THE DEVELOPER'S CONFERENCE

## Java Enterprise

# De monolito para microserviços

## - e algumas descobertas de performance

**Renan Zenkner Roggia**

SAP Cloud Platform Tax Service - Software Engineer

THE DEVELOPER'S CONFERENCE

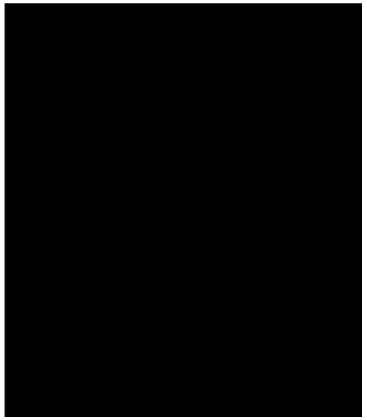
O que vocês fariam se tivessem que aumentar o throughput  do seu web service em 2400%?

## De 1.000 para 25.000

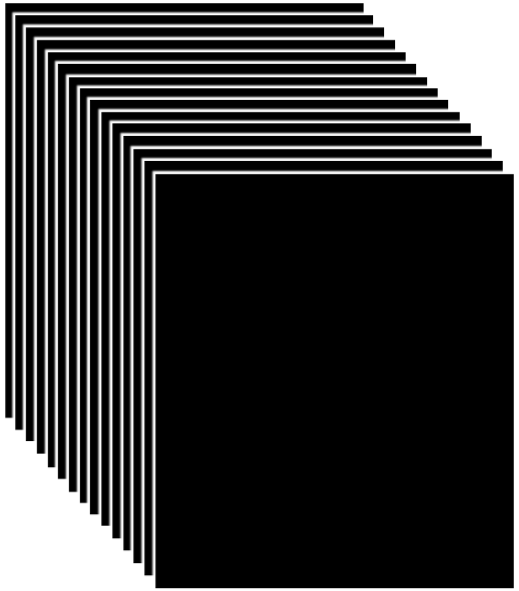
1.000 r/m → 25.000 r/m

* Esses números são meramente ilustrativos
* Proporcionais

## Escalando o monolito



1



25

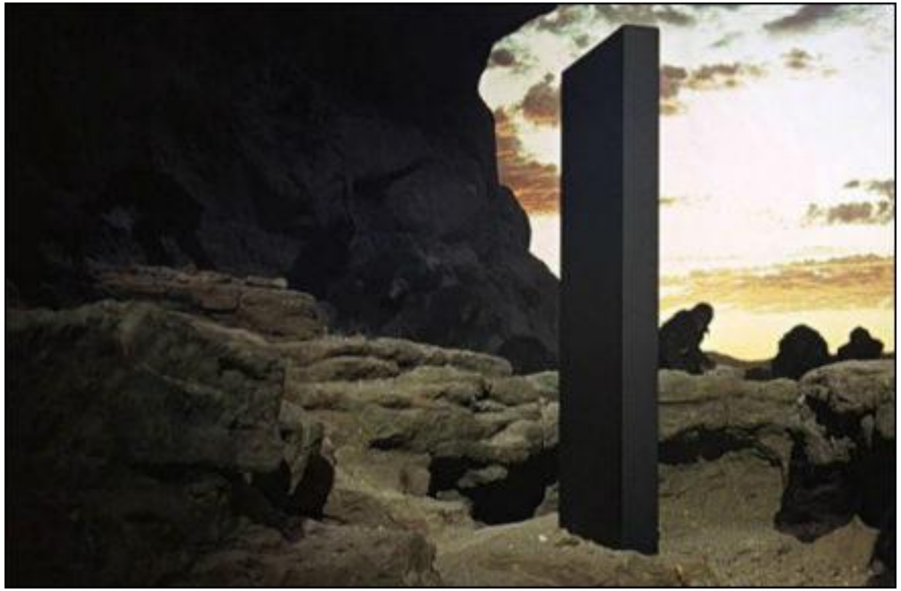
THE DEVELOPER'S CONFERENCE

## Nosso arquitetura de monolito é eficaz?

# Um monolito

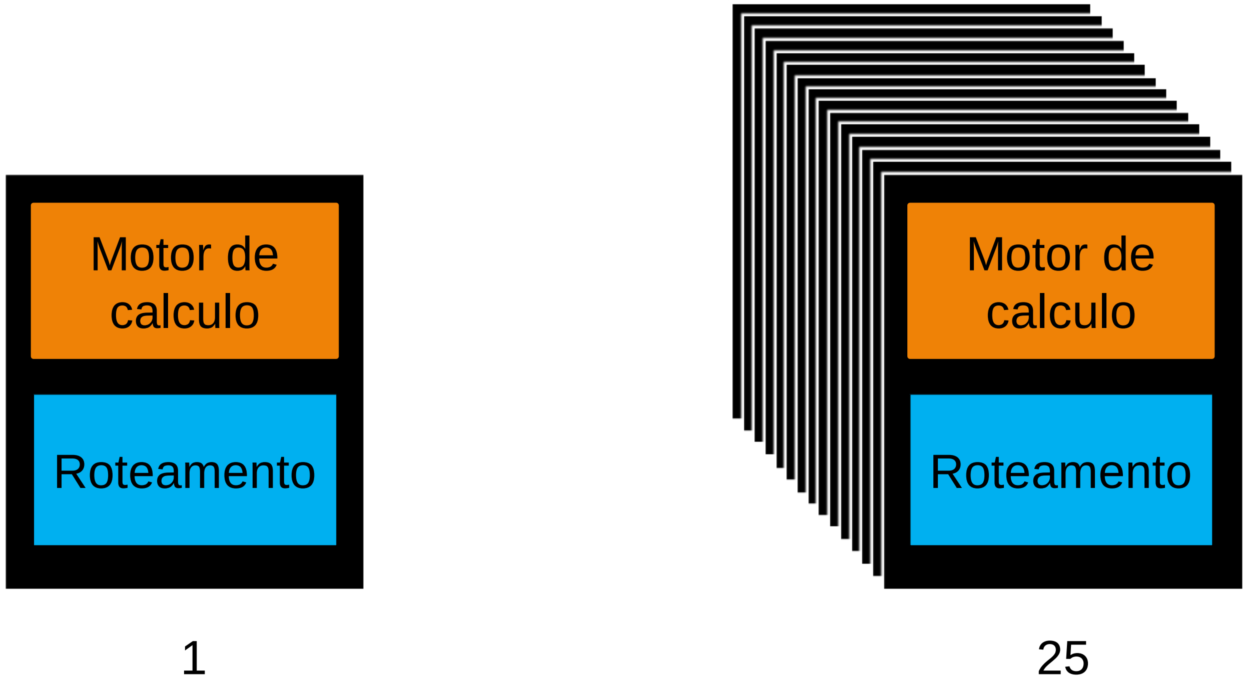
Roteamento:

