kernels via the `@numba.cuda.jit` decorator
- nice examples online https://github.com/ContinuumIO/gtc2017-numba

Numba

CUDA Libraries
Drop-in acceleration

- CUDA versions of classics (cuFFT, cuBLAS), cuRAND,..) leave API almost unchanged
- template library for standard parallel algorithms: Thrust
- neural networks (cuDNN, TensorRT,...)
- ...