Programação Reativa

Spring WebFlux

Definição

- O processamento reativo é um paradigma que permite aos desenvolvedores criar aplicativos **assíncronos** e sem bloqueio que podem lidar com *back-pressure* (controle de fluxo).

- Os sistemas reativos utilizam melhor os processadores modernos. Além disso, a inclusão do ***back-pressure*** na programação reativa garante uma melhor resiliência entre componentes dissociados.

- ***Streaming*** trata-se de uma eficiente técnica arquitetural a qual explora modelos de construção de fluxos de dados, reagindo a eventos na linha do tempo. Seu processamento e transformações de dados são realizadas em um *Pipeline* composto de *Steps*, conhecidos como *Processors*.

Manifesto Reativo

- A programação reativa baseia-se em 4 pilares essenciais:
- **Elástico**: Reage à demanda/carga: aplicações podem fazer uso de múltiplos núcleos e múltiplos servidores;
- **Resiliente**: Reage às falhas; aplicações reagem e se recuperam de falhas de software, hardware e de conectividade;
- **Message Driven:** Reage aos eventos (*event driven*): em vez de compor aplicações por múltiplas *threads* síncronas, sistemas são compostos de gerenciadores de eventos assíncronos e não bloqueantes;
- **Responsivo**: Reage aos usuários: aplicações que oferecem interações ricas e “tempo real” com usuários.

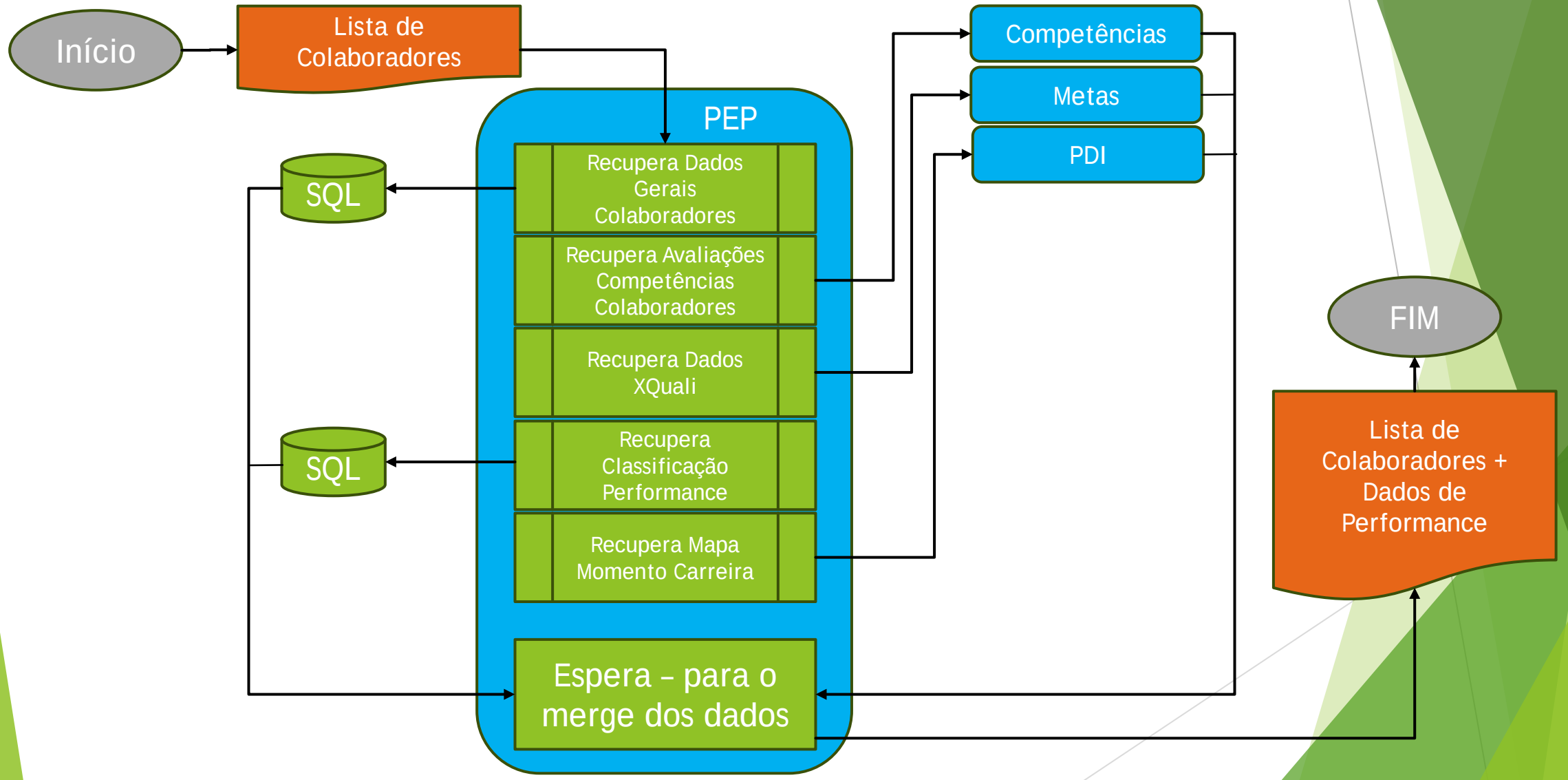
O que é Back-pressure?

