Wrocław
University
of Science
and Technology

Przetwarzanie danych masowych

Wykład 3 – Podstawowe metody zrównoleglania algorytmów uczenia maszynowego. Przetwarzanie synchroniczne i asynchroniczne

dr hab. inż. Tomasz Kajdanowicz, Piotr Bielak, Roman Bartusiak

18.10.2021 r.

HR EXCELLENCE IN RESEARCH

Overview

Zrównoleglanie algorytmów uczenia maszynowego

Przetwarzanie synchroniczne a asynchroniczne

Zrównoleglanie obliczeń a Python

Plan wykładu

Zrównoleglanie algorytmów uczenia maszynowego

Przetwarzanie synchroniczne a asynchroniczne

Zrównoleglanie obliczeń a Python

Motywacja

Zrównoleglanie algorytmów uczenia maszynowego

- większe modele w głębokim uczeniu maszynowym = więcej parametrów
- stosowanie coraz większych zbiorów danych
- ograniczone zasoby sprzętowe
- **rozwiązanie:** zrównoleglenie procesu uczenia na wielu maszynach (akceleratorach)

Podstawowe metody

Zrównoleglanie algorytmów uczenia maszynowego

Wyróżniamy dwa podstawowe podejścia zrównoleglania obliczeń:

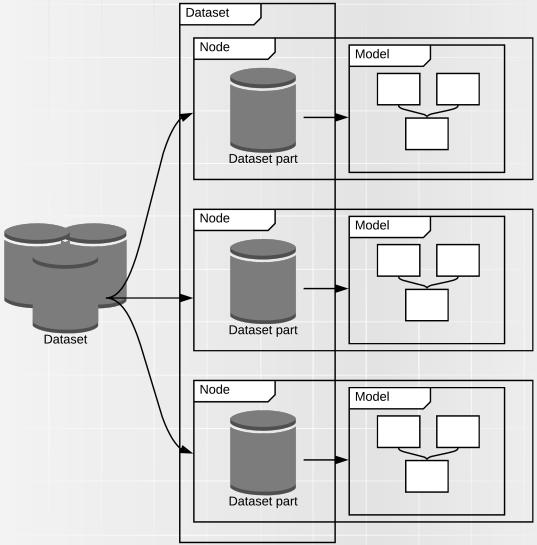
- ze względu na dane (ang. *data parallelism*),
- ze względu na model/zadanie (ang. *task parallelism*).

Zrównoleglanie ze względu na dane)

- każdy węzeł obliczeniowy posiada pełną kopię tego samego modelu
- dane są podzielone pomiędzy węzły
- powtarzane są dwa kroki:
  - uczenie modeli
  - synchronizacja parametrów kopii modelu

Zrównoleglanie ze względu na dane

Zrównoleglanie algorytmów uczenia maszynowego



Zrównoleglanie ze względu na dane – przykład

Na przykładzie uczenia metodą gradientu prostego (ang. *gradient descent*) zastanówmy się dlaczego można go zrównoleglać ze względu na dane – załóżmy, że:

- θ – parametry modelu,
- $\frac{\partial \text{Loss}}{\partial \theta}$ – gradient uzyskany przy uczeniu modelu za pomocą wszystkich n próbek,
- $\frac{\partial l_k}{\partial \theta}$ – gradient uzyskany na k-tym węźle obliczeniowym za pomocą m_k próbek ($m_1 + m_2 + \ldots + m_k = n$),
- (x_i, y_i) – i-ta próbka danych (wejście, etykieta),
- $f(x_i, y_i)$ – koszt dla i-tej próbki,
- k – liczba węzłów obliczeniowych,

Zrównoleglanie ze względu na dane – przykład

$$
\begin{aligned}
\frac{\partial \text{Loss}}{\partial \theta} &= \frac{\partial \left[\frac{1}{n}\sum_{i=1}^{n} f(x_i, y_i)\right]}{\partial \theta} \\
&= \frac{1}{n}\sum_{i=1}^{n} \frac{\partial f(x_i, y_i)}{\partial \theta} \\
&= \frac{m_1}{n}\frac{\partial \left[\frac{1}{m_1}\sum_{i=1}^{m_1} f(x_i, y_i)\right]}{\partial \theta} + \frac{m_2}{n}\frac{\partial \left[\frac{1}{m_2}\sum_{i=m_1+1}^{m_1+m_2} f(x_i, y_i)\right]}{\partial \theta} + \dots \\
&\dots + \frac{m_k}{n}\frac{\partial \left[\frac{1}{m_k}\sum_{i=m_{k-1}+1}^{m_{k-1}+m_k} f(x_i, y_i)\right]}{\partial \theta} \\
&= \frac{m_1}{n}\frac{\partial l_1}{\partial \theta} + \frac{m_2}{n}\frac{\partial l_2}{\partial \theta} + \dots + \frac{m_k}{n}\frac{\partial l_k}{\partial \theta}
\end{aligned}
$$

