Comparações

As comparações independentes da TechEmpower mostram as aplicações **FastAPI** rodando com Uvicorn como um dos *frameworks* Python mais rápidos disponíveis, somente atrás dos próprios Starlette e Uvicorn (utilizados internamente pelo FastAPI). (*)

Mas quando se checa *benchmarks* e comparações você deveria ter o seguinte em mente.

Comparações e velocidade

Ao verificar os *benchmarks*, é comum observar algumas ferramentas de diferentes tipos comparadas como equivalentes.

Especificamente, observa-se Uvicorn, Starlette e FastAPI comparados juntos (entre muitas outras ferramentas).

Quanto mais simples o problema resolvido pela ferramenta, melhor a performance que ela terá. E a maioria dos *benchmarks* não testam as características adicionais fornecidas pela ferramenta.

A hierarquia segue assim:

- **Uvicorn**: um servidor ASGI
  - **Starlette**: (utiliza Uvicorn) um *microframework web*
    - **FastAPI**: (utiliza Starlette) um *microframework* de API com vários recursos adicionais para construção de APIs, com validação de dados, etc.
- **Uvicorn**:
  - Terá a melhor performance, já que ele não tem muito código extra além do servidor em si.
  - Você não conseguiria escrever uma aplicação em Uvicorn diretamente. Isso significa que seu código deveria conter, mais ou menos, todo o código fornecido pelo Starlette (ou **FastAPI**). E se você fizesse isso, sua aplicação final poderia ter a mesma sobrecarga que utilizar um *framework* que minimiza o código e *bugs* da sua aplicação.
  - Se você quer fazer comparações com o Uvicorn, compare com Daphne, Hypercorn, uWSGI, etc. Servidores de Aplicação.
- **Starlette**:
  - Terá a melhor performance, depois do Uvicorn. De fato, Starlette utiliza Uvicorn para rodar. Então, ele provavelmente será "mais lento" que Uvicorn por ter que executar mais código.
  - Mas ele fornece a você as ferramentas para construir aplicações *web* simples, com roteamento baseado em caminhos, etc.
  - Se você quer fazer comparações com o Starlette, compare com Sanic, Flask, Django, etc. *Frameworks Web* (ou *microframeworks*).
- **FastAPI**:

- Do mesmo modo que Starlette utiliza Uvicorn e não pode ser mais rápido que ele, **FastAPI** utiliza o Starlette, então não tem como ser mais rápido do que o Starlette.
- FastAPI fornece mais recursos acima do Starlette. Recursos que você quase sempre precisará quando construir APIs, como validação de dados e serialização. E utilizando eles, você terá uma documentação automática de graça (a documentação automática nem sequer adiciona peso para rodar as aplicações, ela é gerada na inicialização).
- Se você nunca utilizou FastAPI mas utilizou diretamente o Starlette (ou outra ferramenta, como Sanic, Flask, Responder, etc) você teria que implementar toda validação de dados e serialização por conta. Então, sua aplicação final poderia ainda ter a mesma sobrecarga como se fosse desenvolvida com FastAPI. Em muitos casos, a validação de dados e serialização é o maior pedaço de código escrito em aplicações.
- Então, ao utilizar o FastAPI você estará economizando tempo de desenvolvimento, evitará *bugs*, linhas de código, e você provavelmente terá a mesma performance (ou melhor) do que não utilizá-lo (já que você teria que implementar tudo isso em seu código).
- Se você quer fazer comparações com o FastAPI, compare com um *framework* (ou conjunto de ferramentas) para aplicações *web* que forneça validação de dados, serialização e documentação, como Flask-apispec, NestJS, Molten, etc. *Frameworks* com validação de dados automática, serialização e documentação integradas.