1. Motor de calculo interno
   - Processamento alto
2. Motor de calculo externos
   - Alto uso de I/O

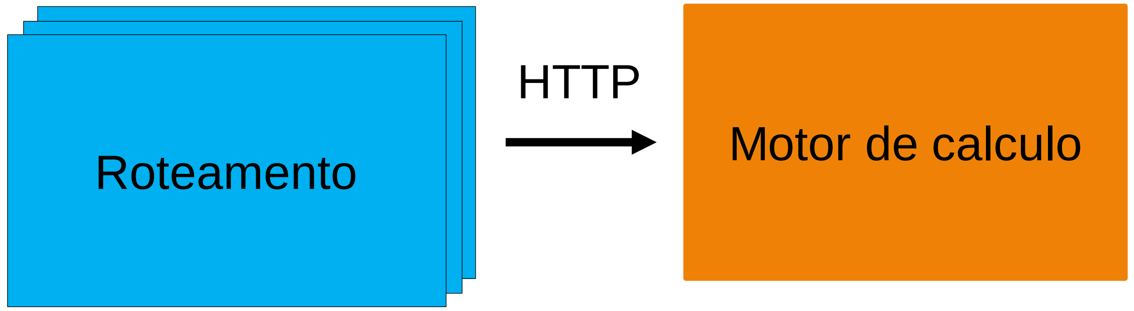


* Foto oficial do monólito

## Desperdiçando de recurso



# Duas metades de um monolito

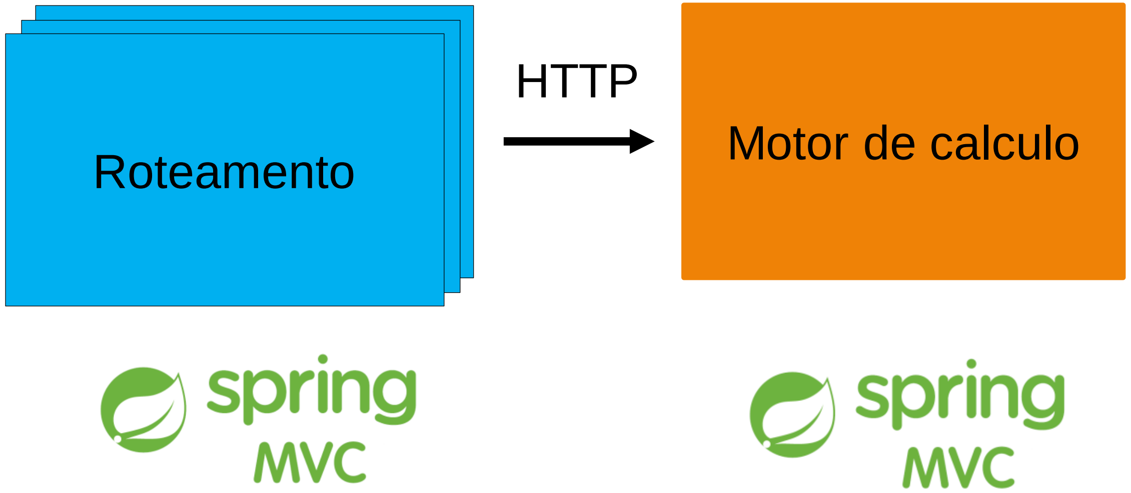


Separando o monolito.

Otimizamos o consumo de recursos, tornamos os ciclos de entrega independentes.

## Duas metades de um monolito

THE DEVELOPER'S CONFERENCE



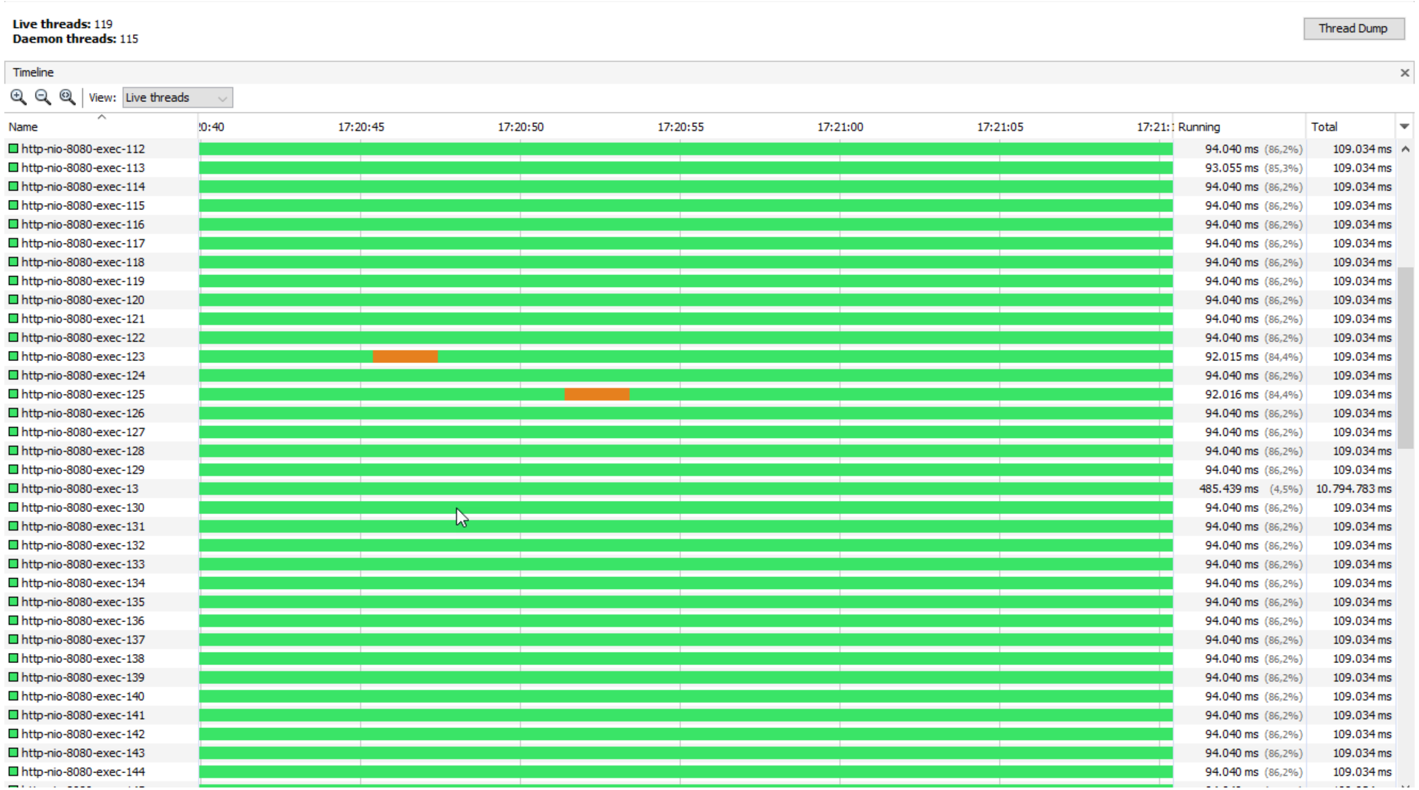
Atualizando a stack de desenvolvimento.

