Projekt zaliczeniowy

Raport końcowy

Zespół Projektowy nr 1
Adrian Rybaczuk 318483
Bartek Cylwik 325457

12 czerwca 2025

Streszczenie

Treść z isoda - temat 2:
Implementacja (wraz z GUI) równoległego / rozproszonego mechanizmu wyliczania indeksów wilgotności (NDMI) i wegetacji (NDVI) roślin na zobrazowaniach satelitarnych (produktach z systemu Copernicus; Sentinel-2) do stwierdzania występowania lub zaniku roślinności (ich kondycji) na wskazanym obszarze.

Spis treści

1 Wprowadzenie **3**

2 Kroki podjęte na początku pracy **4**
2.1 Następnym krokiem będzie uzyskanie dostępu do danych z Sentinel-2 . . . 4
2.2 Research wskaźników NDVI i NDMI 4
2.3 RED, NIR, SWIR czyli B4, B8, B11 4
2.4 Zrównoleglenie / rozproszenie obliczeń 5
2.5 Wybór technologii 5

3 Przebieg procesu w aplikacji **6**

4 Równoległe przetwarzanie **6**
4.1 Które obliczenia i dlaczego postanowiliśmy zrównoleglić 6
4.2 Jak dokonaliśmy zrównoleglenia obliczeń 6

5 Platforma Testowa **7**

6 Jakie wyniki uzyskaliśmy **7**

7 Interpretacja i wnioski **11**
7.1 Wniosek 1: Zoptymalizowany CPU jest najszybszy dla małych i średnich zadań 11
7.2 Wniosek 2: Przetwarzanie równoległe na CPU ponosi znaczący koszt narzutu 11
7.3 Wniosek 3: GPU staje się opłacalne dopiero przy odpowiednio dużym rozmiarze problemu 11

7.4 Wniosek 4: Analiza narzutu GPU i zużycia pamięci 11
7.5 Wniosek ogólny: Wybór metody zależy od kontekstu 11

8 Jak uruchomić program 12
8.1 Uruchamianie skryptów testowych (opcjonalne) 13

Bibliografia 13

1 Wprowadzenie

Celem projektu była implementacja systemu równoległego lub rozproszonego mechanizmu wyliczenia indeksów NDMI i NDVI, czyli wiglotnośći i wilgotności na świecie. Dane jak mówi treść zadania pozyskaliśmy z zobrazowań satelitarnych Sentinel-2. Stworznie porjektu, rozdzieliliśmy na poniższe aspekty:

- Wybranie technologi do realizacji projektu Zdecydowaliśmy na pythona, gdyż sprawdziliśmy, że ma bibloteke do sentinela, no i inne przydatne w wyodrębnianiu/ rekonstrukcji danych.
- Uzyskanie wskaźników NDVI i NDMI
  - Wzory oraz wiedza ogólna jak są wykorzystywane
  - Pobranie niezbędnych wskaźników z zobrazowania satelitarnego Sentinel-2
  - Zapisanie danych pobranych z API w cashu, żeby uniknąć ponownego pobierania danych
  - Identyfikacja potęcjalnych problemów związanych z przetwarzaniem danych
  - Przeprowadzenie obliczeń wskaźników przy sekcji równolełej programu
- Interfejs graficzny
  - Uwierzytelniania z API Sentinelhuba
  - Głowny ekran, w którym możemy wybrac jako parametry okres danych do pobrania, zakres współrzednych jak i sposób wyliczenia współczynników. Rónież i tu pojawią sie obok obszaru dane pokrycie na mapie współczynnikami dla wybranych pikseli obrazu
  - Sekcja wyników, gdzie przeprowadzamy na zgromadzonych danych testy wydajnościowe, w celu porównania obliczeń między cpu a gpu, oraz wpływ ilości wątków.
- Równoległe przetwarzanie
  - Zastosowanie równoleóosci jedynie w obrębie samego liczenia współćzynników, są to tysiące-miliony danych, a GPU jest wydajny gównie w takich przypadkach.
  - Wspomniany wybór pomiedzy przetwarzaniem na CPU a GPU

2 Kroki podjęte na początku pracy

2.1 Następnym krokiem będzie uzyskanie dostępu do danych z Sentinel-2

Do tego celu skorzystaliśmy z API Sentinel-Hub. [1] Udostępnia on dane z satelity poprzez API dopiero po uwierzytelnieniu. W tym celu musielismy przejść przez process opisany w tym odnośniku [2]. Po jego przejsciu mamy dostep do Client ID oraz Client Secret które wykorzystamy do połączenia się z API.

