

The logo consists of the lowercase letters 'kts' in a white, bold, sans-serif font, centered within a solid black square.

kts

Асинхронное программирование python для начинающих

Лабораторная работа

Работа с IO bound операциями

Студент: Студент

Цели

- Понять как ведут себя io bound операции в потоках и процессах в python
- Понять как ведут себя cpu bound операции в потоках и процессах в python
- Сравнить время выполнения реализованных программ
- Понять преимущества асинхронного подхода перед синхронным
- Понять последствия использования синхронных операций в асинхронном коде

Лабораторная работа

Задание 1. Последовательные запросы

Сделать 10 синхронных запросов на сайт <https://api.covidtracking.com/v1/us/current.json>. Для запросов использовать библиотеку [requests](#). Screenshot времени выполнения программы приложите к отчету.

0,62 с.

Задание 2. Запросы в тредах

Сделать 10 запросов на сайт <https://api.covidtracking.com/v1/us/current.json> в разных тредах. Для запросов использовать библиотеку [requests](#). Screenshot времени выполнения программы приложите к отчету.

0,31с.

Задание 3. Запросы в процессах

Сделать 10 запросов на сайт <https://api.covidtracking.com/v1/us/current.json> в разных процессах. Для запросов использовать библиотеку [requests](#). Screenshot времени выполнения программы приложите к отчету.

0,43 с.

Выводы Задания 1-3

При работе с операциями ввода выводы самым медленным подходом является синхронный ввиду того, что каждая следующая операция ждет завершения предыдущий.

В случае использования потоков ситуация сильно меняется и скорость выполнения возрастает пропорционально количеству задействованных потоков. Не смотря на то что GIL не позволяет одновременно выполнять несколько потоков сразу, потоки переключаются между собой пока один находится в ожидании другой производит операцию, что и приводит к уменьшению времени выполнения операций ввода вывода.

При работе с процессами результаты соизмеримы с ризалитами потоков, хотя немного уступают в скорости, это связано с тем что на развертывание процесса требуется время.

Функция которую будем использовать в заданиях связанных с сru bound операциями.

```
def countdown():  
    i = 0  
    begin = time.time()  
    while i < 5_000_000:  
        i += 1  
    print(f'duration: {time.time() - begin}')
```

Задание 4. Синхронное выполнение

10 раз синхронно вызвать функцию **countdown()**. Screenshot времени выполнения программы приложите к отчету.

3,11 с.

Задание 5. Выполнение в тредах

Вызвать функцию **countdown()** в 10 разных тредах. Время выполнения и код программы приложите к отчету.

3,13 с.

Задание 6. Выполнение в процессах

Вызвать функцию **countdown()** в 10 разных процессах. Время выполнения и код программы приложите к отчету.

1,75 с.

Выводы Заданий 4-6

Классический (синхронный) подход при выполнении математических задач оказался сопоставим с потоками. Оба подхода обрабатываются 1м ядром, но в случае потоков - есть расходы на переключение. По этой причине математические задачи выполнять в потоках бессмысленно.

С процессами ситуация кардинально отличается, здесь скорость выполнения операций растет пропорционально количеству свободных ядер, готовых обрабатывать нашу программу.

Задание 7. Сравнение производительности aiohttp и django при увеличении количества параллельных запросов

Запустите команды ab с соответствующими параметрами и заполните таблицу результатов

