## Erlang for Haskellers

Hynek Vychodil (vychodil.hynek@gmail.com)

November 25, 2015

## Erlang: The Name

Bjarne Däcker named language after

- Agner Krarup Erlang (1878 1929) Danish mathematician – queueing theory, unit, . . .
- Ericsson Language

## Erlang: The History I

- 1981 Ericsson research project to find better ways of programming Telecom applications
- 1985 Joe Armstrong joins the lab
- 1987 First Erlang experiments in Prolog
- 1988 Erlang escapes from the lab!
- 1989 Erlang proved  $10 \times$  more productive than PLEX. JAM Joe's Abstract Machine.
- 1990 Erlang: The Movie
- 1993 Distribution to run a homogeneous Erlang system on a heterogeneous hardware. Erlang gets own organisation Erlang System AB in Ericsson and is sold outside Ericsson. Bogumil (Bogdan) Hausman makes BEAM.

## Erlang: The History II

- 1995 Project AXE-N (in C++) collapse, project ADX using Erlang started.
- 1997 Open Telecom Platform (OTP)
- 1998 ADX301 announced the largest functional code industrial software project (1.13 millions lines of Erlang code in 2001).
  - Erlang was banned within Ericsson Radio AB for new product development Erlang escapes Ericsson Open Source Erlang.
- 2001 HiPE High Performance Erlang.
- 2006 SMP BEAM
- 2015 Maps



Erlang: The Thesis

# Making reliable distributed systems in the presence of software errors

Joe Armstrong

The Royal Institute of Technology Stockholm, Sweden December 2003

Erlang: The Movie

Seriously, watch it!

Erlang: The Goal

## Reliability

## Erlang: The Solution

#### Concurrency Oriented Programming Language (COPL)

- Must support processes
- Processes are strongly isolated
- Processes are uniquely identified
- Processes communicates by messages (no shared state)
- Message passing is unreliable (no delivery guarantied)
- Opening of another process
  Opening of another process

## Erlang: The Kind

COPL with processes implemented in functional language

- Strict evaluation
- Dynamic typing
- Strict type inference (only integer to float)
- Single assignment
- Immutable data

## Erlang: The Types

#### Primitive

- Number (Integer, Float)
- Atom
- Bitstring, Binary
- Reference
- Fun
- Port
- Pid

## Erlang: The Types

#### Compound

- Tuple
- Map
- List

#### Sugar

- String List of Integers
- Record Tuple
- Boolean Atom false and true

## Erlang: The Type Check

But we like static type check. No problem, we have dialyzer.

## Erlang: The Syntax

#### Prolog roots

```
-module(par_fact).
-export([calc/1]).
fact(X) \rightarrow fact(X, 1).
fact(0, R) \rightarrow R;
fact(X, R) when X > 0 \rightarrow fact(X-1, R*X).
calc(N) \rightarrow
    Parent = self().
    Pids = [ spawn_link(fun() ->
                       Parent ! {self(), {X, fact(X)}}
                  end)
              || X <- lists:seq(1, N) ],
    [ receive {Pid, R} -> R end || Pid <- Pids ].
```

## Erlang: The Pattern Matching

Best thing after sliced bread. Well Haskellers know little bit.

## Erlang: The Error Handling

#### Classes

- Error
- Exit
- Throw

#### Handling

- try ... catch
- Links (red wire) signals and traps
- Monitors

## Erlang: The Hot Code Swap

- Load module
- ② Call Module:Function

There can be only two running version.

Thanks.
Any questions?