**Aumentamos de 1.000 r/m para 13.000 r/m**

## Nossa aplicação é eficaz?

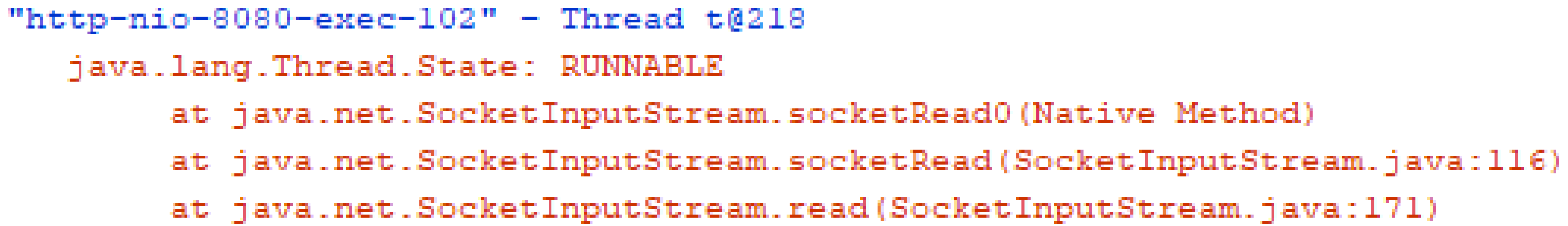
```java
@RestController
public class RoutingController {

    private static final String ENGINE_URL = "http://localhost:8081/route";

    @GetMapping(path = "route", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    public ResponseEntity<String> route(@RequestParam String delay) {

        String uri = UriComponentsBuilder.fromHttpUrl(ENGINE_URL).queryParam("delay", delay).toUriString();

        try {
            String response = new RestTemplate().getForObject(uri, String.class);
            return ResponseEntity.ok().body(response);
        } catch (HttpClientErrorException.BadRequest | HttpServerErrorException.InternalServerError e) {
            return ResponseEntity.badRequest().body(e.getResponseBodyAsString());
        }
    }
}
```

# Síncrono e bloqueante


**Live threads:** 119
**Daemon threads:** 115

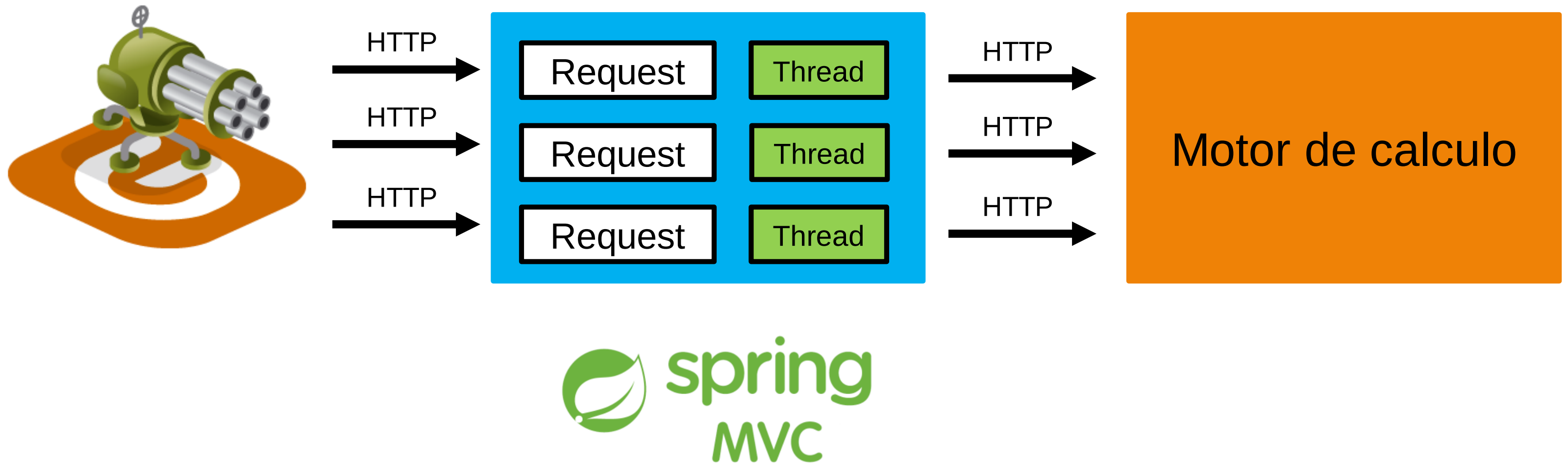
| Name | Running | | Total |
|---|---|---|---|
| http-nio-8080-exec-112 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-113 | 93.055 ms | (85,3%) | 109.034 ms |
| http-nio-8080-exec-114 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-115 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-116 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-117 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-118 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-119 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-120 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-121 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-122 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-123 | 92.015 ms | (84,4%) | 109.034 ms |
| http-nio-8080-exec-124 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-125 | 92.016 ms | (84,4%) | 109.034 ms |
| http-nio-8080-exec-126 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-127 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-128 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-129 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-13 | 485.439 ms | (4,5%) | 10.794.783 ms |
| http-nio-8080-exec-130 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-131 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-132 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-133 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-134 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-135 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-136 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-137 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-138 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-139 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-140 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-141 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-142 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-143 | 94.040 ms | (86,2%) | 109.034 ms |
| http-nio-8080-exec-144 | 94.040 ms | (86,2%) | 109.034 ms |

