Stream Processing

Programming Reactive Systems

Konrad Malawski, Julien Richard-Foy

Stream processing

A rather broad term, however most commonly associated with:

- processing a number (possibly infinite) of elements
- by pushing/pulling them through a "pipeline"
- such pipeline is composed of operations that modify the elements
- operations often expressed as DSL similar to Scala collections (`map`, `flatMap`, `filter`)

You will often find the term "streaming" be rather overloaded by tools and libraries. We will talk about one specific style of streaming in this course.

Motivation for streaming APIs

- Lots of applications are about processing data
- Data sources can be intermittent or unbounded
- Data has to flow throughout distributed nodes
- **stream processing** = manipulation of data whose sources are intermittent and potentially unbounded

Goals of stream processing

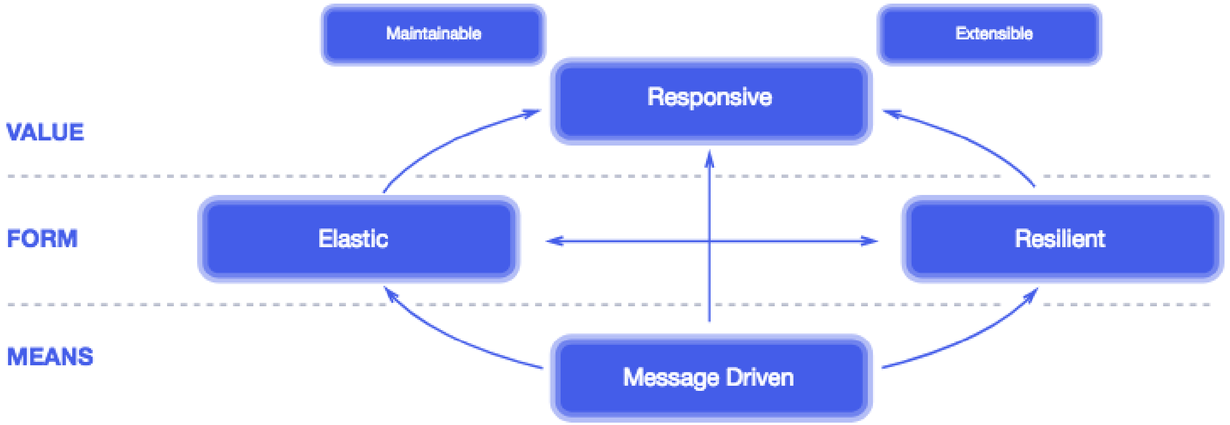
Stream processing aims to address:

- compositional building-blocks
- handle flow-control through such stream pipeline
- process many, possibly infinite, elements at optimal rate

Streaming and the Reactive Manifesto

Since the beginning of this course, we've been looking at the various parts that make a system reactive.

It is important to keep in mind, that *reactive* is not only about streaming, but also the other aspects that were explored before.



Reactive Systems are a super-set of "just" Reactive's streaming aspects.

Asynchronous stream processing

In this course we'll focus on a variant of stream processing that is formalised as the reactive streams specification.

- Reactive Streams were developed independently and collaboratively by Lightbend, Oracle, Pivotal, and RedHat (and more) engineers during 2014-2017
- Reactive Streams are part of Java since Java 9, and there present as the `java.util.concurrent.Flow` interfaces.

Examples

- Tweet stream analysis
- Analysis of logs from a distributed system
- Streaming “APIs” and services in general
- Chat systems, merging many incoming messages to a chat feed
- Grading infrastructure; intermittent unbounded data source: data comes as students submit their solution
- many more...

Streaming and the Reactive Manifesto

Streaming helps in achieving some of the reactive goals, however not all of them. For example, just by having a stream process elements, it does not mean that it is resilient.

- Would it restart processing elements automatically or not?
- Would it let elements be lost, or use snapshotting or acknowledgements to track progress?

These things are core to reactive systems, and indeed easier to achieve once we have streaming, but they are not automatically built-in, just by the fact of using a streaming processing paradigm.

Challenges

To get ourselves into the right problem solving mind-set, consider the following challenges:

- resource efficiency
- flow controlled processing
- failure handling
- separation of business and operational concerns

Summary

In this video we have learnt about:

- the definition of streaming
- how streaming fits into the world of reactive
- what the challenges in streaming systems are