Zrównoleglanie ze względu na dane – przykład

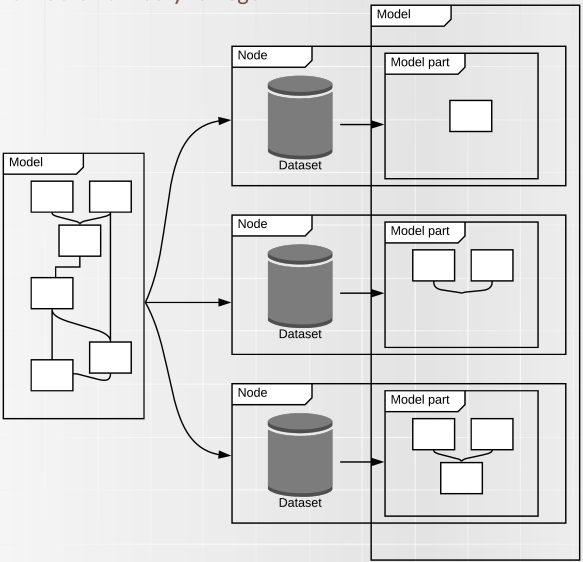
Na każdym węźle używany jest ten sam model, aby wykonać przejście w przód i dla danej paczki danych (ang. *batch*) obliczyć gradient. Informacje o uzyskanych gradientach są wysyłane do głównego węzła obliczeniowego. Obliczana jest (ważona) średnia gradientów, a wynikowy gradient jest używany do aktualizacji wag modelu (kopii modelu na każdym węźle).

Zrównoleglanie ze względu na model/zadanie

- Model jest dzielony i każda część jest przetwarzana na osobnym węźle obliczeniowym.
- Każdy węzeł posiada ten sam zbiór danych.

Zrównoleglanie ze względu na model/zadanie

Zrównoleglanie algorytmów uczenia maszynowego



Zrównoleglanie ze względu na model/zadanie – przykład

- Rozważmy ponownie przykładu z wykorzystaniem głębokiego modelu uczenia maszynowego – np. ResNet50, który posiada łącznie 50 warstw.
- Na obecnie dostępnych zasobach obliczeniowych ten model nie stanowi wyzwania, jednak załóżmy, że każdy nasz węzeł obliczeniowy posiada tylko tyle zasobów aby pomieścić 5 warstw tego modelu. Zakładamy, że posiadamy 10 węzłów, tzn. pierwszy węzeł będzie odpowiedzialny za pierwsze 5 warstw modelu, drugi węzeł za kolejne 5 warstw, itd. Dziesiąty węzeł będzie przetwarzał ostatnie 5 warstw.

Zrównoleglanie ze względu na model/zadanie – przykład

- W każdej iteracji procesu uczenia zaczynamy przejście w przód od pierwszego węzła, którego wyjścia są przesyłane do drugiego węzła itd.
- Funkcja kosztu jest obliczana na dziesiątym węźle. Następnie na podstawie obliczonych gradientów są aktualizowane wagi 5 ostatnich warstw sieci (znajdujących się na tym węźle). Gradienty są propagowane wstecz na dziewiąty węzeł, gdzie na ich podstawie aktualizowane są parametry kolejnych 5 warstw, itd.

Zrównoleglanie ze względu na model/zadanie – przykład

Innym przykładem mogą być modele sieci syjamskich, w których posiadamy dwa moduły sieci o identycznej architekturze. Obliczenia dla każdego modułu mogą być zrównoleglone na dwa węzły obliczeniowe.

Plan wykładu

Zrównoleglanie algorytmów uczenia maszynowego

Przetwarzanie synchroniczne a asynchroniczne

Zrównoleglanie obliczeń a Python

Podział I/O

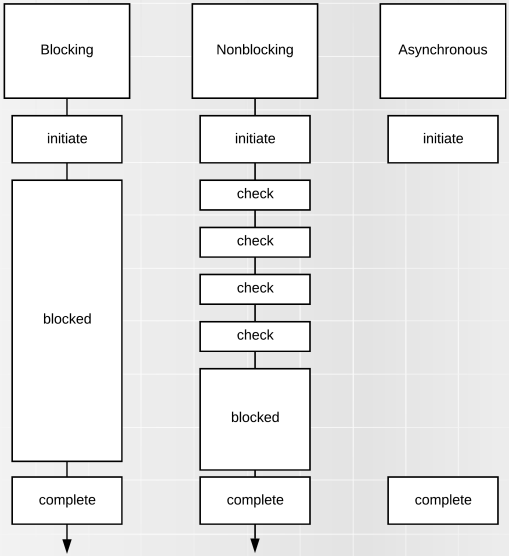
Przetwarzanie synchroniczne a asynchroniczne

Operacje wejścia/wyjścia:

- blokujące,
- nieblokujące,
- asynchroniczne.

