Vanilla Kotlin “Reactive” with Spring WebFlux

Davide Cerbo - Senior Software Engineer @ Alfresco

Davide Cerbo

15+ years as developer

@davide_cerbo

https://medium.com/@davidecerbo

DEVDAY

Once upon a time...

A new innovative service called Kontact.io*!

*The story, all names, characters, and incidents portrayed in this presentation are fictitious. No identification with actual persons (living or deceased), places, buildings, and products is intended or should be inferred. No person or entity associated with this presentation received payment or anything of value, or entered into any agreement, in connection with the depiction of Pivotal products. No developers were harmed in the making of this speaker deck.

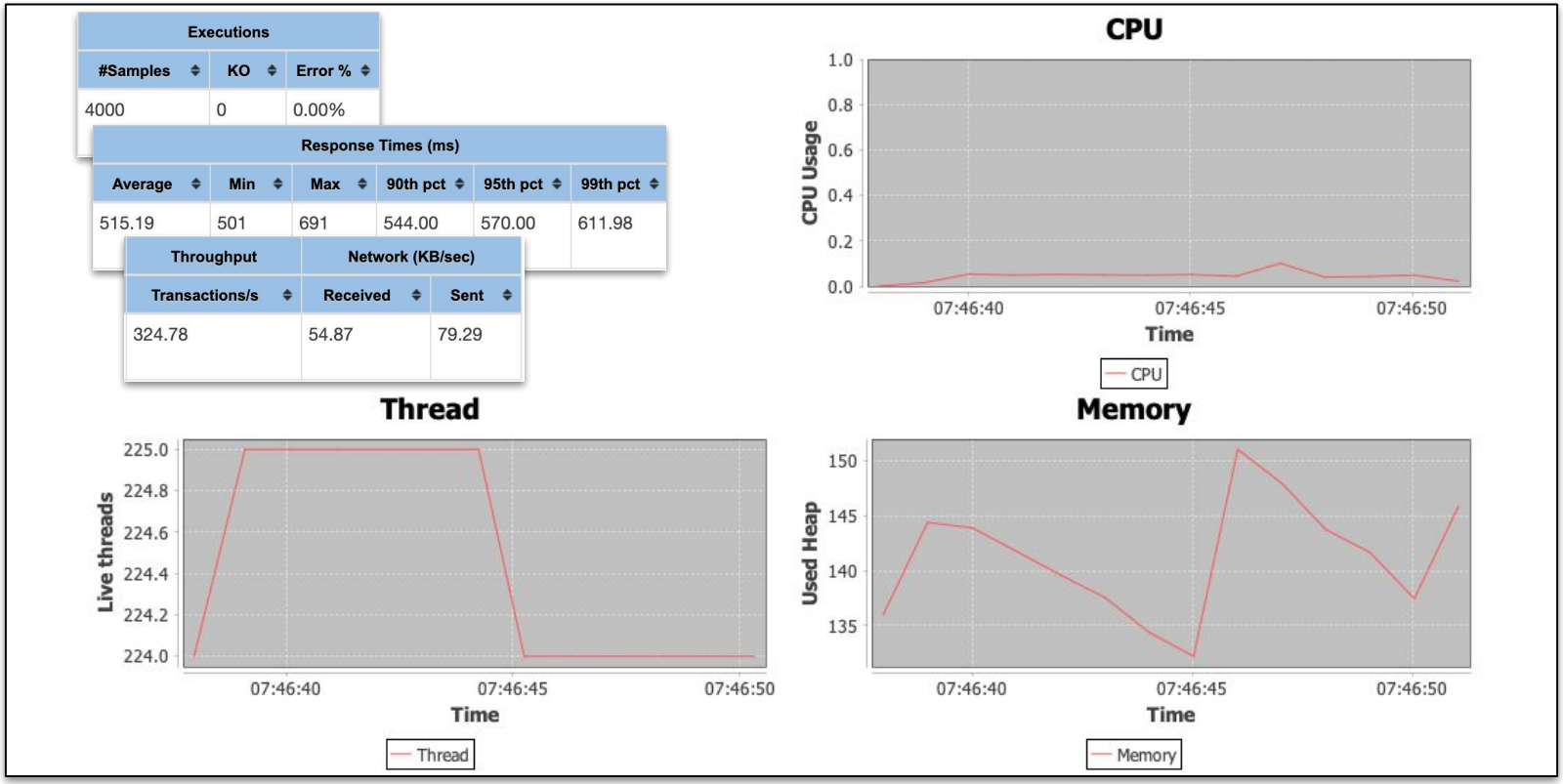
DEMO

Spring MVC (4000 create contacts in 10 seconds)

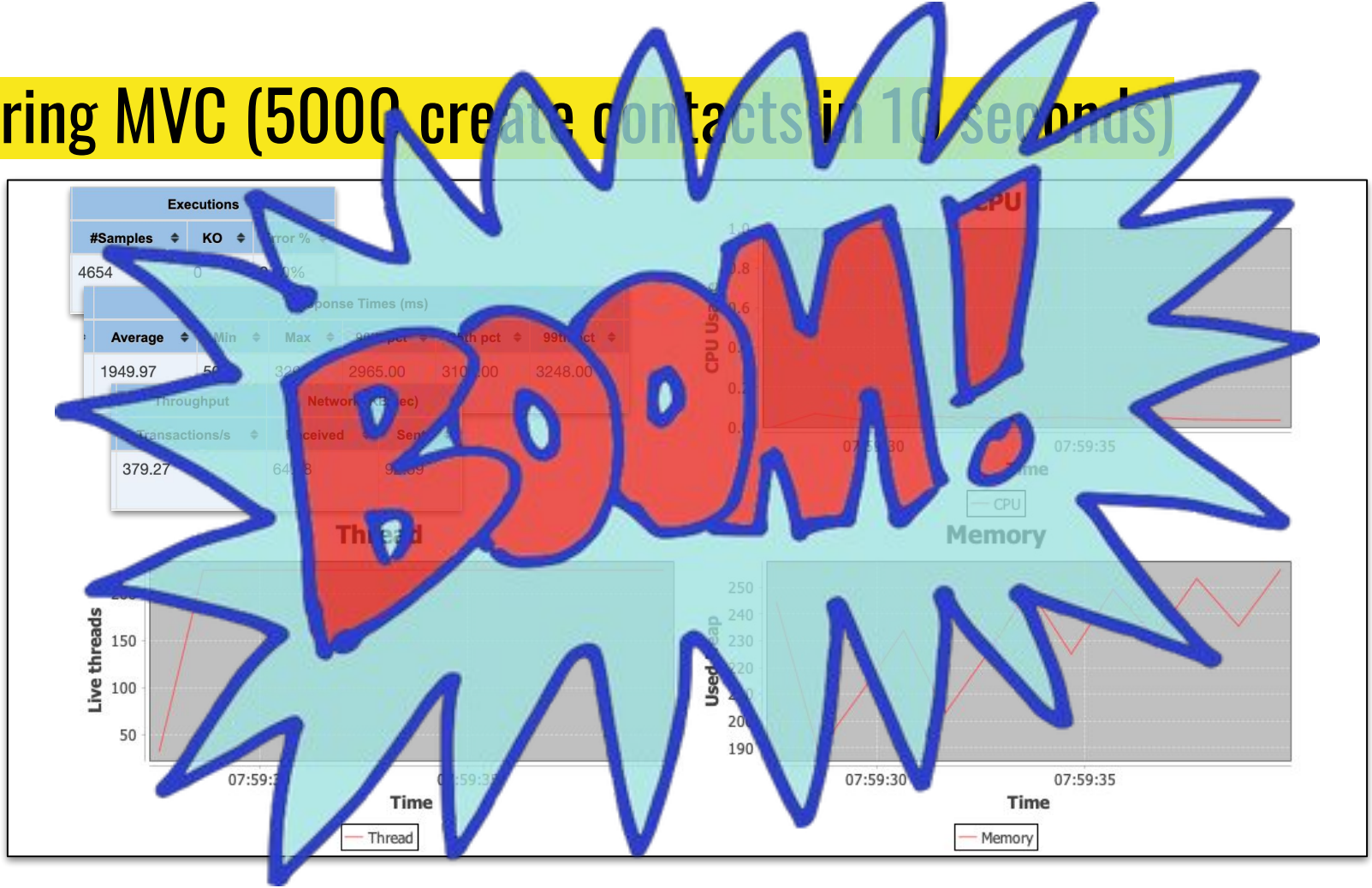
Executions		
#Samples	KO	Error %
4000	0	0.00%

