Elixir Way by revisiting a project

@gzzengwei

Starting

- 👨‍💻 **About Me**
  - long time rubist
  - like engineering/ops stuffs
  - getting into FP/elixir
- 🕵️ **About this talk**
  - an ealry elixir project
  - revisiting / improvement
- 🎥 **Disclaimer**
  - learning in progress
  - spark/poc, not all verified in production

Project backgrounds

- 🚜 An importer for CSV files from external source
- 🛎️ Data source is updated daily (cronjob)
- 🔗 There are up to 50+ csv types (different job types)
- 📈 Data is to dump to db directly for BI purpose (downstream target)
- 🎏 CSV schema changes occasionally (exceptions could happen)
- 🖥️ A simple dashboard is required to integrate in main app (web interface)

Choosing the stack

Ruby / Elixir

Demo

- 👻 Simple job process service
- 🎷 Dummy code to demo some of elixir basic modules

So, we are going to start with something very simple ...

Task

- 🎃 conveniences for spawning and awaiting tasks
- 👺 execute one particular action throughout their lifetime
- 🦖 little or no communication with other processes
- 🏄 convert sequential code into concurrent code by computing a value asynchronously
- Simple usage: `start` and `async`

GenServer

- 🏂 A behaviour module for implementing the server of a client-server relation.
- 🌳 A process to keep state, execute code asynchronously and so on
- 💫 Standard set of interface functions, includes tracing and error reporting

Issues

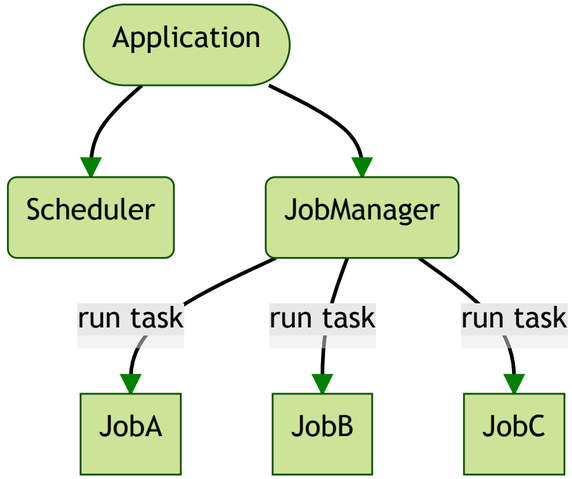
- Often we are running many processes, like many Job types in the example
- When single process dies, and it should not effects others, and its parent process
- process should be isolated when crashed, and restart

Supervisor

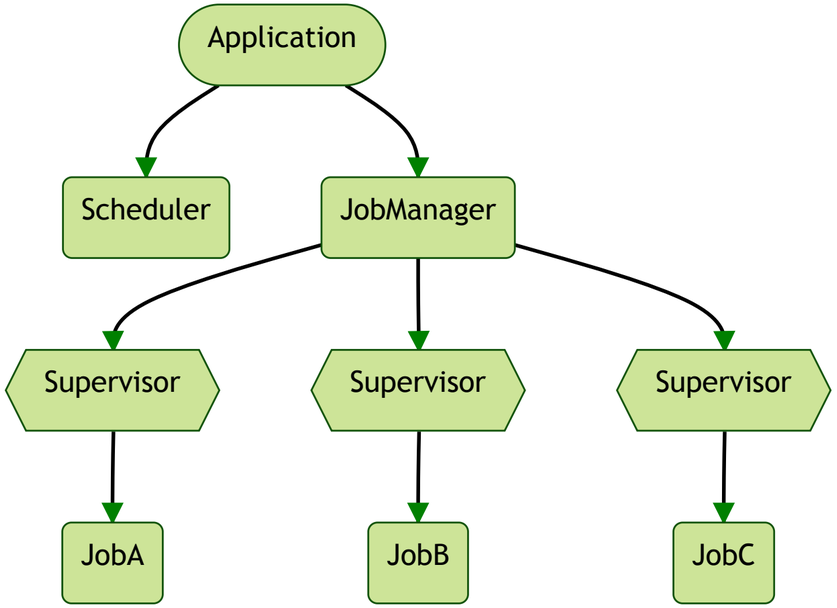
- 🔍 Supervises other processes
- 🌲 supervision tree: hierarchical process structure
- 🔥 provide fault-tolerance/encapsulate how app start/shutdown

Structure Diagram

Current supervision tree



With Supervisor



New Issue

- 💥 Everything works fine, until ...
- 🦀 A big tenant has large csv file(s) in both size and qunatity
- 🥀 Hit by DB error

```
(DBConnection.ConnectionError socket closed (the connection was closed by the pool,
possibly due to a timeout or because the pool has been terminated))
```

