The convergence of stream processing & microservice architecture

Deputy CTO, Lightbend

@viktorklang / v@lightbend.com / viktorklang.com


Lightbend

Overview

History of Data Processing

- Offline Batch processing
- MapReduce
- Lambda Architecture

History of Services

- Client-Server
- CORBA & DCOM
- EJB & N-Tier
- SOA & ESB
- HTTP-API & REST


Today
SAT
MAY

Catch of the day

Live[1] Stream Processing & Microservice Architecture

[1] Near real-time

Shared requirements:

- Available
- Elastically scalable
- Resilient & self-healing
- Loosely coupled & upgradeable

Reactive Systems[2]

Available (responsive), under failure (resilience) and under load (elasticity), by being message-driven.

[2] https://www.reactivemanifesto.org/

Reactive Streams[3]

Provides a standard for asynchronous stream processing with non-blocking back pressure.

[3] www.reactive-streams.com & java.util.concurrent.Flow (JDK9+)

Ingredients

- Stream processing stages: Stateless & Stateful
- Microservices: RPC-style & Evented
- Domain Events with Schema/Codec

#define Events[4]

- Immutable recordings of observations
- Can be disregarded but not be retracted once accepted
- Can be invalidated but (typically) not be deleted
- Knowledge accrues with every processed event

[4] Learn more: https://speakerdeck.com/jboner/designing-events-first-microservices

Services (evented)

$$(Behavior', Effects[(Event*, State']) = Behavior(State, Event)$$


Realization
evented services are stream processing stages

The near future:

Convergence

Is this the solution?

- Alice: So we we lift evented services into the stream processing pipeline ...
- Bob: ... and then we're done!
- <record scratch>
- Narrator: They're forgetting the request-reply use-case which RPC services excel at.

Services (rpc)

$$(Behavior', Effects[State']) = Behavior(State, Command)$$

The improved solution

- lift the service into a stream stage
- lift input Events into Commands and submit
- capture responses & convert into output Events

What if we could derive microservices from data streams?

Ingestion

endpoints

Simple Projection

endpoints

Complex Projection

endpoints

Unification

- Microservices are a part of a streaming pipeline
- A pipeline can now be exposed as Microservices
- We can independently upgrade parts of the pipeline

Developing converged applications

- Deployment Orchestration
- Events-First DDD & EventStorming
- (Meta)Data-driven Continuous Deployment
- Use of the right tools for each component


Beyond

Extend to the Edges

End of stream

Explore what we do at Lightbend:

- Lightbend Fast Data Platform
  - https://www.lightbend.com/products/fast-data-platform
- Lightbend Reactive Platform
  - https://www.lightbend.com/products/reactive-platform
- Lightbend Enterprise Suite
  - https://www.lightbend.com/products/enterprise-suite

Thank you for attending!

The convergence of stream processing & microservice architecture

✓

Deputy CTO, Lightbend

@viktorklang / v@lightbend.com / viktorklang.com