CUDA — Laboratorium 01

Sprawozdanie

Mateusz Łopaciński

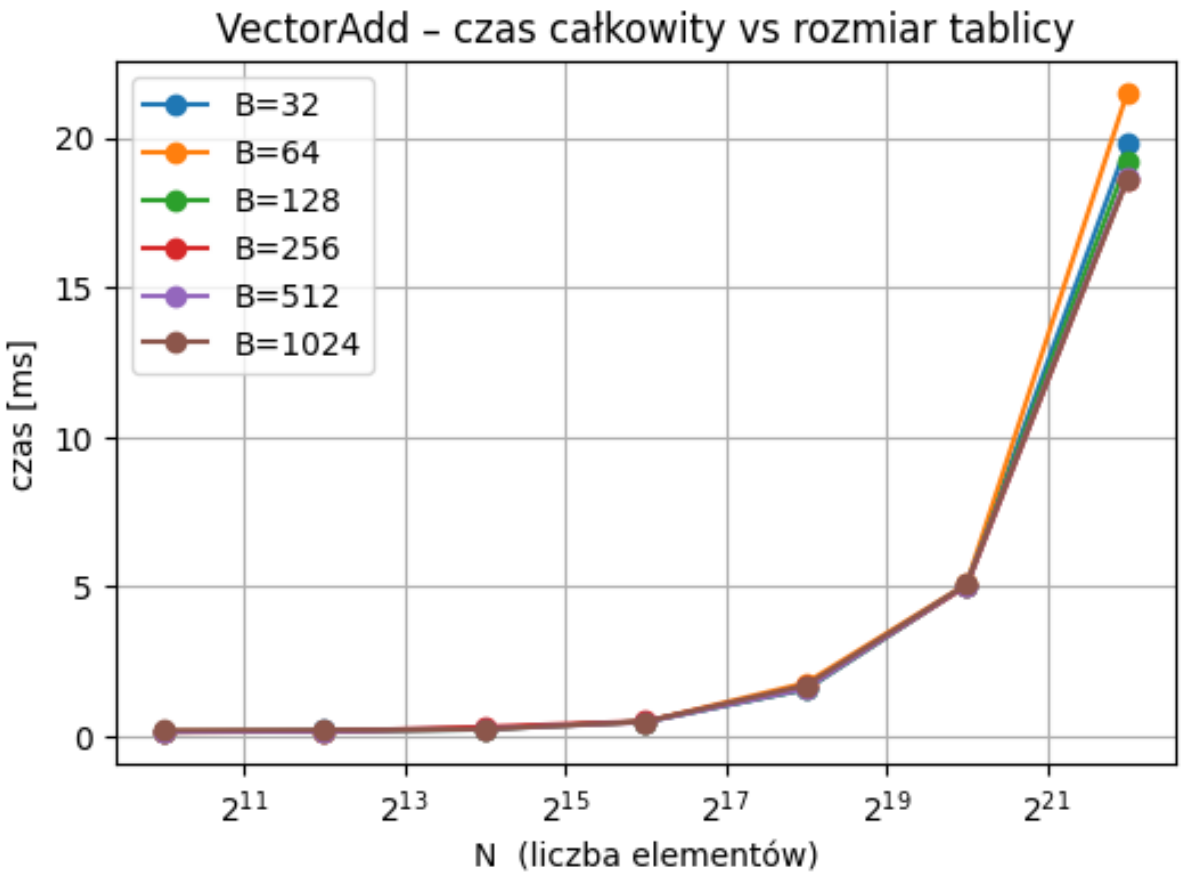
22 czerwca 2025

Spis treści

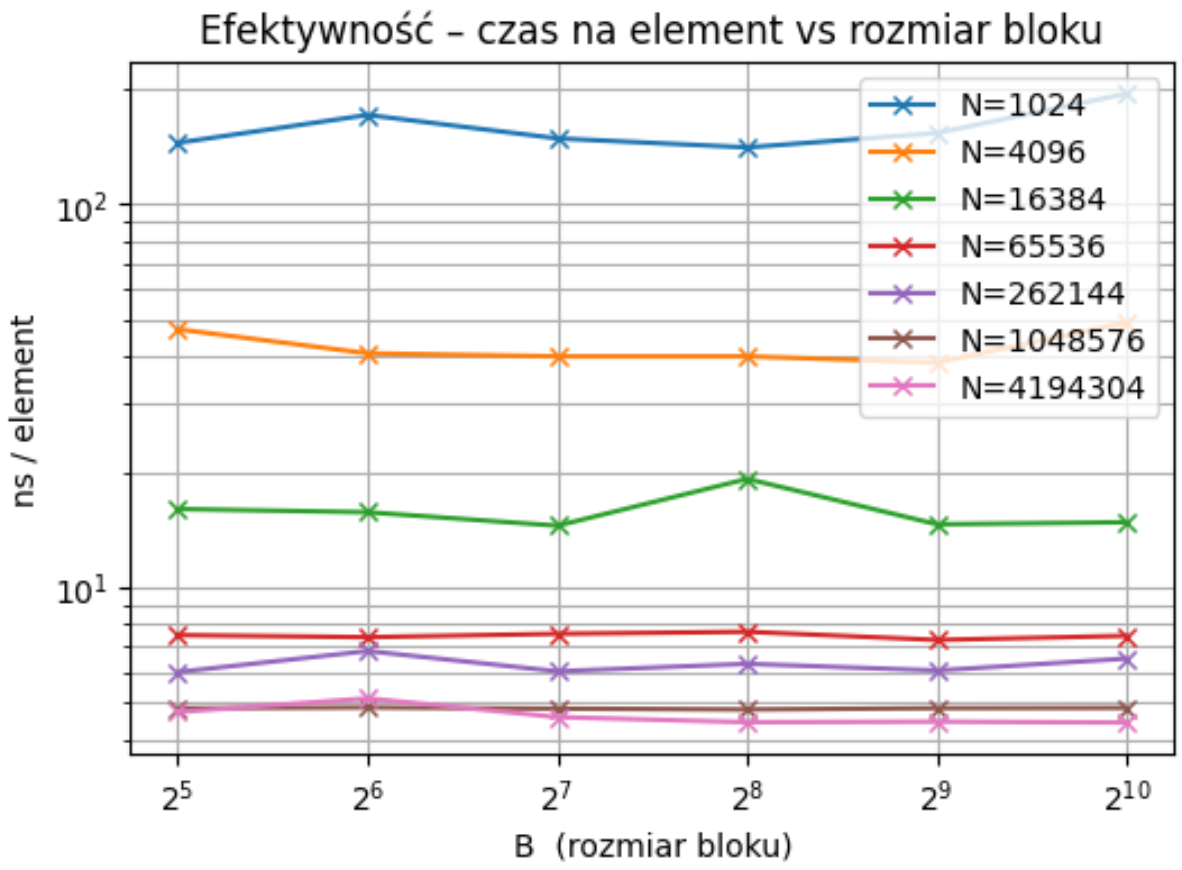
1 Środowisko testowe

- **GPU:** Nvidia Tesla T4 (16 GB, arch. SM_75) — Google Colab
- **Toolkit:** CUDA 12.5 (`nvcc V12.5.82`)
- Pomiary czasu: `cudaEventElapsedTime`

