Beyond the spinning wheel

Building robust and resilient APIs

About me

- 15+ years of experience of building software
- Principal software engineer and trainer @ Sorsix 8+ years
- 8+ years of experience in Academia as TA
- I like building software (web, mobile, desktop):
  - PHP, APS.NET, Python (Django), Java (Play Framework)
  - Spring/Kotlin with Angular/TypeScript
- I like learning programming languages
  - Scala, Haskell, Elixir
  - Rust, Go

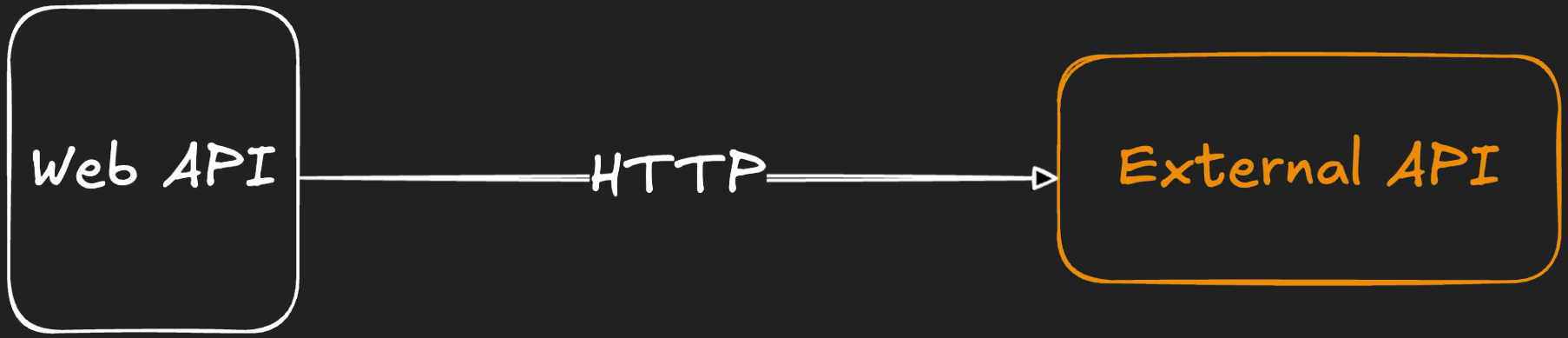
This talk is NOT about

- Scalability
- Redundancy

It is about how the performance, stability and resilience of a single server are affected by some of the most common stability anti-patterns.

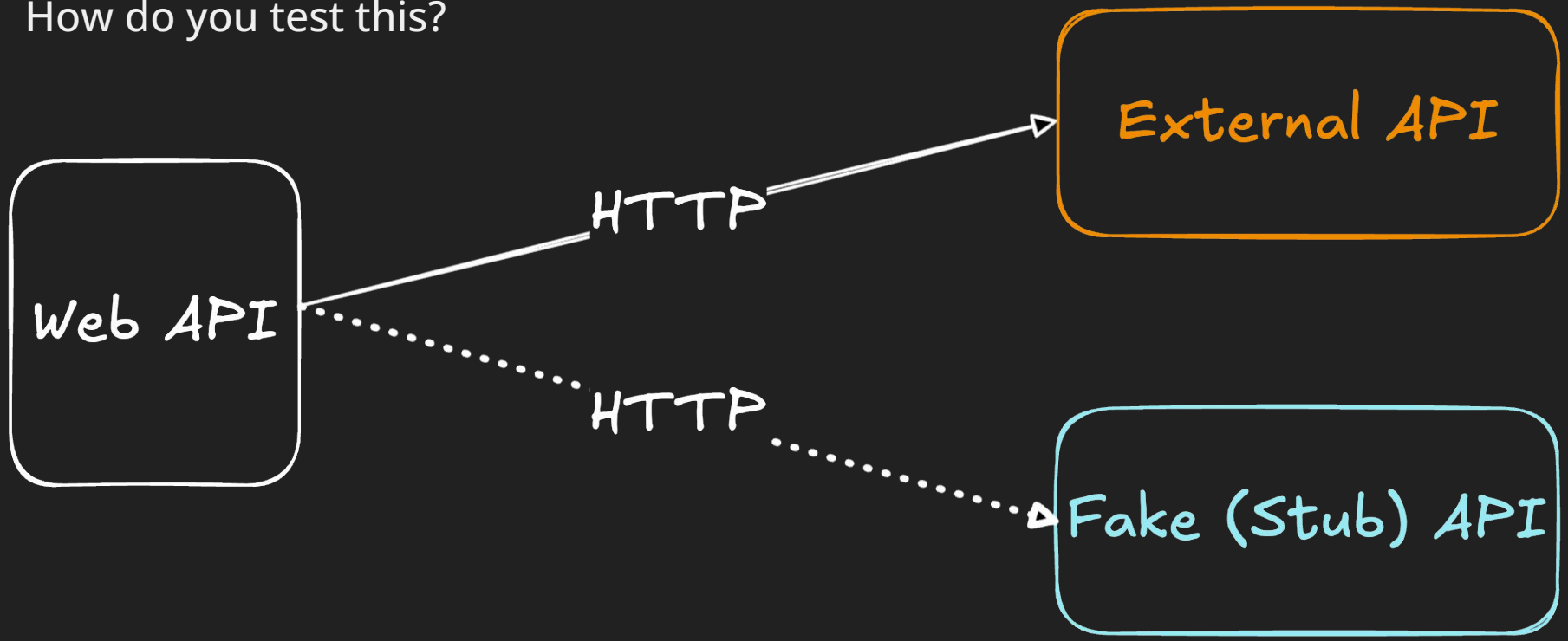
The story...

Based on true events



The story...

How do you test this?



The story...

Everything is working, everything is great...

You deploy to PRODUCTION.

The story...

And, then one day, the phone rings (you get a ticket).

