PROGRAMOWANIE FUNKCYJNE

Bartosz Górka
ITI 127228

Plan prezentacji

- *Języki funkcyjne - co to takiego?*
- *Erlang*
- *Elixir*
- *Aplikacje i rozwiązania programistyczne*
- *Języki funkcyjne w uczeniu maszynowym*

Programowanie funkcyjne

- *Odmiana programowania deklaratywnego*
- *Programowanie za pomocą wyrażeń, nie instrukcji*
- *Unikanie zmian stanu i mutacji*
- *Eliminacja side-effects poprzez idempotentne funkcje*
- *Wywodzi się z rachunku lambda*

Języki funkcyjne

- *Scala*
- *R (statystyka)*
- *Haskell*
- *Erlang*
- *Elixir*
- *Lisp*
- *JavaScript*, Python*, Ruby**

Założenia

Funkcja wyższego rzędu

- *Funkcja przyjmuje bądź zwraca jako argument inną funkcję*
- *Dostarczanie generycznych algorytmów, parametryzowanych przez programistę*

Założenia

Czyste funkcje i wyrażenia

- *Bez skutków ubocznych (pamięć bądź operacje I/O)*
- *Używane do optymalizacji kodu*
- *Dla niezależnych, kolejność wykonania może być optymalizowana*

Założenia

Rekurencja

- *Większość języków pozwala na nieskończoną* rekurencję*
- *Dane nie podlegają mutacji - zwracane poprzez wynik i akumulowane*

Założenia

Przejrzystość referencji

- *Brak operacji przypisania*
- *Wartość zmiennej nigdy nie ulega zmianie po zdefiniowaniu*
- *Eliminacja efektów ubocznych, referencji*

Object-Oriented Programming patterns?

OO pattern/principle	FP equivalent
• Single Responsibility Principle	• Functions
• Open/Closed principle	• Functions
• Dependency Inversion Principle	• Functions, also
• Interface Segregation Principle	• Functions
• Factory pattern	• You will be assimilated!
• Strategy pattern	• Functions again
• Decorator pattern	• Functions
• Visitor pattern	• Resistance is futile!

Seriously, FP patterns are different

Functional programming design patterns by Scott Wlaschin [NDC London 2014]

OOP in Functional Programming by Wojtek Mach [ElixirConfEU 2016]

FP ~~vs~~ and OOP

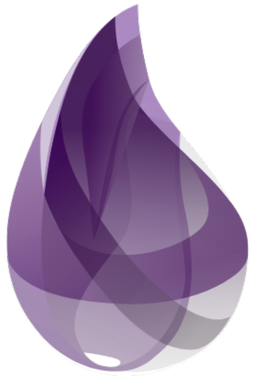
Zastosowania w przemyśle

- *Erlang - Ericsson [Open Telecom Platform]*
- *Facebook, Amazon, Heroku*
- *WhatsApp*
- *Scala - MapReduce*

Erlang

- *Język programowania dla rozwiązań współbieżnych*
- *Obsługa rozproszonych systemów wymagających długotrwałej pracy*
- *Wysoka odporność na awarie*
- *Wspiera hot-swamping kodu*
- *Twórca Joe Armstrong, rok 1986*

Elixir



- *Język programowania dla rozwiązań współbieżnych, skalowalnych*
- *Funkcyjny, dynamiczny*
- *Idea lekkich procesów*
- *Procesy izolowane, wymiana poprzez komunikaty*
- *Uruchamiany na maszynie wirtualnej Erlanga*
- *Twórca José Valim, rok 2012*

Elixir

- ***Typy danych**: Integers, Floats, Atoms (Symbols), Ranges*
- ***Kolekcje**: Tuples, Linked Lists, Binaries, Maps*
- ***Typy systemowe**: PIDs, Ports*
- *Funkcje*
- *Pattern Matching*
- *Guards*
- *“Magiczny” |>*

Ciąg Fibonacciego

```erlang
-module(fib).
-export([fib/1]).

fib(1) -> 1;
fib(2) -> 1;
fib(N) -> fib(N - 2) + fib(N - 1).
```

