Tutorial 1

Organization, CUDA Setup, Assignment 1

Organization

- Max
  - mxkl@uni-bremen.de
  - MZH 3530
- Thomas
  - hudo@uni-bremen.de
  - MZH 3490
- Discord: https://discord.gg/xyxc7Rff
  - Write me a message on Discord to get access (hudo259)
  - Or ask in the general channel

Organization

- Assignment every 2 weeks
  - New assignment Wednesday
  - Turn in Sunday 11:59pm before next assigment
  - Groups of 3
  - Email tutor (subject include MPA, last name of group members, assignment number)
    - First 3 assigments to me (hudo@uni-bremen.de)
    - Last 3 assigments to Max (mxkl@uni-bremen.de)

- Don’t package your binaries and other intermediate output. Just send the .cu/.h/.c/.cpp files and the PDF (containing the non-programming exercises and preferably some basic explanation of your code, if necessary)
- Code and theoretical tasks will be discussed in next subsequent tutorial.

CUDA Setup

- Software
  1. Install Cmake
  2. Install CUDA (see the guides linked in exercise 0 of the first assignment!)
     1. Windows: Install Visual Studio (>=2019).
     2. Linux: Use VSCode or another editor (We recommend you use your package manager to install the CUDA Toolkit/SDK…)
  3. Optional: You can also just use the official Nvidia Docker image (you may need to install Cmake as root on it first… –u root):

```
docker run -it --rm --gpus all --volume $(pwd):$(pwd):rw --user $(id -u):$(id -g) --net=host nvidia/cuda:12.3.2-devel-ubuntu22.04
```

- Hardware
  - Personal computer with NVIDIA GPU (Windows 11, Visual Studio 2019 and 2022)
  - CGVR-Lab laptops (MZH 3590)
    - Lab entry
      - with chip (send me a message on Discord to request access rights)
        - Admin: Sabine Dolhs (sdolhs@uni-bremen.de)
      - Call someone on the floor and ask for entry
    - PIN for lab: [In person/ask in Discord]
    - Account: stud1; Password: You can find it on the laptops

Assignment 1

- (Install CUDA & compiler)
- CUDA
  - Extension of C++ language
  - *.cu-Files, compiled with nvcc
  - Provided CMake file for our assignments.
- Basic GPU memory management
  - cudaMalloc(void **ptrToDevPtr, size_t size);
  - cudaMemcpy(void *dst, void *src, size_t cnt, cudaMemcpyKind kind);
    - Return cudaError_t
    - Alternative: cudaGetLastError() returns last raised error (for asynchronous calls)
  - cudaFree(void *ptr);

Assignment 1

- A note on compilation:
- Windows:
  - Option 1: Start Visual Studio and open the project “from CMake” and build it.
  - Option 2: Navigate to folder containing CMakeLists.txt, open a terminal and:
    - mkdir build
    - cd build
    - cmake .. –G “Visual Studio 16 2019” –A x64 (or another Visual Studio version)
    - open the generated .sln file in Visual Studio and select the cudaMemcpyAndMalloc as primary solution and build it or: cmake --build .
- Linux: Like Windows option 2 but instead of last step:
  - Just: cmake --build .
  - Use Nsight for VSCode to automate this(?)