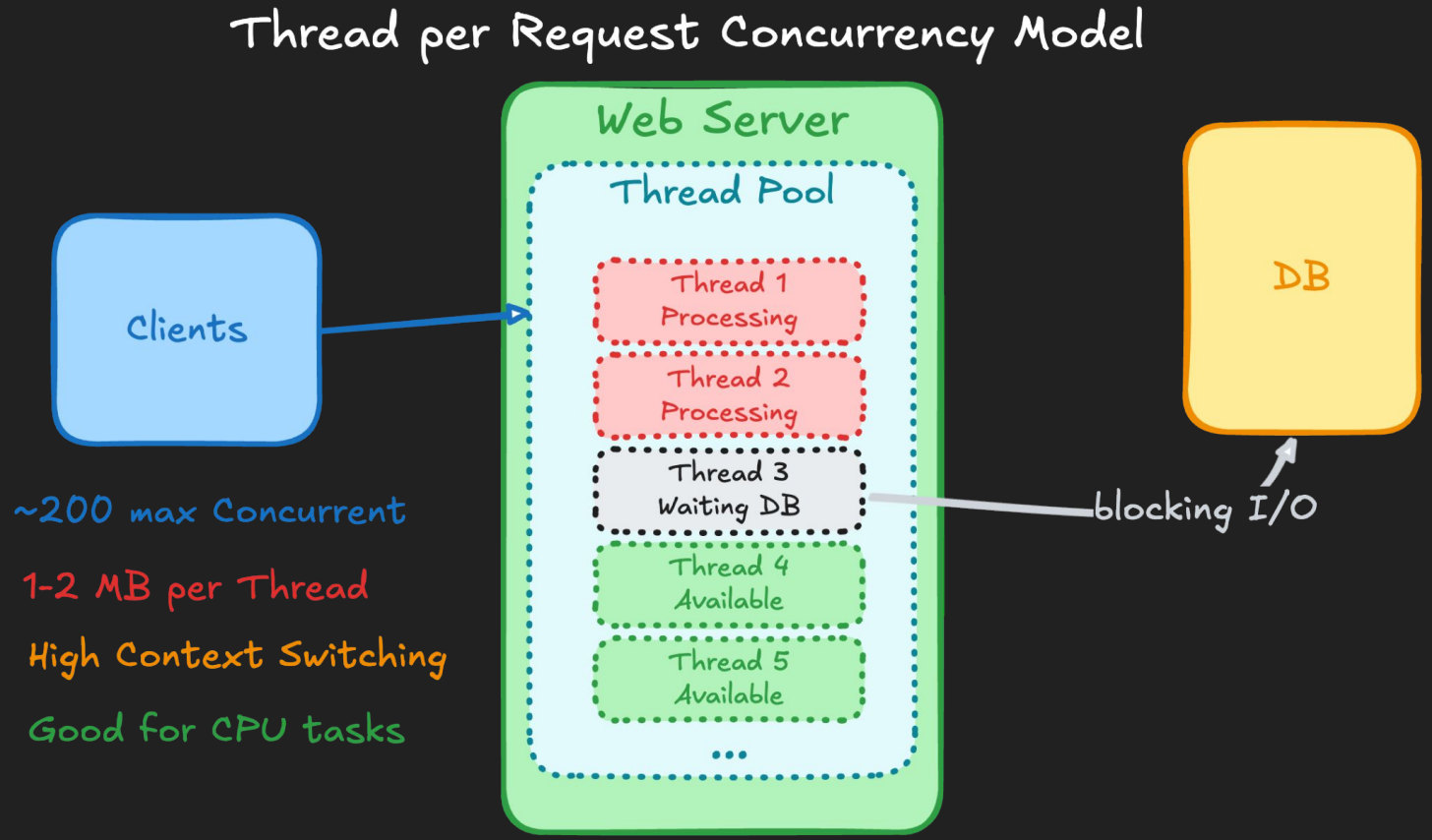
The ticket is:

“the app is not working, the users are seeing a spinning wheel???!!!”

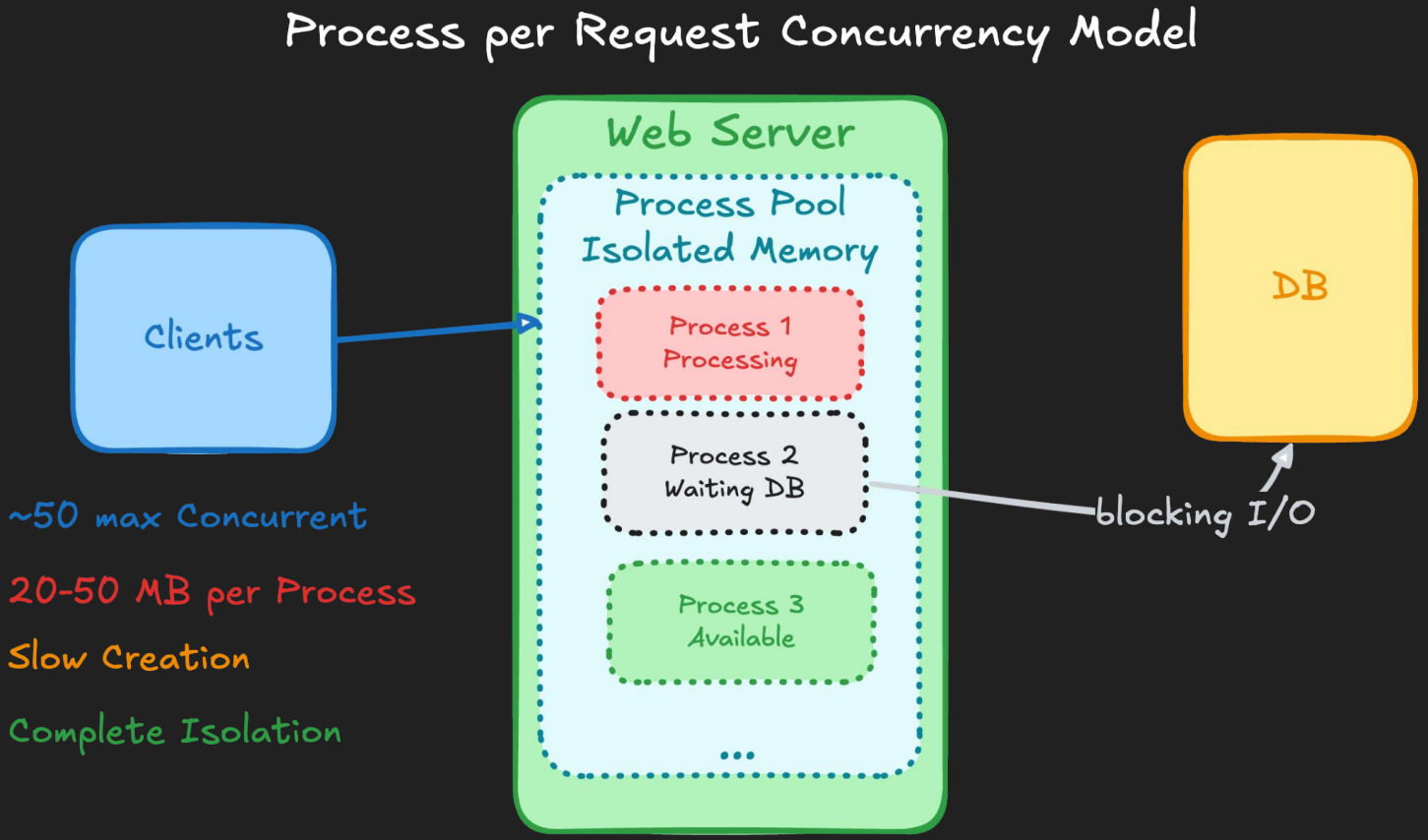
Web servers concurrency

- Web applications can be used by many users at the same time
- Web frameworks handle with concurrency using one of the three different models