## Síncrono e bloqueante



```
"http-nio-8080-exec-102" - Thread t@218
   java.lang.Thread.State: RUNNABLE
        at java.net.SocketInputStream.socketRead0(Native Method)
        at java.net.SocketInputStream.socketRead(SocketInputStream.java:116)
        at java.net.SocketInputStream.read(SocketInputStream.java:171)
```

# One thread per request model



*“Waiting within the servlet is an inefficient operation as it is a blocking operation that consumes a thread and other limited resources.”*

Java Servlet Specification 3.1

THE DEVELOPER'S CONFERENCE

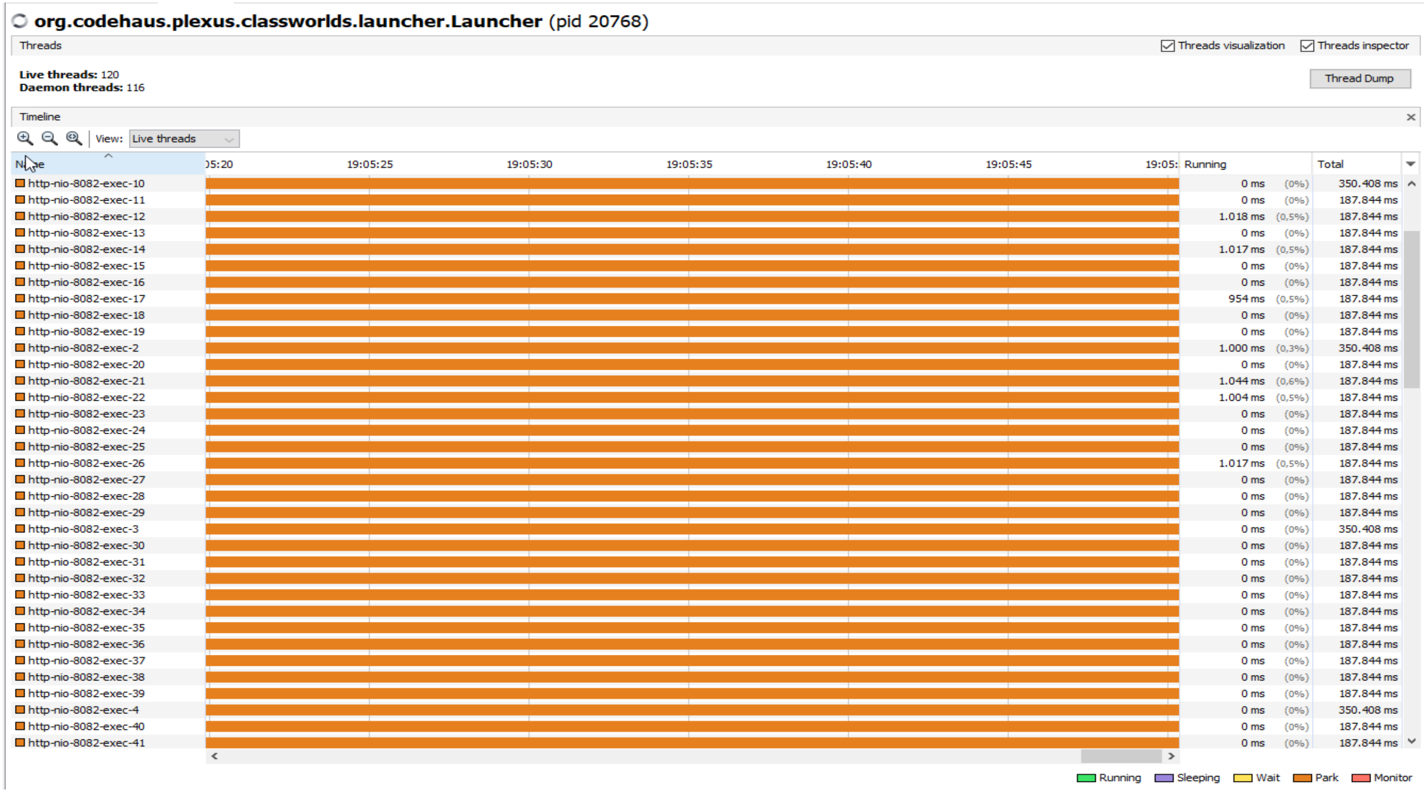
```java
@GetMapping(path = "anotherRoute", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public DeferredResult<ResponseEntity<String>> anotherRoute(@RequestParam String delay) {
    DeferredResult<ResponseEntity<String>> response = new DeferredResult<>();

    ForkJoinPool.commonPool().submit(() -> {
        String uri = UriComponentsBuilder.fromHttpUrl(ENGINE_URL).queryParam("delay", delay).toUriString();

        try {
            return ResponseEntity.ok().body(new RestTemplate().getForObject(uri, String.class));
        } catch (HttpClientErrorException.BadRequest | HttpServerErrorException.InternalServerError e) {
            return ResponseEntity.badRequest().body(e.getResponseBodyAsString());
        }

    });

    return response;
}
```

THE DEVELOPER'S CONFERENCE

```java
private static WebClient webClient = WebClient.create(ENGINE_URL);

@GetMapping(path = "route", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public Mono<String> route(@RequestParam String delay) {

    return webClient.get().uri("/route?delay=" + delay)
            .header(HttpHeaders.CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE).retrieve()
            .onStatus(HttpStatus::is4xxClientError, e -> Mono.error(new RuntimeException("e")))
            .onStatus(HttpStatus::is5xxServerError, e -> Mono.error(new RuntimeException("e")))
            .bodyToMono(String.class);

}
```