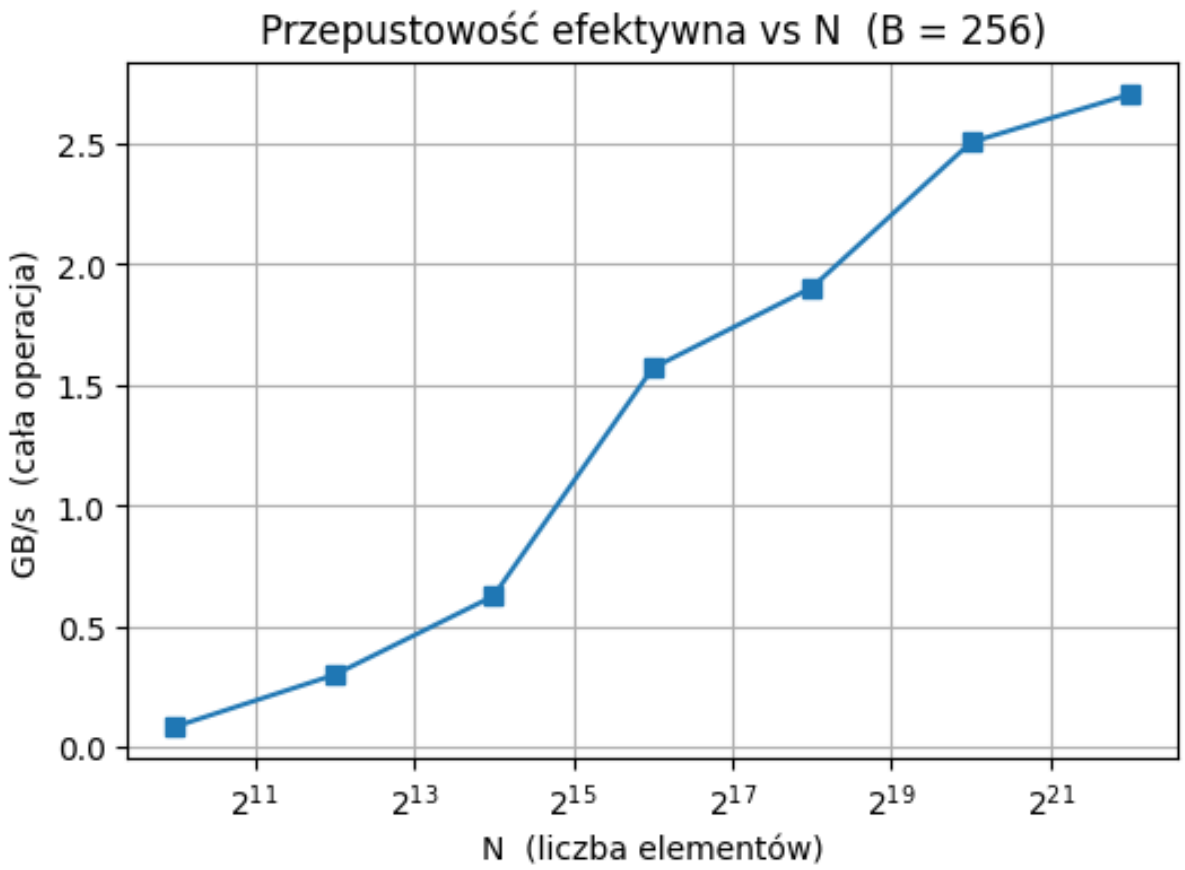
2 Ćwiczenie 1 — Dodawanie wektorów



Rysunek 1: Efektywność — czas na element w zależności od rozmiaru bloku.



