

SPRING WEBFLUX VS. VIRTUAL THREADS: UMA BATALHA PELA SOBREVIVÊNCIA



SUMÁRIO

- O que é Spring Webflux?
- Por que o Virtual Threads pode vencer essa batalha?
- Conclusão e QA



O QUE É
SPRING WEBFLUX?

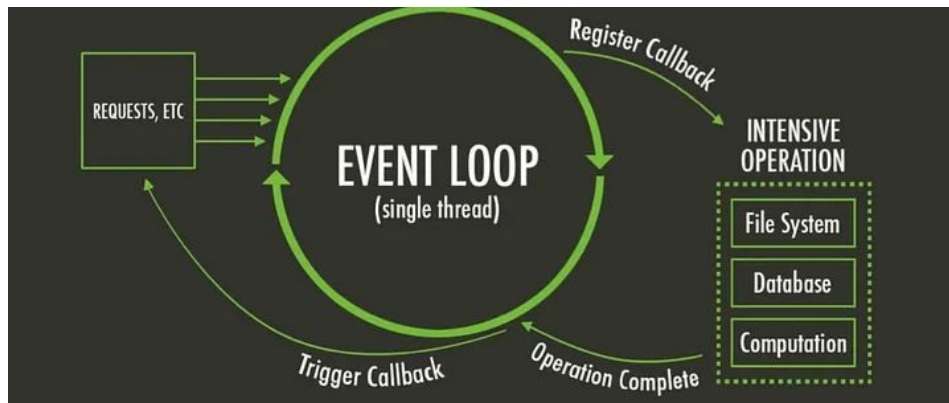
O QUE É?

- Framework Web
- Assíncrono
- IO Não-bloqueante
- Paradigma Reativo
- Usa Event-Loop



PARADIGMA REATIVO

- *Data-stream*
- Eventos



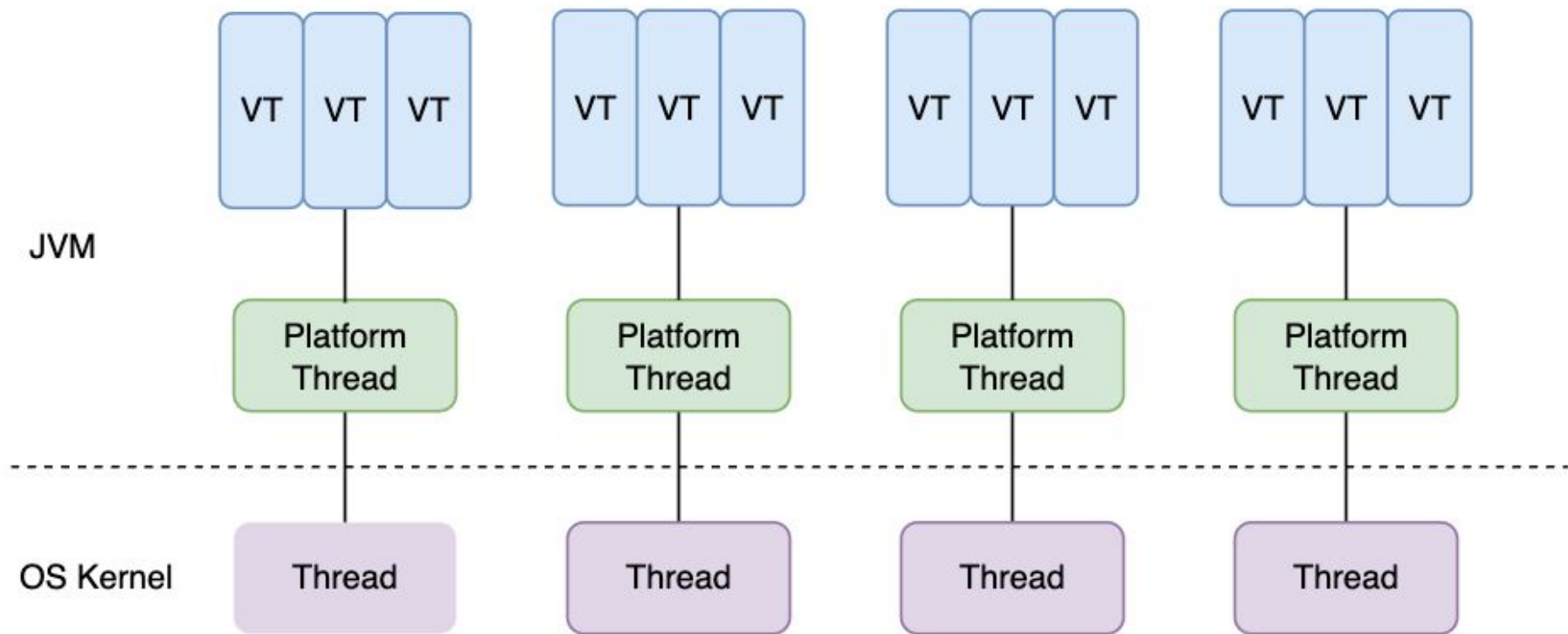
Fonte: <https://dev.to/khaosdoctor/node-js-por-baixo-dos-panos-3-um-mergulho-no-event-loop-38l9>