Podział I/O

Przetwarzanie synchroniczne a asynchroniczne



Dodatkowa uwaga – POSIX I/O vs HPC

Przetwarzanie synchroniczne a asynchroniczne

- POSIX jest stanowy, system operacyjny śledzi na bieżąco wszystkie deskryptory plików
- POSIX wnosi sporo dodatkowych metadanych (potecjalnie niepotrzenych)
- POSIX pozwala na uzyskanie silnej spójności danych – po zapisie danych, można je odczytać

Dodatkowa uwaga – POSIX I/O vs HPC

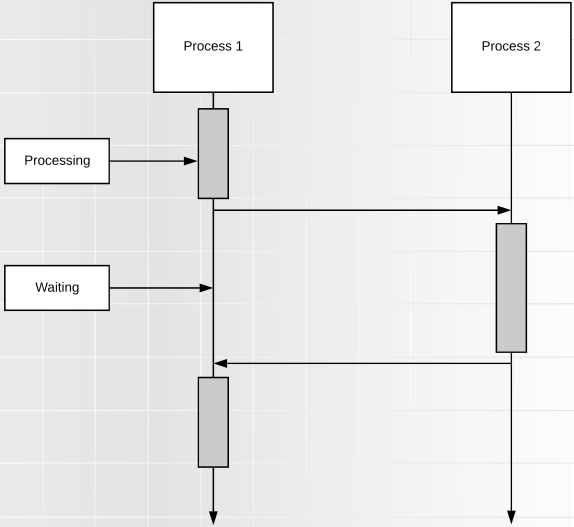
Przetwarzanie synchroniczne a asynchroniczne

- HPC = High Performance Computing
- programy HPC zapewniają, że dwa procesy nie będą pisać do tego samego pliku
- w HPC spójność jest ograniczona do mniejszego podzbioru węzłów (zamiast całego klastra obliczeniowego)
- dla zainteresowanych – `https://opensource.com/article/20/6/linux-noatime`

Przetwarzanie synchroniczne

Przetwarzanie synchroniczne a asynchroniczne

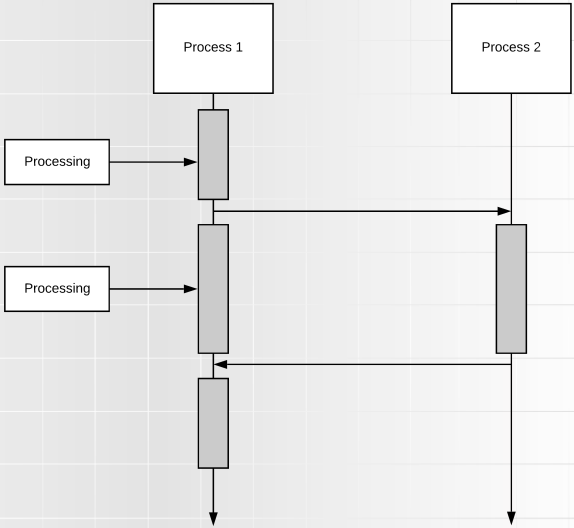
- wykonanie zapytania
- oczekiwanie na odpowiedź
- dalsze przetwarzanie



Przetwarzanie asynchroniczne

Przetwarzanie synchroniczne a asynchroniczne

- wykonanie zapytania
- dalsze przetwarzanie
- wykorzystaj wynik zapytania gdy się pojawi
- dalsze przetwarzanie



Plan wykładu

Zrównoleglanie algorytmów uczenia maszynowego

Przetwarzanie synchroniczne a asynchroniczne

Zrównoleglanie obliczeń a Python

Wprowadzenie

Zrównoleglanie obliczeń a Python

- Z perspektywy programisty dostępnych jest kilka sposobów zrównoleglania obliczeń – np. wątki oraz procesy.
- Zazwyczaj zaleca się używanie wątków.
- Wątki posiadają m.in. współdzielony obszar pamięci, a ich utworzenie jest mało kosztowne obliczeniowo.
- Procesy są w pełni od siebie odseparowane, natomiast utworzenie procesu jest bardziej kosztowe obliczeniowo niż utworzenie wątku (m.in. ze względu na kopiowanie obszaru pamięci).
- Czy tak jest również w Pythonie?