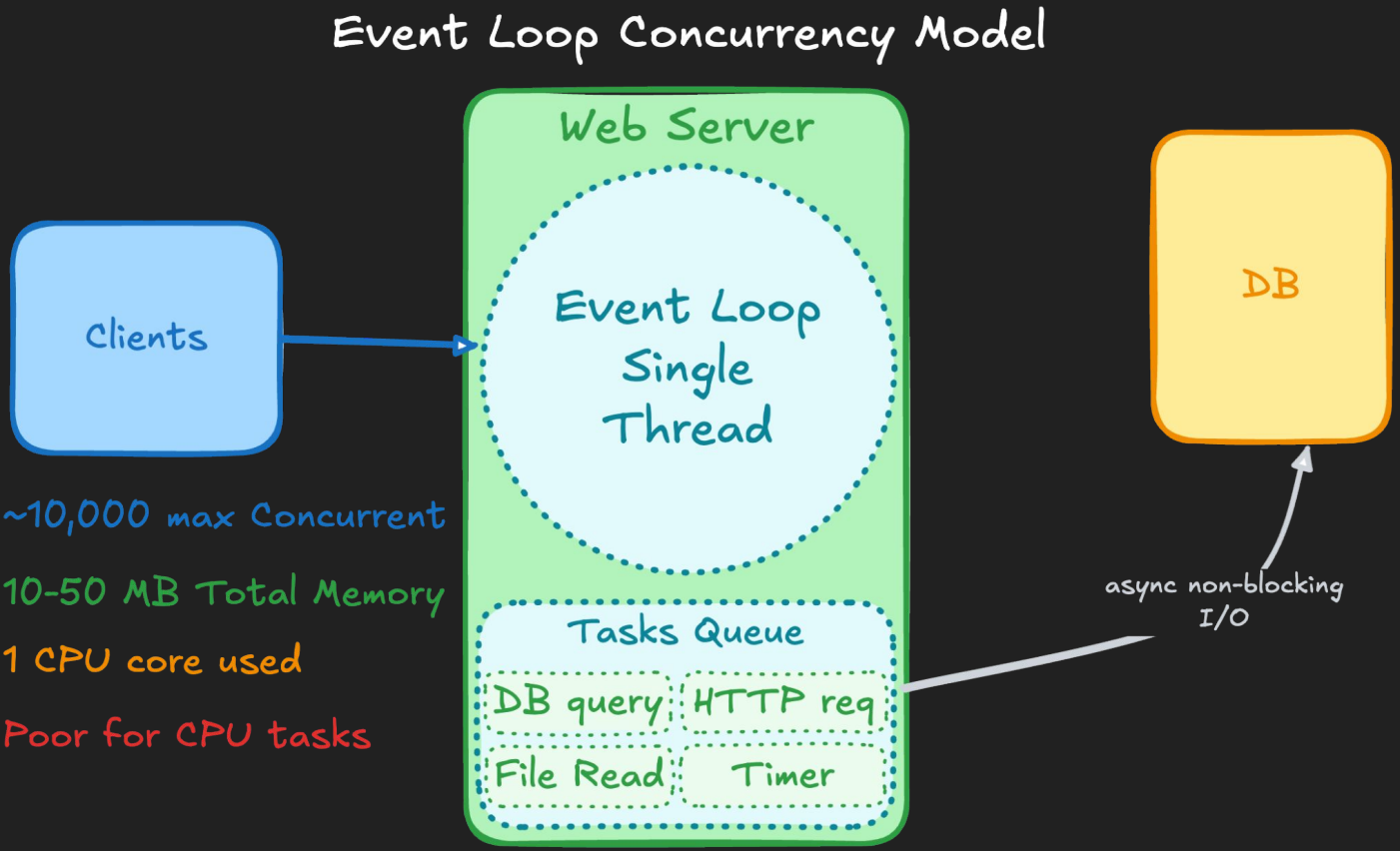
(1/3) Web servers concurrency models



(2/3) Web servers concurrency models



(3/3) Web servers concurrency models



What went wrong?

- To find the root cause, we must reproduce the failing scenario
- How to do this?
- Test harness
  - Component isolation
  - Simulated environment
  - Test execution (load testing)
  - Result analysis
- Gatling
  - Open workload model
    https://gatling.io/blog/workload-models-in-load-testing#the-open-workload-model

Case 1

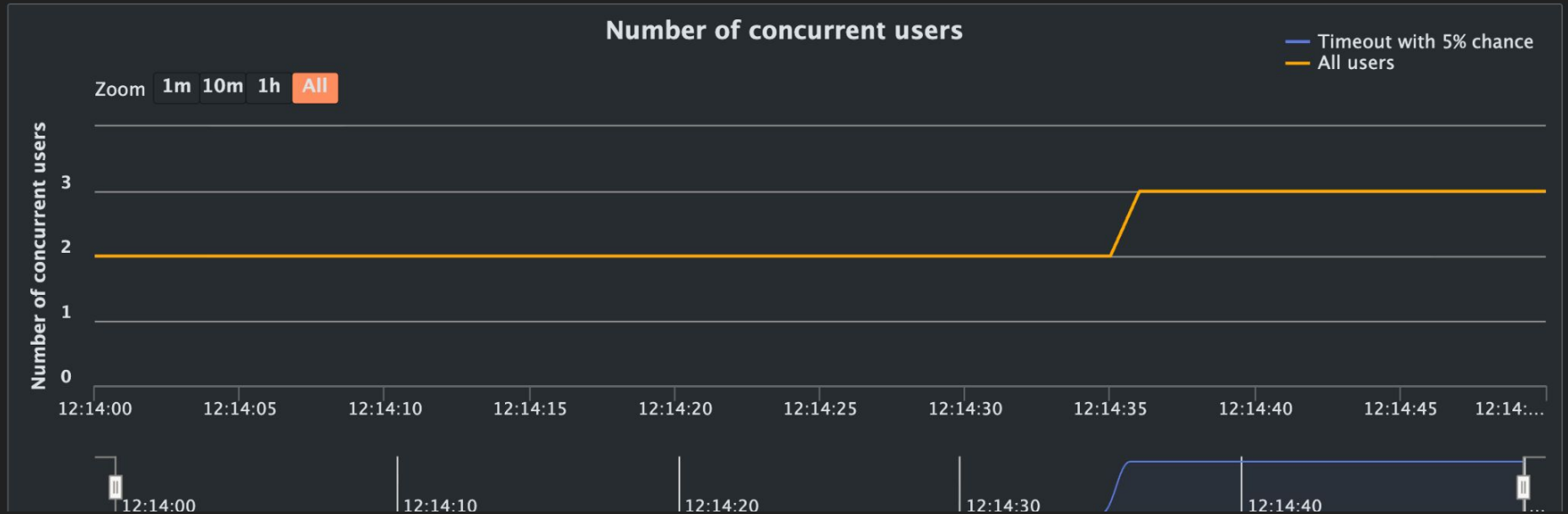
Stats

Executions

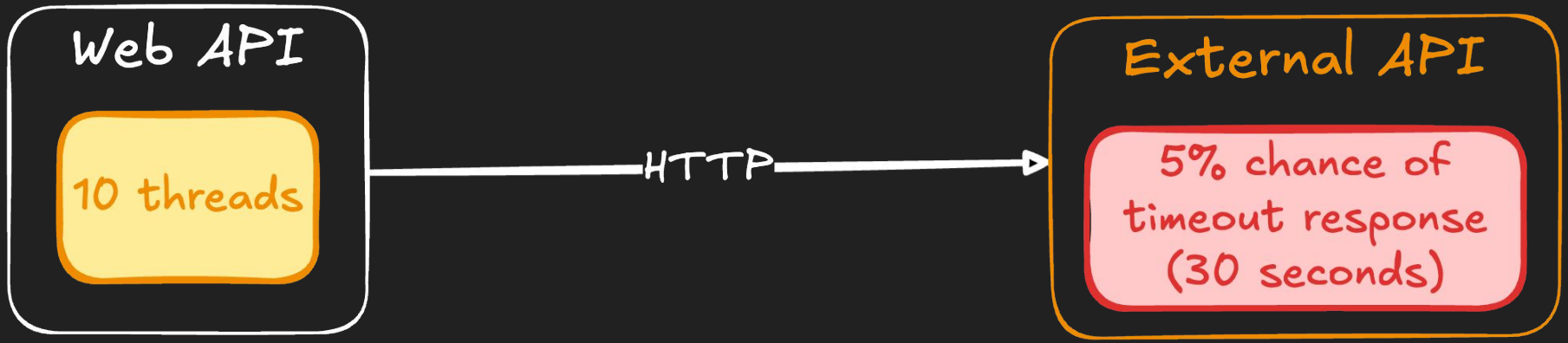
	Total	OK	KO
Total count	750	750	-
Mean count/s	14.71	14.71	-

Response Time (ms)