https://www.baeldung.com/spring-webflux-backpressure

- É a capacidade de um *Consumer* sinalizar ao *Producer* que a taxa de emissão é maior do que ele pode suportar. Assim, utilizando este mecanismo, o *Consumer* obtém controle sobre a velocidade com que os dados são emitidos. Com isso, o *Subscriber* controla o fluxo de dados do *Publisher*.

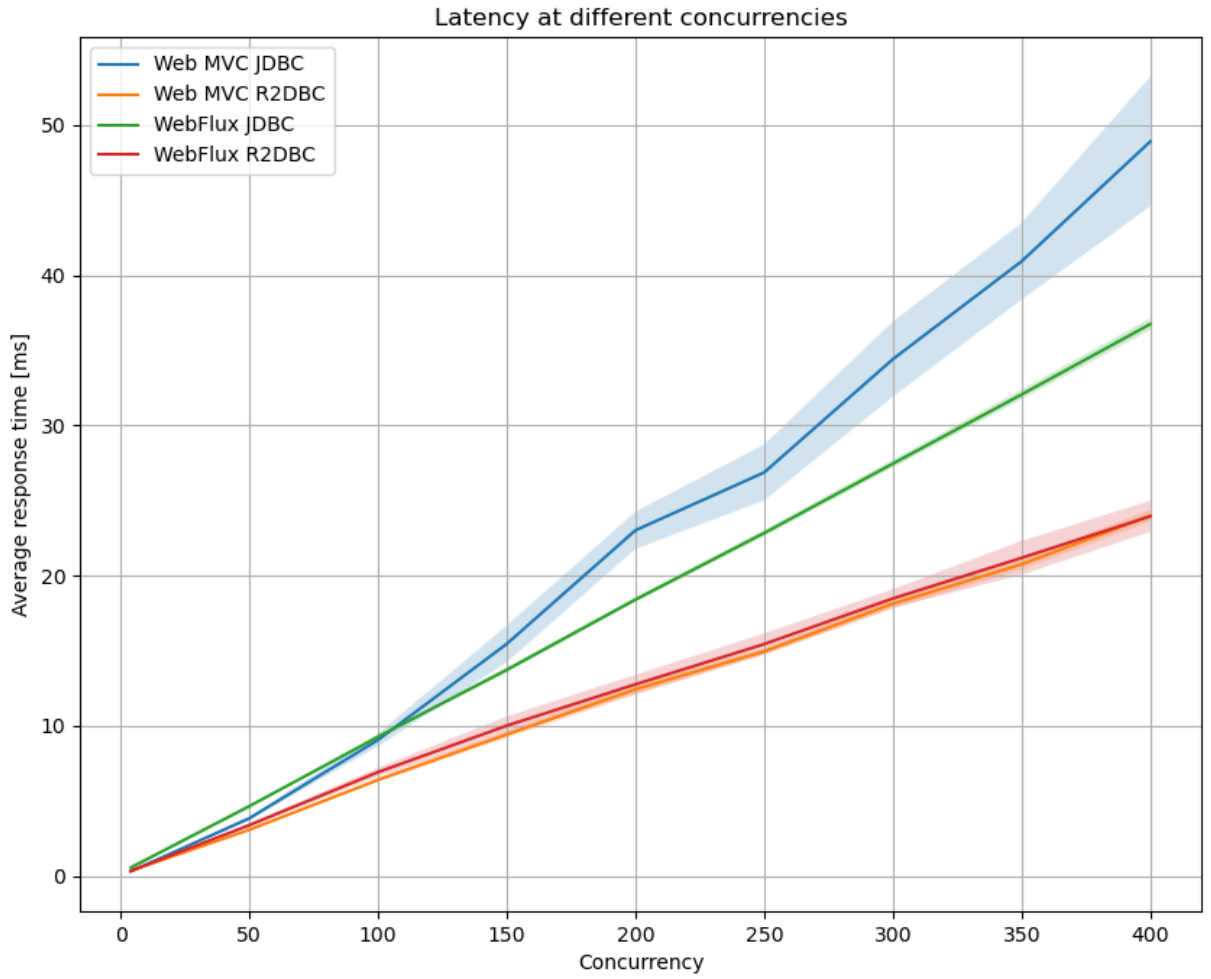
- Devido à natureza não bloqueante da Programação Reativa, o servidor não envia o fluxo completo de uma só vez. Ele pode enviar os dados simultaneamente assim que estiverem disponíveis. Assim, o cliente espera menos tempo para receber e processar os eventos.

