Dive into Elixir

When programming is fun again

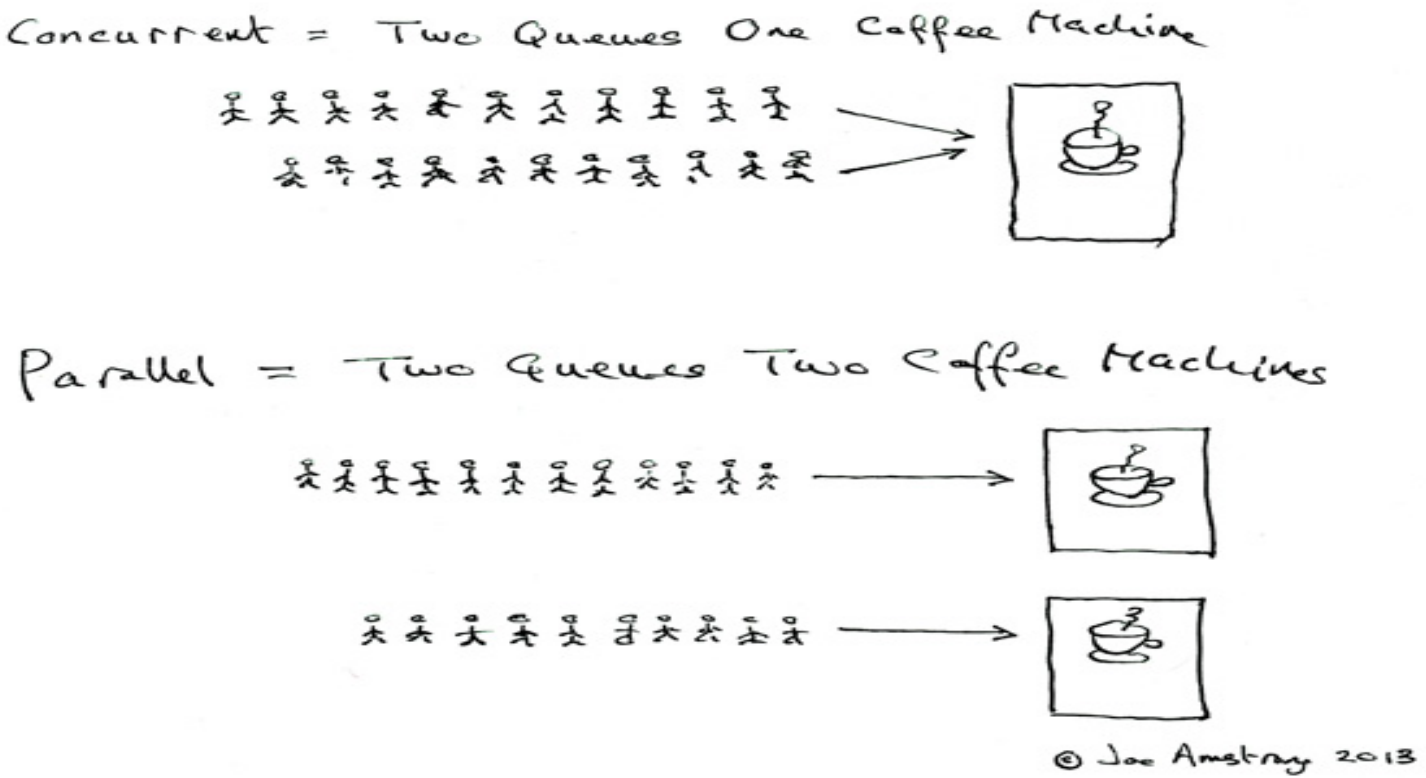
By Adrian Magdas

Co-Founder & Lead Developer @ Crafting Software
adrian@craftingsoftware.com
@zenCrafter
linkedin.com/in/adrianmagdas

Agenda

1. Functional programming principles
2. Why Elixir & Erlang
3. Elixir language basics
4. Anatomy of an Elixir application

Concurrency vs. parallelism



Functional programming principles

- Pure functions and higher order functions
- Immutability and persistent data types
- Support for recursion or CSP
- Pattern matching
- Focus on data transformation, pipe is king
- = is a match operator, if the match succeeds, it binds a value to some name(s)

Why Erlang

- Philosophy of Erlang: we assume that errors will occur, processes will crash and machines WILL FAIL
- Focus on cure instead of prevention
- Error handling: remote detection and handling of errors
- 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

Why Erlang

- Preemptive scheduling & share nothing
- SMP support since 2006
- Built-in fault tolerance: 99.99999% uptime
- 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

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
- Better model for managing state than objects

Elixir tools

- mix
- lex
- observer
- credo
- dialyzer

Take-aways

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

Elixir&Erlang: suited for software systems, long lived and complex

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