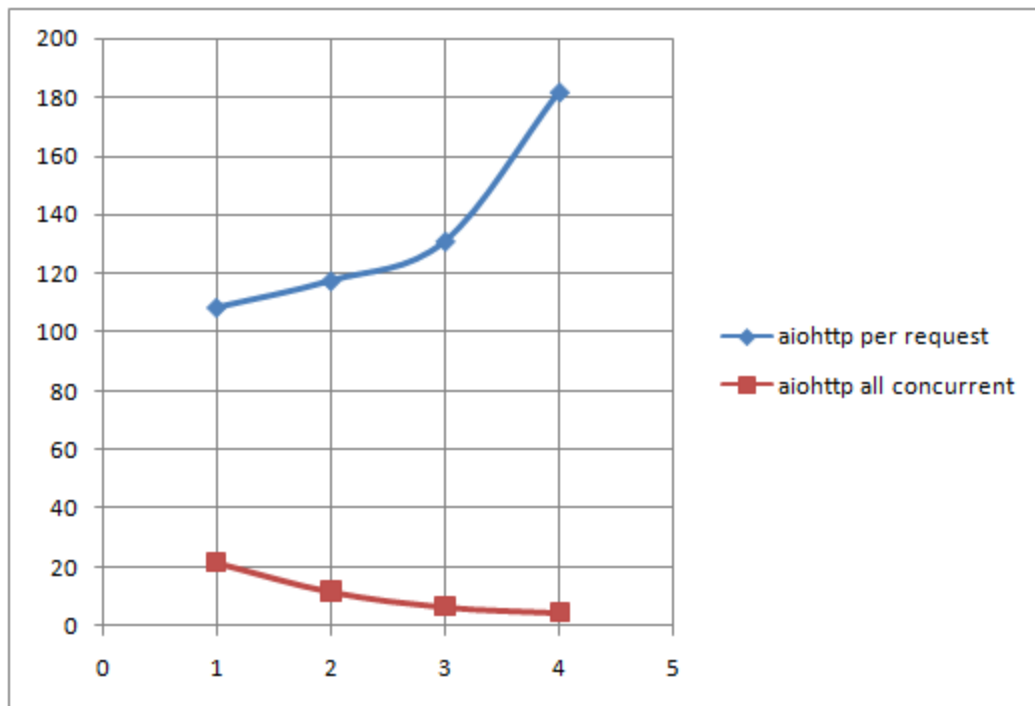
Aiohttp

ab -n 100 -с <Число параллельных соединений> <http://127.0.0.1:8088/>

Обработка 100 запросов

Число параллельных соединений	per request	all concurrent	Rps
5	108,518	21,704	46,08
10	117,603	11,760	85,03
20	131,037	6,552	152,63
40	181,808	4,545	220,01

Постройте графики по полученным значениям и приложите его к отчету.