Use Case: Relatório de Performance



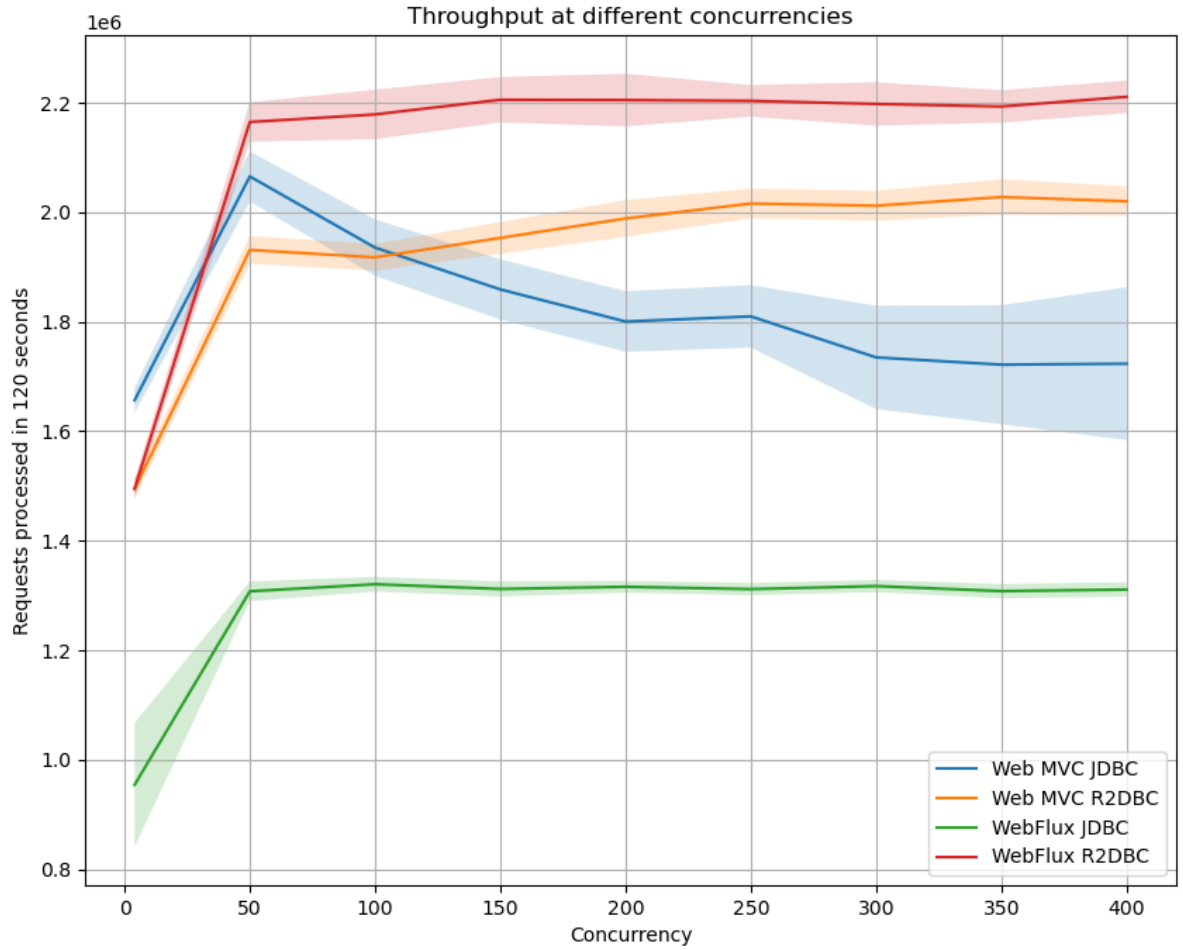
Response Time

https://technology.amis.nl/software-development/performance-and-tuning/spring-blocking-vs-non-blocking-r2dbc-vs-jdbc-and-webflux-vs-web-mvc



