

1.st Task in *HPC-I*

Deadline: Nov 7, 2023, 11:59pm

GPU computing using CUDA

1. Start with CUDA

Install CUDA on your computer

- manjaro-Linux: `sudo pacman -S nvidia nvidia-utils cuda` (hints¹)
- ubuntu-Linux: `sudo apt install nvidia-cuda-toolkit cuda` (hints²)
- Running Ubuntu in Windows via WSL2 requires appropriate drivers which is described in general by Microsoft³ and in detail by NVIDIA⁴.

and check

- Compiler: `nvcc --version`
- GPU/CUDA: `nvidia-smi` (or `nvidia-smi -L`)
- example Code⁵

```
cd CUDA/firstSteps
nvcc data_mv_GH.cu
./a.out
```

2. Your first CUDA Code

Copy the *CUDA* directory into your directory and download⁶

the Code from [HaSh19] and use `float` as data type for all tasks.

Start with your copy of file *data_mv_GH.cu*:

[5 pts]

- The given code realizes on GPU $\underline{b} := \underline{a}$ followed by $++\underline{b}$.
Extend the code with a kernel function for $\underline{c} := \underline{a} + \underline{b}$. Check the result on the host.
- Compare your code with code *vector_addition_gpu_thread_block.cu*⁷ from [HaSh19, §1].
- Extend your code with two kernel functions for $\underline{b} := \ln(\underline{a})$ and $\underline{c} := \exp(\underline{b})$. Check the result \underline{c} on the host.
- Write a second main function that uses unified memory⁸ (intro⁹) instead of the `malloc-cudaMalloc-cudaMemcpy` framework. Have a look at code *unified_memory.cu*¹⁰ from [HaSh19, p.70f] on how to use `cudaMallocManaged`.
- Add timing for the kernel calls, see general performance metrics¹¹ on GPU.

¹<https://miloserdov.org/?p=4181>

²<https://linuxconfig.org/how-to-install-cuda-on-ubuntu-20-04-focal-fossa-linux>

³<https://docs.microsoft.com/en-us/windows/win32/direct3d12/gpu-cuda-in-wsl>

⁴<https://docs.nvidia.com/cuda/wsl-user-guide/index.html>

⁵http://imsc.uni-graz.at/haasegu/Lectures/GPU_CUDA/CUDA.zip

⁶<https://github.com/PacktPublishing/Learn-CUDA-Programming>

⁷https://github.com/PacktPublishing/Learn-CUDA-Programming/tree/master/Chapter02/02_memory_overview/02_vector_addition/vector_addition_gpu_thread_block.cu

⁸<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#um-unified-memory-programming-hd>

⁹<https://devblogs.nvidia.com/unified-memory-cuda-beginners/>

¹⁰https://github.com/PacktPublishing/Learn-CUDA-Programming/blob/master/Chapter02/02_memory_overview/06_unified_memory/unified_memory.cu

¹¹<https://devblogs.nvidia.com/how-implement-performance-metrics-cuda-cc/>

- By using CPU-timing (`clock()` or `std::chrono::system_clock`). Don't forget the call `cudaDeviceSynchronize()` to wait until the kernel functions finish on GPU.
- By using `cudaEvent_t`, see example.

3. Reduction We need frequently a reduction function, i.e., to compare two vectors on the GPU in the previous tasks.

Have a look at codes in *CUDA/skalar* computing the inner product of two vectors.