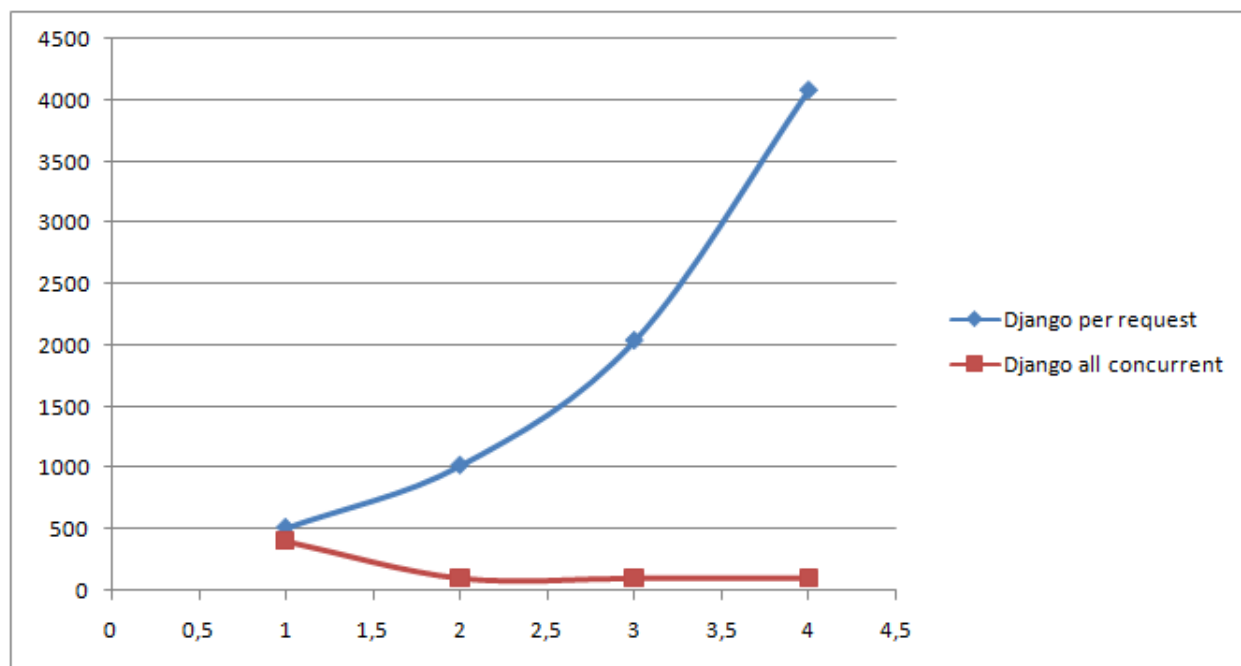
Django

ab -n 100 -c <Число параллельных соединений> <http://127.0.0.1:8089/>

Обработка 100 запросов

Число параллельных соединение	per request	all concurrent	rsp
5	512,147	102,429	9,76
10	1021,499	102,150	9,79
20	2042,556	102,128	9,79
40	4091,326	102,283	9,78

Постройте графики по полученным значениям и приложите его к отчету.



Выводы

При увеличении количества одновременных клиентов в aiohttp время на выполнение запроса незначительно увеличивается, при этом количество запросов которые может выдержать сервер увеличивается.

У Django время на выполнение запроса значительно больше чем у aiohttp, и способность принять запросы не меняется от нагрузки.

Задание 8. Сравнение производительности aiohttp и django при увеличении времени io bound операции

Одновременно изменяйте параметр sleep в aiohttp и django. Изменяемый параметр: время сна. PS: не забывайте перезапускать веб-серверы, иначе параметры не будут меняться.