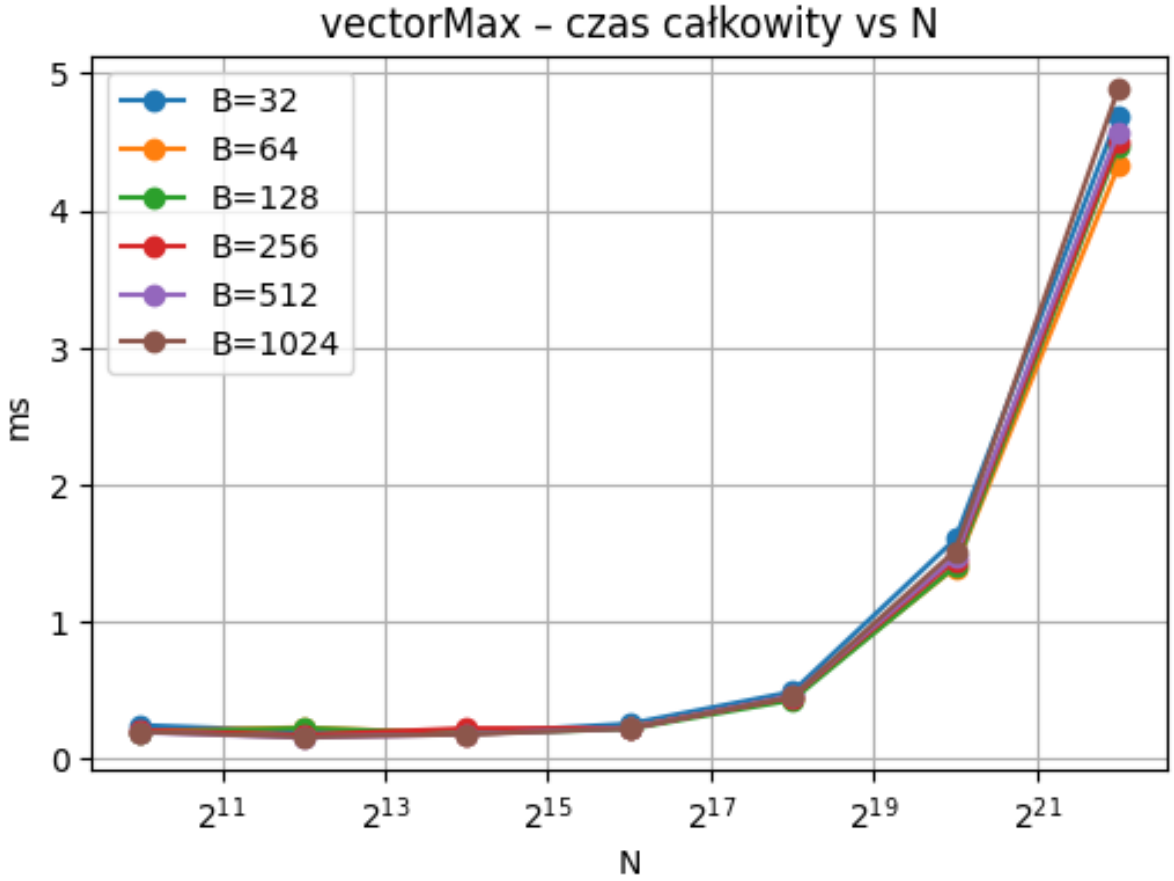
Rysunek 2: Czas całkowity operacji w funkcji rozmiaru tablicy.



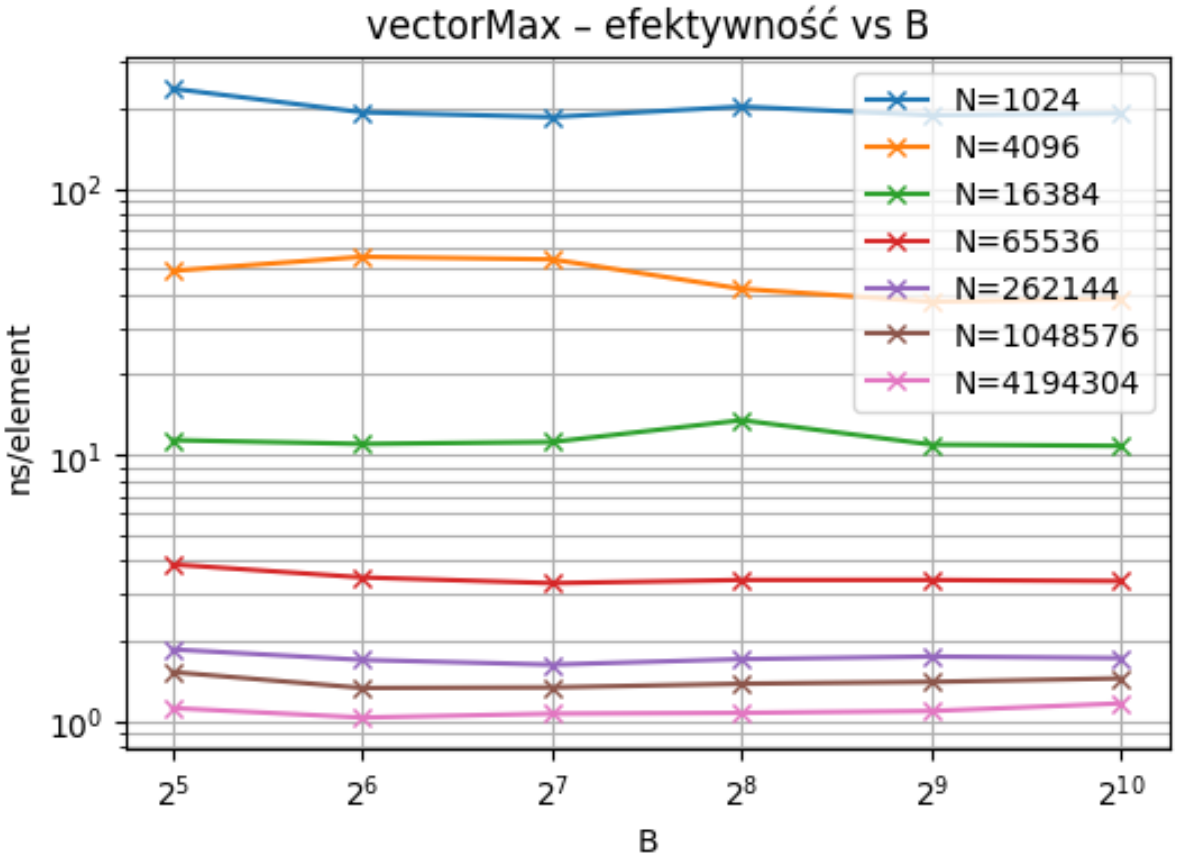
Rysunek 3: Przepustowość efektywna pamięci dla $B = 256$.