- 🍿 Data reach peak once cronjob triggers, and flood the DB with all the concurrent tasks
- 🚦 DB simple cannot handle the load

Revisit the workload

- 🍄 sequential: taking too long and waste of infra resouces
- 🚗 concurrent without control: too much pressures on downstream service in short time
- 🎸 The job actually split into 2 parts
  - **Upstream**: Data fetching(read rows from csv file(s))
  - **Downstream**: Import the rows to db
- 🛫 Split the workflow to (multi) stages that we have control of

GenStage

- 🛸 data-exchange steps that send and/or receive data from other stages
- 🥽 producer: stage sends data; consumer: it receives data;
- 🦷 some stages can be both producer and consumer
- 🌸 back-pressure mechanism: consumer is sending demand upstream, the producer will emit items

Data flow:

[Producer] – data --> [ProducerConsumer] – data --> [Consumer]

Demand flow:

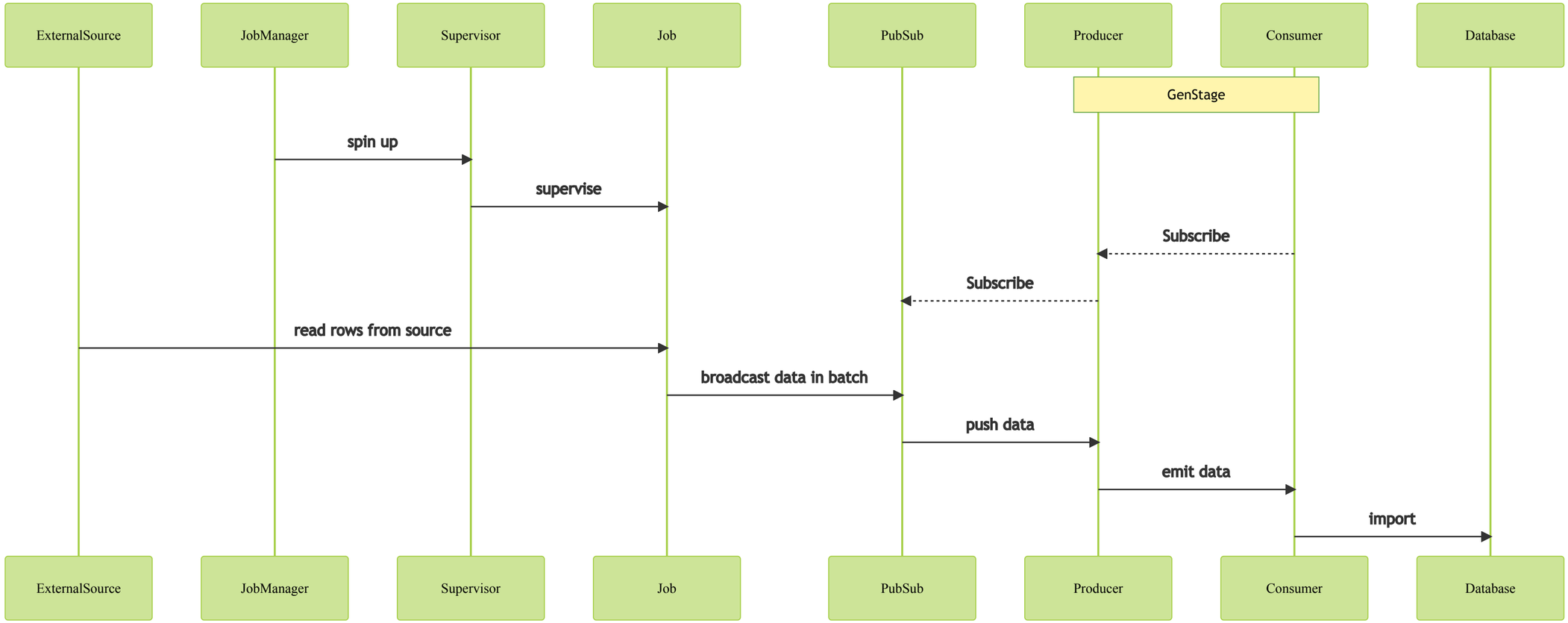
[Producer] <-- ask for data – [ProducerConsumer] <-- ask for data – [Consumer]

Let’s do some spark

- 🐠 **producer** : sending rows from csv file(s)
- 🐢 **consumer** : Importing data to db
- 🐯 issue: data source is passive(trigger by cronjob)
- 🍀 we need something in between reading data in files and emitting data

First version: Pubsub

- 🐙 Why Pubsub
- 🐣 A buffer between Job/Producer



Buffering demand

- 💦 Handle cases that:
  - events arrive and there are no consumers
  - consumers send demand and there are not enough events
- 🌴 link: https://hexdocs.pm/gen_stage/GenStage.html#module-buffering-demand