- Subscriber / Publisher / Subscription



**DAMN IT, THIS IS TOO
HARD!**

VIRTUAL THREADS



MILLHÕES DE THREADS!!!
DE GRAÇA!!!

AH! AGORA EU ENTENDI



**AGORA TUDO FAZ
SENTIDO**

POR QUE VIRTUAL THREADS PODE VENCER ESSA BATALHA?

- Agora com VT, a limitação no número de threads pro Spring MVC praticamente acabou!
- A idéia é conseguir a mesma escala do Webflux, porém com a facilidade do Spring MVC!



MAS SERÁ MESMO????

PARA PARA PARA PARA

memecrunch.com

corbis.

**SPRING
WEBFLUX**

**SPRING
MVC COM VT**

BENCHMARK

BENCHMARK

50 concurrent connections		
Measurement	SpringBoot (VT)	SpringBoot Webflux
Time taken (secs)	125.8 _s	82.6 _s
Requests per second	7948	12105
Minimum latency (ms)	0.61 _{ms}	0.43 _{ms}
10 pct. latency (ms)	5.47 _{ms}	3.08 _{ms}
25 pct. latency (ms)	5.68 _{ms}	3.22 _{ms}
Median latency (ms)	5.89 _{ms}	3.6 _{ms}
Mean latency (ms)	6.29 _{ms}	4.13 _{ms}
75 pct. latency (ms)	6.14 _{ms}	4.76 _{ms}
90 pct. latency (ms)	6.74 _{ms}	5.72 _{ms}
99 pct. latency (ms)	11.82 _{ms}	8.99 _{ms}
Maximum latency (ms)	3319.79 _{ms}	96.45 _{ms}
Avg. CPU usage (%)	295 %	401 %
Avg. Memory usage (MB)	545 MB	661 MB

100 concurrent connections		
Measurement	SpringBoot (VT)	SpringBoot Webflux
Time taken (secs)	129.3 _s	80.6 _s
Requests per second	7734	12406
Minimum latency (ms)	0.55 _{ms}	0.46 _{ms}
10 pct. latency (ms)	11.77 _{ms}	6.71 _{ms}
25 pct. latency (ms)	12.03 _{ms}	7.1 _{ms}
Median latency (ms)	12.32 _{ms}	7.44 _{ms}
Mean latency (ms)	12.93 _{ms}	8.06 _{ms}
75 pct. latency (ms)	12.71 _{ms}	8.58 _{ms}
90 pct. latency (ms)	13.63 _{ms}	9.67 _{ms}
99 pct. latency (ms)	21.61 _{ms}	17.04 _{ms}
Maximum latency (ms)	3290.81 _{ms}	125.28 _{ms}
Avg. CPU usage (%)	300 %	389 %
Avg. Memory usage (MB)	548 MB	689 MB