Response Times (ms)					
Average	Min	Max	90th pct	95th pct	99th pct
515.19	501	691	544.00	570.00	611.98

Throughput	Network (KB/sec)	
Transactions/s	Received	Sent
324.78	54.87	79.29



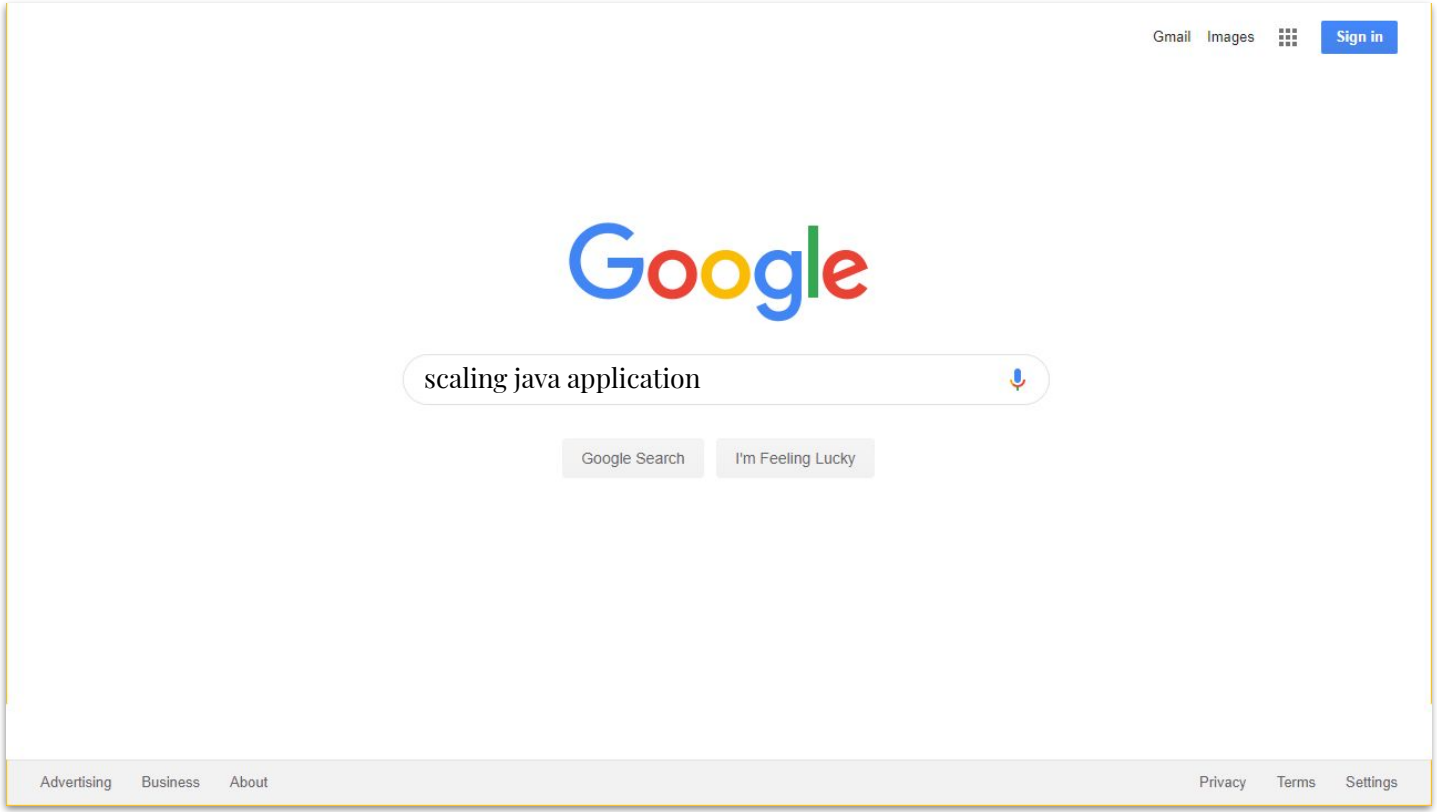
Spring MVC (5000 create contacts in 10 seconds)



Doesn’t scale...too many long requestes!

The default thread pool is 200 that was why we have 400 requests/sec for 500ms response time.

Don’t panic! Use Google!

