Enterprise Architecture

Reactive Apps

A new programming paradigm
“REACTIVE”

The Reactive Manifesto

- Check:
  - https://www.reactivemanifesto.org/

The Reactive Manifesto

- Responsive – systems should respond in a timely manner
- Message Driven – systems should use async message-passing between components to ensure loose coupling
- Elastic – systems should scale and stay responsive under high load
- Resilient – systems should stay responsive when some components fail

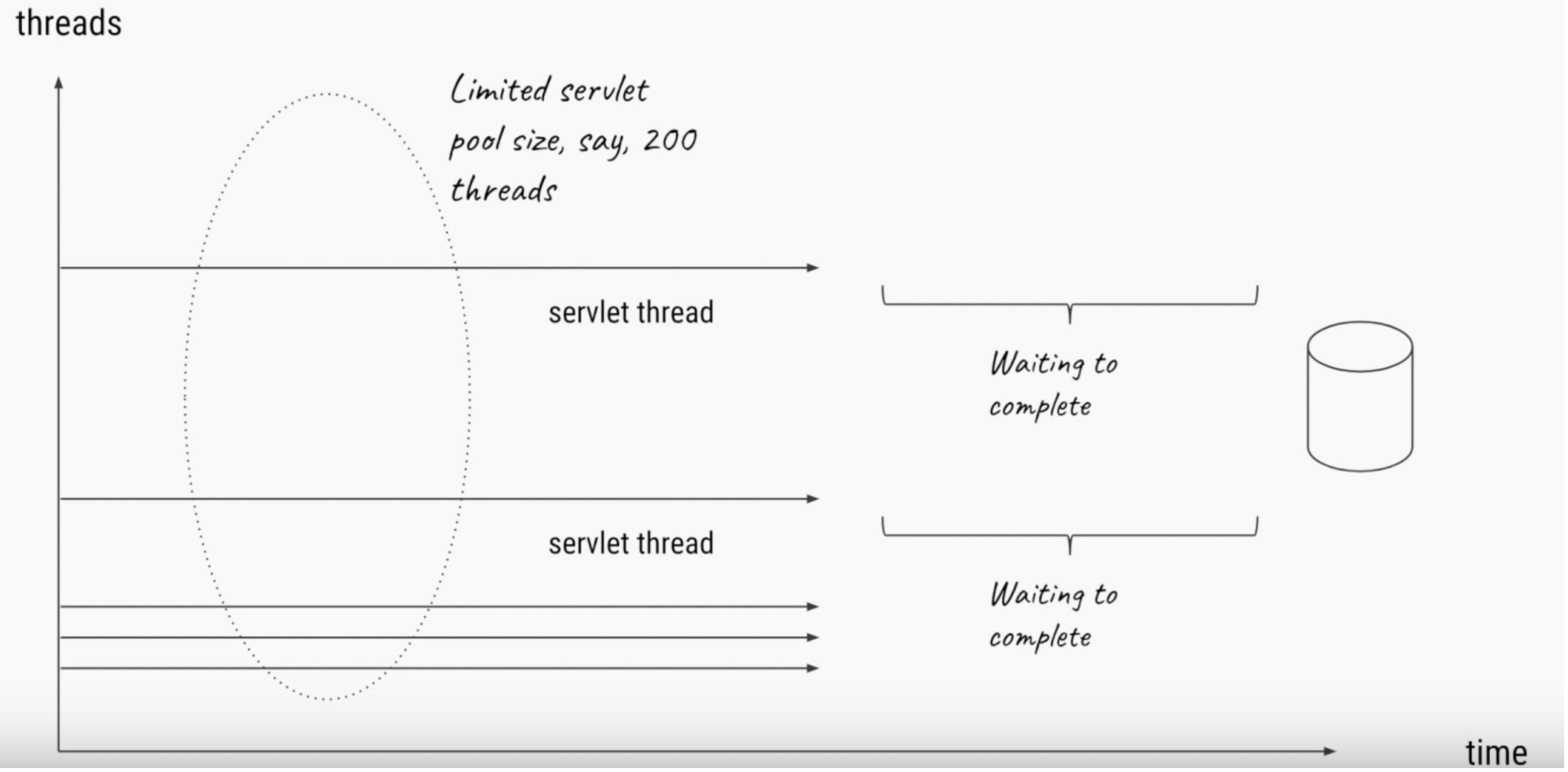
Functional programming

- Building software by composing **pure functions**, avoiding shared state, mutable data, and side-effects.

