**Akka vs. Spring Reactor Initial Comparison:**

**I used for both a mass of Fake data of 200000 records for each.**

**Both use the Reactive Streams Specification (Reactive Streams)**

**Reactive Streams is an initiative to provide a standard for asynchronous flow processing with non-blocking return pressure. This includes efforts focused on runtime environments (JVM and JavaScript) as well as network protocols.**

**What we want is asynchronous and non-blocking. A group of developers of companies like Netflix, Pivotal, RedHat, etc. met and converged to something called The Reactive Streams Specification.**

****

**Akka framework**

Akka is an Opensource framework.

Akka is implemented from the agent model (originated in 1973 by Carl Hewitt), which provides a model for dealing with concurrent computing.

Akka, the unit of execution is the Actor and his microservices are built as actors. An actor is lightweight and there may be several million actors per GB of heap memory.

The actor is an object that encapsulates state and behavior and communicates exclusively by exchanging messages that are placed in the recipient's mailbox. Below is a diagram that illustrates this.

Simpler Concurrent & Distributed Systems

Actors and Streams let you build systems that scale up, using the resources of a server more efficiently, and out, using multiple servers.

Resilient by Design

Building on the principles of The Reactive Manifesto Akka allows you to write systems that self-heal and stay responsive in the face of failures.

High Performance

Up to 50 million msg/sec on a single machine. Small memory footprint; ~2.5 million actors per GB of heap.

Elastic & Decentralized

Distributed systems without single points of failure. Load balancing and adaptive routing across nodes. Event Sourcing and CQRS with Cluster Sharding. Distributed Data for eventual consistency using CRDTs.

Reactive Streaming Data

Asynchronous non-blocking stream processing with backpressure. Fully async and streaming HTTP server and client provides a great platform for building microservices. Streaming integrations with Alpakka.

**Time metric using 200000 records for both:**

**Akka:**

-------------------------------------------------------------------------------

Test set: com.lightbend.akka.sample.AkkaQuickstartTest

-------------------------------------------------------------------------------

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.766 sec

INITIAL TIME: [05/10/2019 10:10:49.708]

FINAL TIME: [05/10/2019 10:11:03.081]

Average 13.5 seconds

****

**Spring Reactor**

Like Akka Spring is also an Opensource framework.

Project Reactor is Spring's implementation of The Reactive Specification and it's specifically favored by the [Spring Webflux](https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html) module, although you can use it with other modules like [RxJava](https://github.com/ReactiveX/RxJava" \t "_blank).

The idea is to operate Asynchronously with Backpressure using Publishers and Subscribers.

Here, we're being introduced to several new concepts! Let's explain them one by one:

* **Publisher** - A Publisher is a provider of a potentially unbounded number of elements.
* **Subscriber** - A Subscriber listens to that Publisher, asking for new data. Sometimes, it's also referred to as a Consumer.
* **Backpressure** - The ability of the Subscriber to let the Publisher how many requests can it handle at the time. So it's the Subscriber that is responsible for the flow of the data, not the Publisher as it just provides the data.

The Reactor Project offers 2 types of publishers. These are considered the main building blocks of Spring Webflux:

* **Flux** - is a publisher that produces 0 to N values. It could be unbounded. Operations that return multiple elements use this type.
* Mono - is a publisher that produces 0 to 1 value. Operations that return a single element use this type.

**Time metric using 200000 records for both:**

**Spring Reactor:**

-------------------------------------------------------------------------------

Test set: reactivewebconsumer.reactivewebconsumer.ReactiveWebConsumerApplicationTests

-------------------------------------------------------------------------------

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.558 s - in reactivewebconsumer.reactivewebconsumer.ReactiveWebConsumerApplicationTests

INITIAL TIME: 2019-05-10T12:53:24.858Z

FINAL TIME: 2019-05-10T12:59:54.453Z

Average 30 secomds

**Conclusion:**

My personal conclusion, Akka is a more robust framework, because I would choose Akka:

- Best performance

- Higher data load support

- Faster

The Spring Reactor, because I would choose:

- Ease of implementation

- Greater Productivity in Developing

- Easy maintanance