	Total	OK	KO
Min	2	2	-
50th percentile	57	57	-
75th percentile	74	74	-
95th percentile	98	98	-
99th percentile	118	118	-
Max	152	152	-
Mean	14.71	14.71	-
Standard Deviation	25	25	-


Number of concurrent users
Timeout with 5% chance
All users
Zoom
1m
10m
1h
All
Number of concurrent users
3
2
1
0
12:14:00
12:14:05
12:14:10
12:14:15
12:14:20
12:14:25
12:14:30
12:14:35
12:14:40
12:14:45
12:14:00
12:14:10
12:14:20
12:14:30
12:14:40

Case 2


Web API
10 threads
HTTP
External API
5% chance of timeout response (30 seconds)

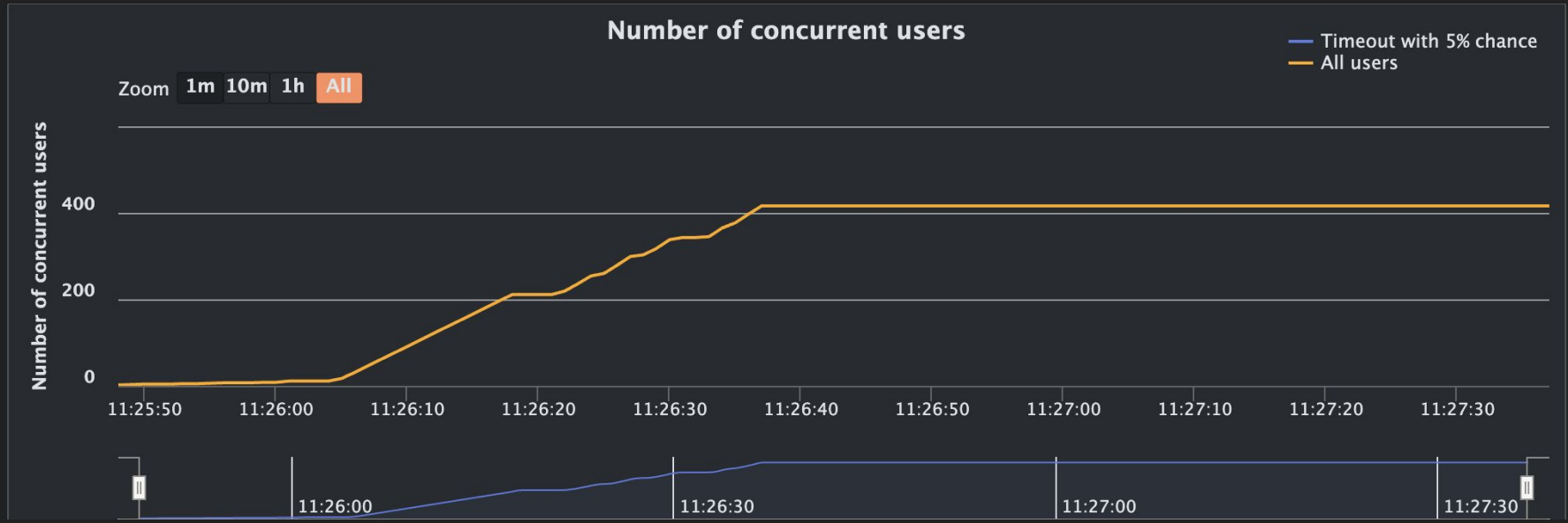
Stats

Executions

	Total	OK	KO
Total count	750	557	193
Mean count/s	6.82	5.06	1.75

Response Time (ms)

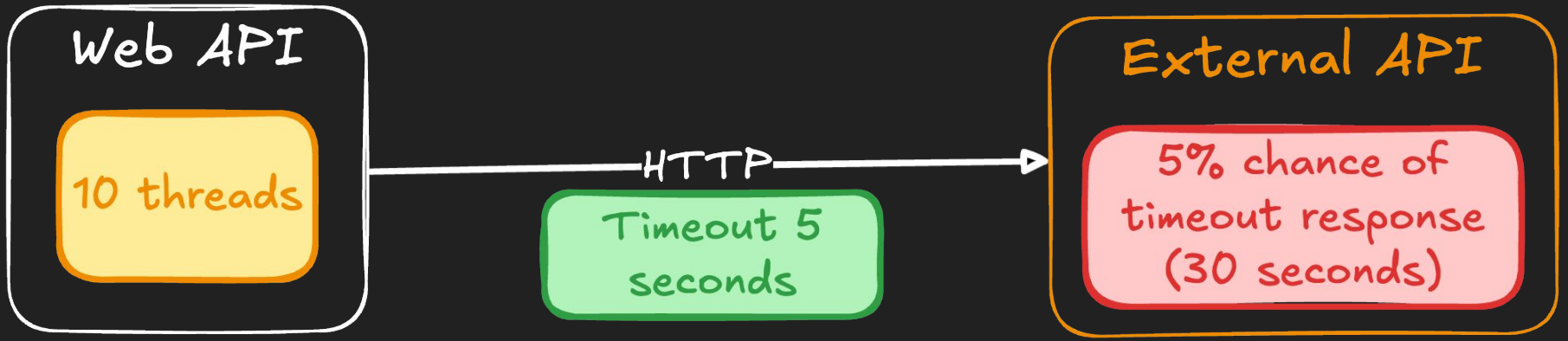
	Total	OK	KO
Min	4	4	60000
50th percentile	35789	18030	60001
75th percentile	58578	38212	60001
95th percentile	60001	56102	60002
99th percentile	60003	58029	60002
Max	60004	58068	60004
Mean	6.82	5.06	1.75
Standard Deviation	23825	20039	1


Number of concurrent users
Timeout with 5% chance
All users
Zoom
1m
10m
1h
All
Number of concurrent users
400
200
0
11:25:50
11:26:00
11:26:10
11:26:20
11:26:30
11:26:40
11:26:50
11:27:00
11:27:10
11:27:20
11:27:30
11:26:00
11:26:30
11:27:00
11:27:30

Timeout

- Timeout is one of the most basic but essential stability pattern.
- You should never wait indefinitely for anything:
  - HTTP calls
  - Email service (SMTP)
  - Resources synchronization (locking)
- HTTP clients do not have default timeout

Timeout


Web API
10 threads
HTTP
Timeout 5 seconds
External API
5% chance of timeout response (30 seconds)