API posiada reate limiting. [3] Z tego też powodu zamierzamy wykorzystać cache do przechowywania pobranych już danych. Z natury projektu i dania sobie możliwości testowania wyników zamierzamy przechowywać nieprzetworzone dane zamiast obliczonych już wyników. Pomoże nam to przeprowadzić testy i wprowadzić poprawki w przyszłości.

2.2 Research wskaźników NDVI i NDMI

NDVI (Normalized Difference Vegetation Index) [4] jest prostym wskaźnikiem ilościowym. Służy on do klasyfikacji wegetacji roślin. Jego wartość mieści się w przedziale od -1 do 1: czyli od mocnego występowania wody do bujnej flory

Wzór NDVI :

$$\text{NDVI} = \text{Index}(\text{NIR}, \text{RED}) = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}}$$

W przypadku danych Sentinel-2, indeks ten wyznaczamy na podstawie kanałów B8 (NIR) i B4 (RED):

$$\text{NDVI} = \text{Index}(\text{B8}, \text{B4}) = \frac{\text{B8} - \text{B4}}{\text{B8} + \text{B4}}$$

NDMI (Normalized Difference Moisture Index) [5] -znormalizowany wskaźnik wilgotności, który do wyznaczenia wilgotności wykorzystuje pasma NIR i SWIR. W tym samym przedziale co NDVI, wskazuje od suchych obszarów do mocno wilgotnych.

Wskaźnik NDMI definiujemy wzorem:

$$\text{NDMI} = \text{Index}(\text{NIR}, \text{SWIR}) = \frac{\text{NIR} - \text{SWIR}}{\text{NIR} + \text{SWIR}}$$

W przypadku danych Sentinel-2, indeks ten wyznaczamy na podstawie kanałów B8 (NIR) i B11 (SWIR):

$$\text{NDMI} = \text{Index}(\text{B8}, \text{B11}) = \frac{\text{B8} - \text{B11}}{\text{B8} + \text{B11}}$$

2.3 RED, NIR, SWIR czyli B4, B8, B11

RED czerwony kanał światła, silnie odbijany przez martwe liście. Wykorzystywany do identyfikacji typów roślinności, gleb, obszarów zabudowanych itd.

Dla Sentinel-2 to B4: [6]

- Rozdzielczość: 10 m/px
- Centralna długość fali: 665 nm
- Szerokość pasma: 30 nm

NIR (Near Infrared) to bliska podczerwień, która dobrze obrazuje linie brzegowe oraz zawartość biomasy.

W Sentinel-2 kanał NIR odpowiada pasmu B8: [7]

- Rozdzielczość: 10 m/px
- Centralna długość fali: 842 nm
- Szerokość pasma: 115 nm

SWIR (Short-Wave Infrared) jest to fala wysokowrażliwa na zawartość wody w obiektach. Dlatego jest dobrym wskaźnikiem wilgotności.

W Sentinel-2 kanał SWIR odpowiada pasmu B11: [8]

- Rozdzielczość: 20 m/px
- Centralna długość fali: 1610 nm
- Szerokość pasma: 130 nm

Poczatkowo zakladalismy koniecznosc recznego przeskalowania pasm Sentinel-2 o roznych rozdzielczościach, aby je ujednolicić. Jednak API Sentinel Hub automatycznie dostarcza je w najwyzszej wspolnej rozdzielczosci (10 m/px), co uproscilo przetwarzanie danych.

2.4 Zrównoleglenie / rozproszenie obliczeń

W związku z tym, że chcieliśmy, by aplikacja działała na jednym komputerze, wybraliśmy ścieżkę zrównoleglenia obliczeń. Zadanie obliczania wskaźników polega na wykonaniu prostych operacji arytmetycznych dla milionów pikseli.

