

# 並列分散処理 最終レポート

チーム E 大城 慶知 眞榮城 隆守 伊波卓浩 宮良友也

July 21, 2018

## 最終報告書に載せるやつ(あとで消すやつ)

演習の背景、目的、方法、結果、考察を A410 ページ以内で適切にまとめる。個々のメンバーの役割分担を明記する。記載がない場合、あるいは、曖昧な場合には、減点の対象となる。例えば、あるタスクを複数名で担当した場合でも、個々のメンバーの役割をできる限り区別して説明する。最終報告書にはソースコードの github リポジトリも記載する。

## 1 演習の背景

「講義で説明した並列分散処理を実践し、他者に有益な情報となるように共有せよ。」という課題が渡された。

b3 前期はメンバーが忙しく時間も取れないので軽量かつ有益な情報をということで、GPU マシンを使った並列処理を断腸の思いで断念し、Python における非同期処理を用いた I/O の並列処理を行うことにした。

## 2 目的

Python のスレッドと concurrent を用いて並列化を行うとともに、async await の使うことで非同期処理を行いさらに速度向上を目指す。

## 3 Python の事前知識

### 3.1 スレッドの制約

Python では、GIL(Global Interpreter Lock) と呼ばれる制約がある。GIL とは、Figure 1 と Figure2 のように Python を実行した際に一つだけしかスレッドのリソースを起動できない制約である。そのため、Python の CPU バウンドの並列処理はプロセスを使って、I/O バウンドの並列処理はスレッドを行う事が多い。

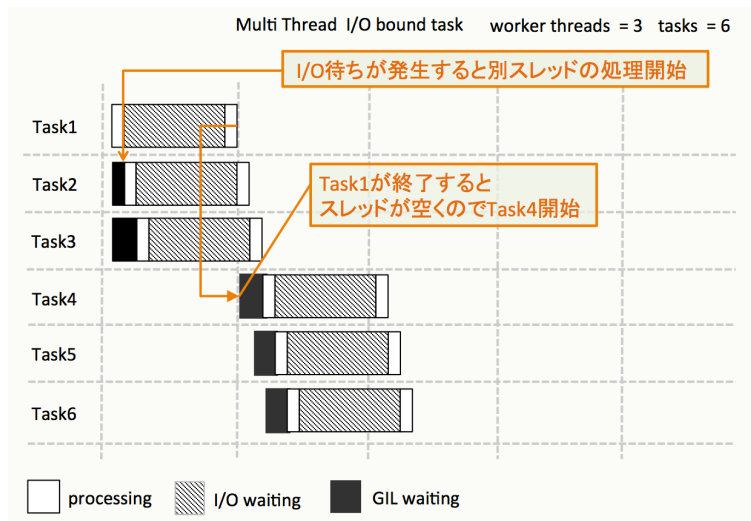


Figure 1: マルチスレッド I/O バウンド

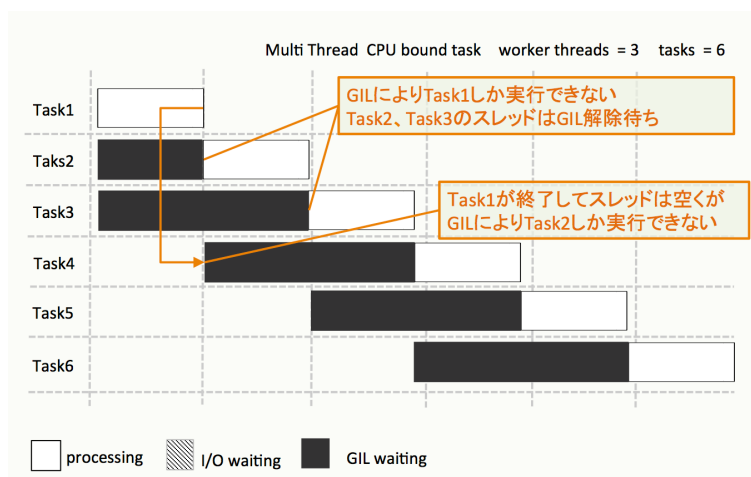


Figure 2: マルチスレッド CPU バウンド

## 3.2 async と await

コード 1: FizzBuzz using async-await

```

1 import asyncio
2
3 def fizzbuzz(i):
4     if i == 15:
5         return 'FizzBuzz'
6     if i % 5 == 0:
7         return 'Buzz'
8     if i % 3 == 0:
9         return 'Fizz'
10    return str(i)
11
12 async def task_fizzbuzz(prefix):
13     for x in range(1, 31):

```

```

14         await asyncio.sleep(0.3)
15         print(prefix + "{}:".format(str(x)) + fizzbuzz(x))
16     return None
17
18 loop = asyncio.get_event_loop()
19 # コルーチン個生成 1000
20 tasks = asyncio.wait([task_fizzbuzz(str(i) + ":") for i in range(1, 1000)])
21 loop.run_until_complete(tasks)
22 loop.close()

```

### 3.3 プロセスを用いた

## 4 実験方法

HTTP の GET を用いて実験を行った。GET を複数回実行する場合を考えると、逐次処理の場合ではレスポンスがあるまで次の GET を送信することができない。これを並列処理によりレスポンスを待つことなく次の GET を実行した。これにより効率よく GET を実行し、結果を受け取ることができると想定した。

コード 2: シンプレクス法プログラム

```

1  # Example 2: asynchronous requests
2  import asyncio
3  import requests
4  import time
5
6  start = 0
7
8  async def main():
9      start = time.time()
10
11     for num in range(4):
12         loop = asyncio.get_event_loop()
13         futures = [
14             loop.run_in_executor(
15                 None,
16                 requests.get,
17                 'https://ie.u-ryukyu.ac.jp/'
18             )
19             for i in range(5)
20         ]
21         for response in await asyncio.gather(*futures):
22             print(time.time() - start)
23             pass
24
25 loop = asyncio.get_event_loop()
26 loop.run_until_complete(main())

```

## 5 実行結果

ああ

こうなりました。

しゅごiiiiiiiiiiiiiiiiiiiiいつつ

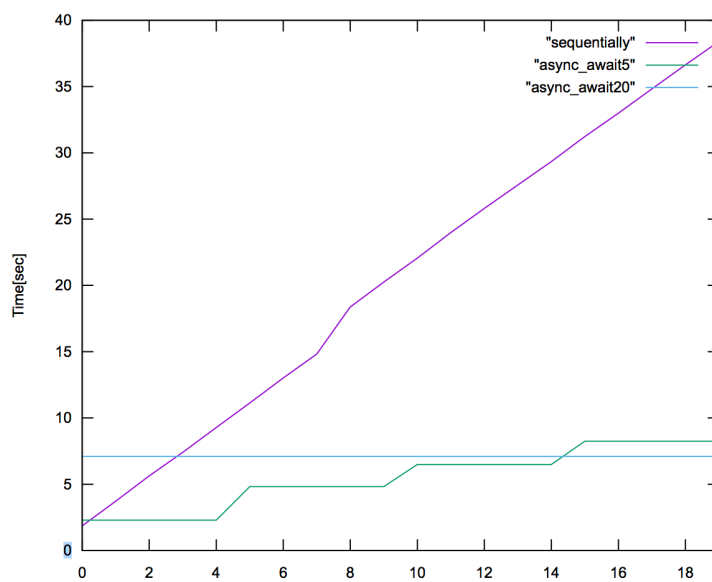


Figure 3: 実行結果

## 6 考察

## 7 感想・意見

### GitHub の URL

<https://github.com/e165719/ParallelDistributedProcessing>