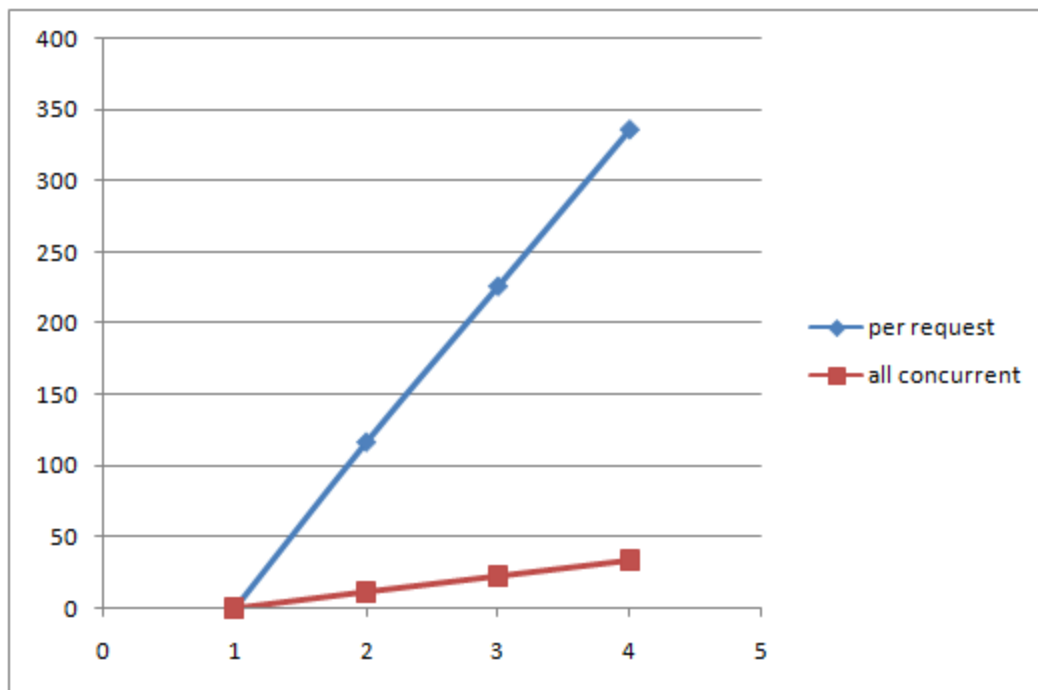
Aiohttp

ab -n 100 -c 10 <http://127.0.0.1:8088/>

Обработка 100 запросов

Время сна (сек.)	per request	all concurrent	rsp
0.1	117.076	11.637	85.93
0.2	226.506	22.651	44.15
0.3	336.514	33.651	29.72

Постройте графики по полученным значениям и приложите его к отчету.



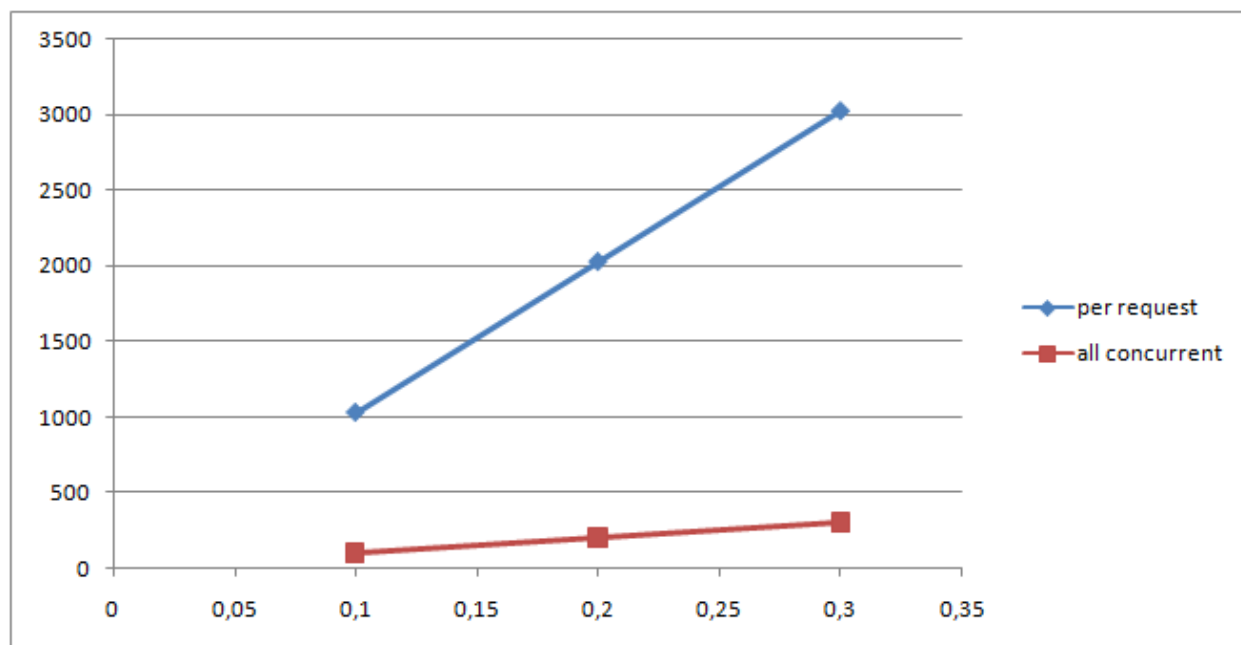
Django

ab -n 100 -c 10 <http://127.0.0.1:8089/>

Обработка 100 запросов

Время сна (сек.)	per request	all concurrent	rsp
0.1	1025.761	102.576	9.75
0.2	2026.275	202.628	4.94
0.3	3025.702	302.570	3.31

Постройте графики по полученным значениям и приложите его к отчету.



