Seminar Work (KI/KPAR)

Lecturer: prof. Zbyšek Posel

Information

Date: 13. 05. 2025

Terms:
- The language for seminar work is English.
- Seminar work contains program part (codes in Python) and text part (document Word/pdf with details).
- Cooperation is allowed on program part.
- Text part is submitted individually.
- Text part contains:
  - i) topic description
  - ii) details including description of solutions, simplified code layout or workflow,
  - iii) results (Figures, tables etc.),
  - iv) Final report including literature

Deadline: No later than 04. 07. 2025

Not seminar work nor its corrections will be accepted after the deadline.

Each student has one particular topic listed below.

Name	Topic
Kopecký Jakub	Calculation of Fourier transformation in CUDA using Numba package
Kotlan Petr	Evaluation of autocorrelation function with CUDA kernel
Trejdlová Anna	N-dimensional integration using mean value Monte Carlo integration
Priban Lukáš	Strong and weak scaling of parallel text processing using GPU

In addition, students in distance form of education

- **choose their own topic independently,**
- **can submit their work no later than 21. 09. 2025**

1. Calculation of Fourier transformation in CUDA using Numba package

Write the CUDA kernel for calculating the Discrete Fourier transformation (DFT), where DFT calculates the contribution of k-th frequency as follows

$$\overline{S_k} = \sum_{n=0}^{N-1} s_n e^{-i\frac{2\pi}{N}nk}$$

Evaluate the following matrix equation

$$\begin{pmatrix} \overline{S_0} \\ \overline{S_1} \\ \overline{S_2} \\ \vdots \\ \overline{S_{N-1}} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & W^1 & W^1 & \dots & W^{N-1} \\ 1 & W^2 & W^3 & \dots & W^{N-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W^{N-1} & W^{N-2} & \dots & W^1 \end{pmatrix} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ \\ s_{N-1} \end{pmatrix}$$

where $W = e^{-i\frac{2\pi}{N}}$ using DFT or FFT variants.

- In case you choose DFT, implement any standard method for evaluation the matrix equation and construct the W matrix by yourself.
- In case you choose FFT, implement Cooley Tukey algorithm for fast evaluation of matrix equation.
- Compare the results with one of the options listed below
  - Numpy *fft* implementation
  - *cuFFT* CUDA function
  - *cupy.fft* from CuPy package

Deliver these graphical outputs:

- Original function and its spectra obtained from DFT, FFT and from Numpy fft.
- Comparison of scaling behaviour (Numpy, DFT, FFT) with increasing *N*.

2. Evaluation of autocorrelation function with CUDA kernel

Write the CUDA kernel for calculation of autocorrelation function using Python language and Numba package.

Choose an arbitrary function $f(x)$ where $\{x_i\}_{1\ldots N}$ are N function points and calculate autocorrelation function ACF as

$$\mathrm{ACF(lag)} = \sum_{lag=0}^{\frac{N}{2}} \sum_{i=1}^{N-lag} f(x_i) \cdot f\left(x_{i+\mathrm{lag}}\right) \Big/ \sum_{i=1}^{N} f(x_i) \cdot f(x_{i+0})$$

Compare the calculation with parallel implementation in Python and with serial version (can be that one implemented in Numpy package).

Deliver these graphical outputs:

- Arbitrary function and its ACF.
- Comparison of ACFs from serial, parallel and CUDA. Must be the same!
- Comparison of calculation times for CUDA, parallel and serial version as a function of N variable.

Discuss the effectivity of implementation using CUDA kernel and determine the borderline where the CUDA approach is beneficial.

3. N-dimensional integration using mean value Monte Carlo integration

Use the Monte Carlo mean value method to calculate integrals in 1 to 10 dimensions. Consider the following integral

$$I = \int_{-3}^{3} e^{-x_1^2} \mathrm{d}x_1 \, , \int_{-3}^{3}\int_{-3}^{3} e^{-x_1^2-x_2^2} \mathrm{d}x_1 \mathrm{d}x_2 \, , \int_{-3}^{3}\int_{-3}^{3}\int_{-3}^{3} e^{-x_1^2-x_2^2-x_3^2} \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}x_3 \, , \ldots$$

The Monte Carlo mean value method approximates the exact value of the n-dimensional integral as follows.

$$I = \int_{\Omega} f(\bar{x}) \, \mathrm{d}\bar{x} \approx \frac{V(\Omega)}{N} \sum_{i=1}^{N} f(\bar{x}_i)$$

where N is the number of n-dimensional vectors $\bar{x}_i$ that are generated in the space $\Omega \in R^n$ and $V(\Omega)$ is the volume of the space given by the function $f(\bar{x})$.

Discuss your GPU implementation as follows

- compare the outputs and time consumption up to dimension 3 with the built-in functions of Numpy and Scipy packages
- show the time complexity of the implementation up to dimension 10

4. Strong and weak scaling of parallel text processing using GPU

In parallel, use the GPU to process a coherent and comprehensible text of at least 4000 characters without spaces (equivalent to approximately 1 A4 page at font size 11 and line 1). Read the text from an external txt or word file and perform the following analysis.

- Statistics of the occurrence of individual letters (do not distinguish upper- and lower-case letters, do not distinguish individual characters).
- Statistics on the occurrence of each letter at the beginning of a sentence (following the period).

Discuss your GPU implementation as follows.

- Compare the speed with the parallel implementation on the CPU
- Perform strong and weak scaling and discuss the limits of your GPU implementation