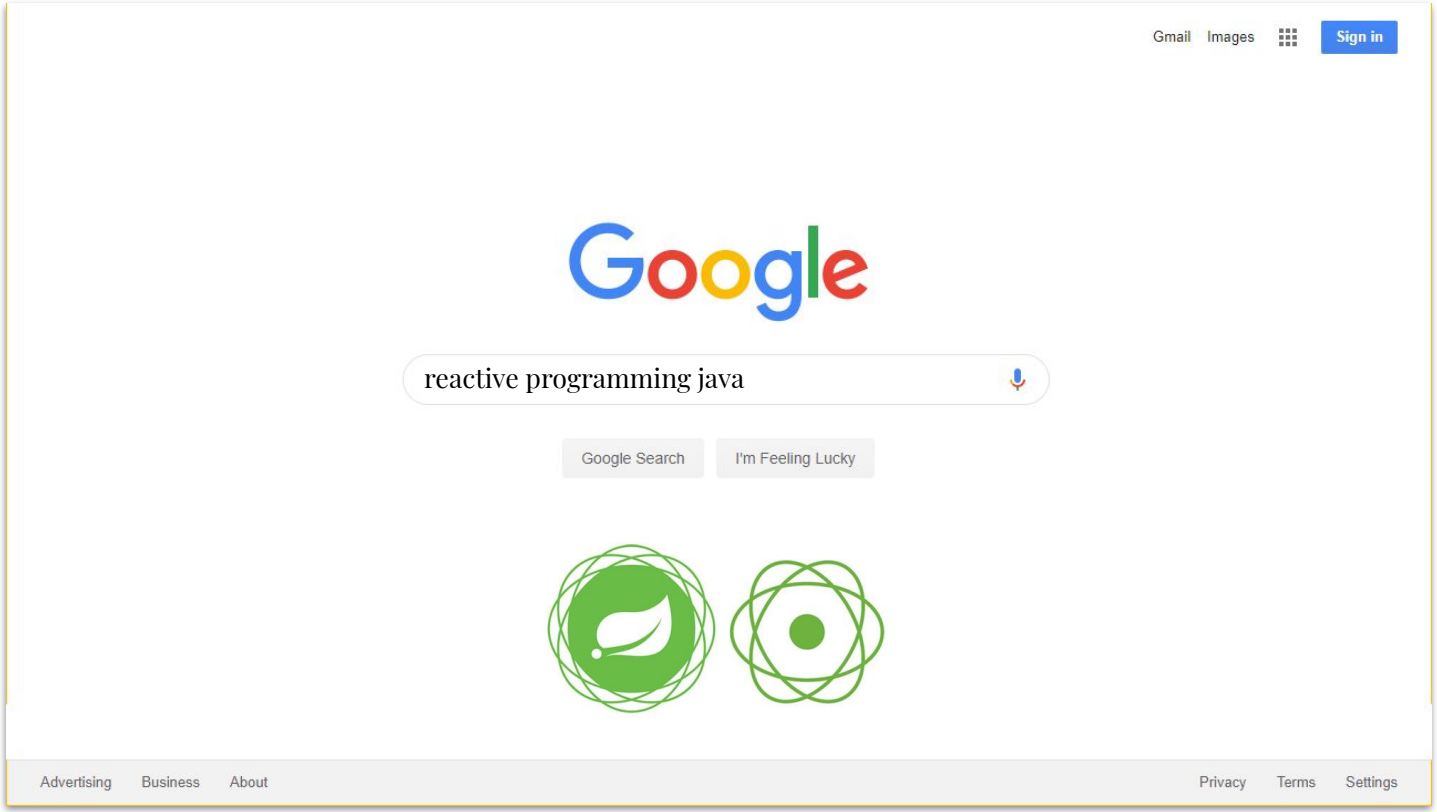


What can I do?

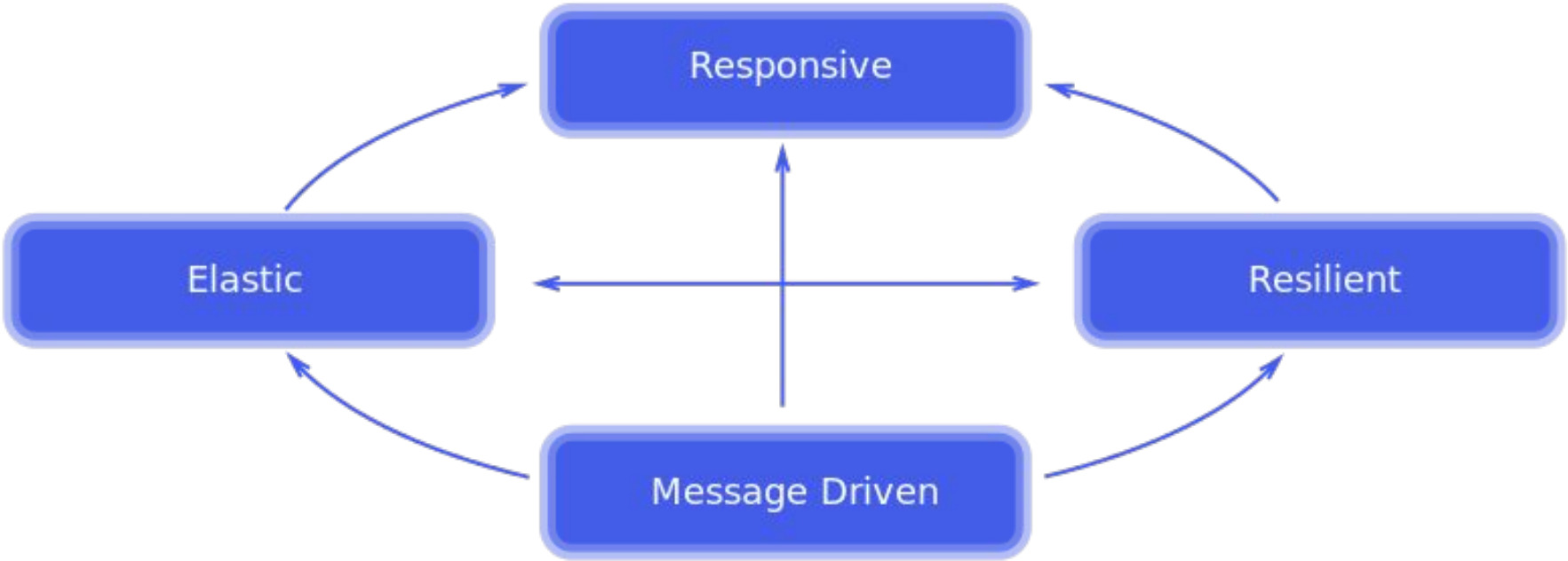
- Increase the thread pool size
- DeferredResult or CompletableFuture with Servlet
- Reactive programming

https://medium.com/@filia.aleks/microservice-performance-battle-spring-mvc-vs-webflux-80d39fd81bf0

Don’t panic! Use Google! (again)



Reactive Systems



Reacting to the future of application architecture by Grace Jansen
https://www.youtube.com/watch?v=5NuvR8_hCxw

From imperative to Reactive

- **Imperative**: tell what to do and how to do it.

```
for (String a : list) { if(...) { ... } else { ...} }
```

- **Declarative**: tell what to do and **not** how to do it.

```
list.stream().forEach{...}
```

- **Functional**: declarative + HOC + functional composition + lazy evaluation.