Выводы

При увеличении времени выполнения io-операций aiohttp, увеличивается время на выполнение запроса, уменьшается способность одновременно обслуживать несколько запросов, при этом показатели Django падают по сравнению с aiohttp в разы

Задание 9. Сравнение производительности aiohttp и django при увеличении количества параллельных запросов, если в aiohttp использовать синхронную операцию.

В коде aiohttp приложения замените `await asyncio.sleep(0.1)` на вызов `time.sleep(0.1)`

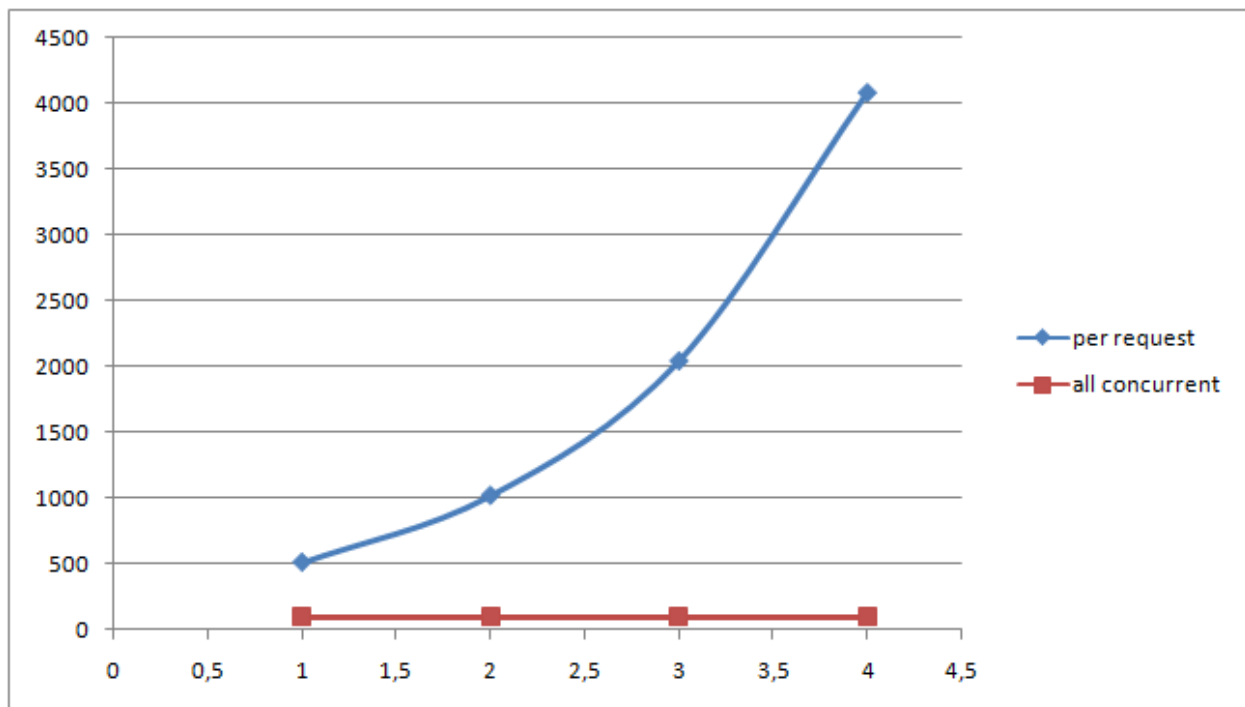
Запустите команды `ab` с соответствующими параметрами и заполните таблицу результатов

`ab -n 100 -c <Число параллельных соединений> http://127.0.0.1:8088/`

Обработка 100 запросов

Число параллельных соединений	per request	all concurrent	rsp
5	510.606	102.121	9.79
10	1017.942	101.794	9.82
20	2038.707	101.935	9.81
40	4074.766	101.869	9.82

Постройте графики по полученным значениям и приложите его к отчету.



Выводы

При выполнении в коде синхронных операций `aiohttp` по производительности приравнивается к Django.