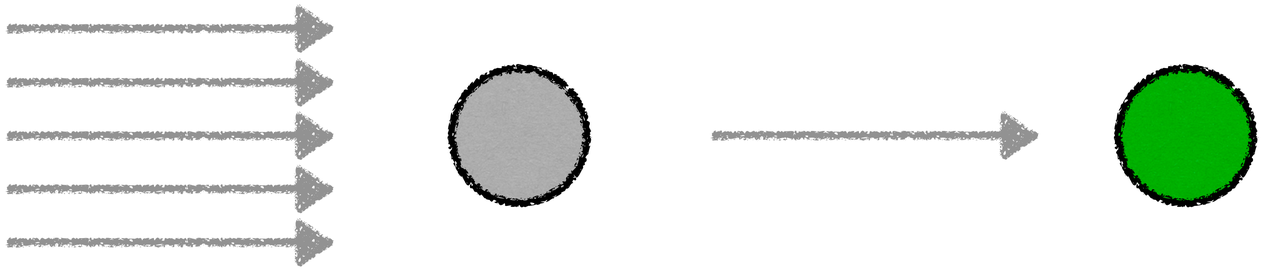
```elixir
defmodule Fibonacci do
  def fib(0), do: 0
  def fib(1), do: 1
  def fib(n), do: fib(n-1) + fib(n-2)
end
```