Zakładaliśmy, że obliczenia na GPU, ze względu na ich masowo równoległą architekturę, będą najwydajniejszym rozwiązaniem. Aby się o tym przekonać, dodajemy do naszej aplikacji testy, przedstawiające w outpucie analizę związaną z różnymi strategiani obliczeniowymi:

1. **Przetwarzanie na GPU** z wykorzystaniem języka Taichi, aby zbadać potencjał masowego zrównoleglenia.
2. **Równoległe przetwarzanie na CPU** z wykorzystaniem modułu `multiprocessing` i pamięci dzielonej, aby ocenić skalowalność na wielordzeniowych procesorach.
3. **Jednowątkowe przetwarzanie na CPU** z wykorzystaniem wysoce zoptymalizowanej biblioteki NumPy, służące jako nasz główny punkt odniesienia.

2.5 Wybór technologii

Jako że określiliśmy już dokładniej ramy projektu możemy potwierdzić wybrane przez nas technologie. Ogólny język implementacji to Python. Do pobierania danych z api wykorzystaliśmy bibliotekę SentinelHub. Która udosptępnia już gotowe funkcje pozwalające zautoryzować się z api oraz pobrać dane. Do gui wykorzystaliśmy Tkinter podstawową i dosyć prostą bibliotekę do tworzenia interfejsu graficznego. Do prezentacji mapy wykorzystaliśmy TkinterMapView. Do zrównoleglenia obliczeń skorzystamy z Taichi Lang [9] którego zaembedujemy w pythonie. Jest to język programowania osadzony w pythonnie stworzony do wysokowydajnych obliczeń. Jego zaletą jest to że kod napisany raz przekąpiluje się na wiele różnych platform.

3 Przebieg procesu w aplikacji

Aplikacja została zaprojektowana z myślą o prostocie i intuicyjności obsługi. Interakcja użytkownika z systemem przebiega w kilku logicznych krokach:

1. **Uwierzytelnianie:** CliendId i ClientSecret oraz własne hasło. Przy nastepnym logowaniu wystarczy hasło.
2. **Główny widok mapy:** Po pomyślnym zalogowaniu, użytkownik przechodzi do głównego interfejsu, który składa się z dwóch głównych paneli: interaktywnej mapy satelitarnej na któej może wybrać współrzędne oraz pole na okres pobieranych danych. Z boku jest panel na wyniki obliczeń indeksów.
3. **Wybór sposobu liczenia:**] W górnym panelu dostępne są kontrolki do wyboru zakresu dat oraz metody przetwarzania (GPU lub CPU).
4. **Pobieranie danych:** Jest przycisk "Fetch Data"gdzie aplikacja wysyła zapytanie do API Sentinel Hub o pasma B4, B8, B11 oraz maskę danych dla widocznego na ekranie obszaru. Pobrane dane są zapisywane w lokalnym folderze
5. **Obliczenia i wizualizacja:** Po pobraniu danych, aktywują się przyciski "Calculate NDVI"i "Calculate NDMI". Kliknięcie jednego z nich uruchamia obliczenia w osobnym wątku (aby nie blokować interfejsu), wykorzystując wybraną metodę. Po zakończeniu obliczeń, z prawje strony wyświetlana jest heatmapa wskaźnika wraz z legendą.
6. **Moduł testowy:** Z widoku mapy można przejść do dedykowanego panelu testowego, który pozwala na uruchomienie serii testów wydajnościowych i wygenerowanie wykresów porównujących zaimplementowane metody obliczeniowe.

4 Równoległe przetwarzanie

4.1 Które obliczenia i dlaczego postanowiliśmy zrównoleglić

Głównym kandydatem do zrównoleglenia w naszym projekcie były obliczenia wskaźników NDVI i NDMI. Decyzja ta wynikała z charakterystyki problemu:

- **Niezależność danych (Data Parallelism):** Wartość wskaźnika dla każdego piksela obrazu satelitarnego jest obliczana wyłącznie na podstawie wartości pasm dla tego samego piksela. Nie ma żadnych zależności między sąsiednimi pikselami.
- **Duża skala problemu:** Obrazy satelitarne mogą składać się z milionów lub więcej pikseli. Wykonywanie operacji na tak dużej liczbie danych w sposób sekwencyjny się mija z celem.
- **Powtarzalność operacji:** Ten sam, prosty wzór matematyczny $(a-b)/(a+b)$ jest stosowany dla każdego piksela.