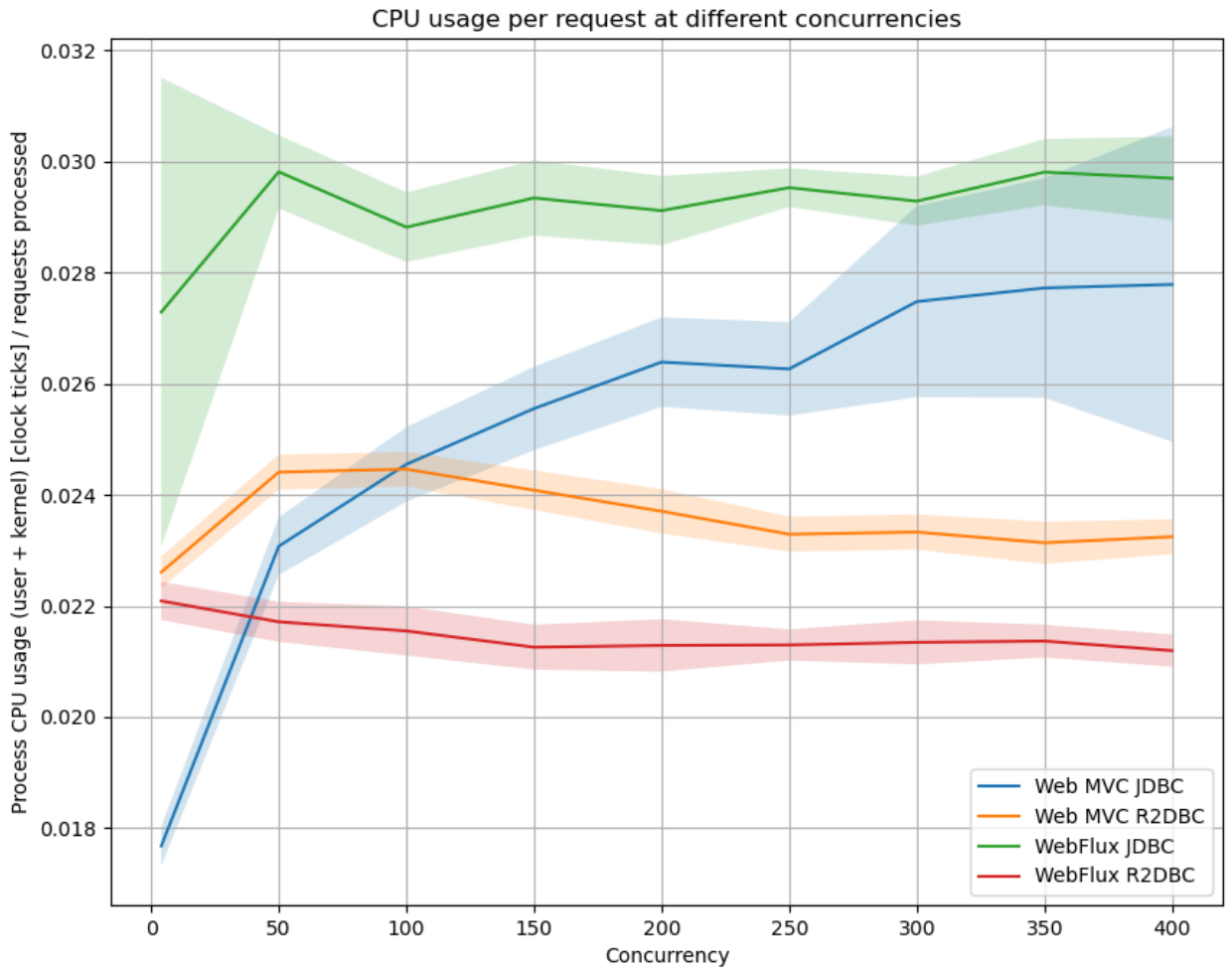
THROUGHPUT

https://technology.amis.nl/software-development/performance-and-tuning/spring-blocking-vs-non-blocking-r2dbc-vs-jdbc-and-webflux-vs-web-mvc