- **Reactive**: functional + message driven + elastic + responsive + resilience.

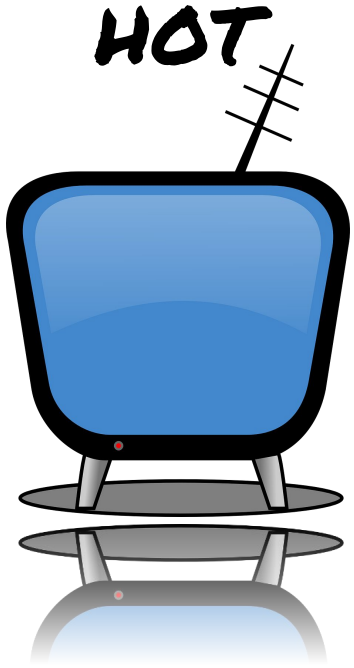
```
Flux.from(list)
    .doOnNext(...)
    .doOnError(...)
    .doOnSuccess(...)
    .doOnComplete(...)
    .subscribe()
```

From Functional to Reactive Programming, Venkat Subramaniam https://www.youtube.com/watch?v=U_NgcAg7jyY

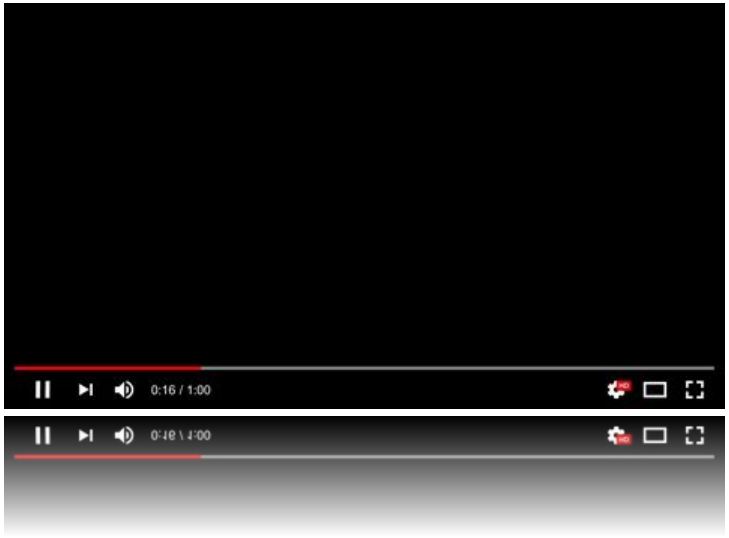
Reactive Programming

Reactive programming, is a non-blocking alternative to traditional programming solutions, working entirely with event driven data streams and functional programming concepts to manipulate streams and considering the failure as first citizen, in any kind of scenario.

Hot Vs. Cold subscribing



COLD



Backpressure

Backpressure = client message control

Server sent event

Server-sent events - Wikipedia

https://en.wikipedia.org › wiki › Server-sent_events ▾ Traduci questa pagina

Server-Sent Events (SSE) is a **server** push technology enabling a client to receive automatic updates from a **server** via HTTP connection. The **Server-Sent Events** EventSource API is standardized as part of HTML5 by the W3C.

Overview · Web browsers · Libraries · Java

Keep attention to your environment!

1. Reactive programming needs reactive libraries!
   https://projectreactor.io/docs/core/release/reference/#faq.wrap-blocking
2. Use Blockhound
   https://medium.com/@domenicosibilio/blockhound-detect-blocking-calls-in-reactive-code-before-its-too-late-6472f8ad50c1
3. R2DBC
   https://r2dbc.io/

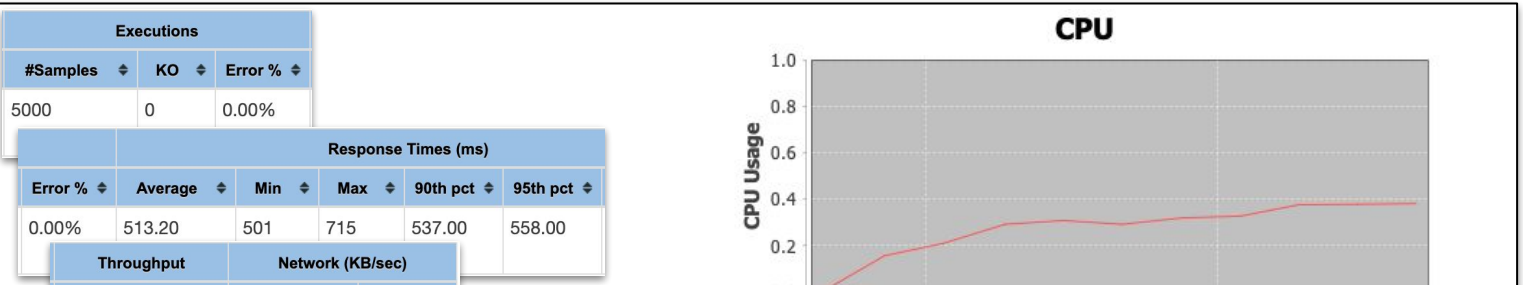
DEMO

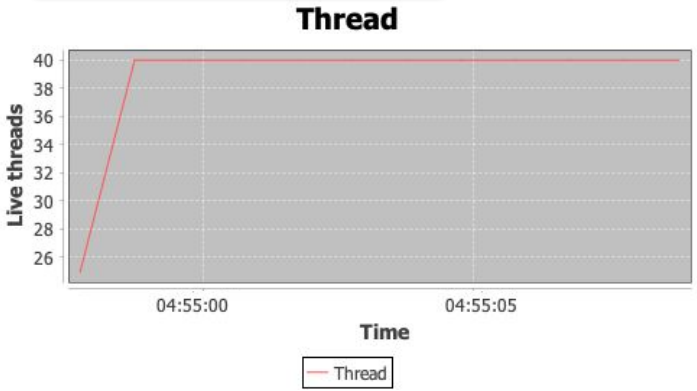
Webflux + Reactor (5000 create contacts in 10 seconds)

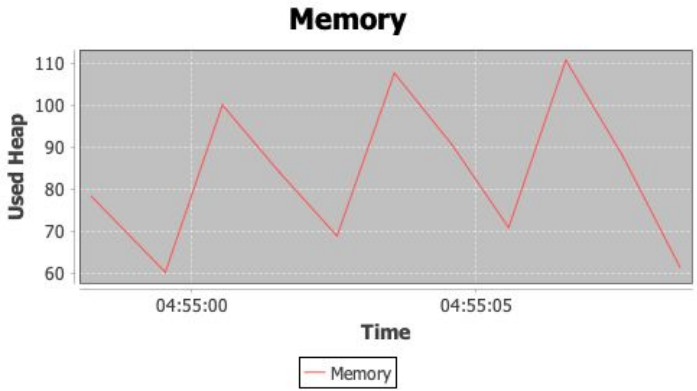
Executions		
#Samples	KO	Error %
5000	0	0.00%