4.2 Jak dokonaliśmy zrównoleglenia obliczeń

Aby przeprowadzić kompleksową analizę, zaimplementowaliśmy trzy różne strategie obliczeniowe:

Przetwarzanie na GPU z użyciem Taichi Lang

Wykorzystaliśmy Taichi jako kompilator JIT do równoległego przetwarzania danych na GPU, co pozwoliło nam błyskawicznie wykonywać obliczenia $\frac{a-b}{a+b}$ na tysiącach pikseli jednocześnie.

Równoległe przetwarzanie na CPU z użyciem 'multiprocessing'

Tutaj dzięki mechanizmowi pamięci dzielonej mogliśmy równolegle przetwarzać dane na wielu rdzeniach CPU bez kosztownego kopiowania.

Jednowątkowe przetwarzanie na CPU z użyciem NumPy

Stworzyliśmy także szybki, jednowątkowy baseline w NumPy, korzystający z natywnych, zoptymalizowanych funkcji w C i Fortranie.

5 Platforma Testowa

Testy wydajnościowe zostały przeprowadzone na jednolitej platformie sprzętowej dostepny widok z głownego dashbordu naszej aplikacji. Poniższa specyfikacja dokumentuje konfigurację komputera użytego do testów.

System Operacyjny: Windows 10 (AMD64)

Procesor (CPU): AMD Ryzen 9 3900X

Szczegóły CPU: 12 rdzeni fizycznych / 24 wątki, Taktowanie bazowe: 3.8 GHz, Cache L2: 6 MB, Cache L3: 64 MB

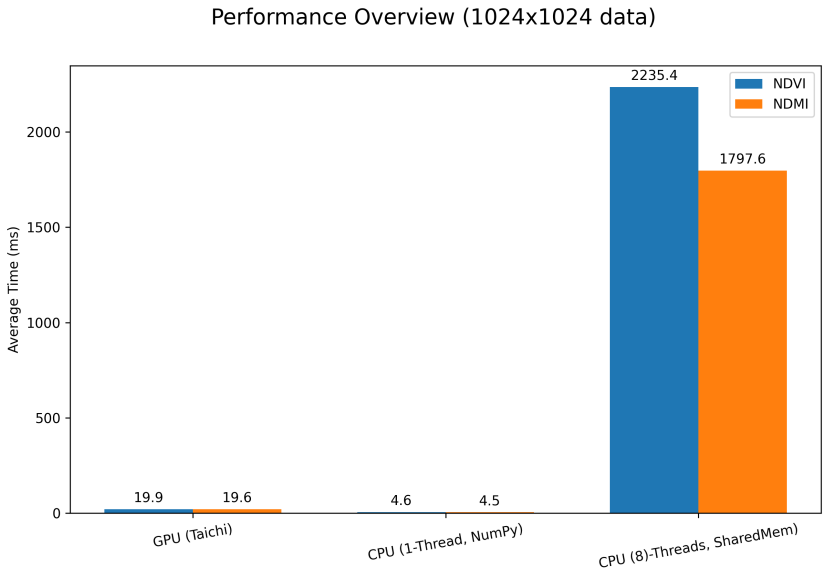
Pamięć RAM: 64 GB DDR4

Karta Graficzna (GPU): AMD Radeon RX 6800 XT

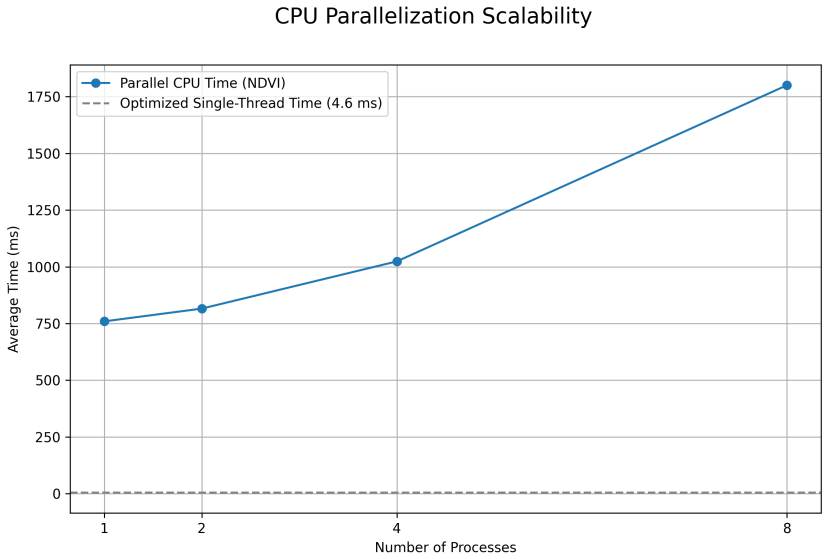
Uwaga: Na wydajność, zwłaszcza w przypadku obliczeń na GPU, mają również wpływ czynniki nieujęte w powyższej specyfikacji, takie jak wersja i przepustowość magistrali PCIe, szybkość pamięci VRAM oraz ogólne obciążenie systemu podczas testów.