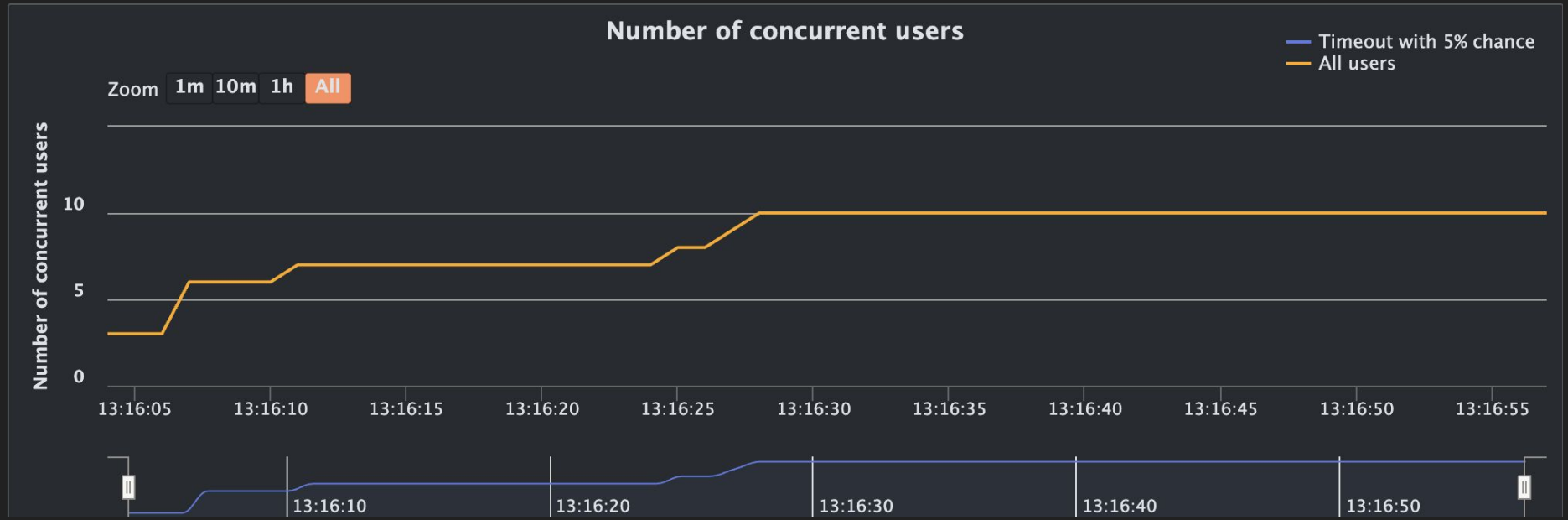
Stats

Executions

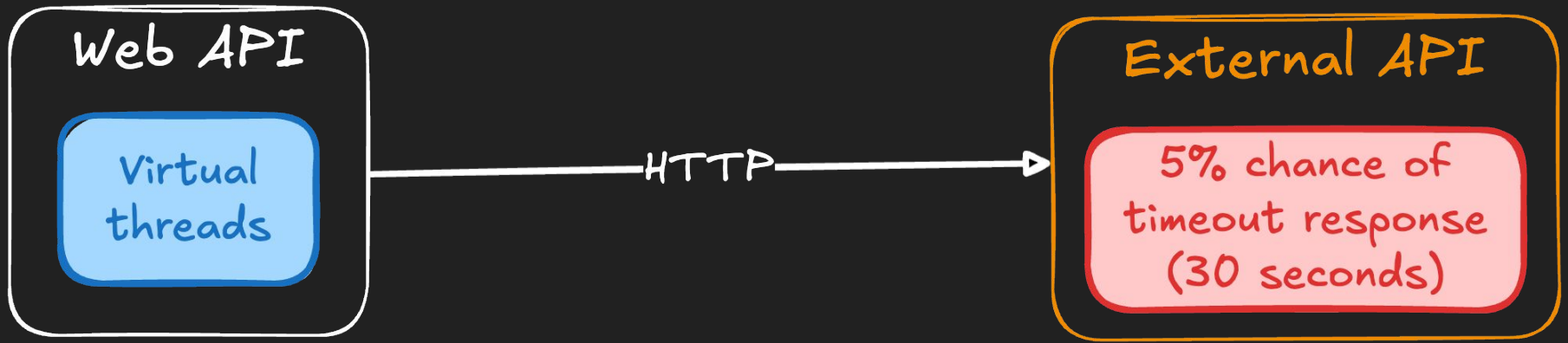
	Total	OK	KO
Total count	750	750	-
Mean count/s	13.89	13.89	-

Response Time (ms)

	Total	OK	KO
Min	2	2	-
50th percentile	61	61	-
75th percentile	81	81	-
95th percentile	1222	1222	-
99th percentile	5012	5012	-
Max	5023	5023	-
Mean	13.89	13.89	-
Standard Deviation	1099	1099	-


Number of concurrent users
Timeout with 5% chance
All users
Zoom
1m
10m
1h
All
Number of concurrent users
10
5
0
13:16:05
13:16:10
13:16:15
13:16:20
13:16:25
13:16:30
13:16:35
13:16:40
13:16:45
13:16:50
13:16:55
13:16:10
13:16:20
13:16:30
13:16:40
13:16:50

Other “solutions”



10 threads

Stats

Executions

	Total	OK	KO
Total count	750	557	193
Mean count/s	6.82	5.06	1.75

Response Time (ms)

	Total	OK	KO
Min	4	4	60000
50th percentile	35789	18030	60001
75th percentile	58578	38212	60001
95th percentile	60001	56102	60002
99th percentile	60003	58029	60002
Max	60004	58068	60004
Mean	6.82	5.06	1.75
Standard Deviation	23825	20039	1

Virtual threads (event loop)