	Response Times (ms)				
Error %	Average	Min	Max	90th pct	95th pct
0.00%	513.20	501	715	537.00	558.00

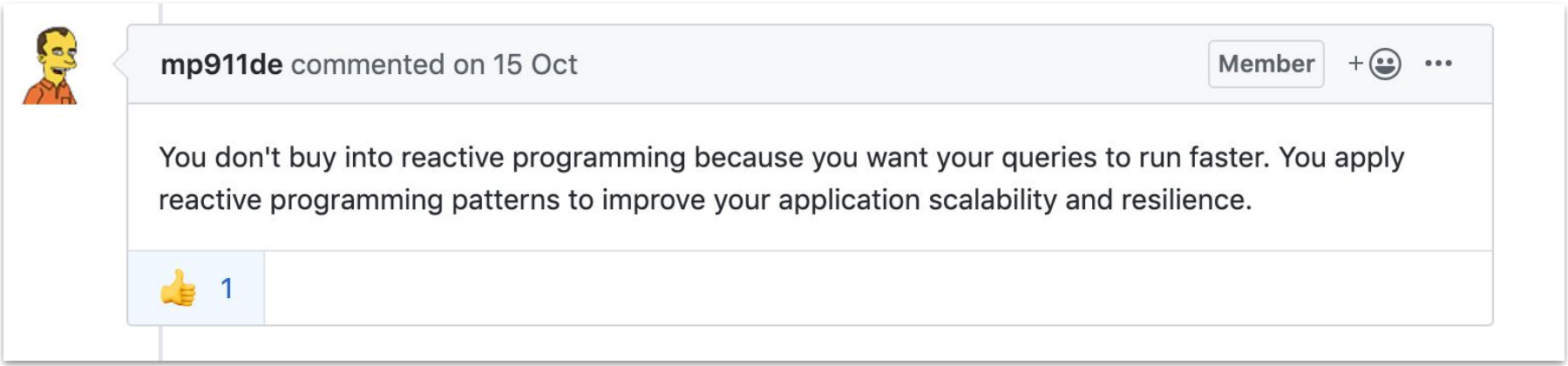
Throughput	Network (KB/sec)	
Transactions/s	Received	Sent
485.96	55.05	118.64







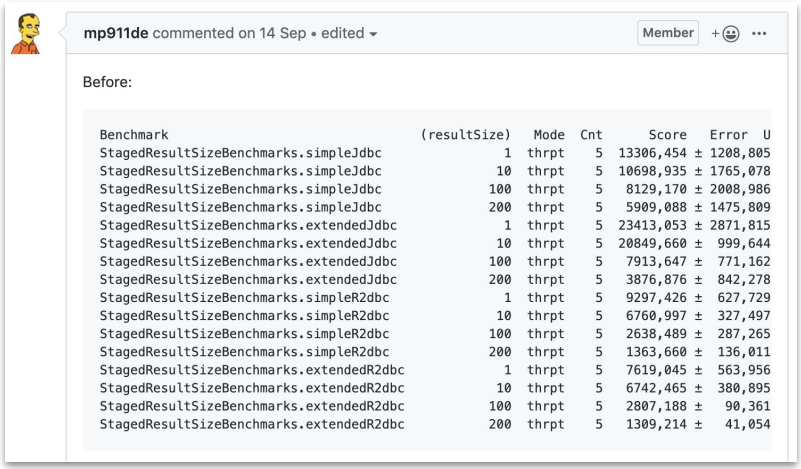
WTF!?!?!



https://medium.com/@filia.aleks/r2dbc-vs-jdbc-19ac3c99fafa

https://github.com/spring-projects/spring-data-r2dbc/issues/203

Serious performance tests!



mp911de commented on 14 Sep • edited

Member

Before:

```
Benchmark                                    (resultSize)   Mode  Cnt      Score   Error  U
StagedResultSizeBenchmarks.simpleJdbc                   1  thrpt    5  13306,454 ± 1208,805
StagedResultSizeBenchmarks.simpleJdbc                  10  thrpt    5  10698,935 ± 1765,078
StagedResultSizeBenchmarks.simpleJdbc                 100  thrpt    5   8129,170 ± 2008,986
StagedResultSizeBenchmarks.simpleJdbc                 200  thrpt    5   5909,088 ± 1475,809
StagedResultSizeBenchmarks.extendedJdbc                 1  thrpt    5  23413,053 ± 2871,815
StagedResultSizeBenchmarks.extendedJdbc                10  thrpt    5  20849,660 ±  999,644
StagedResultSizeBenchmarks.extendedJdbc               100  thrpt    5   7913,647 ±  771,162
StagedResultSizeBenchmarks.extendedJdbc               200  thrpt    5   3876,876 ±  842,278
StagedResultSizeBenchmarks.simpleR2dbc                  1  thrpt    5   9297,426 ±  627,729
StagedResultSizeBenchmarks.simpleR2dbc                 10  thrpt    5   6760,997 ±  327,497
StagedResultSizeBenchmarks.simpleR2dbc                100  thrpt    5   2638,489 ±  287,265
StagedResultSizeBenchmarks.simpleR2dbc                200  thrpt    5   1363,660 ±  136,011
StagedResultSizeBenchmarks.extendedR2dbc                1  thrpt    5   7619,045 ±  563,956
StagedResultSizeBenchmarks.extendedR2dbc               10  thrpt    5   6742,465 ±  380,895
StagedResultSizeBenchmarks.extendedR2dbc              100  thrpt    5   2807,188 ±   90,361
StagedResultSizeBenchmarks.extendedR2dbc              200  thrpt    5   1309,214 ±   41,054
```