6 Jakie wyniki uzyskaliśmy

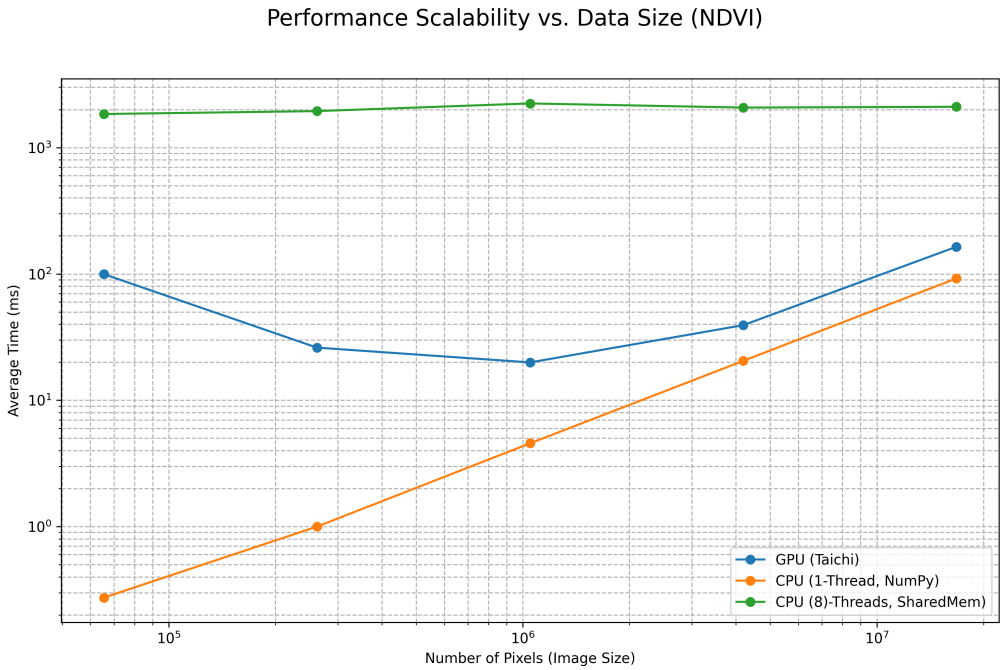
W celu oceny wydajności zaimplementowanych metod, przeprowadziliśmy serię testów, które polegały na mierzeniu czasu wykonania obliczeń dla danych o różnej rozdzielczości oraz analizie zużycia pamięci. Oto otrzymane wykresy:



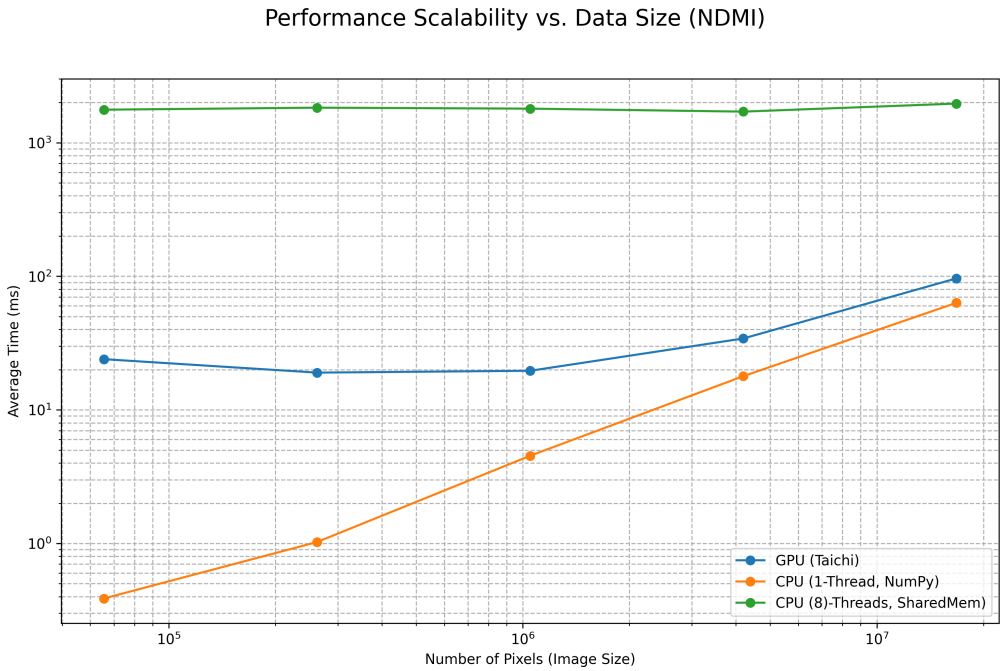
Rysunek 1: Przegląd wydajności dla danych 1024x1024.



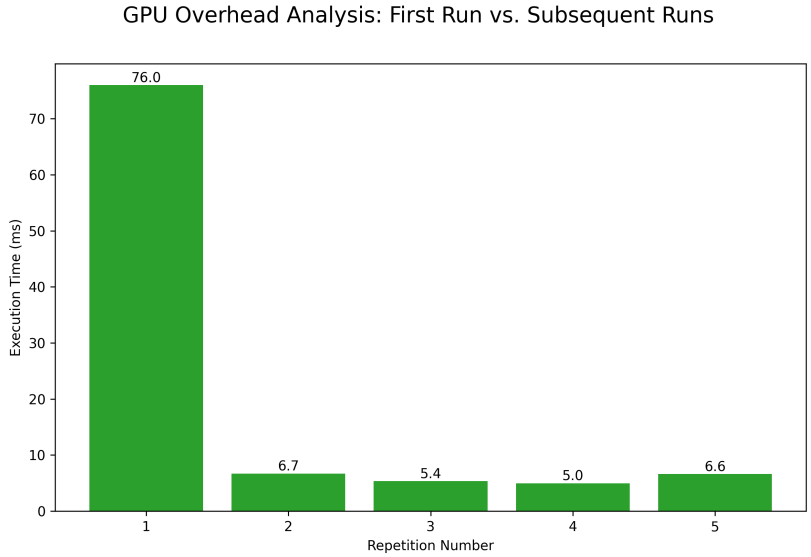
Rysunek 2: Analiza skalowalności implementacji równoległej na CPU.



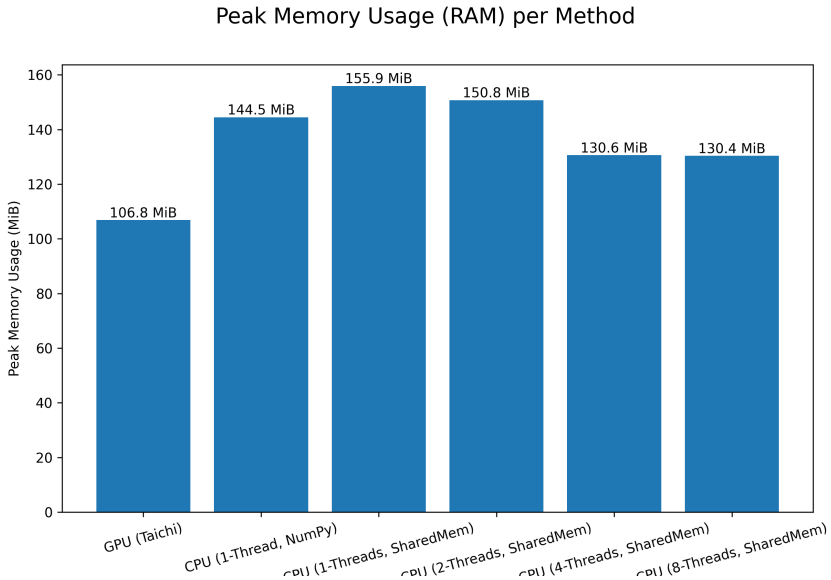
Rysunek 3: Skalowalność wydajności w zależności od rozmiaru danych (NDVI).