# Assíncrono e não bloqueante

THE DEVELOPER'S CONFERENCE



**org.codehaus.plexus.classworlds.launcher.Launcher** (pid 20768)

Threads ☑ Threads visualization ☑ Threads inspector

**Live threads:** 120
**Daemon threads:** 116

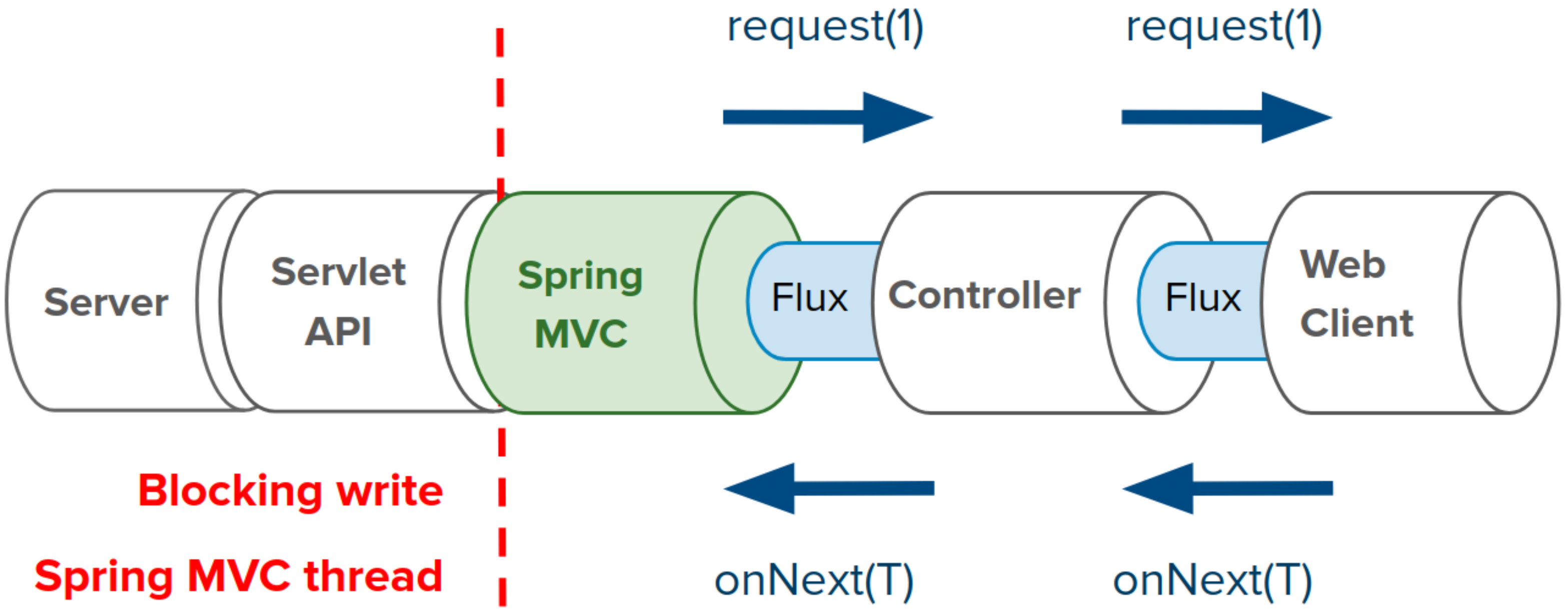
Thread Dump

Timeline

View: Live threads

| Name | Running | | Total |
|---|---|---|---|
| http-nio-8082-exec-10 | 0 ms | (0%) | 350.408 ms |
| http-nio-8082-exec-11 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-12 | 1.018 ms | (0,5%) | 187.844 ms |
| http-nio-8082-exec-13 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-14 | 1.017 ms | (0,5%) | 187.844 ms |
| http-nio-8082-exec-15 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-16 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-17 | 954 ms | (0,5%) | 187.844 ms |
| http-nio-8082-exec-18 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-19 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-2 | 1.000 ms | (0,3%) | 350.408 ms |
| http-nio-8082-exec-20 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-21 | 1.044 ms | (0,6%) | 187.844 ms |
| http-nio-8082-exec-22 | 1.004 ms | (0,5%) | 187.844 ms |
| http-nio-8082-exec-23 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-24 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-25 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-26 | 1.017 ms | (0,5%) | 187.844 ms |
| http-nio-8082-exec-27 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-28 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-29 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-3 | 0 ms | (0%) | 350.408 ms |
| http-nio-8082-exec-30 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-31 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-32 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-33 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-34 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-35 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-36 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-37 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-38 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-39 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-4 | 0 ms | (0%) | 350.408 ms |
| http-nio-8082-exec-40 | 0 ms | (0%) | 187.844 ms |
| http-nio-8082-exec-41 | 0 ms | (0%) | 187.844 ms |

🟩 Running 🟪 Sleeping 🟨 Wait 🟧 Park 🟥 Monitor



Source: [https://www.infoq.com/articles/Servlet-and-Reactive-Stacks-Spring-Framework-5/](https://www.infoq.com/articles/Servlet-and-Reactive-Stacks-Spring-Framework-5/)

## Alternativas ao Servlet API?

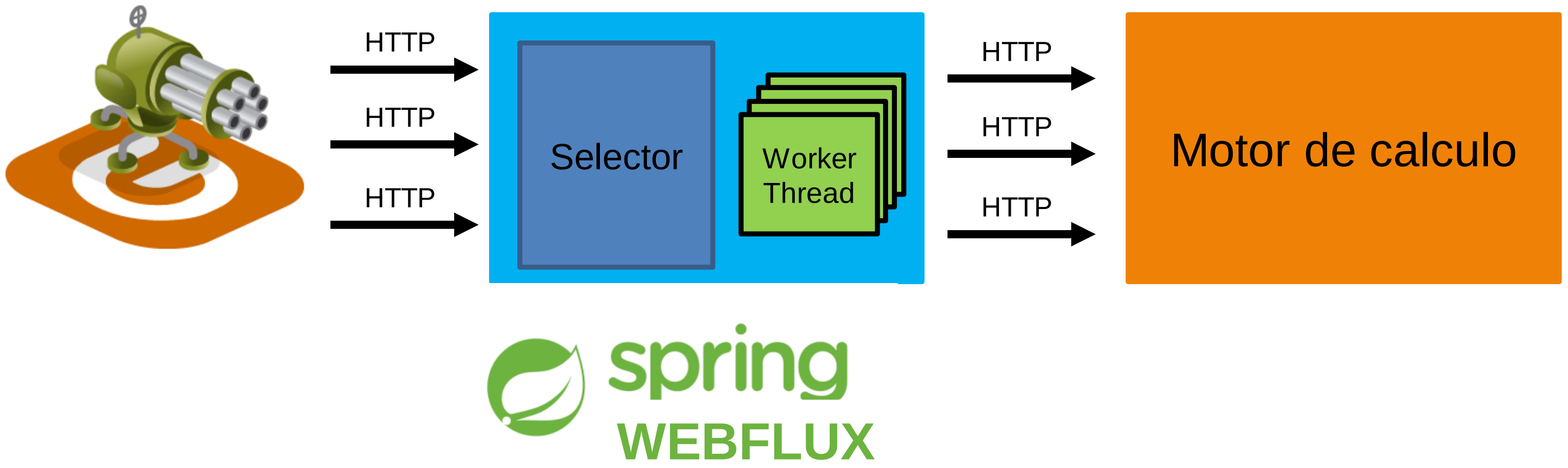
## Spring WebFlux

THE DEVELOPER'S CONFERENCE

*"For a non-blocking web stack to handle concurrency with a small number of threads and scale with fewer hardware resources."*