After:

```
Benchmark                                    (resultSize)   Mode  Cnt      Score      Error
StagedResultSizeBenchmarks.simpleJdbc                   1  thrpt    5  13306,454 ± 1208,805
StagedResultSizeBenchmarks.simpleJdbc                  10  thrpt    5  10698,935 ± 1765,078
StagedResultSizeBenchmarks.simpleJdbc                 100  thrpt    5   8129,170 ± 2008,986
StagedResultSizeBenchmarks.simpleJdbc                 200  thrpt    5   5909,088 ± 1475,809
StagedResultSizeBenchmarks.extendedJdbc                 1  thrpt    5  23413,053 ± 2871,815
StagedResultSizeBenchmarks.extendedJdbc                10  thrpt    5  20849,660 ±  999,644
StagedResultSizeBenchmarks.extendedJdbc               100  thrpt    5   7913,647 ±  771,162
StagedResultSizeBenchmarks.extendedJdbc               200  thrpt    5   3876,876 ±  842,278
StagedResultSizeBenchmarks.simpleR2dbc                  1  thrpt    5  10713,126 ± 1283,533
StagedResultSizeBenchmarks.simpleR2dbc                 10  thrpt    5  10046,317 ±  784,692
StagedResultSizeBenchmarks.simpleR2dbc                100  thrpt    5   5079,499 ± 1738,124
StagedResultSizeBenchmarks.simpleR2dbc                200  thrpt    5   3875,689 ±  752,872
StagedResultSizeBenchmarks.extendedR2dbc                1  thrpt    5  11612,125 ± 2114,247
StagedResultSizeBenchmarks.extendedR2dbc               10  thrpt    5   9109,973 ± 1904,214
StagedResultSizeBenchmarks.extendedR2dbc              100  thrpt    5   5314,664 ±  503,736
StagedResultSizeBenchmarks.extendedR2dbc              200  thrpt    5   3507,015 ±  539,890
```

https://github.com/r2dbc/r2dbc-postgresql/pull/158

Coroutine

Coroutine ≃ light-weight thread

- They are like threads, they run in parallel, wait for each other and they communicate.
- They are cheap, we can create many of those without having performance issues.
- They are executed in a thread pool.
- A thread can handle more than one coroutine.
- Thread became free until a coroutine is in waiting state. When the coroutine will return active, it doesn't get the old thread, but it will use a free thread in the pool.

https://github.com/Kotlin/kotlinx.coroutines/blob/master/coroutines-guide.md
https://proandroiddev.com/approaching-kotlin-coroutines-an-extensive-feature-concurrent-programming-in-kotlin-eaaa19b003d2

Coroutines are humble

```
fun main(args: Array<String>) = runBlocking {
    val jobs = List(100_000) {
        launch {
            delay(1000L)
            print(".")
        }
    }
    jobs.forEach { it.join() }
}
```

```
fun main(args: Array<String>)  {
    val jobs = List(100_000) {
        thread(start = true) {
            Thread.sleep(
            p
        }
    }
    jobs.forEach { it.join() }
}
```

OUT OF MEMORY!!!

Coroutine: suspend, async / await

```
fun main(args: Array<String>) = runBlocking<Unit> {
    val time = measureTimeMillis {
        val one = doSomethingUsefulOne()
        val two = doSomethingUsefulTwo()
        println("The answer is ${one + two}")
    }
    println("Completed in $time ms")
}

suspend fun doSomethingUsefulOne(): Int {
    delay(1000L)
    return 13
}

suspend fun doSomethingUsefulTwo(): Int {
    delay(1000L)
    return 29
}
```

~2 sec.

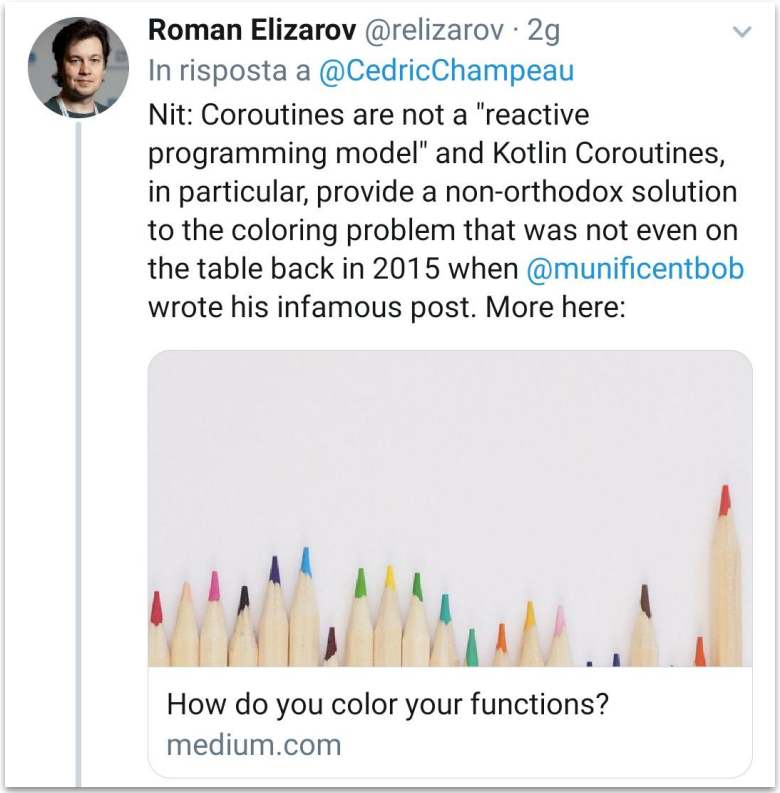
```
fun main(args: Array<String>) = runBlocking<Unit> {
    val time = measureTimeMillis {
        val one = async { doSomethingUsefulOne() }
        val two = async { doSomethingUsefulTwo() }
        println("The answer is ${one.await() + two.await()}")
    }
    println("Completed in $time ms")
}

suspend fun doSomethingUsefulOne(): Int {
    delay(1000L)
    return 13
}

suspend fun doSomethingUsefulTwo(): Int {
    delay(1000L)
    return 29
}
```

~1 sec.

Coroutine means reactive?



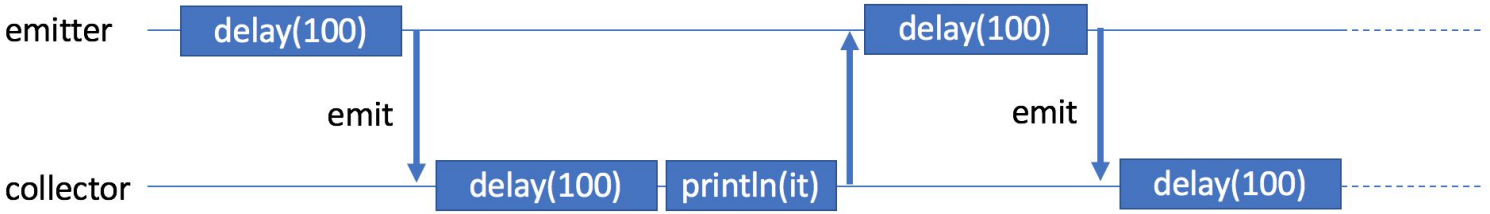
Structured concurrency

- The main task will complete when all children concurrent tasks complete.
- The lifetime of a thread is bound to a particular syntactic construct, typically a scope or a code block.
- Formulated in 2016 by Martin Sústrik (creator of ZeroMQ).

```
fun main() = runBlocking {
    launch {
        println("start parent")
        launch {
            (1..10).forEach {
                delay(100)
                println("a $it")
            }
        }
        launch {
            (1..10).forEach {
                delay(200)
                println("b $it")
                cancel()
            }
        }
        delay(500)
        cancel()
    }
    println("end")
}
```

