Сравним aiohttp и Django

Мы не будем сравнивать rps, в этом нет никакого смысла. Внутри django много написанных модулей, которые замедляют ее работу, но позволяют программистам быстрее писать код. В aiohttp на старте такого нет, и уже поэтому он будет быстрее.

Поэтому мы:

- Сравним динамику изменения rps при увеличении числа параллельных запросов
- Сравним динамику изменения rps при увеличении времени aiobound-операции
- Посмотрим, как деградирует производительность асинхронного сервера aiohttp при использовании синхронных операций.

Для выполнения практики нам понадобится mercury из предыдущей лабораторной работы.

Для сравнения будем использовать 2 функции из aiohttp и django соответственно:

aiohttp, функция handle

```
# servers/aio/server.py

import asyncio
import time

from aiohttp import web


async def handle(request):
    await asyncio.sleep(0.2)
    return web.Response(text='Hello, Anonymous')

app = web.Application()
app.add_routes([web.get('/', handle),
                web.get('/{name}', handle)])

if __name__ == '__main__':
    web.run_app(app, port=8088, host='0.0.0.0')
```

django, функция index

```
# servers/dj/dj/urls.py
import time

from django.contrib import admin
from django.http import HttpResponse
from django.urls import path


def index(request):
    time.sleep(0.1)
    return HttpResponse(status=200, content='Hello, Anonymous')


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index)
]
```

Будем использовать sleep, так как он повторяет поведение синхронной и асинхронной операции ввода и вывода — io bound операции.