Actor based concurrency with Elixir

...

Get ready for soft real-time systems

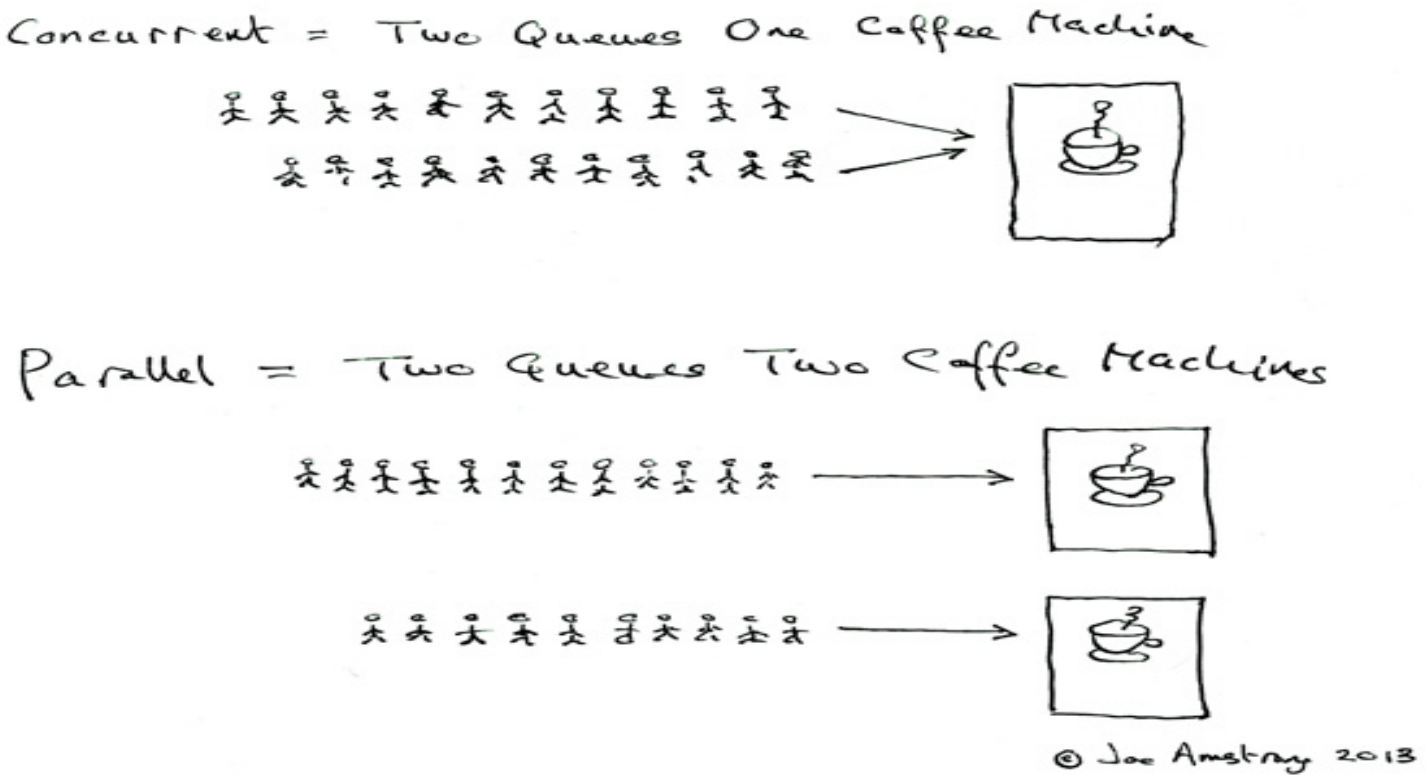
By Adrian Magdas

Co-Founder & Lead Developer @ Crafting Software Innovation
adrian@craftingsoftware.com

Agenda

1. Concurrency vs. parallelism
2. Why Functional Programming
3. Actor model concurrency
4. Why Elixir & Erlang
5. Short introduction to Elixir
6. Elixir agents
7. OTP patterns for handling agents: GenServer & Supervisor

Concurrency vs. parallelism



Why Functional Programming

- **Handling concurrency in imperative languages with shared mutable state is hard to get right, a lot of well known issues: race conditions, deadlocks**
- **FP avoids the problems associated with shared mutable state by forcing immutability**
- **Usual properties for a functional language**: functions as first class citizens, referential transparency, immutability, persistent data structures, pattern matching, pure functions, data separated from transformations

Actor model concurrency

The actor model in computer science is a mathematical model of concurrent computation that treats "actors" as the universal primitives of concurrent computation. In response to a message that it receives, an actor can: make local decisions, create more actors, send more messages, and determine how to respond to the next message received. Actors may modify private state, but can only affect each other through messages (avoiding the need for any locks). (Wikipedia)

Why Erlang

- **Erlang and Elixir are impure, dynamically typed functional languages**
- **Both run on the BEAM (Erlang VM) which has built-in concurrency, distribution and scalability, actors are part of the language itself**
- **SMP support since 2006**
- **Built-in fault tolerance: 99.99999% uptime**

Why Erlang(cont)

- **Battle tested: ~50% of all mobile comms go through Erlang boxes**
- **Happy coincidence: the solution to telecom problems for reliability and scalability are a perfect match for the same problems the web and IOT are facing**
- **2 million web servers vs. 1 server with 2 million connections**

Why Elixir

- **Modern language with solid additions to Erlang: protocols, macros, sigils, pipe operator, great tooling**
- **Beauty and expressiveness of Python & Ruby, power and awesomeness of Erlang**
- **Compatible with existing Erlang infrastructure and libraries**

Short introduction to Elixir

Let’s see some syntax and language data types to be able to understand the examples

CODE

Elixir agents

- **Agents in Elixir are regular light weight processes**
- **In Elixir, all code runs inside processes. Processes are isolated from each other, run concurrent to one another and communicate via message passing.**
- **Messages are sent asynchronously and stored in mailboxes**
- **Used for keeping state or running asynchronous code**
- **CODE**

GenServer

A behaviour module for implementing the server of a client-server relation.

A GenServer is a process like any other Elixir process and it can be used to keep state, execute code asynchronously and so on. The advantage of using a generic server process (GenServer) implemented using this module is that it will have a standard set of interface functions and include functionality for tracing and error reporting. It will also fit into a supervision tree.

Supervisor

- **Process that supervises other processes(child processes)**
- **Fits into a supervision tree**
- **Multiple supervision strategies: :one_for_one, :one_for_all, :rest_for_one, :simple_one_for_one**

Take-aways

Functional programming: functions as first class citizens, referential transparency, immutability, persistent data structures, pattern matching, pure functions, data separated from transformations

Actor based concurrency: light weight processes, no shared state between actors, process linking and monitoring, all the communication happens by sending messages

OTP: patterns for handling process creation, supervision strategies and fault-tolerance support

There are no silver bullets :)

Thank you

Develop with passion!

Resources

https://pragprog.com/book/elixir/programming-elixir

http://elixir-lang.org/

http://elixirschool.com/

http://learnyousomeerlang.com/what-is-otp

http://learnyousomeerlang.com/the-hitchhikers-guide-to-concurrency

https://joearms.github.io/2013/04/05/concurrent-and-parallel-programming.html