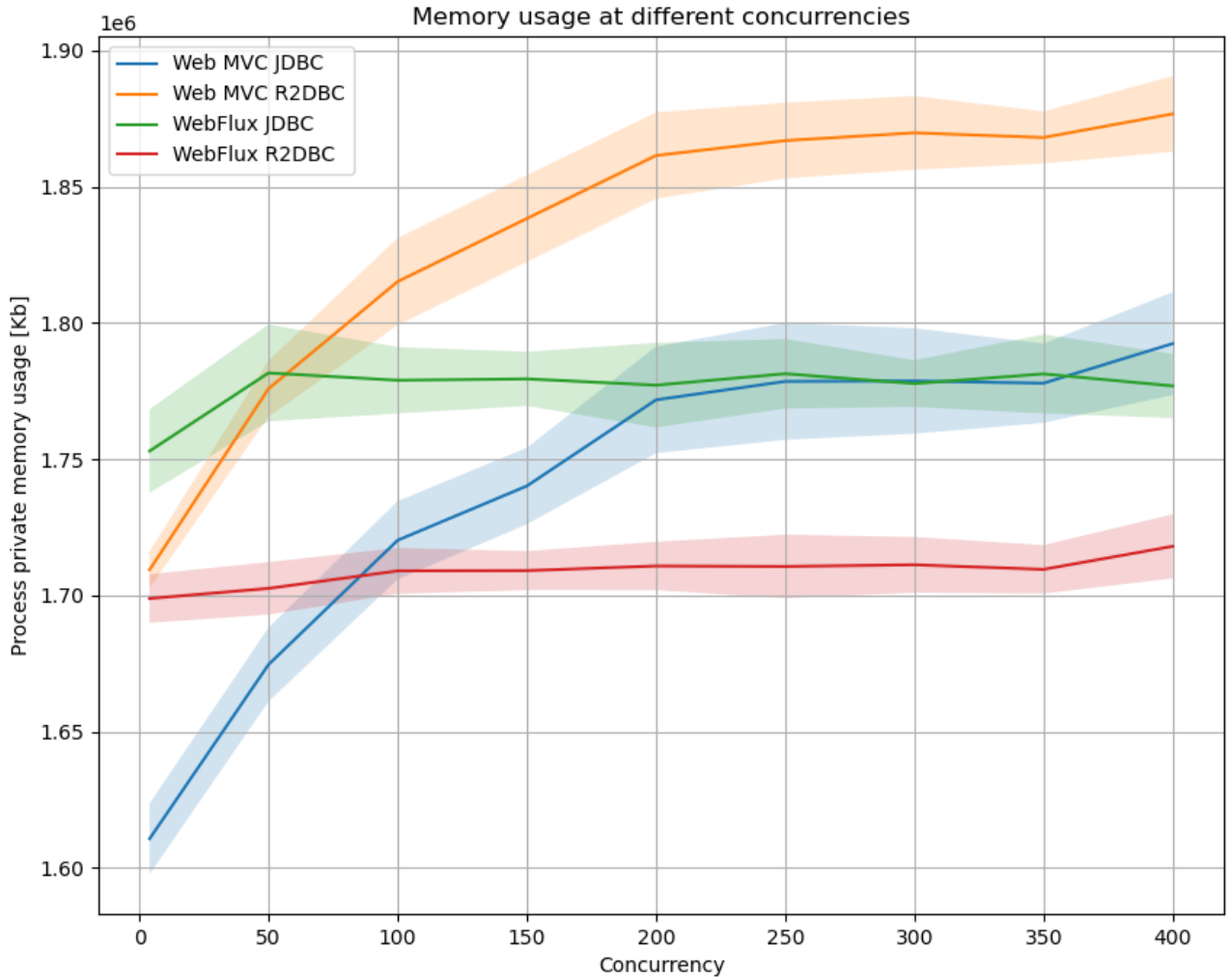
CPU usage per request

https://technology.amis.nl/software-development/performance-and-tuning/spring-blocking-vs-non-blocking-r2dbc-vs-jdbc-and-webflux-vs-web-mvc



MEMORY

https://technology.amis.nl/software-development/performance-and-tuning/spring-blocking-vs-non-blocking-r2dbc-vs-jdbc-and-webflux-vs-web-mvc



Spring Boot

Reactive Stack

Spring WebFlux is a non-blocking web framework built from the ground up to take advantage of multi-core, next-generation processors and handle massive numbers of concurrent connections.

Servlet Stack

Spring MVC is built on the Servlet API and uses a synchronous blocking I/O architecture with a one-request-per-thread model.

Reactive Stack	Servlet Stack
Netty, Servlet 3.1+ Containers	Servlet Containers
Reactive Streams Adapters	Servlet API
Spring Security Reactive	Spring Security
Spring WebFlux	Spring MVC
Spring Data Reactive Repositories Mongo, Cassandra, Redis, Couchbase, R2DBC	**Spring Data Repositories** JDBC, JPA, NoSQL

Mão na massa!

- https://github.com/leandro-jesus-eng/crud-person-webflux

- **Leandro de Jesus**
- **leandro.jesus.eng@gmail.com**
- ***Engenheiro de Software na TCS***
- Mestre em Ciência da Computação
- Doutor Meio Ambiente e Desenvolvimento Regional