Reactive Programming

- Asynchronous
- Event-driven
- Non-blocking
- Low footprint:
  - Less Memory
  - Small number of threads (easy to scale vertically within the JVM)

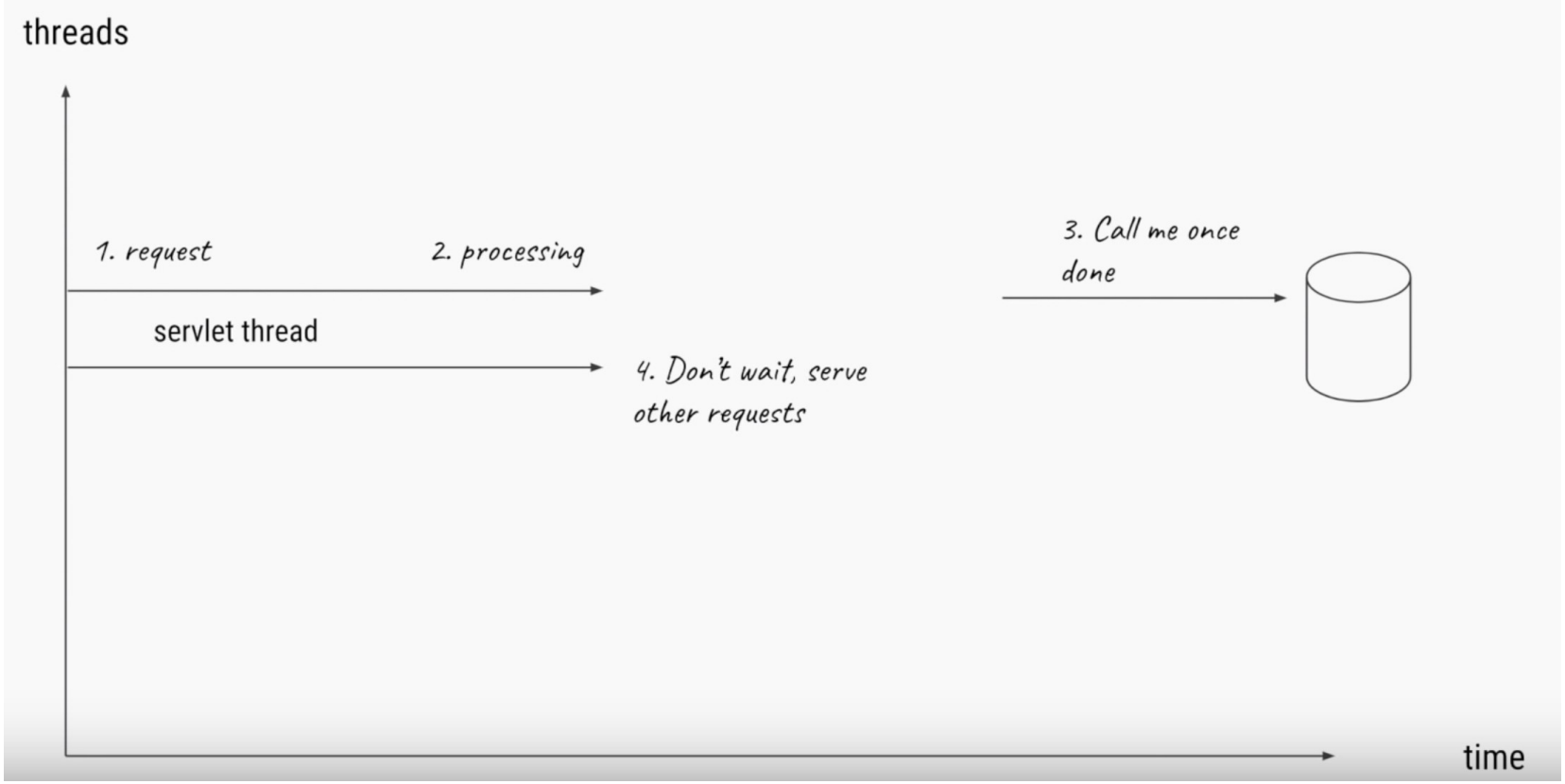
Non-Blocking



https://www.youtube.com/watch?v=M3jNn3HMeWg

Blocking IO operation

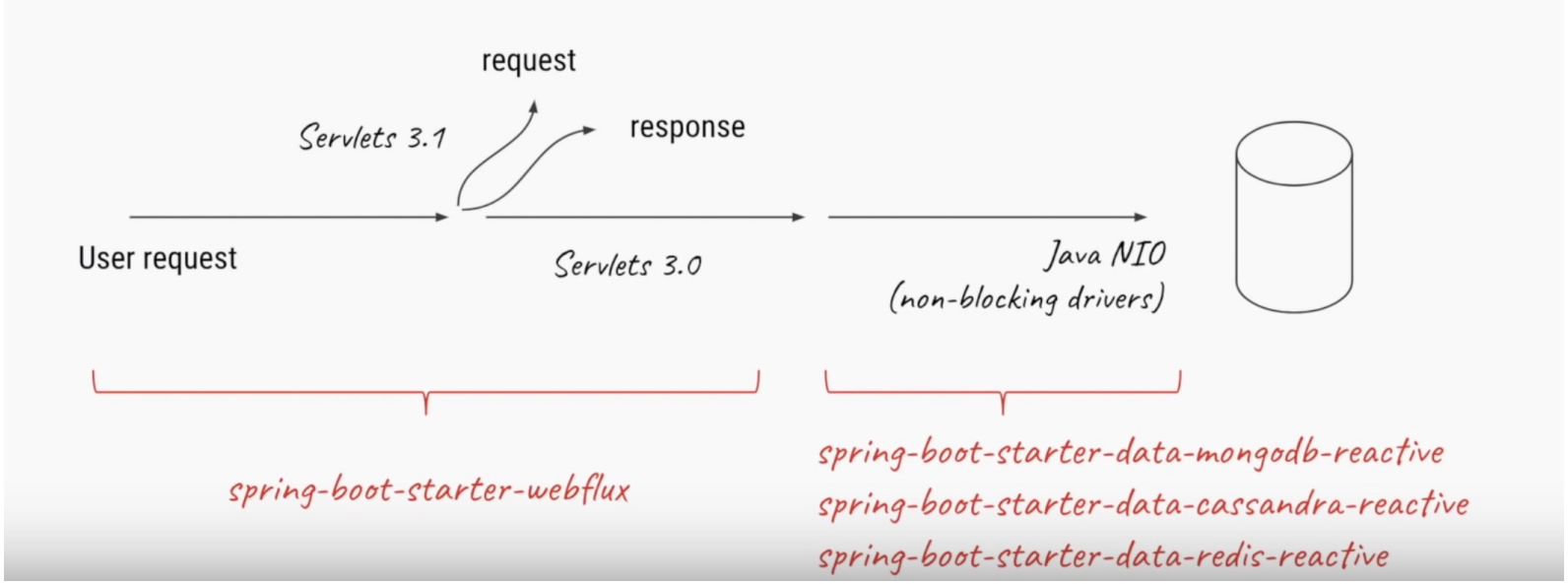
Non-Blocking



https://www.youtube.com/watch?v=M3jNn3HMeWg

Non blocking operation

Non-Blocking



https://www.youtube.com/watch?v=M3jNn3HMeWg

Non blocking DB drivers

Back Pressure

- Drinking from the fire hose ?!

http://renandhiedi.com/workwith/wrong-with-me/

Back Pressure

- Ensure producers don’t overwhelm consumers.
- A pipeline of reactive components between A DB to restful HTTP service :
  - When the HTTP connection is too slow → the DB (sender) slow/stop pushing data until network capacity frees up.

Key Benefits

- Applications are much more resilient and predictable under load or during scaling
  - Due to low threads/memory usage
- Suitable for streaming APIs
- In scenarios where there’s a lot of unpredictable latencies between components
  - Back pressure optimizes the flow rate and minimizes failures

Cons

- Learning curve
- To fully utilize, the whole pipeline must be reactive as well
- May be too much for simple CRUD operations
- Not suitable for CPU intensive operation
  - Non blocking (e.g. IO) doesn’t apply

Reactive Programming style

- Reactive Programming: Why It Matters | Andre Staltz
  - https://www.youtube.com/watch?v=49dMGC1hM1o