Rysunek 4: Skalowalność wydajności w zależności od rozmiaru danych (NDMI).



Rysunek 5: Analiza narzutu inicjalizacyjnego (warm-up) dla GPU.



Rysunek 6: Szczytowe zużycie pamięci RAM przez proces główny.

7 Interpretacja i wnioski

Przeprowadzone testy porpzez wykresy, pozwalają na sformułowanie kluczowych wniosków dotyczących wydajności i zasadności stosowania różnych technik przetwarzania równoległego dla naszego problemu.

7.1 Wniosek 1: Zoptymalizowany CPU jest najszybszy dla małych i średnich zadań

Jak widać na Rysunku 1, dla standardowego zadania (obraz 1024x1024), wysoce zoptymalizowana, jednowątkowa implementacja oparta na NumPy jest ewidentnie najszybsza. Jest tak bo obliczenia są na tyle proste, że koszt narzutu związany z przeniesieniem danych do GPU lub zarządzaniem pulą procesów na CPU jest znacznie większy niż zysk z samego zrównoleglenia.

7.2 Wniosek 2: Przetwarzanie równoległe na CPU ponosi znaczący koszt narzutu

Rysunek 2 doskonale ilustruje problem narzutu w ‘multiprocessing‘. Czas wykonania dla jednego procesu w puli jest drastycznie wyższy niż dla wersji jednowątkowej bez puli. Wynika to z konieczności tworzenia nowych procesów, konfiguracji pamięci dzielonej i komunikacji międzyprocesowej. Mimo że implementacja poprawnie skaluje się wraz z dodawaniem kolejnych procesów (czas maleje), nigdy nie jest w stanie zniwelować początkowego narzutu i pokonać wydajności czystego NumPy dla tego zadania.

7.3 Wniosek 3: GPU staje się opłacalne dopiero przy odpowiednio dużym rozmiarze problemu

Analiza skalowalności w zależności od rozmiaru danych (Rysunki 3 i 4) jest najważniejszym wynikiem projektu. Obserwujemy, że linia wydajności GPU - niebieska ma mniejsze nachylenie niż linia CPU - pomareńczowa, co oznacza, że lepiej radzi sobie z rosnącym obciążeniem. Widzimy wyraźny **punkt przełamania (break-even point)** w okolicach $2 \cdot 10^6$ pikseli. Poniżej tego progu, zoptymalizowany CPU jest szybszy. Powyżej tego progu, zysk z masowego zrównoleglenia na GPU zaczyna przewyższać stały koszt transferu danych, czyniąc GPU metodą wydajniejszą dla bardzo dużych zbiorów danych.

7.4 Wniosek 4: Analiza narzutu GPU i zużycia pamięci

Rysunek 5 potwierdza istnienie kosztu "rozgrzewki"GPU. Pierwsze uruchomienie jest znacznie wolniejsze z powodu kompilacji JIT. Rysunek 6 pokazuje z kolei kompromisy w zarządzaniu pamięcią. Metoda GPU odciąża systemową pamięć RAM, gdyz przenosi dane do VRAM.

7.5 Wniosek ogólny: Wybór metody zależy od kontekstu

Nie istnieje jedna, uniwersalnie najlepsza metoda. Wybór optymalnej strategii zależy od konkretnego przypadku użycia i konkretnych parametró wejściowych:

- **Interaktywna analiza małych obszarów:** Wyraźnie widać, że najlepszym wyborem jest zoptymalizowany, jednowątkowy CPU (NumPy) ze względu na minimalny czas latencji.
- **Przetwarzanie wsadowe bardzo dużych obrazów:** Dla zadań przekraczających "punkt przełamania", GPU oferuje najwyższą przepustowość i jest metodą preferowaną.
- **Efektywność energetyczna:** Dla zadań, gdzie CPU jest szybsze, jest ono również znacznie bardziej efektywne energetycznie.

Wyniki te prezentują, jak kluczowe jest zrozumienie charakterystyki problemu (złożoność obliczeniowa vs. obciążenie I/O) oraz narzutów platformy przy projektowaniu systemów o wysokiej wydajności.