- *skalar_3_fast.cu*: The data management is similar to the previous task. The inner product calculation follows Mark Harris' presentation in *CUDA_intro/reduction_Mark_Harris.pdf* with its own kernel functions.

```
nvcc --ptxas-options=-v -restrict skalar_3_fast.cu
```

- *skalar_4.cu*: Uses cuBLAS for inner product calculation.

```
nvcc -restrict skalar_4.cu -lcublas
```

- *skalar_5.cu*: Uses the Thrust¹² library for vector management in combination with STL-algorithms. See the documentation¹³.

```
nvcc --ptxas-options=-v -restrict skalar_5.cu
```

Your tasks for realizing $\underline{c} == \underline{a}$ on GPU:

[8 pts]

- Write a kernel function (e.g.: `equal`) for comparing $\underline{c} == \underline{a}$. You need a reduction operation similar to the inner product in *CUDA/skalar/skalar_3_fast.cu*.
- See also the reduction kernel in [HaSh19, §3, p.117-126] and the code versions¹⁴ of it.
- Copy and Rewrite your code from Task 2 by using Thrust. You might use only `thrust::reduce()`, see [NLS14] for combining unified memory with Thrust. You might also rewrite all vector operations regarding Thrust.
- ?? Can we call cuBLAS routines as `cublasDdot` with unified memory vectors or/and Thrust vector ??

4. Using cuBLAS 1-3 The basic idea of these tasks consist in applying cuBLAS¹⁵ (BLAS) calls for all vector and/or matrix operations. Check also the runtime for reasonable matrix sizes.

Start with columnwise stored dense matrices (example *CUDA/densmatrices_libs*. Use `float` as data type. See also [HaSh19, §8, p.320ff] and its code¹⁶.

[8 pts]

- BLAS1¹⁷: Realize $\underline{y} := \alpha \underline{x} + \underline{y}$; $\underline{x} := \alpha \underline{x} + \underline{y}$; $\underline{z} := \alpha \underline{x} + \beta \underline{y}$; $\langle \underline{x}, \underline{y} \rangle$; $\| \underline{x} \|^2$;
- BLAS2: Realize $\underline{r} = M * \underline{x}$; $\underline{r} = M^T * \underline{x}$ with a non-symmetric dense real matrix M .
- Realize $\underline{r} = T \cdot \underline{x}$ with a tridiagonal matrix $T = [-1, 2, -1]$ (assuming non-constant sub-/diagonal entries). Take advantage of the matrix structure.
- BLAS3: Realize $A = M * M^T$ and compare the result of $\underline{z} := A * \underline{x}$ with $\underline{y} := M * (M^T * \underline{x})$.
- Extract the diagonal from a dense matrix M with a BLAS1-function.

¹²<https://developer.nvidia.com/thrust>

¹³<https://docs.nvidia.com/cuda/thrust/index.html>

¹⁴https://github.com/PacktPublishing/Learn-CUDA-Programming/tree/master/Chapter03/03_cuda_thread_programming

¹⁵<https://developer.nvidia.com/cublas>

¹⁶https://github.com/PacktPublishing/Learn-CUDA-Programming/tree/master/Chapter08/08_cuda_libs_and_other_languages/01_sgemm

¹⁷http://www.netlib.org/blas/#_level_1

References

- [HaSh19] Jaegeun Han and Bharatkumar Sharma. *Learn CUDA Programming*. Packt> (2019); buy¹⁸; download code¹⁹, download figures²⁰
- [NSLS14] Dan Negrut, Radu Serban, Ang Li and Andrew Seidl. *Unified Memory in CUDA 6: A Brief Overview and Related Data Access/Transfer Issues*. TR-2014-09²¹ (2014)
- [ChAn17] Andrzej Chruszczyk and Jacob Anders *Matrix computationson the GPUCUBLAS, CUSOLVER and MAGMA by examples*. NVIDIA (2017); download book²².

¹⁸<https://www.packtpub.com/eu/application-development/cuda-cookbook>

¹⁹<https://github.com/PacktPublishing/Learn-CUDA-Programming>

²⁰https://static.packt-cdn.com/downloads/9781788996242_ColorImages.pdf

²¹https://www.researchgate.net/publication/326920953_Unified_Memory_in_CUDA_6_A_Brief_Overview_and_Related_Data_AccessTransfer_Issues

²²<https://developer.nvidia.com/sites/default/files/akamai/cuda/files/Misc/mygpu.pdf>