Source: https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html

# Spring WebFlux



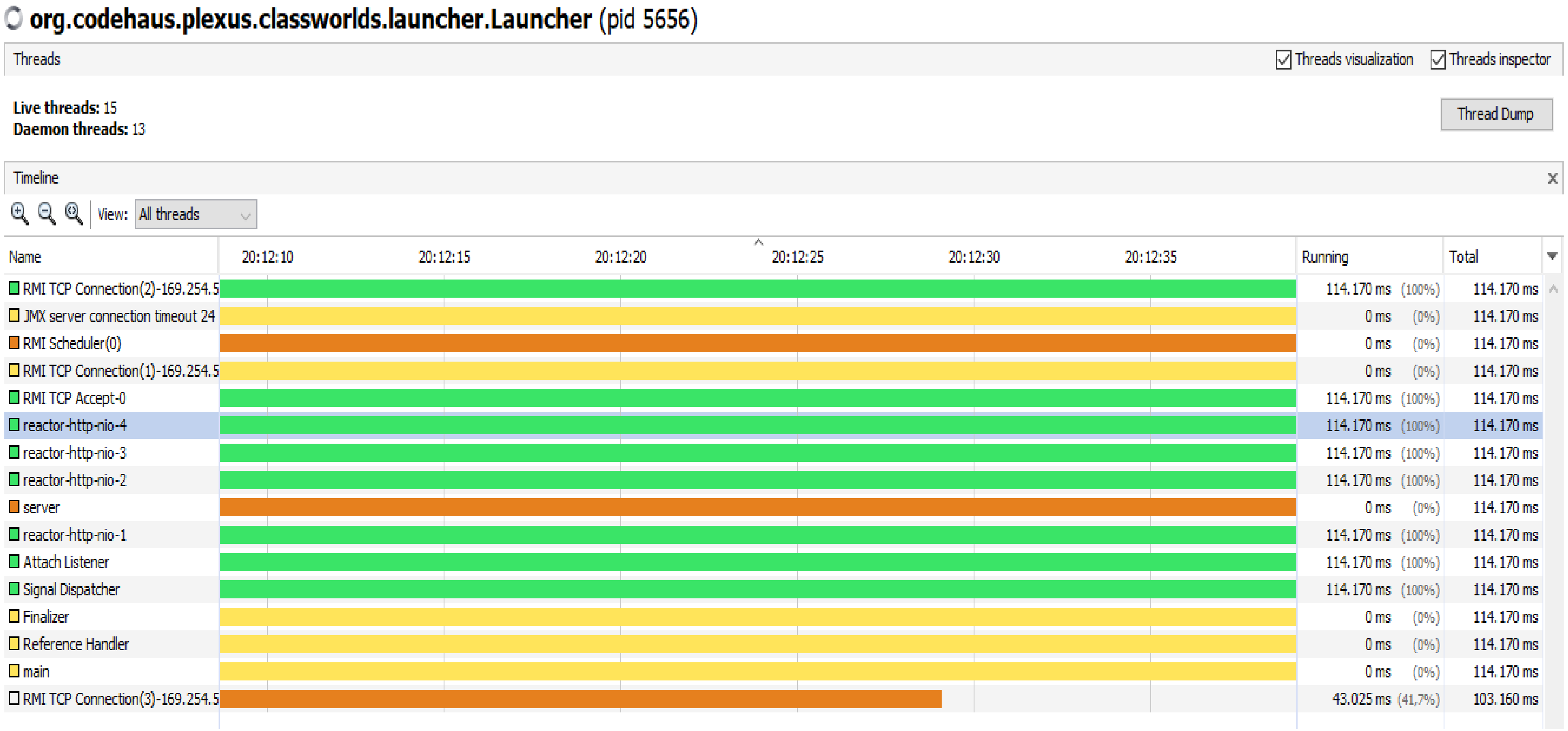
THE DEVELOPER'S CONFERENCE

```
private static WebClient webClient = WebClient.create(ENGINE_URL);

@GetMapping(path = "route", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public Mono<String> route(@RequestParam String delay) {

    return webClient.get().uri("/route?delay=" + delay)
            .header(HttpHeaders.CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE).retrieve()
            .onStatus(HttpStatus::is4xxClientError, e -> Mono.error(new RuntimeException("e")))
            .onStatus(HttpStatus::is5xxServerError, e -> Mono.error(new RuntimeException("e")))
            .bodyToMono(String.class);

}
```

```java
@Configuration
public class RoutingHandler {

    private static WebClient webClient = WebClient.create("http://localhost:8081");

    @Bean
    public RouterFunction<?> routes() {
        return RouterFunctions.route().GET("/route", request -> {
            Optional<String> delay = request.queryParam("delay");

            return webClient.get().uri("/route?delay=" + delay.get())
                    .header(HttpHeaders.CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE).retrieve()
                    .bodyToMono(String.class).flatMap(body -> ServerResponse.ok().syncBody(body));

        }).build();
    }
}
```

# Event Loop

THE DEVELOPER'S



org.codehaus.plexus.classworlds.launcher.Launcher (pid 5656)

Threads ☑ Threads visualization ☑ Threads inspector

**Live threads:** 15
**Daemon threads:** 13

Thread Dump

Timeline

View: All threads

| Name | 20:12:10 … 20:12:35 | Running | | Total |
|---|---|---|---|---|
| 🟩 RMI TCP Connection(2)-169.254.5 | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟨 JMX server connection timeout 24 | 🟨 | 0 ms | (0%) | 114.170 ms |
| 🟧 RMI Scheduler(0) | 🟧 | 0 ms | (0%) | 114.170 ms |
| 🟨 RMI TCP Connection(1)-169.254.5 | 🟨 | 0 ms | (0%) | 114.170 ms |
| 🟩 RMI TCP Accept-0 | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟩 reactor-http-nio-4 | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟩 reactor-http-nio-3 | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟩 reactor-http-nio-2 | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟧 server | 🟧 | 0 ms | (0%) | 114.170 ms |
| 🟩 reactor-http-nio-1 | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟩 Attach Listener | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟩 Signal Dispatcher | 🟩 | 114.170 ms | (100%) | 114.170 ms |
| 🟨 Finalizer | 🟨 | 0 ms | (0%) | 114.170 ms |
| 🟨 Reference Handler | 🟨 | 0 ms | (0%) | 114.170 ms |
| 🟨 main | 🟨 | 0 ms | (0%) | 114.170 ms |
| ⬜ RMI TCP Connection(3)-169.254.5 | 🟧 | 43.025 ms | (41,7%) | 103.160 ms |

Mudando pra Spring Webflux e o paradigma para reactive programming.

