Optymalizacja Kodu Na Różne Architektury
HOW TO OPTIMIZE GEMM
raport

Iwo Szczepaniak

14 Kwietnia 2024

1 Procesor użyty do wykonania zadania

1.1 Parametry procesora

Platform: Desktop Laptop	**Product Family:** AMD Ryzen™ Processors	**Product Line:** AMD Ryzen™ 7 Mobile Processors with Radeon™ Graphics
# of CPU Cores: 8	**# of Threads:** 16	**Max. Boost Clock:** Up to 4.2GHz
Base Clock: 2.9GHz	**L2 Cache:** 4MB	**L3 Cache:** 8MB
Default TDP: 45W	**AMD Configurable TDP (cTDP):** 35-54W	**Processor Technology for CPU Cores:** TSMC 7nm FinFET
CPU Socket: FP6	**Max. Operating Temperature (Tjmax):** 105°C	**Launch Date:** 1/6/2020

Rysunek 1: Parametry procesora

1.2 Liczba GFLOPS

Informacje o procesorze znajdują się w pliku /proc/cpuinfo oraz w specyfikacji procesora [1]. Dodatkowym źródłem informacji o naszym procesorze mogą być również inne materiały - na przykład artykuł z wikipedii pozwalający ustalić liczbę FLOPs [2].

W przypadku CPU wykorzystanego do obliczeń, istotne parametry to:
FLOPs_per_cycle = 8
GHz_clock_rate = 4,2

Tak więc teoretyczne GFLOPS to:

$$GFLOPS = FLOPs\_per\_cycle * n\_processors * GHz\_clock\_rate =$$

$$= 8 * 1 * 4,2 = 33,6$$

W celu optymalizacji zastosowano flagi: -O2 -msse3 -march=znver2 [3].

2 Opis poszczególnych optymalizacji

- MMult1: Dodano makro X oraz funkcję AddDot.
- MMult2: Zmiana kroku dla zmiennej j na co 4.

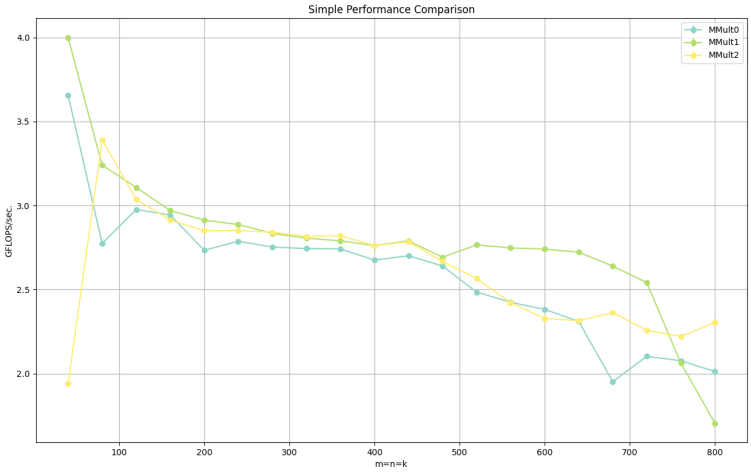
MMult_1x4 oznacza liczenie bloków po 4 elementy na raz używając funkcji AddDot1x4. MMult_4x4 oznacza obliczanie bloków 4x4 - 16 elementowych na raz używając funkcji AddDot4x4.

- MMult_Nx4_3: Przeniesienie wielu wywołań funkcji AddDot do funkcji AddDotNx4.

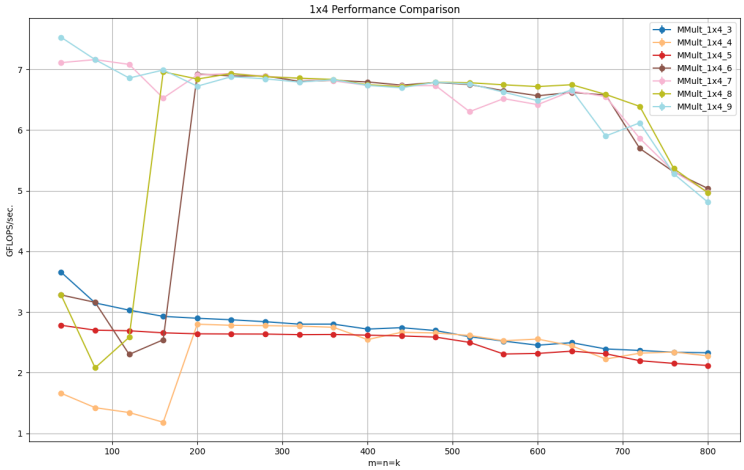
- MMult_Nx4_4: Usunięcie funkcji AddDot, przeniesienie jej kodu do wnętrza AddDotNx4.
- MMult_Nx4_5: Zredukowanie 4 pętli do jednej poprzez połączenie optymalizacji z poprzedniej wersji w jedną pętlę.
- MMult_Nx4_6: Użycie rejestrów dla zmiennych A i C.
- MMult_Nx4_7: Zastosowanie wskaźników do zmiennej B dla efektywniejszego dostępu.
- MMult_Nx4_8: Zmiana kroku pętli na 4 oraz wykorzystanie rejestrów dla zmiennej B.
- MMult_Nx4_9: Zmiana kroku dla aktualizacji wskaźników B na 4
- MMult_4x4_10: Wykorzystanie wektorów __m128d i operacji wektorowych.
- MMult_4x4_11: Podział algorytmu na bloki oraz dodanie funkcji InnerKernel.
- MMult_4x4_12: Stworzenie funkcji PackMatrixA w celu obserwacji nieefektywnego pakowania danych.
- MMult_4x4_13: Uproszczenie pakowania danych poprzez zmniejszenie ilości wywołań PackMatrixA.
- MMult_4x4_14: Dodanie funkcji PackMatrixB oraz zmiana kroku na 4 w PackMatrixA.
- MMult_4x4_15: Uproszczenie pakowania danych poprzez zmniejszenie ilości wywołań PackMatrixB.