300 concurrent connections		
Measurement	SpringBoot (VT)	SpringBoot Webflux
Time taken (secs)	125.6 _s	79.8 _s
Requests per second	7964	12535
Minimum latency (ms)	0.6 _{ms}	1.15 _{ms}
10 pct. latency (ms)	35.01 _{ms}	20.95 _{ms}
25 pct. latency (ms)	35.58 _{ms}	21.98 _{ms}
Median latency (ms)	36.14 _{ms}	22.94 _{ms}
Mean latency (ms)	37.66 _{ms}	23.94 _{ms}
75 pct. latency (ms)	36.9 _{ms}	24.36 _{ms}
90 pct. latency (ms)	41.72 _{ms}	26.51 _{ms}
99 pct. latency (ms)	63.56 _{ms}	51.94 _{ms}
Maximum latency (ms)	4133.15 _{ms}	192.47 _{ms}
Avg. CPU usage (%)	294 %	391 %
Avg. Memory usage (MB)	567 MB	920 MB

<https://medium.com/deno-the-complete-reference/springboot-virtual-threads-vs-webflux-performance-comparison-for-jwt-verify-and-mysql-query-ff94cf251c2c>

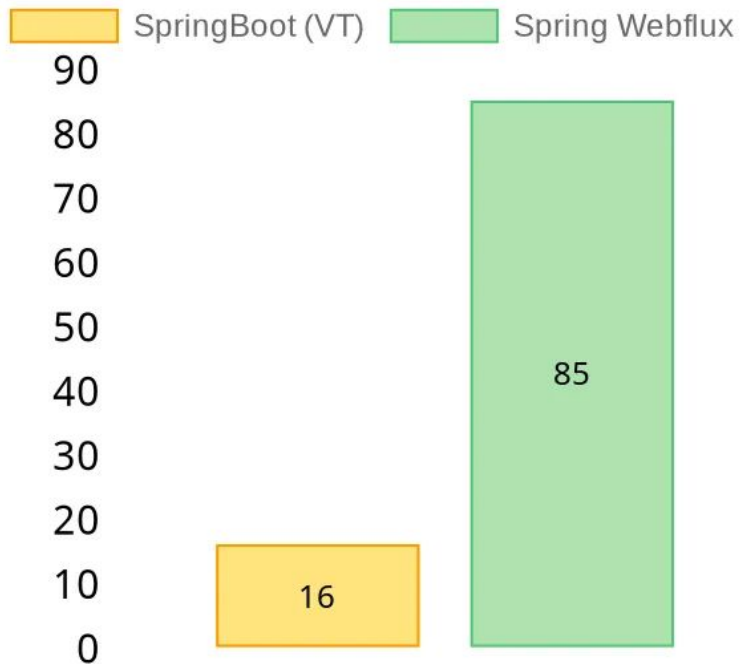
BENCHMARK

A scorecard is generated from the results using the following formula. For each measurement, get the winning margin. If the winning margin is:

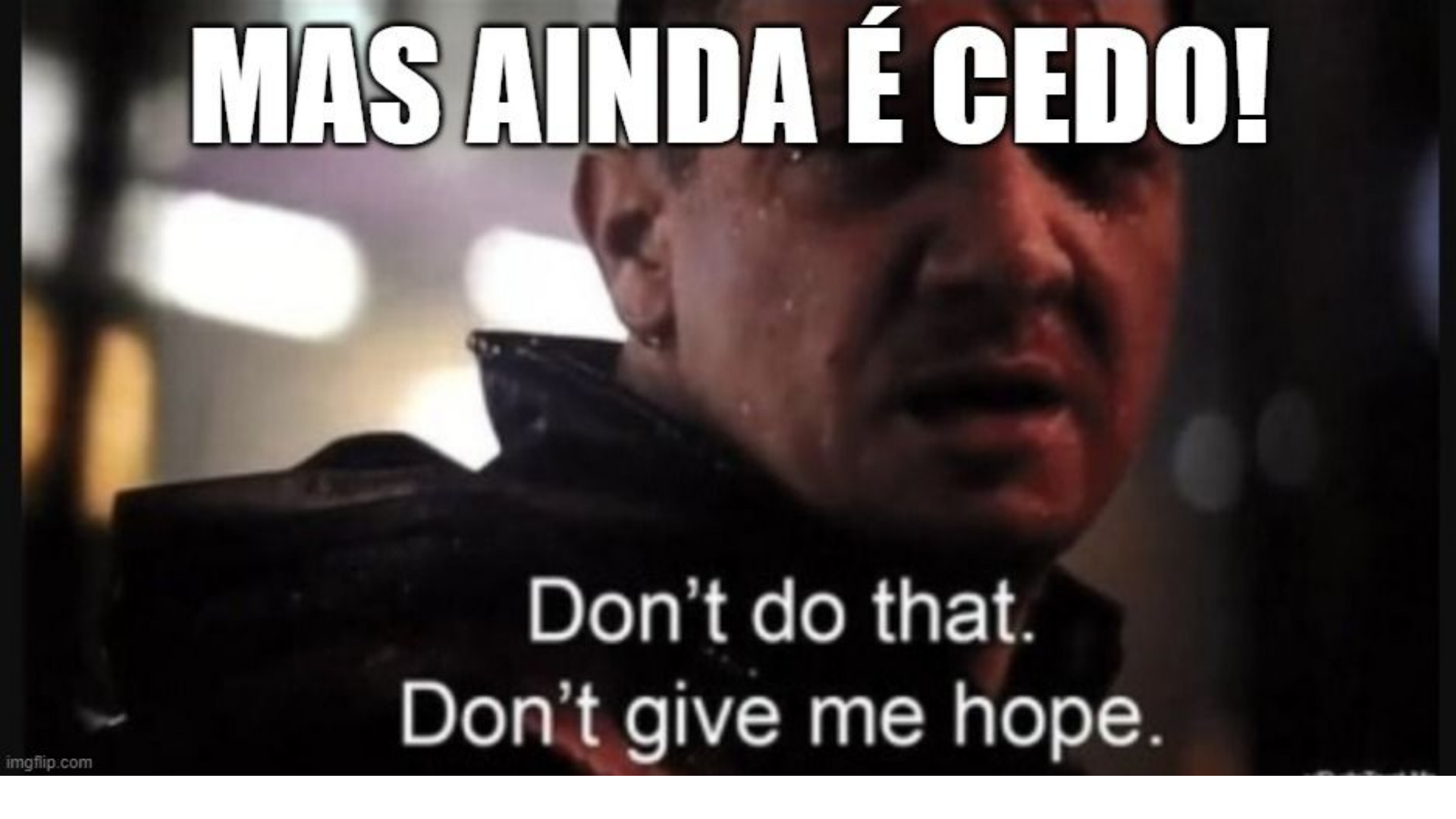
- $< 5\%$, no points are given
- between 5% and 20%, 1 point is given to the winner
- between 20% and 50%, 2 points are given to the winner
- $> 50\%$, 3 points are given to the winner

BENCHMARK

Scorecard



On an average (RPS)
SpringBoot Webflux
is **57%** faster than
SpringBoot (VT)



MAS AINDA É CEDO!

Don't do that.
Don't give me hope.

Spring Boot 3.2.0 available now

RELEASES | MORITZ HALBRITTER | NOVEMBER 23, 2023 | 0 COMMENTS

On behalf of the Spring Boot team and everyone that has contributed, I am pleased to announce that Spring Boot 3.2.0 has been released and is available from Maven Central.

This release adds a significant number of new features and improvements. For full [upgrade instructions](#) and [new and noteworthy features](#) please see the [release notes](#).

What's new in 3.2

The highlights of the 3.2 release include:

- 
- Support for Virtual Threads
 - Initial support for JVM Checkpoint Restore (Project CRaC)
 - SSL Bundle Reloading
 - A lot of Observability Improvements
 - Support for RestClient
 - Support for JdbcClient
 - Support for Jetty 12
 - Support for Spring for Apache Pulsar
 - SSL bundle support for Kafka and RabbitMQ
 - Reworked Nested Jar handling
 - Docker Image Building improvements

OBRIGADO!!



Pedro Cavaleiro
Staff Software Engineer
International Career Mentor
<https://pedrocavaleiro.com>



Cristiano Gregio
Senior Software Engineer
Java Specialist
[linkedin.com/in/cristiano-gregio](https://www.linkedin.com/in/cristiano-gregio)