Flow

A cold asynchronous data stream that sequentially emits values and completes normally or with an exception.



https://medium.com/@elizarov/kotlin-flows-and-coroutines-256260fb3bdb

Comparing Flow & Flux

```
flow {

   emit(“abc”)

}

.catch{ ... }

.collect { ... }

.onCompletion { ... }
```

```
Flux.create{

   it.next(“abc”)

}

.onError(...)

.subscribe { ... }

.onComplete { ... }
```

From Flux to Flow and viceversa

```
val flowA = flowOf(1, 2, 3)
    .map { it + 1 }
    .flowOn(ctxA)
    .collect
```

```
val fluxA = Flux.just(1, 2, 3)
    .map { it + 1 }
    .subscribeOn(ctxA)
    .subscribe()
```

kotlinx.coroutines.Dispatchers:

- Main
- Default
- Single
- IO

```
val flowA = flowA.asFlow()
```

Buffer & Conflate

```
fun foo(): Flow<Int> = flow {
    for (i in 1..3) {
        delay(100)
        println("Emitting $i")
        emit(i)
    }
}

fun main() = runBlocking<Unit> {
    val time = measureTimeMillis {
        foo()
            .buffer() or .conflate()
            .collect { value ->
                delay(300)
                println("Collecting $value")
            }
    }
    println("Collected in $time ms")
}
```

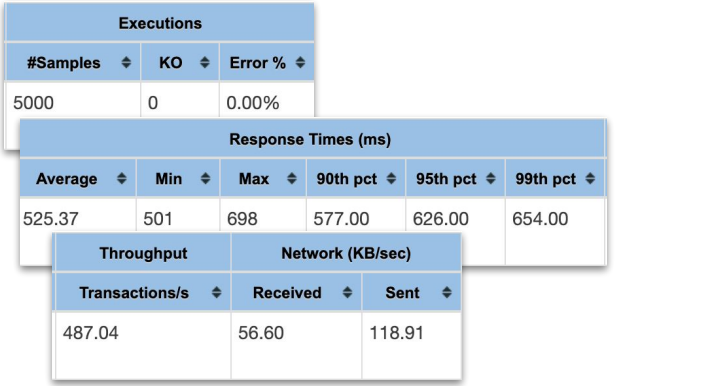
DEMO

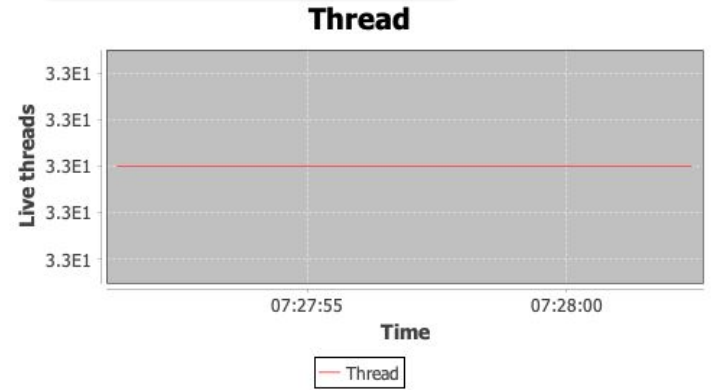
Webflux + Coroutine (5000 create contacts in 10 seconds)

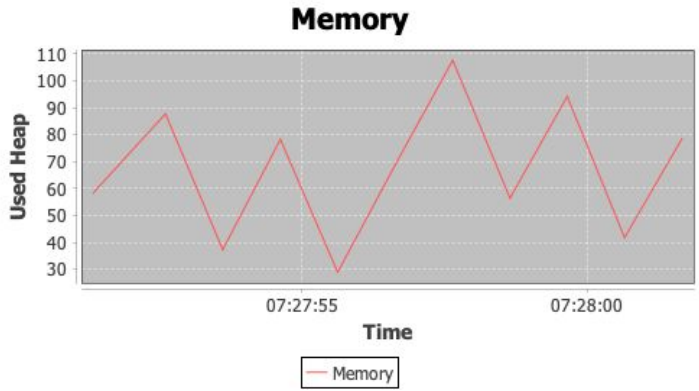
Executions		
#Samples	KO	Error %
5000	0	0.00%

Response Times (ms)					
Average	Min	Max	90th pct	95th pct	99th pct
525.37	501	698	577.00	626.00	654.00

Throughput	Network (KB/sec)	
Transactions/s	Received	Sent
487.04	56.60	118.91

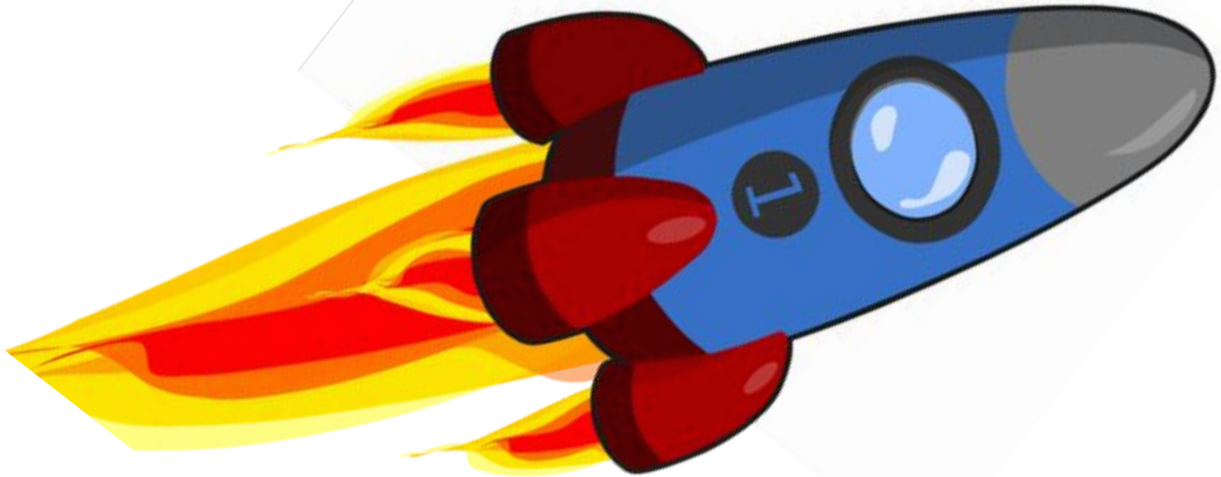






I'm growing up!

- I need to scale!
- I need to communicate with other microservices!



RSocket

RSocket is a binary protocol for use on byte stream transports such as TCP, WebSockets, and Aeron.