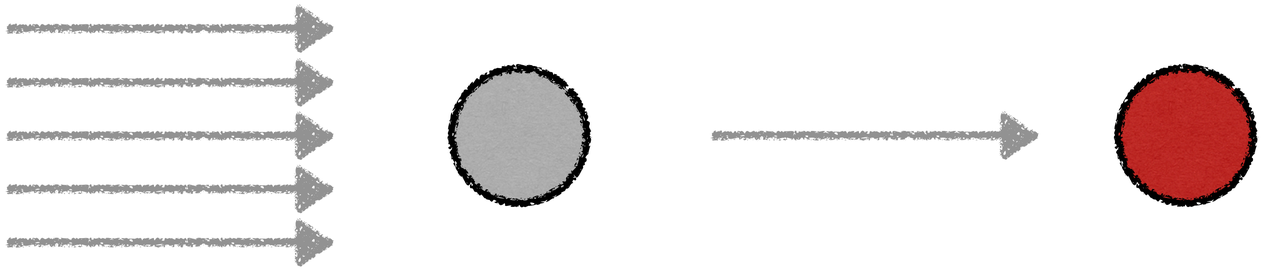
Sumowanie z cache

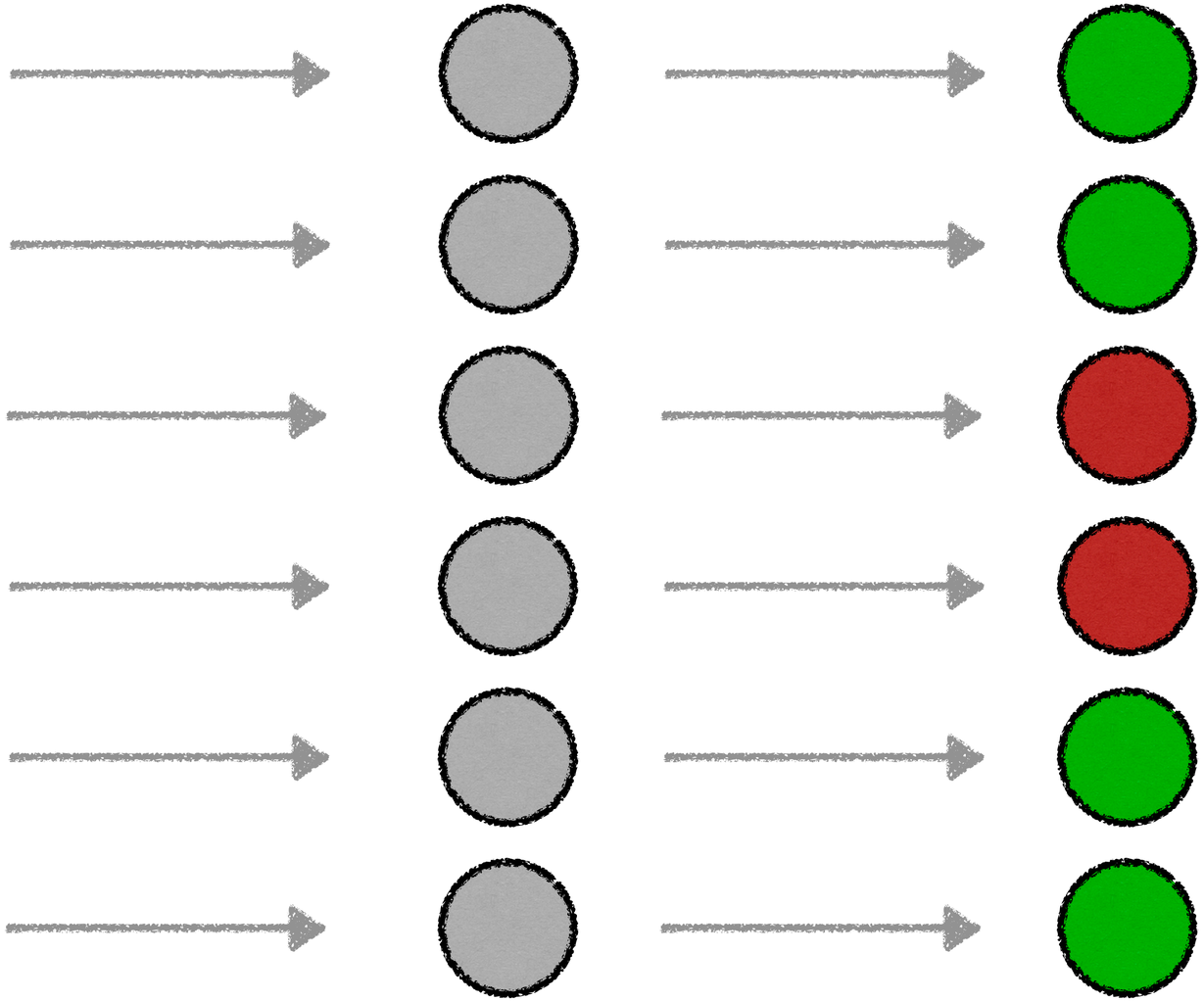
```
-module(sum_server).
-behaviour(gen_server).
-export([
start/0, sum/3,
init/1, handle_call/3, handle_cast/2, handle_info/2,
terminate/2,
code_change/3
]).

start() -> gen_server:start(?MODULE, [], []).
sum(Server, A, B) -> gen_server:call(Server, {sum,
A, B}).
init(_) -> {ok, undefined}.
handle_call({sum, A, B}, _From, State) -> {reply, A +
B, State}.
handle_cast(_Msg, State) -> {noreply, State}.
handle_info(_Info, State) -> {noreply, State}.
terminate(_Reason, _State) -> ok.
code_change(_OldVsn, State, _Extra) -> {ok, State}.
```

```
defmodule SumServer do
 use GenServer

 def start() do
    __MODULE__
    |> GenServer.start(nil)
 end

 def sum(server, a, b) do
  GenServer.call(server, {:sum, a, b})
 end

 def handle_call({:sum, a, b}, _from, state) do
  {:reply, a + b, state}
 end
end
```