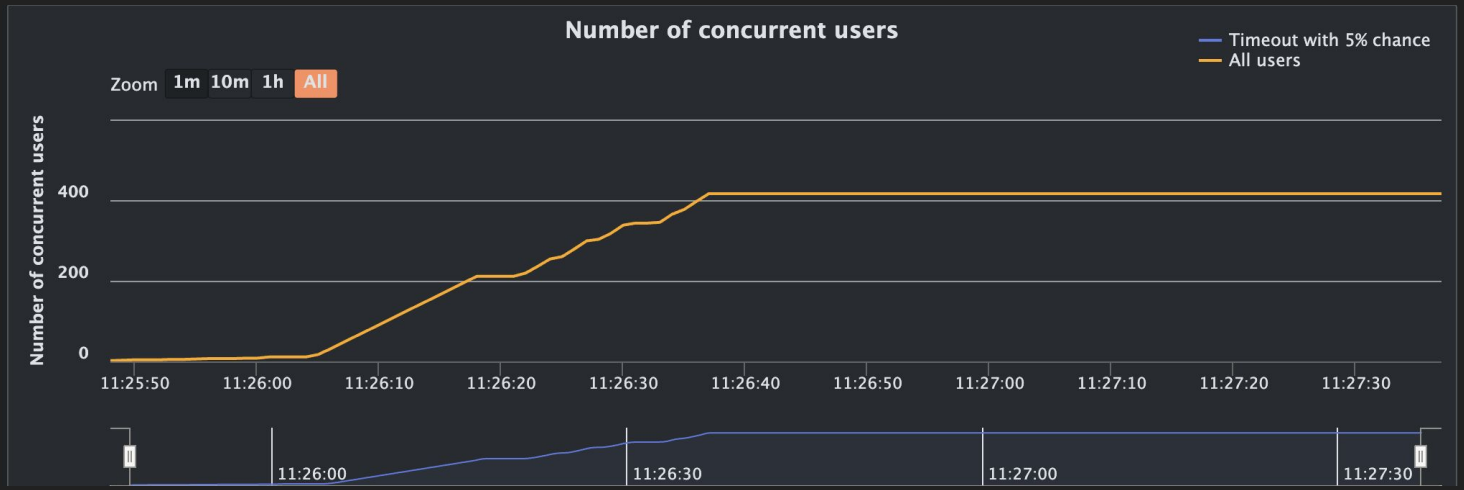
Stats

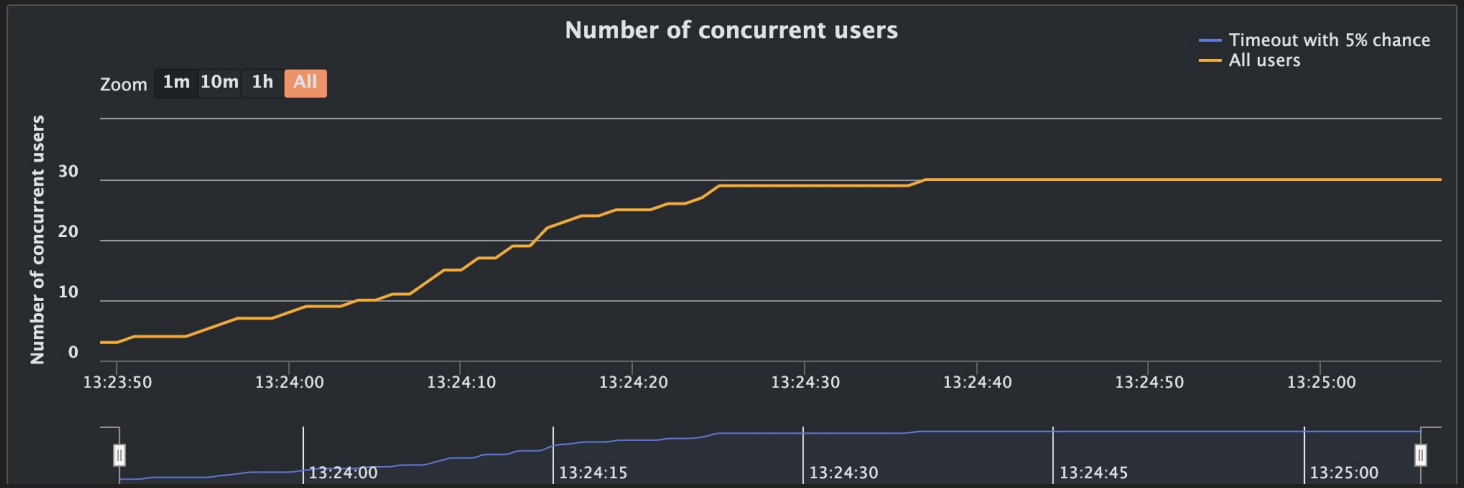
Executions

	Total	OK	KO
Total count	750	750	-
Mean count/s	9.49	9.49	-

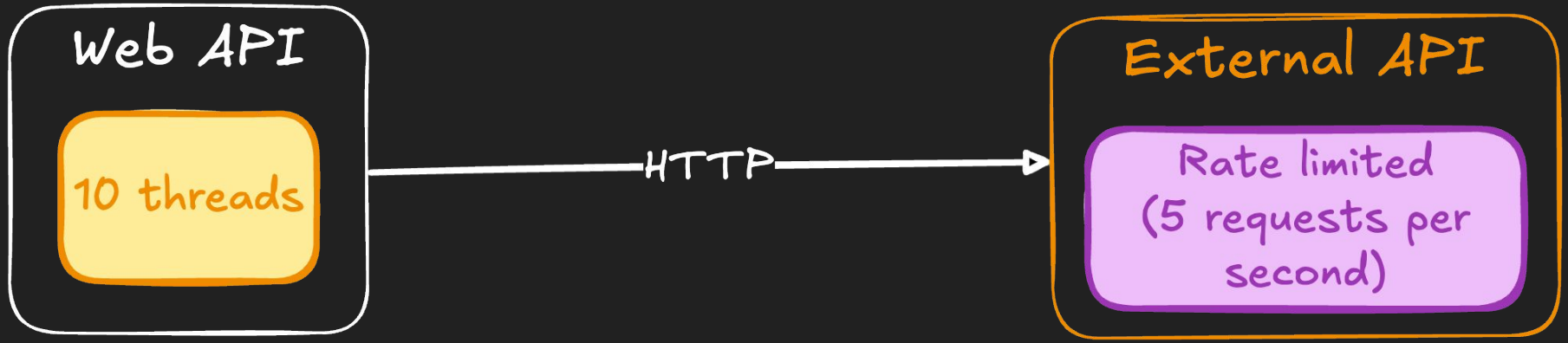
Response Time (ms)

	Total	OK	KO
Min	2	2	-
50th percentile	58	58	-
75th percentile	77	77	-
95th percentile	7472	5246	-
99th percentile	30017	29982	-
Max	30026	30026	-
Mean	9.49	9.49	-
Standard Deviation	6488	6488	-


Number of concurrent users
Timeout with 5% chance
All users
Zoom 1m 10m 1h All
Number of concurrent users
400
200
0
11:25:50
11:26:00
11:26:10
11:26:20
11:26:30
11:26:40
11:26:50
11:27:00
11:27:10
11:27:20
11:27:30
11:26:00
11:26:30
11:27:00
11:27:30


Number of concurrent users
Timeout with 5% chance
All users
Zoom 1m 10m 1h All
Number of concurrent users
30
20
10
0
13:23:50
13:24:00
13:24:10
13:24:20
13:24:30
13:24:40
13:24:50
13:25:00
13:24:00
13:24:15
13:24:30
13:24:45
13:25:00

Unbalanced capabilities



The external API is not returning an error (429 Too Many Requests) when the rate limit is exceeded, instead it was slowing down.

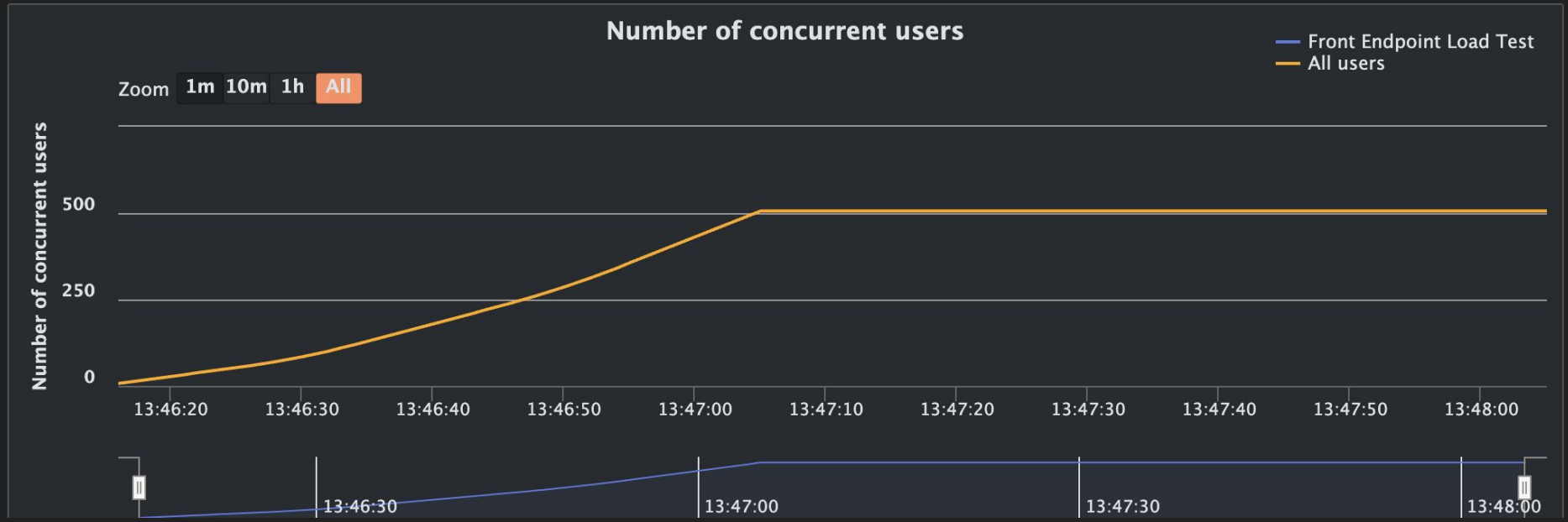
Stats

Executions

	Total	OK	KO
Total count	750	468	282
Mean count/s	6.82	4.25	2.56

Response Time (ms)