Dla dużych tablic ($N \geq 2^{19}$) uzyskano przepustowość 2–2.6 GB/s, co odpowiada ograniczeniu PCIe w Colabie. Optymalny rozmiar bloku: 128–256 wątków.

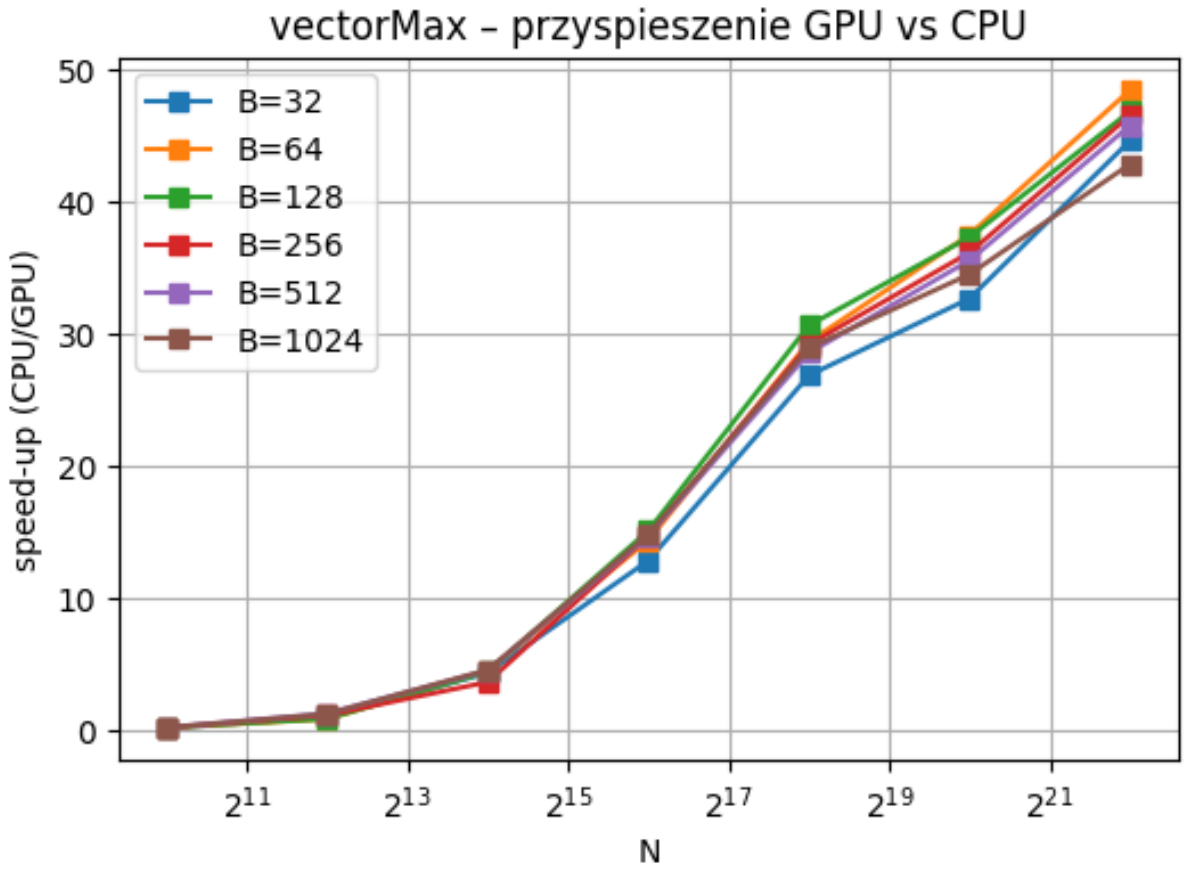
3 Ćwiczenie 2 — Maksimum w wektorze



Rysunek 4: Czas całkowity (GPU) w funkcji rozmiaru wektora.



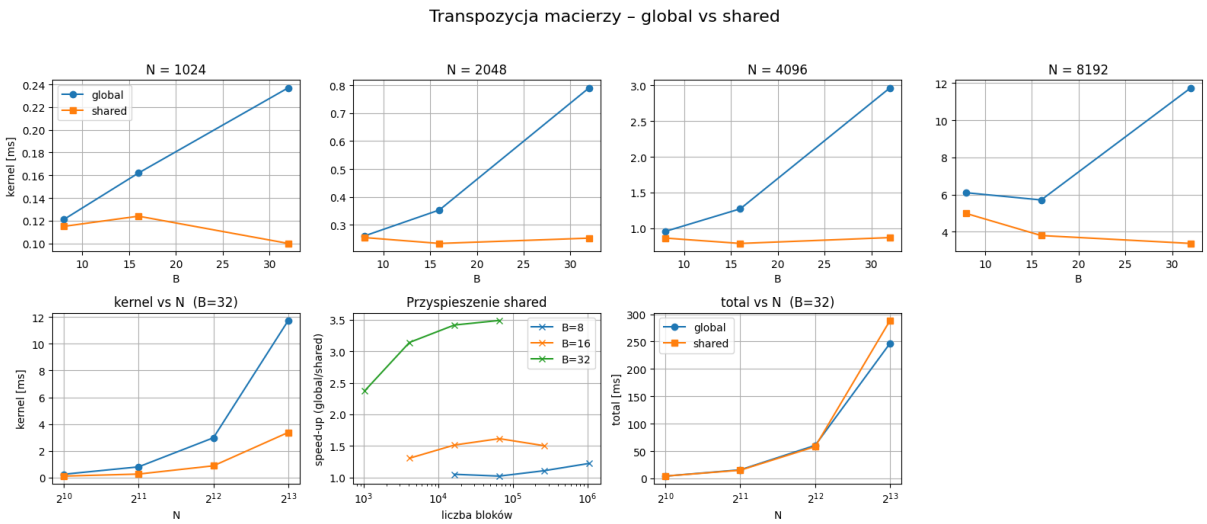
Rysunek 5: Efektywność — czas na element vs rozmiar bloku.



Rysunek 6: Przyspieszenie GPU względem jednowątkowego CPU.

Przy $N \approx 2^{21}$ zanotowano przyspieszenie rzędu 45–50 $\times$. Kernel redukcji najwydajniejszy dla bloków 128–256 wątków.

4 Ćwiczenie 3 — Transpozycja macierzy



Rysunek 7: Porównanie kernela *global-only* i *shared-tile*.

Z tile'em w shared-mem kernel przyśpiesza 2–3 ×; w czasie całkowitym (z kopiami HD) zysk wynosi ok. 15 %.

5 Wnioski końcowe

- Zadania transfer-bound (ćw. 1) ogranicza przepustowość magistrali.
- Redukcje (ćw. 2) zyskują na GPU dopiero dla dużych N.
- W transpozycji (ćw. 3) kafelek w shared-mem redukuje nie-coalesced odczyty, co przynosi ponad 2 × przyspieszenie kernela.