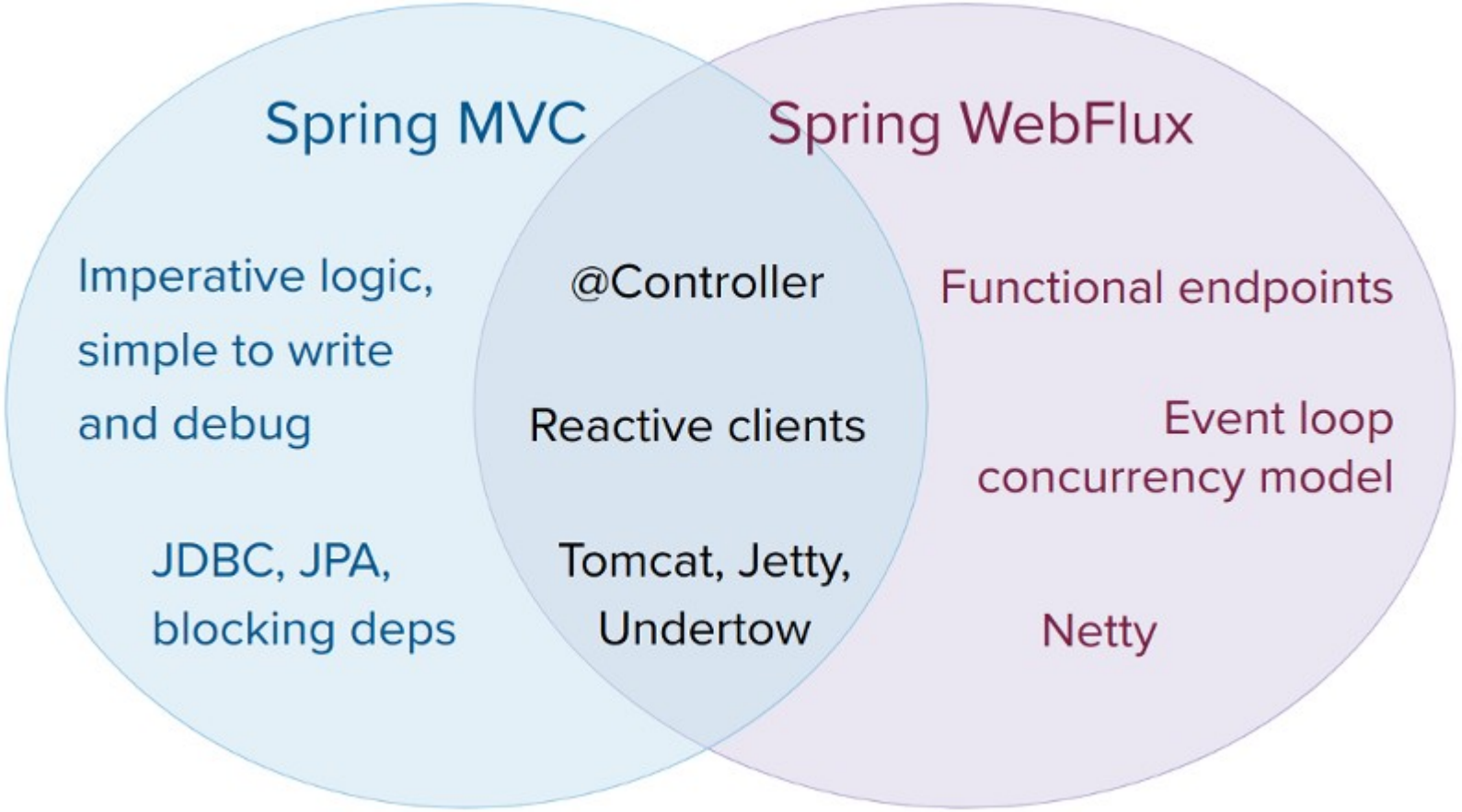
Reactive Streams

- Java Spec:
  - https://www.reactive-streams.org/
- Most of frameworks are built on top of it
- Non Blocking, Back pressure
- Low latency, high throughput

Reactive frameworks

- Check Rx*: http://reactivex.io/
  - RxJava (Obervable)
- Spring WebFlux
  - Based on Reactor: https://projectreactor.io/
- Scala: AKKA (Actor Model)

Classical MVC vs reactive



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

Do I need reactive?

- How to choose ?
  - https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html#webflux-framework-choice

Example reactive service in spring Webflux

- Got time ? , watch GOTO 2019 • Reactive Spring | Josh Long
  - https://www.youtube.com/watch?v=49dMGC1hM1o
- Note: This is an extensive example, you won’t need all of it.
  - https://developer.okta.com/blog/2018/09/24/reactive-apis-with-spring-webflux

Reactive Programming

- Learn to think in reactive way:
  - The introduction to Reactive Programming you've been missing by André Staltz
    - https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
- Seriously read it!