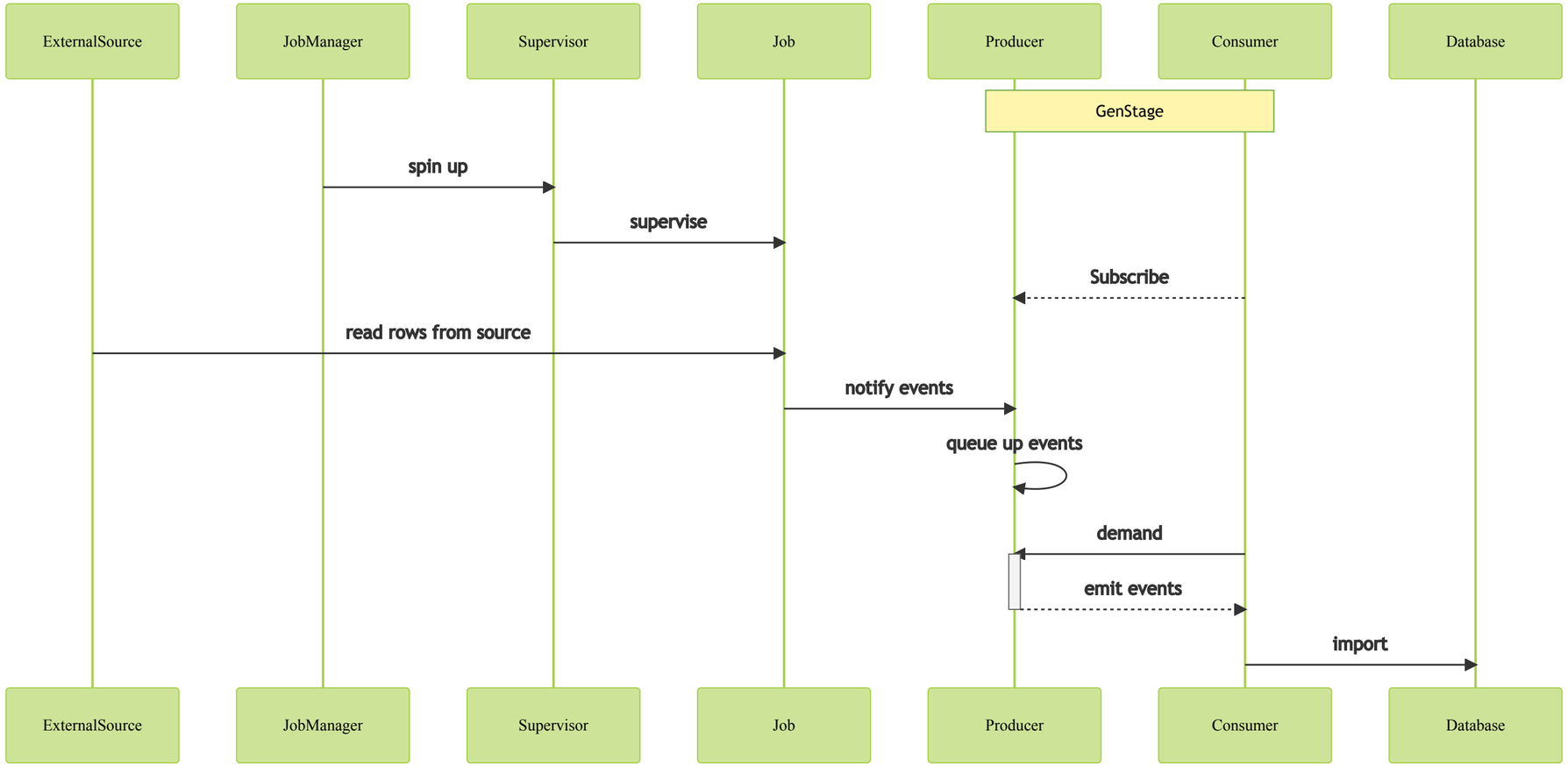
BroadcastDispatcher

- 🐚 Accumulates demand from all consumers before broadcasting events to all of them.
- 🦢 Guarantees that events are dispatched to all consumers within demand range
- Advanced features like `:selector` consumer only subscribe to certain events

Buffering demand (cont.)

BroadcastDispatcher with queue

- 🍋 Use erlang `:queue` module (FIFO)
- 🍏 Control over the events and demand by tracking this data locally



Summary

- 🥯 More understanding Elixir tools
- 🌈 Tools are powerful but they might require learning curve
- 💐 Confidence in its performance
- 🛠️ More to explorer
  - Broadway: Concurrent and multi-stage data ingestion and data processing
  - Commanded: CQRS/EventSourcing

Reference

Book by *Svilen Gospodinov*, **Concurrent Data Processing in Elixir**

Thank You

Questions ?