**Aumentamos de 13.000 r/m para 19.000 r/m**

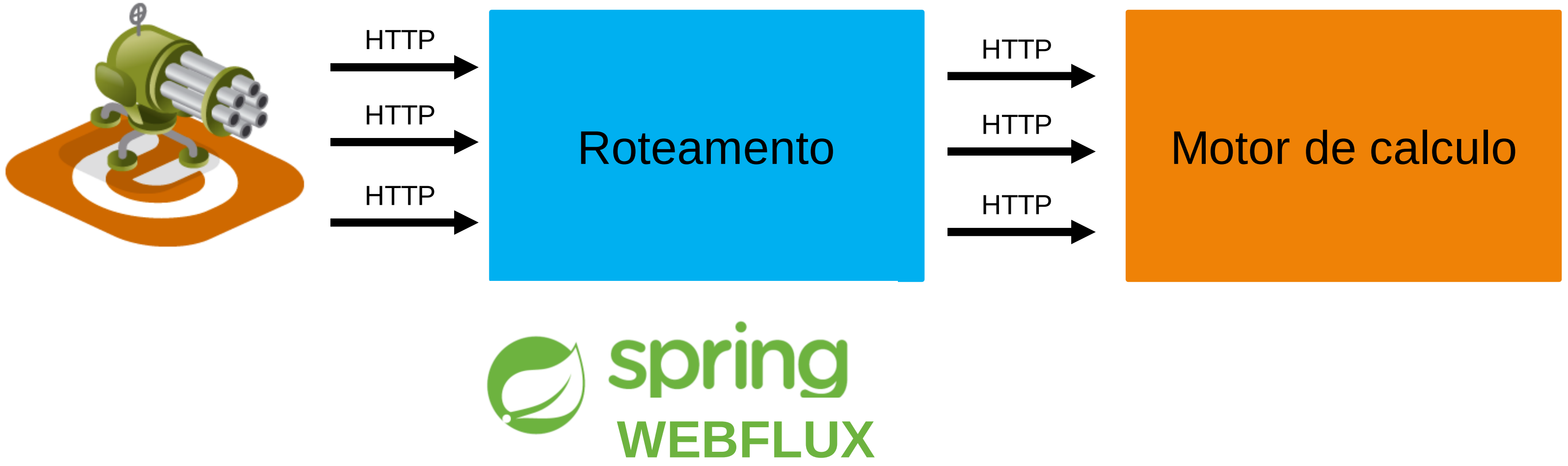
Nossa arquitetura é eficaz.

Nossa aplicação é eficaz.