Dla N = {1,4}

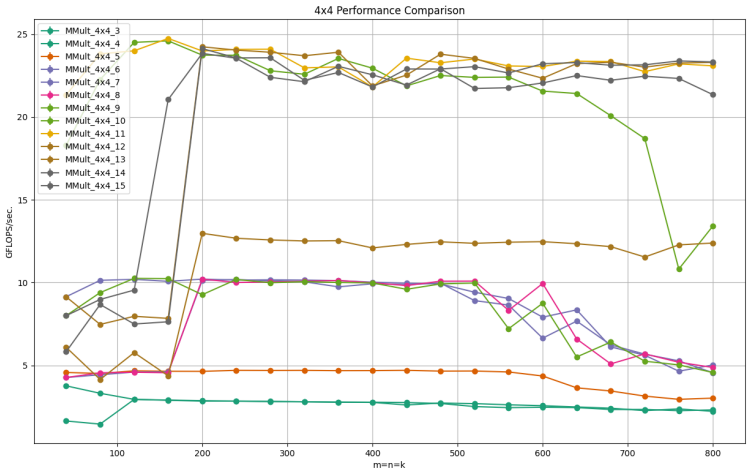
3 Wyniki optymalizacji



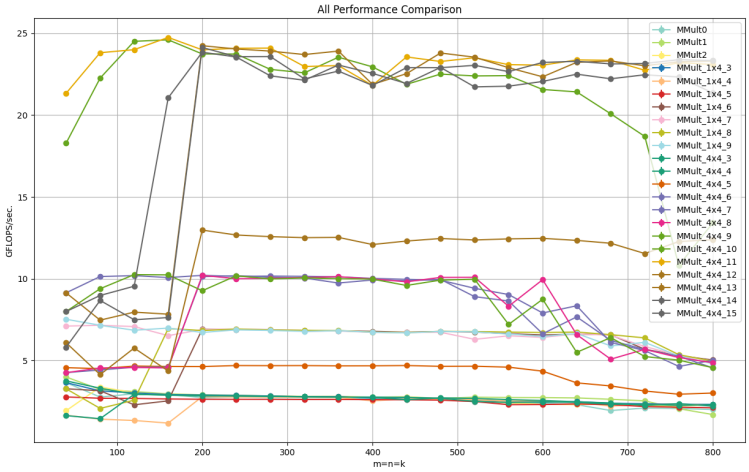
Rysunek 2: Wykres porównujący MMult0(bez optymalizacji) z MMult1 i MMult2



Rysunek 3: Wykres porównujący wyniki MMult_1x4



Rysunek 4: Wykres porównujący wyniki MMult_4x4



Rysunek 5: Wykres zbiorczy porównujący wyniki wszystkich optymalizacji

4 Podsumowanie

Pierwsze dwie optymalizacje(MMult1, MMult2) okazały się mało efektywne - nie przyniosły one znaczącej różnicy w ilości FLOPS. Podobnie pierwsze 3 z obu kategorii MMult_1x4_n i MMult_4x4_n(dla n od 3 do 5) - wszystkie zwracają wyniki między 3 a 4, jednak optymalizacja MMult4x4_5 nieco wyłamuje się z tego trendu dając wyniki zbliżone do 5. Następne 3 optymalizacje(od 6 do 9) przynoszą dla obu kategorii ponownie około dwukrotną poprawę wyników(tutaj zaczyna się większy rozdźwięk między 1x4 a 4x4, z racji że dwukrotna poprawa w pierwszym przypadku to około 7 FLOPS, a w drugim już około 10). Ostatnie 6 optymalizacji - zastosowanych tylko dla MMult_4x4(od 10 do 15) - przynoszą następną około dwukrotną poprawę ilości FLOPS. Najlepszy wynik FLOPS wynoszący 24.74924 jest nieznacznie niższy od przewidywanego teoretycznie 33,2. To znaczy, że uzyskaliśmy 74,5% wydajności trybu turbo procesora. Co zaskakujące, najlepsze efekty dla optymalizacji widać między rozmiarami 200 a 700 - dla mniejszych i większych rozmiarów tablic uzyskana korzyść znacznie spada dla większości optymalizacji.

Literatura

[1] Specyfikacja procesora AMD Ryzen 7 4800H, `https://www.amd.com/en/support/apu/amd-ryzen-processors/amd-ryzen-7-mobile-processors-radeon-graphics/amd-ryzen-7-4800h`

[2] Artykuł na Wikipedii o FLOPS, `https://en.wikipedia.org/wiki/FLOPS#Floating-point_operations_per_clock_cycle_for_various_processors`

[3] Strona Gentoo Wiki o procesorach AMD, `https://wiki.gentoo.org/wiki/Ryzen`