It enables the following symmetric interaction models via async message passing over a single connection:

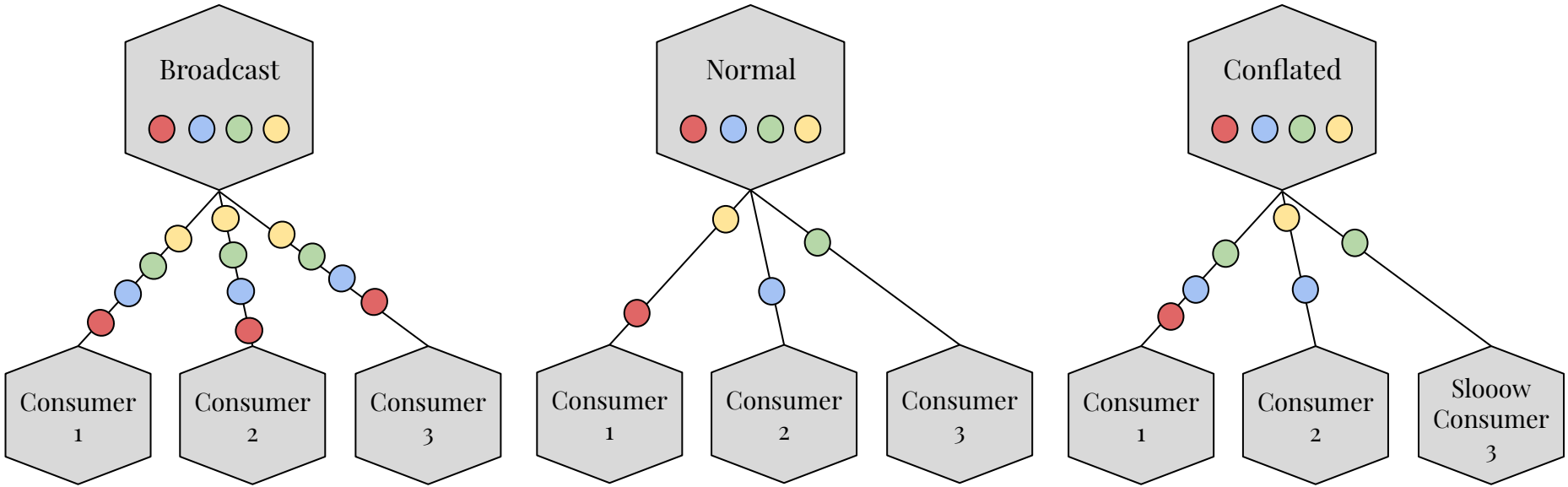
- request/response (stream of 1)
- request/stream (finite stream of many)
- fire-and-forget (no response)
- channel (bi-directional streams)

It supports:

- Session resumption
- Security
- Backpressure

Channels

Channels are **hot** asynchronous streams of data.



Null pointer when connecting to RSocket #24088

Edit New issue

Closed **jesty** opened this issue 13 hours ago · 3 comments

jesty commented 13 hours ago • edited by rstoyanchev

I'm trying RSocket and I'm using:

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.2.BUILD-SNAPSHOT</version>
        <relativePath/>
</parent>
```

My client code looks like:

```
requester
        .route("test")
        .retrieveFlow<Int>()
        .collect({
            println("Receiving $it")
        })
```

Starting from today I'm facing this exception:

```
        at java.util.Objects.requireNonNull(Objects.java:203) ~[na:1.8.0_202]
        at reactor.core.publisher.FluxSource.<init>(FluxSource.java:46) ~[reactor-core-3
        at reactor.core.publisher.Flux.wrap(Flux.java:9869) ~[reactor-core-3.3.1.BUILD-S
        at reactor.core.publisher.Flux.from(Flux.java:961) ~[reactor-core-3.3.1.BUILD-SN
        at io.rsocket.RSocketRequester.requestChannel(RSocketRequester.java:174) ~[rsock
        at io.rsocket.util.RSocketProxy.requestChannel(RSocketProxy.java:50) ~[rsocket-c
        at io.rsocket.util.MultiSubscriberRSocket.lambda$requestChannel$3(MultiSubscribe
        at reactor.core.publisher.FluxDefer.subscribe(FluxDefer.java:46) ~[reactor-core-
        at reactor.core.publisher.Flux.subscribe(Flux.java:8128) ~[reactor-core-3.3.1.BU
        at kotlinx.coroutines.reactive.PublisherAsFlow.collect(ReactiveFlow.kt:77) ~[kot
        at kotlinx.coroutines.flow.FlowKt__TransformKt$onEach$$inlined$unsafeTransform$1
        at kotlinx.coroutines.flow.FlowKt__CollectKt.collect(Collect.kt:30) ~[kotlinx-co
        at kotlinx.coroutines.flow.FlowKt.collect(Unknown Source) ~[kotlinx-coroutines-c
```

Assignees

rstoyanchev

Labels

in: messaging

type: bug

Projects

None yet

Milestone

5.2.2

Notifications Customize

Unsubscribe

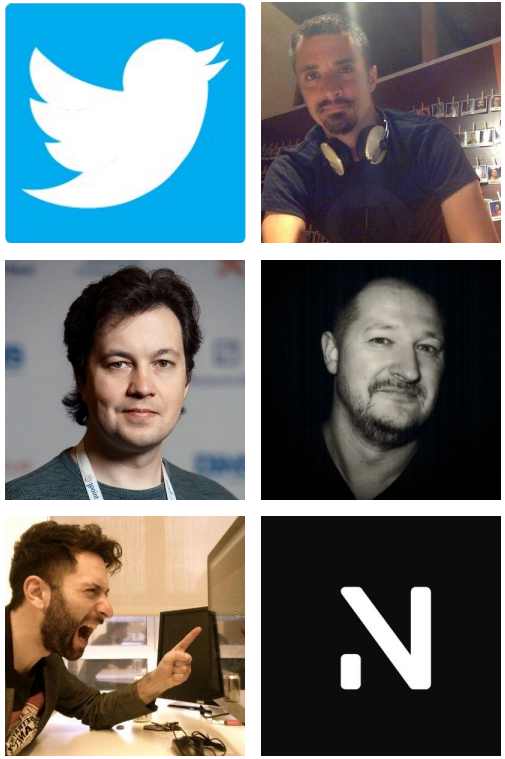
You're receiving notifications because you were mentioned.

3 participants

DEMO

People to follow

- https://twitter.com/sdeleuze
- https://twitter.com/relizarov
- https://twitter.com/mp911de
- https://twitter.com/smaldini
- https://twitter.com/netifi_inc



Questions?

https://github.com/jesty/kotlin-reactive

ps: we are hiring!

@davide_cerbo
https://medium.com/@davidecerbo