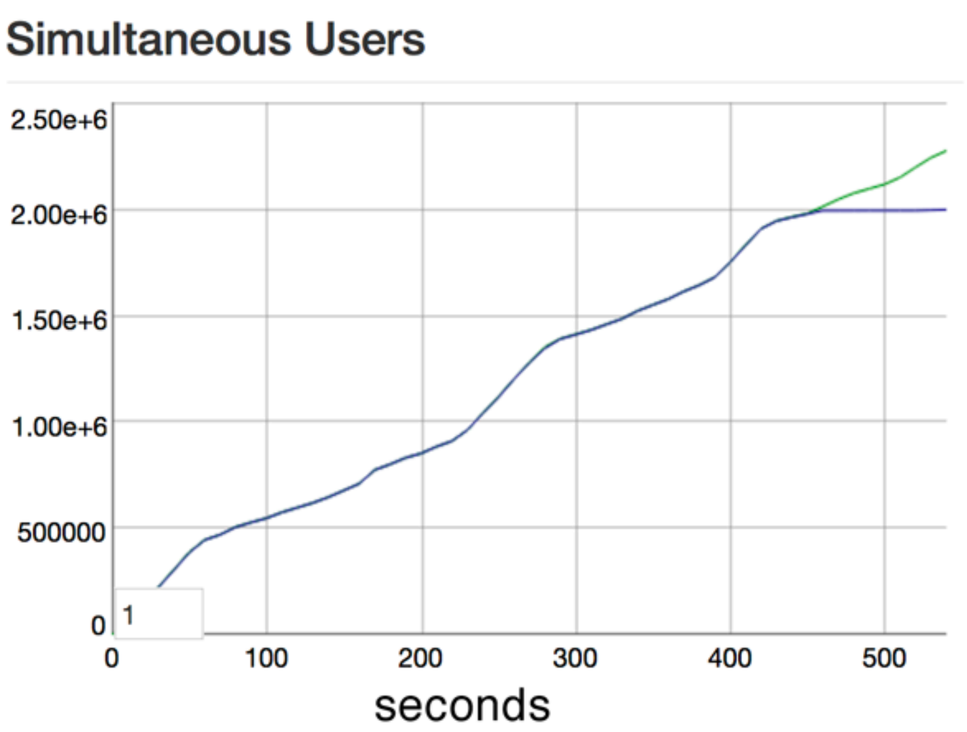
Procesy

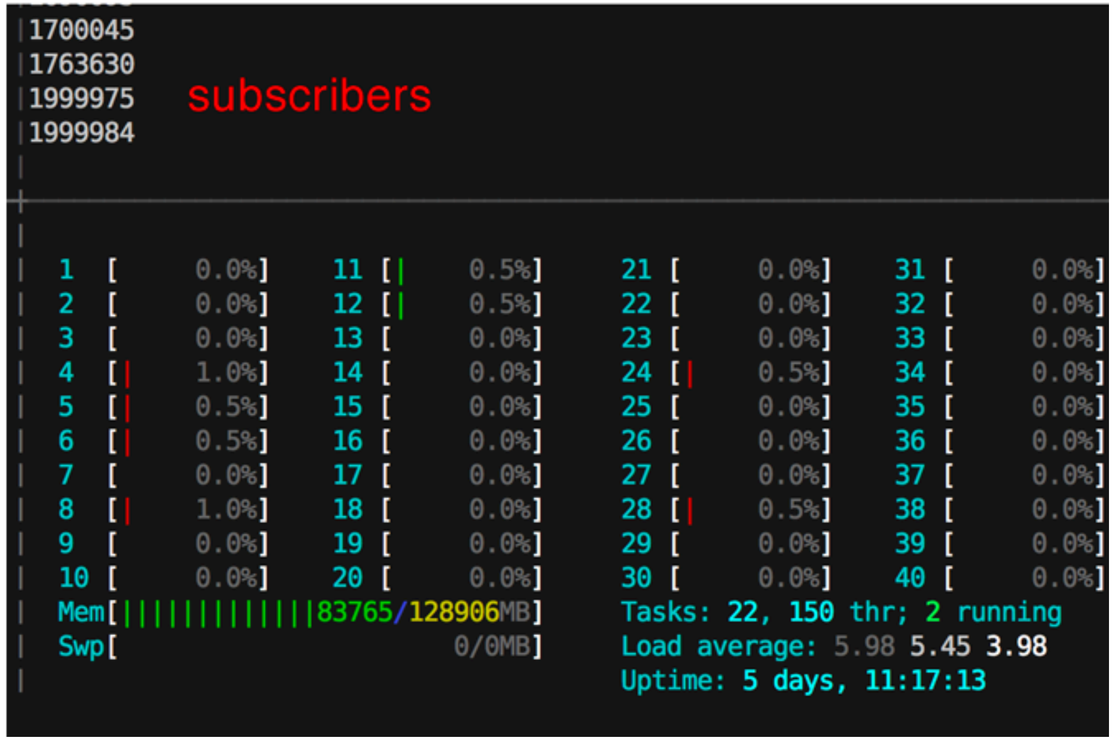






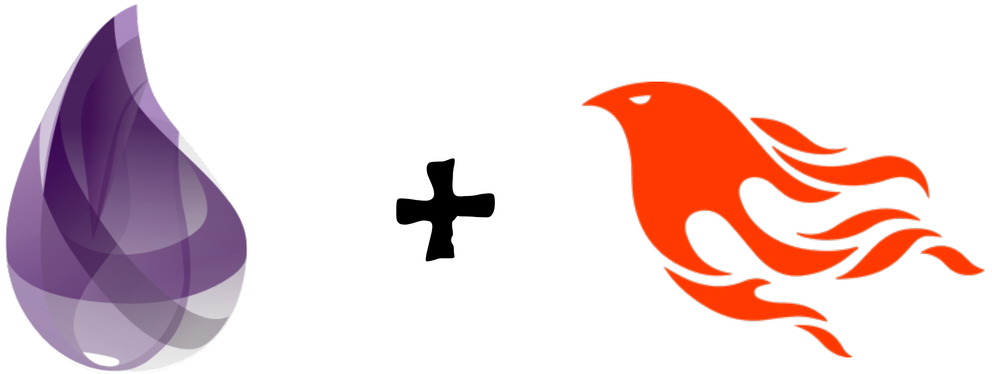
Phoenix Framework

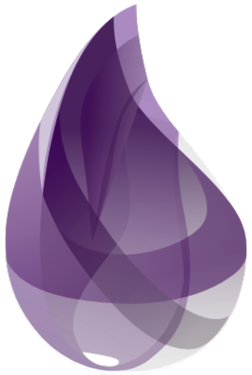


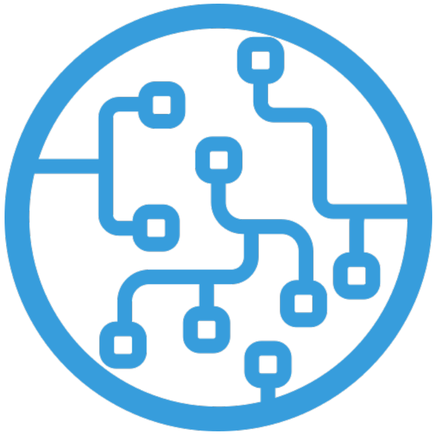


The Road to 2 Million Websocket Connections in Phoenix

BartoszGorka96







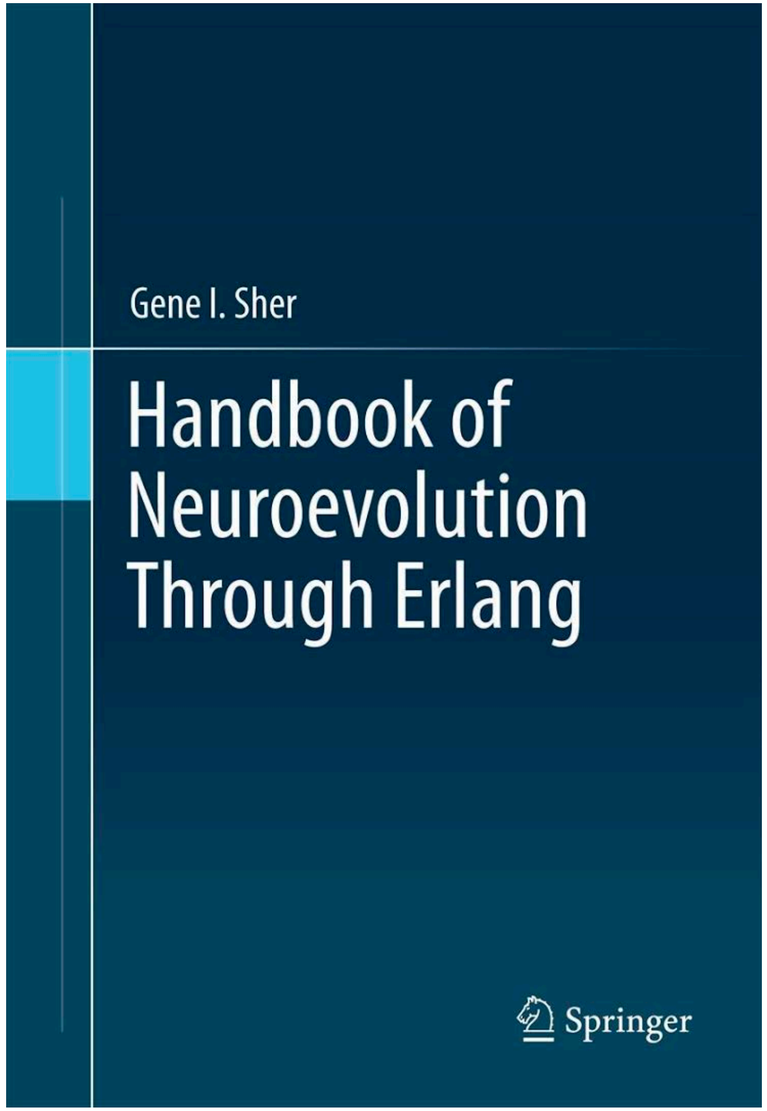
TensorFlex

- ***Tensorflow** dla języka Elixir*
- *Powstały w ramach Google Summer of Code 2018*
- *Inspiracja frameworkiem **Keras** dla języka Python*
