# CUDA — Laboratorium 02

Sprawozdanie

Mateusz Łopaciński

23 czerwca 2025

## Spis treści

# 1 Redukcja (Zadanie 1)

## Konfiguracja

- Dwa programy: `reduction_global` oraz `reduction_shared`.
- Wektor wejściowy $N = 2^{24}$, 30 iteracji kernela.

## Wyniki

| Wariant | Czas [ms] | Przepustowość [GB/s] |
|---|---|---|
| Global memory | 14.36 | 4.67 |
| Shared memory | **1.59** | **42.20** |

Tabela 1: Redukcja: pamięć globalna vs. współdzielona

**Komentarz.** Przeniesienie danych do *shared–memory* zmniejsza ruch w global–memory, co przyspiesza kernel ok. 9 razy.

# 2 Warp divergence (Zadanie 2)

## Warianty

Sekwencyjne i przeplatane indeksowanie wątków w redukcji.

## Wyniki

| Kernel | Czas [ms] |
|---|---|
| Sequential | **2.48** |
| Interleaved | 3.01 |

Tabela 2: Wpływ kolejności wątków

**Komentarz.** Dla testowego rozmiaru dane w bankach pamięci są sąsiadujące; wariant sequential ma mniej konfliktów banków.

# 3 Loop unrolling (Zadanie 3)

## Testowane konfiguracje

- Kerneli: **CG** (co-operative groups) i **WP** (warp primitives).
- Każdy z opcją kompilatora `#pragma unroll` ON/OFF.

### Wyniki

| Kernel | Unroll | Czas [ms] |
|---|---|---|
| CG | ON | 0.814 |
| CG | OFF | 0.814 |
| WP | ON | 0.808 |
| WP | OFF | **0.802** |

Tabela 3: Loop unrolling — niewielki wpływ na czas

**Komentarz.** Przy redukcji ograniczanej przepustowością pamięci różnice sterowania pętlą są marginalne.

## 4 Operacje atomowe (Zadanie 4)

### Strategie

1. **Simple** — pełne atomiki w global–mem.
2. **Block atomic** — jeden `atomicAdd` na blok.
3. **Warp atomic** — redukcja warp → atomik blokowy.

### Wyniki

| Wariant | Czas [ms] |
|---|---|
| Simple | 35.23 |
| Block atomic | **0.806** |
| Warp atomic | **0.806** |

Tabela 4: Redukcja liczby globalnych atomików

**Komentarz.** Już jeden atomik na blok ( 100× mniej operacji) daje ponad 40-krotne przyspieszenie; dodatkowa redukcja warpowa nie zmienia wyniku w tej skali.

## 5 Histogram (Zadanie 5)

### Parametry testu

Jedna seria danych (65 536 elementów) i dwa rozmiary histogramu $B$.

### Wyniki kernela `naive_histo`

| $B$ | shared [$\mu$s] | atomic [$\mu$s] | Szybszy |
|---|---|---|---|
| 16 | 11.17 | 10.85 | atomic |
| 1024 | 10.62 | 11.49 | shared |

Tabela 5: Czasy bez kopiowania HD

**Komentarz.** Dla małego $B$ kontencja atomików jest minimalna — wariant atomowy wypada odrobinę lepiej. Przy $B = 1024$ licznik w global–mem staje się gorącym punktem, więc wersja z pamięcią współdzieloną zyskuje 8 %.

# Podsumowanie

Największy wpływ na wydajność ma redukcja transakcji global–memory (shared-mem, eliminacja atomików). Optymalizacje sterujące (unrolling, układ wątków) dają zyski dopiero gdy wąskie gardło pamięciowe jest usunięte.