8 Jak uruchomić program

Aby uruchomić aplikację, należy postępować zgodnie z poniższymi krokami:

1. Sklonować repozytorium projektu i przejść do jego głównego katalogu.
2. Utworzyć i aktywować środowisko wirtualne z użyciem Pythona 3.11:

   ```
   python3.11 -m venv venv
   source venv/bin/activate
   ```

   (Na Windows zamiast `source venv/bin/activate` należy użyć `venv\Scripts\activate`.)
3. Zainstalować wszystkie wymagane biblioteki za pomocą menedżera pakietów pip:

   ```
   pip install -r requirements.txt
   ```

4. Stworzyć plik `.env` w głównym katalogu projektu. Plik ten będzie zawierał **jedynie** sól kryptograficzną, która jest niezbędna do odszyfrowania danych logowania. Dla celów testowych i odtworzenia wyników, należy użyć poniższej wartości:

   ```
   SENTINEL_SALT="385
   ↪ d0b98fb81618a8d5165a2c25300e26217c22661ca9ada7cde877075184219
   ↪ "
   ```

5. Przygotować własne dane uwierzytelniające (Client ID oraz Client Secret) do API Sentinel Hub. Instrukcje, jak je uzyskać, znajdują się w oficjalnej dokumentacji: https://docs.sentinel-hub.com/api/latest/api/overview/authentication/.
6. Przy pierwszym uruchomieniu, aplikacja poprosi o podanie uzyskanych danych oraz o hasło, które posłuży do ich zaszyfrowania i bezpiecznego przechowania w pliku `sentinel_config.enc`. Dla celów ewaluacji projektu, w repozytorium znajduje się już gotowy plik `sentinel_config.enc`, do którego hasło zostanie udostępnione osobno.
7. Uruchomić główną aplikację za pomocą polecenia z **głównego katalogu projektu**:

   ```
   python -m src.main
   ```

   Uwaga: Użycie flagi `-m` jest kluczowe, ponieważ zapewnia, że Python poprawnie rozpoznaje strukturę pakietów w folderze `src`.

8.1 Uruchamianie skryptów testowych (opcjonalne)

Aby przeprowadzić testy wydajności i pamięci, należy najpierw wygenerować dane testowe, a następnie uruchomić skrypt profilujący. Oba skrypty należy uruchamiać z **głównego katalogu projektu**.

1. Generowanie danych testowych o różnej skali:

   ```
   python -m src.scripts.generate_scaled_data
   ```

2. Uruchomienie testów zużycia pamięci (wymaga `memory-profiler`):

   ```
   python -m memory_profiler src.scripts.run_memory_tests
   ```

Bibliografia

[1] Sentinel Hub. *Sentinel-2 API Documentation*. URL: https://docs.sentinel-hub.com/api/latest/data/sentinel-2-l2a/ (term. wiz. 11. 06. 2024).

[2] Sentinel Hub. *Sentinel-2 API Documentation Authentication*. URL: https://docs.sentinel-hub.com/api/latest/api/overview/authentication/ (term. wiz. 11. 06. 2024).

[3] Sentinel Hub. *Sentinel-2 API Documentation Rate Limiting*. URL: https://docs.sentinel-hub.com/api/latest/api/overview/rate-limiting/ (term. wiz. 11. 06. 2024).

[4] Sentinel Hub. *Normalized Difference Vegetation Index (NDVI)*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/ndvi/ (term. wiz. 11. 06. 2024).

[5] Sentinel Hub. *Normalized Difference Moisture Index (NDMI)*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/ndmi/ (term. wiz. 11. 06. 2024).

[6] Sentinel Hub. *Sentinel-2 Band Documentation B4*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/bands/#band-4 (term. wiz. 11. 06. 2024).

[7] Sentinel Hub. *Sentinel-2 Band Documentation B8*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/bands/#band-8 (term. wiz. 11. 06. 2024).

[8] Sentinel Hub. *Sentinel-2 Band Documentation B11*. URL: https://custom-scripts.sentinel-hub.com/custom-scripts/sentinel-2/bands/#band-11 (term. wiz. 11. 06. 2024).

[9] Taichi Lang. *Taichi Lang Documentation*. URL: https://www.taichi-lang.org/ (term. wiz. 11. 06. 2024).