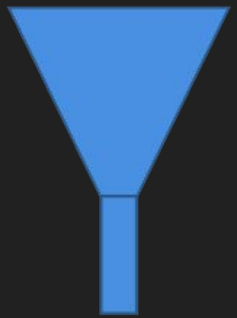
	Total	OK	KO
Min	290	290	60000
50th percentile	46705	27345	60001
75th percentile	60001	43398	60001
95th percentile	60002	56507	60002
99th percentile	60002	59288	60002
Max	60009	59880	60009
Mean	6.82	4.25	2.56
Standard Deviation	20698	17491	1


Number of concurrent users
Front Endpoint Load Test
All users
Zoom
1m
10m
1h
All
Number of concurrent users
500
250
0
13:46:20
13:46:30
13:46:40
13:46:50
13:47:00
13:47:10
13:47:20
13:47:30
13:47:40
13:47:50
13:48:00
13:46:30
13:47:00
13:47:30
13:48:00

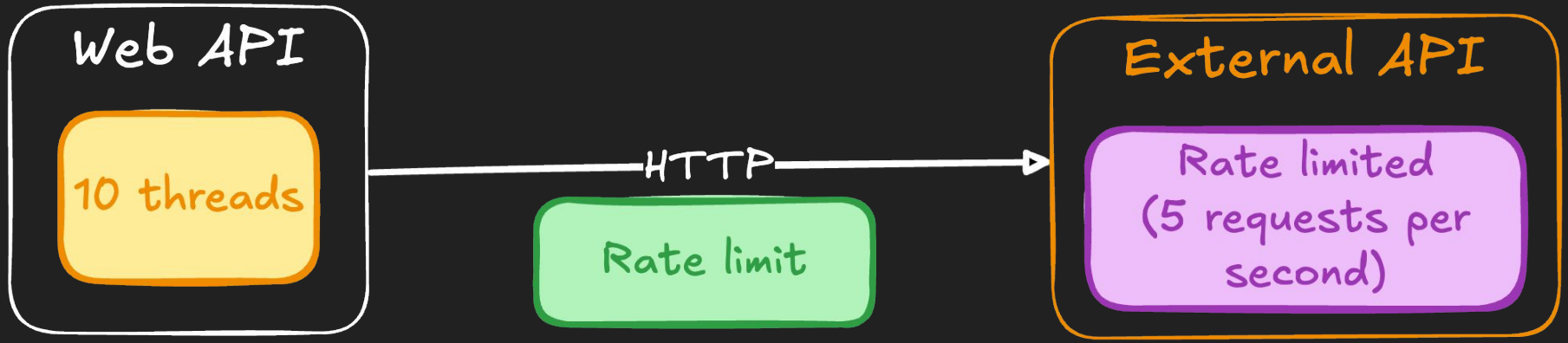
Fail fast

Enforcing a rate limiter on the calling site.

A **rate limiter** is a system component or mechanism that controls **the number of requests** a client can make to a server **within a specific time frame**.



Rate limiter on the calling side



Without rate limiting

Stats

Executions

	Total	OK	KO
Total count	750	468	282
Mean count/s	6.82	4.25	2.56

Response Time (ms)

	Total	OK	KO
Min	290	290	60000
50th percentile	46705	27345	60001
75th percentile	60001	43398	60001
95th percentile	60002	56507	60002
99th percentile	60002	59288	60002
Max	60009	59880	60009
Mean	6.82	4.25	2.56
Standard Deviation	20698	17491	1