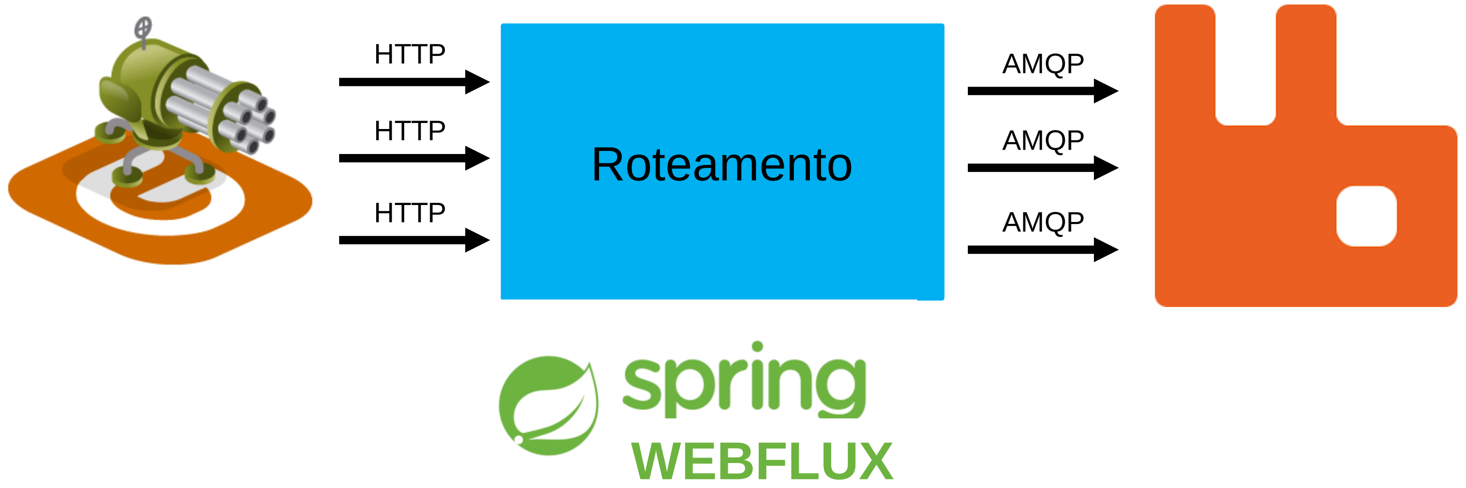
THE DEVELOPER'S CONFERENCE

## A integração dos microserviços é eficaz?

# HTTP


HTTP
HTTP
HTTP
Roteamento
HTTP
HTTP
HTTP
Motor de calculo
spring
WEBFLUX

## AMQP

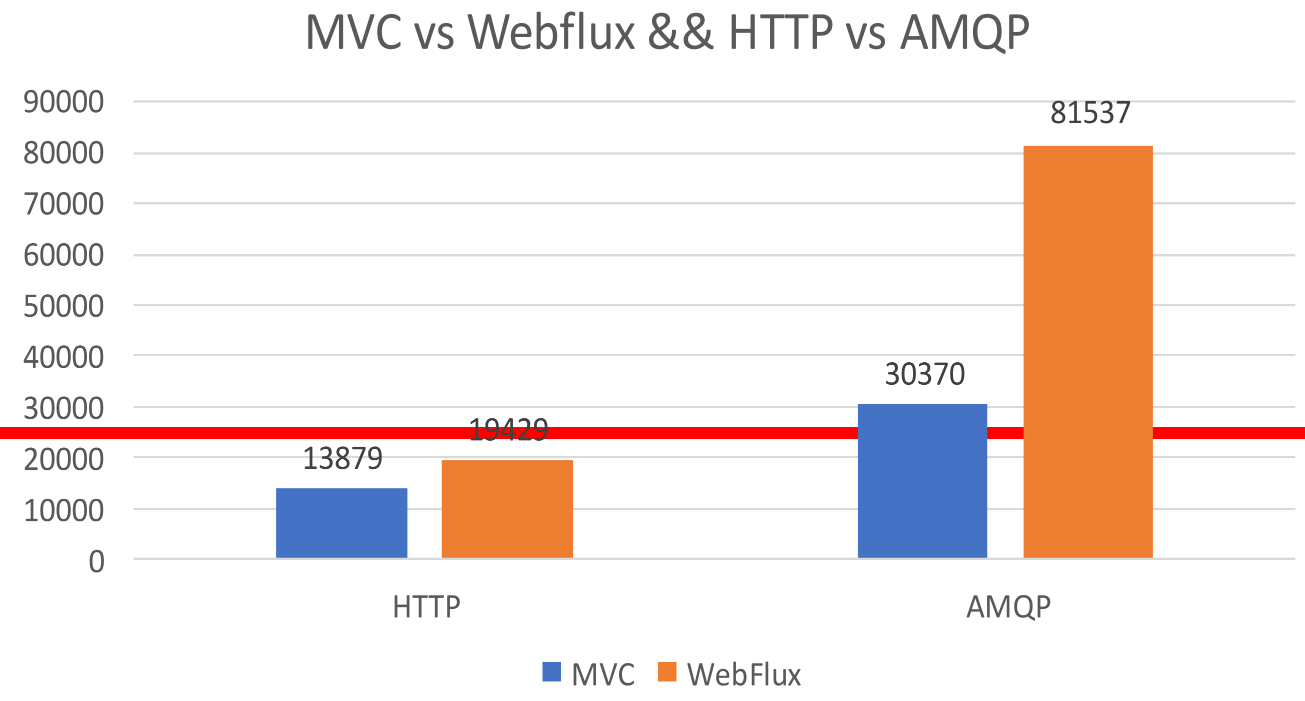


Adicionando um serviço de mensageria para a comunicação dos microserviços.

**Aumentamos de 19.000 r/m para 81.000 r/m**

## Concluindo

THE DEVELOPER'S CONFERENCE

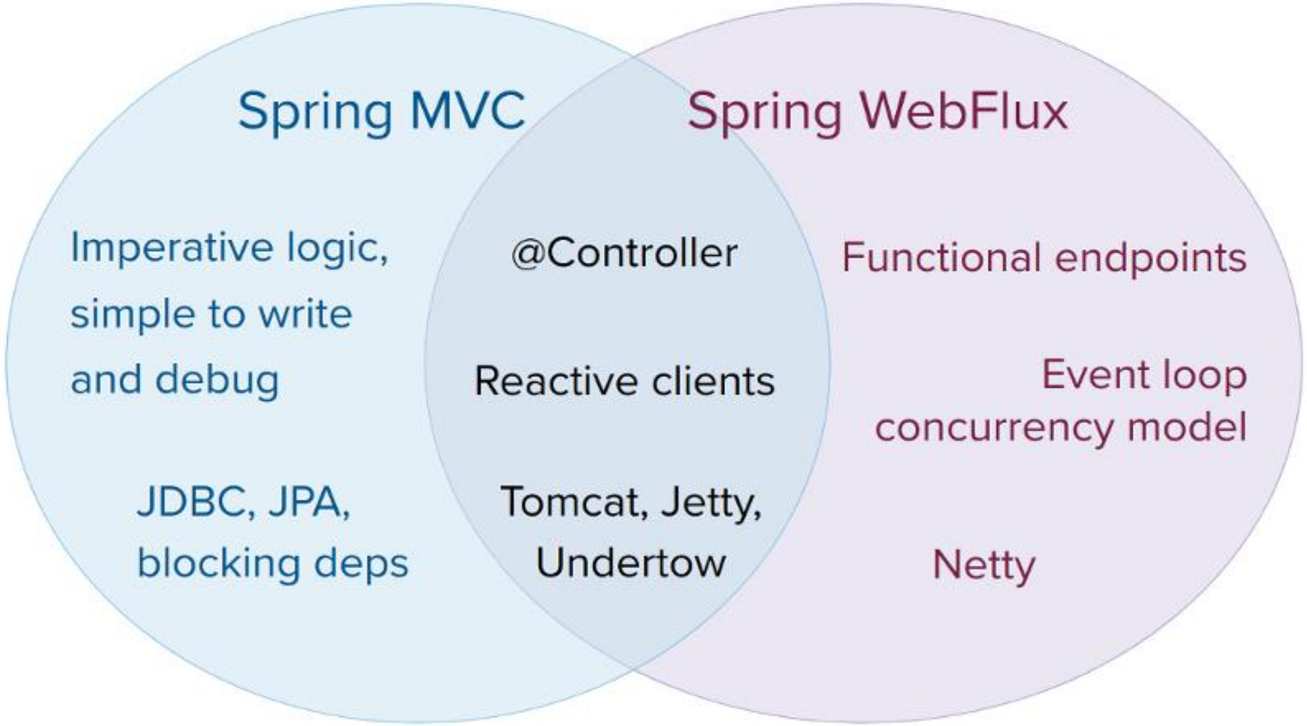


## Concluindo

THE DEVELOPER'S CONFERENCE

Realmente precisamos de Spring WebFlux?

# Concluindo


Spring MVC
Spring WebFlux
Imperative logic, simple to write and debug
JDBC, JPA, blocking deps
@Controller
Reactive clients
Tomcat, Jetty, Undertow
Functional endpoints
Event loop concurrency model
Netty

THE DEVELOPER'S CONFERENCE

renanzr@gmail.com

/RRoggia

Códigos e baselines
de mvc e webflux