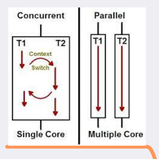
Java Reactive

0. Basic concepts

Parallelism vs Concurrency



Parallelism is about doing lot of things at once.

- Tools
  - Threads
  - Thread Pool
    - ExecuterService
    - ForkJoinPool
    - Custom Thread Pools (eg: Web Servers)
    - Requires >1 cpu cores.

Concurrency is about dealing with lot of things at once

- Shared resource is to be accessed/updated
- Multiple tasks need to coordinate
- Tools
  - Locks / synchronized
  - Atomic classes
  - Concurrent data structures
    - ConcurrentHashMap
    - BlockingQueue
  - CompletableFuture
  - CountDownLatch / Phaser / CyclicBarrier / Semaphore, etc

Funcional Programming

It's a paradigm that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data.

Combine Reactive Programming and Functional programming to write code in a faster, clearer and more understandable way.

Streams API

Streams let you process data in a declarative way

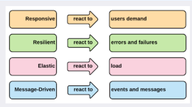
Pipelining: Chain of various stream operations.

The main goal

Code is more waiting than working

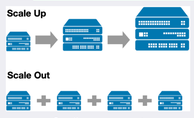
To maximize the use rate of modern multicore CPUs and, more precisely, of the threads computing for their use.

Reactive Manifesto



https://www.reactivemanifesto.org/

- Responsive
  - The system responds in a timely manner if possible
  - This is key for usability and utility
  - Consistent response times
- Resilience
  - The system stays responsive when facing a failures
  - Performance, endurance and security are all facets of resiliency
- Elastic
  - The system stays responsive under varying workload
  - Design with no contention points or bottlenecks
  - Ability to share or replicate components and distribute inputs among them



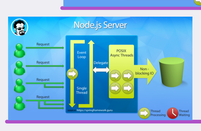
Scaling up or out

- Message-Driven
  - Reactive applications rely on asynchronous message-passing to establish a boundary between components
  - Ensure loose coupling, isolation and location transparency.
  - Delegate failures as messages
  - Non-blocking communication allows recipients to only consume resources while active

Nonblocking

- Vs Blocking
  - Single Item: T
  - Many Items: Stream<T>
- Project Reactor
  - Single Item: Mono <T>
  - Many Items: Flux<T>
  - https://projectreactor.io/
- Blocking

- Other Frameworks

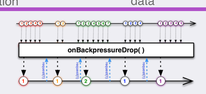


Back pressure (Throttling)

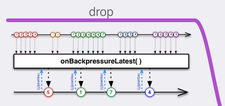
Problem: Source of information overflows the consumer with data

Solution

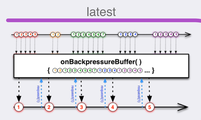
- Mechanism
  - drop



  - latest



  - buffer



- Provides resilience

Cold vs Hot Sources of information

A cold source of information start running upon subscription

A hot source of information is always emitting data, even before a subscription is active

Cold Publisher

Hot Publisher

Otros

- Multiple Subscribers
- Less latency
- More tolerance of failure
- Threads
  - Using very few threads
  - Low-level threading
  - Thread safety

1. Definitions

Reactive Programming

Wikipedia: In computing, reactive programming is a programming paradigm oriented around data flows and the propagation of change

- Stream is an internal iterator functional composition and Lazy evaluation.
- A stream is a sequence of ongoing events ordered in time
- Declarative, Event based programming
- Decompose problem into multiple, asynchronous discrete steps.

Example simple: Spreadsheet

- Define a formula in a cell.
- How the change is propagated when you change the value of one cell.

Reactive Streams

Standard for asynchronous stream processing with Non-blocking and Back pressure

It's more than just async and non blocking execution

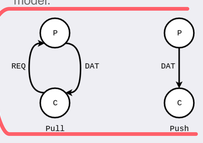
- Advanced time scheduling
- Flexible Scheduling
- Back pressure control
- Resilience on errors

With the imperative model we pull and process things synchronously using iterators.

With reactive programming we take that and create push model.

2. Java Streams vs Reactive Streams

Pull vs Push Data



Data only vs 3 Channel

- Data
- Error
- Complete

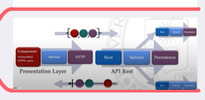
Exceptions vs Circuit breaks

- Error is just another form of data in reactive programming.
  - First Class Citizen
- Exceptions are imperative style ideas.

Secuential vs Parallel

Single Pipeline vs Multiple subscribers

Synch vs Async



When to use

- Java Streams
  - Collection Processing
  - Batch computational jobs
  - Queries on already available data
    - Find the transaction with the highest value
    - Find the green apples over 150 grams.
- Observables
  - On demand computation
  - Updating calculations based upon a series o events
  - I/O workloads
  - Eg:
    - Aggregate saved video positions stored on different micro services.
    - Monitor stock market price ticks until the price is below 100
    - Aggregating temperature sensor information to control heating

3. Features

Java 9 Reactive Stream Api

java.util.concurrent.Flow

- Publisher
  - A potentially unbounded source of events.
- Subscriber
  - Receives events.
- Subscription
  - Control the processing + back pressure of the events.
- Processors
  - Both a Publisher and Subscriber.

Used by Libraries

JDK Standardization of the Reactive Stream Protocol.

Enables greater adoption by having it in the core JDK.

https://github.com/reactive-streams/reactive-streams-jvm

Some rules

- If a publisher fails it must signal an on Error
- If a Publisher terminates successfully (finite stream) it must signal an on Complete
- If a Subscription is cancelled its Subscriber MUST eventually stop being signaled
- Publisher.subscribe MAY be called as many times as wanted but MUST be with a different Subscriber each time
- A subscriber MUST signal demand via Subscription.request(long n) to receive onNext signals

4. Api

RxJava

Used directly by Programmers

Why RxJava

- Callbacks were too low level an API
- Completable Future only deals with a single value.
- What if you are trying process a series of values on the fly.
- 3 channel vs 2 channel
- Reactive Stream vs CompletableFuture/Promises
  - CompletableFuture<List<V>>
    - Wait for all elements in the list to be available.
  - Observable<V>
    - Start processing one by one on arrival
- Actors require you to buy into a whole actor framework and are also a lower lever API.

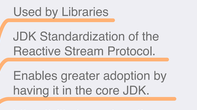
Alternative Implementations

Frameworks/Libs

- Project Reactor - Spring
- AkkaStreams - Akka
- Vert.x
- RxNetty

Reactive socket.io

- Web socket protocol with back pressure.
- https://rsocket.io/



Reactive Systems

An architectural style that allows multiple applications to work together.

https://projectreactor.io/

https://spring.io/reactive

Spring WebFlux

- Mono
- Flux