With rate limiting

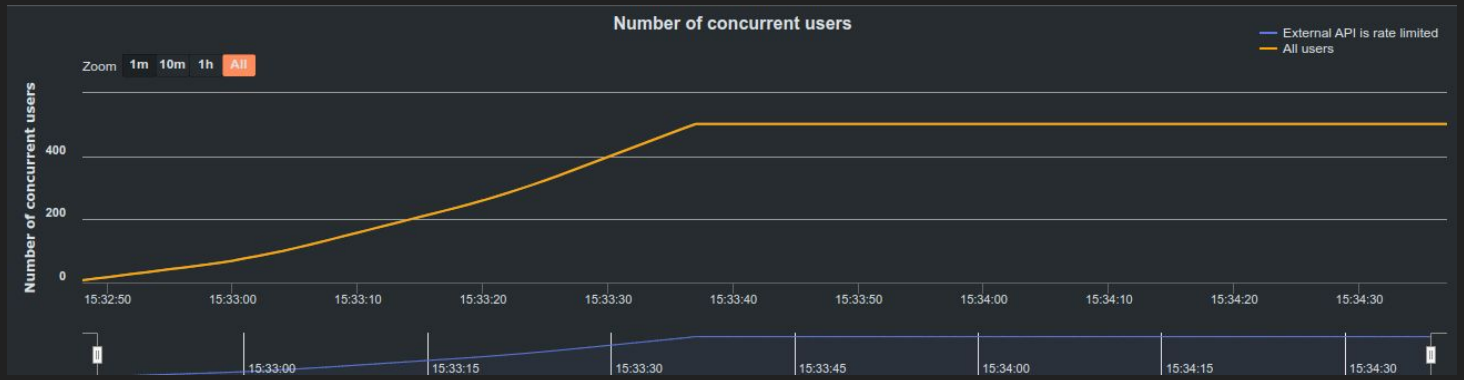
Stats

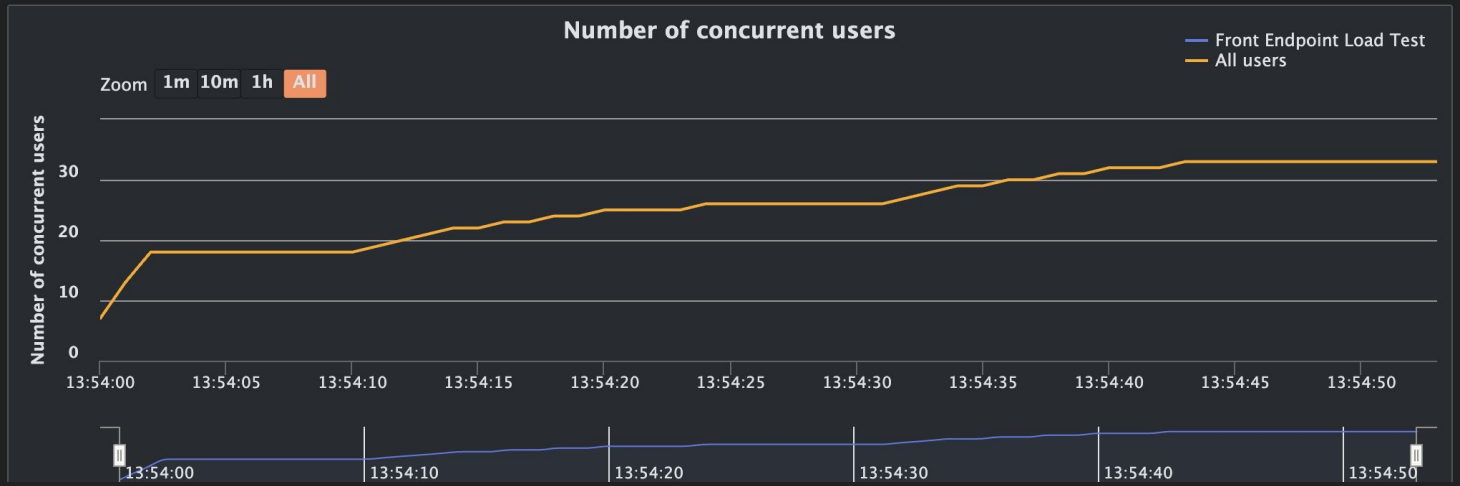
Executions

	Total	OK	KO
Total count	750	260	490
Mean count/s	13.89	4.81	9.07

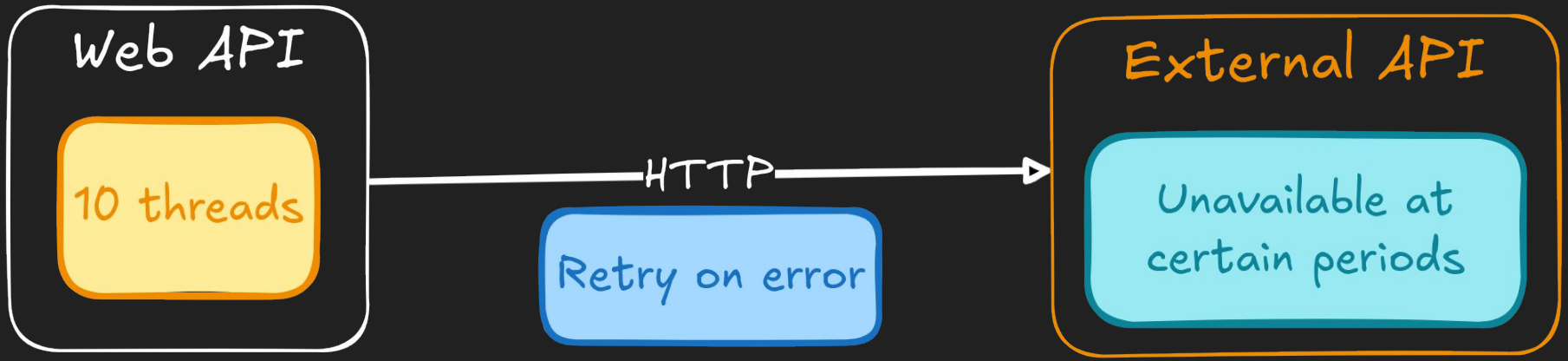
Response Time (ms)

	Total	OK	KO
Min	306	306	1002
50th percentile	1008	2644	1007
75th percentile	2345	2951	1008
95th percentile	3140	3337	1009
99th percentile	3499	3530	1010
Max	3608	3608	1013
Mean	13.89	4.81	9.07
Standard Deviation	813	557	2

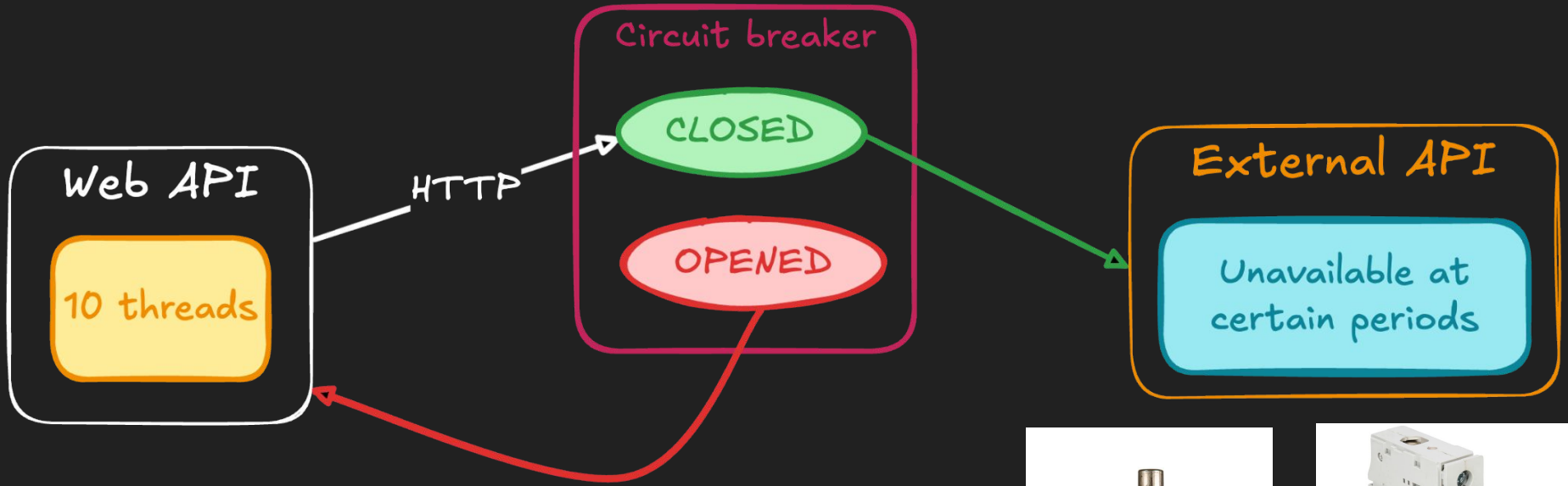

Number of concurrent users
External API is rate limited
All users
Zoom 1m 10m 1h All
Number of concurrent users
400
200
0
15:32:50
15:33:00
15:33:10
15:33:20
15:33:30
15:33:40
15:33:50
15:34:00
15:34:10
15:34:20
15:34:30
15:33:00
15:33:15
15:33:30
15:33:45
15:34:00
15:34:15
15:34:30


Number of concurrent users
Front Endpoint Load Test
All users
Zoom 1m 10m 1h All
Number of concurrent users
30
20
10
0
13:54:00
13:54:05
13:54:10
13:54:15
13:54:20
13:54:25
13:54:30
13:54:35
13:54:40
13:54:45
13:54:50
13:54:00
13:54:10
13:54:20
13:54:30
13:54:40
13:54:50

Case 5


Web API
10 threads
HTTP
Retry on error
External API
Unavailable at certain periods

Circuit breaker



Stats

Executions

	Total	OK	KO
Total count	750	730	20
Mean count/s	12.5	12.17	0.33

Response Time (ms)

	Total	OK	KO
Min	4	4	15565
50th percentile	11588	11091	31282
75th percentile	20882	20112	31665
95th percentile	28254	28088	32047
99th percentile	31150	28875	32047
Max	32047	30618	32047
Mean	12.5	12.17	0.33
Standard Deviation	10763	10619	7752

Stats

Executions

	Total	OK	KO
Total count	750	392	358
Mean count/s	15	7.84	7.16

Response Time (ms)

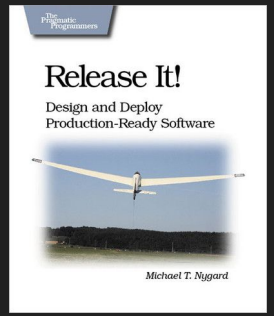
	Total	OK	KO
Min	2	3	2
50th percentile	5	5	3393
75th percentile	3559	5	7355
95th percentile	12802	10	14475
99th percentile	16252	8139	16276
Max	28224	28224	16408
Mean	15	7.84	7.16
Standard Deviation	4373	2687	4756

Rate Limiting and Circuit Breaker Libraries

Technology	Rate Limit	Circuit Breaker
JVM	Resilience4j	Resilience4j
.NET	Polly	Polly
Node.js	express-rate-limit (bottleneck)	Opossum

Key takeaways

- The root cause is most commonly cause by one of the following stability anti-patterns:
  - timeouts
  - rate limiting
  - errors.
- Always try to reproduce the failing state
- “Small” changes can have significant effect on the performance
- Do the post-mortem and learn from your mistakes or even better, from other people (me) mistakes
- To read: **Release It!** - Michael T. Nygard



Questions

Thank you for your attention (is all we need)