- *NIFs (Native Implemented Functions) we współpracy z Tensorflow C API*
- *Efektywny i przystępny kod elixirowy dla uczenia mazynowego*

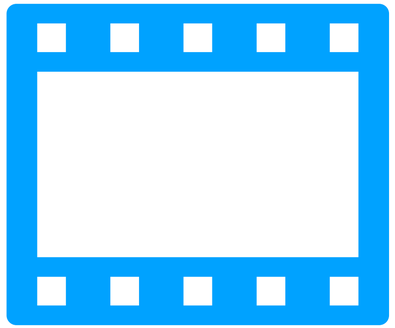
Problemy

- *Wolna arytmetyka zmiennopozycyjna w BEAM*
- *Użycie NIFs likwiduje ten problem, lecz tracimy współbieżność*
- *Mała społeczność pracująca nad ML*
- *Brak SciPy, Numpy albo ich podobieństw*

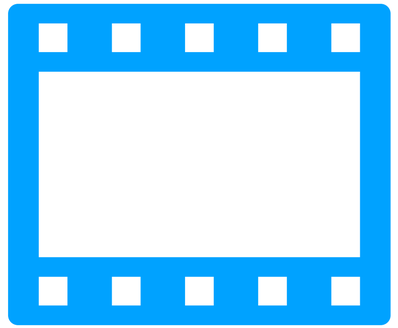
Problemy


Gene I. Sher
Handbook of
Neuroevolution
Through Erlang
Springer

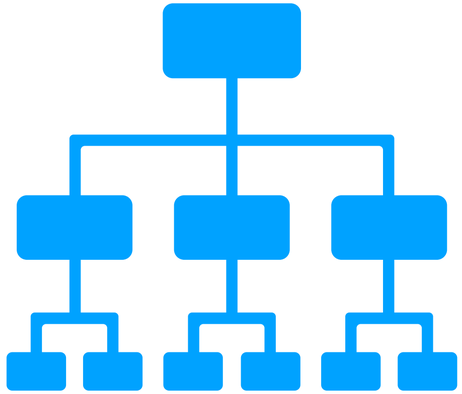
Eksperymenty



Machine Learning with Elixir and Phoenix by Eric Weinstein
ElixirConf EU 2017



Intro to Machine Learning by Jesse J. Anderson
Code Beam SF 2018



Deep Learning with Elixir:
Building and Training a Multi-Layered Neural Network

BartoszGorka96

Pytania?

Feedback?

Dziękuję za uwagę!

BartoszGorka96

Referencje

- *What is Erlang*
- *OTP in Erlang*
- *Elixir Language*
- *FP is not opposite to OOP*
- *Google Summer of Code 2018*
- *Tensorflex introduction*
- *Elixir in Action, second edition* *ISBN 9781617295027*