Przetwarzanie sekwencyjne

Zrównoleglanie obliczeń a Python

```python
import time


def countdown(n):
    while n > 0:
        n -= 1


def main():
    n = 50000000

    st = time.time()
    countdown(n)
    end = time.time()

    print('Processing took:', end - st, '(s)')


if __name__ == '__main__':
    main()
```

Czas wykonania: **3.09 [s]**

Przetwarzanie zrównoleglone za pomocą 2 wątków

Zrównoleglanie obliczeń a Python

```
1  from threading import Thread
2  import time
3
4
5  def countdown(n):
6      while n > 0:
7          n -= 1
8
9
10 def main():
11     n = 50000000
12
13     t1 = Thread(target=countdown, args=(n//2,))
14     t2 = Thread(target=countdown, args=(n//2,))
15
16     st = time.time()
17     t1.start(); t2.start()
18     t1.join(); t2.join()
19     end = time.time()
20
21     print('Processing took:', end - st, '(s)')
22
23
24 if __name__ == '__main__':
25     main()
```

Czas wykonania: **5.37 [s]**!

GIL

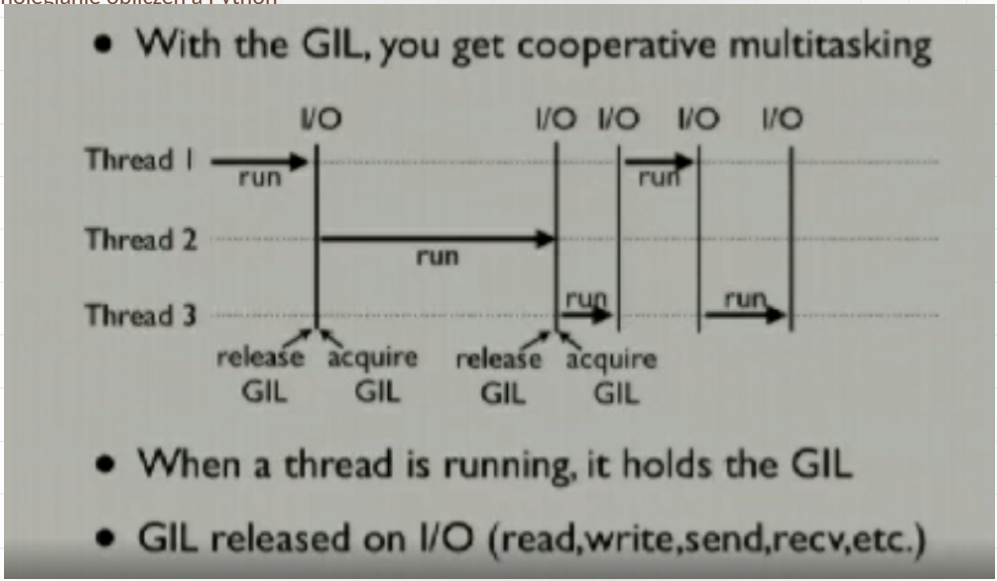
Zrównoleglanie obliczeń a Python

- głównym powodem takiego zachowania jest **GIL – Global Interpreter Lock**,
- od Pythona 3.2 wiele zostało poprawione,
- kilka dni temu powstała pierwsza działająca wersja bez GILa:

  https://mail.python.org/archives/list/python-dev@python.org/thread/ABR2L6BENNA6UPSPKV474HCS4LWT26GY/
- GIL zapewnia, że tylko jeden wątek może być wykonywany na raz przez interpreter,
- dlaczego? ułatwiło to implementację interpretera (zarządzanie pamięcią, wywołania zewnętrznych funkcji napisanych w C itd.),

Tylko jeden wątek na raz może przetrzymywać GIL

Zrównoleglanie obliczeń a Python



Podsumowanie

Zrównoleglanie obliczeń a Python

- problem głównie występuje dla wątków z dużym wykorzystaniem procesora, a małą liczbą operacji wejścia/wyjścia (ang. *CPU bound*),
- w Pythonie nie ma dobrego algorytmu szeregowania wątków,
- może to doprowadzić do sytuacji, w której tylko jeden wątek będzie uruchomiony, a reszta będzie oczekiwać (tzw. zagłodzenie),
- w połączeniu ze specjalnym mechanizmem sprawdzania (***check*** *mechnism*) w implementacji interpretera Pythona, można zaobserwować spore spowolnienie czasu działania programów używających wątki do zrównoleglania obliczeń

Wrocław University of Science and Technology

Przetwarzanie danych masowych

Wykład 3 – Podstawowe metody zrównoleglania algorytmów uczenia maszynowego. Przetwarzanie synchroniczne i asynchroniczne

dr hab. inż. Tomasz Kajdanowicz, Piotr Bielak, Roman Bartusiak

18.10.2021 r.