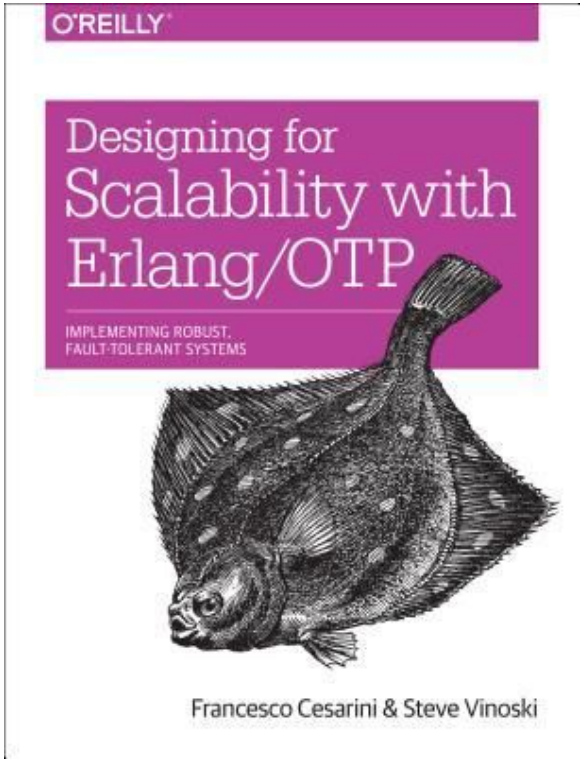
Francesco Cesarini
@francescoC

Founder & Technical Director @
Erlang Solutions

WhatsApp's Secret Sauce

An Introduction to Erlang

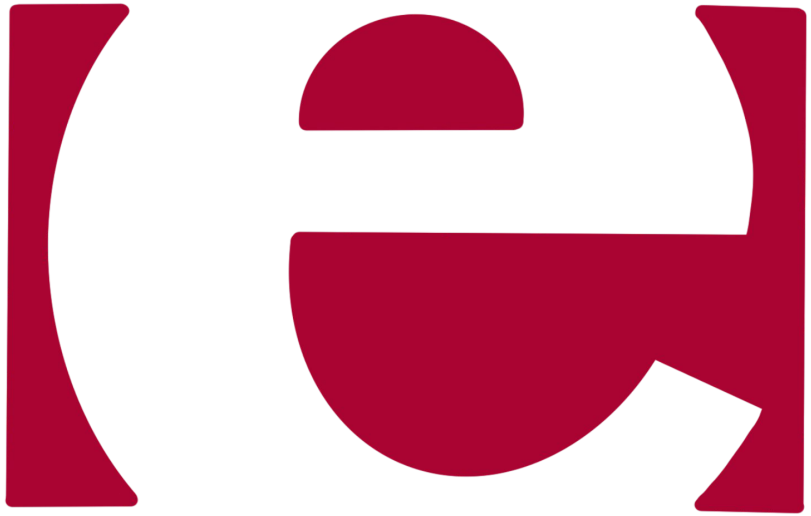


ERICSSON

Erlang
SOLUTIONS

UNIVERSITY OF
OXFORD

Erlang
SOLUTIONS



ERLANG

Java

Erlang SOLUTIONS

PROGRAMMING LANGUAGE ECOSYSTEMS HAVE TO BE:

PREDICTABLE

EASY TO USE

EASY TO MAINTAIN

WhatsApp Acquisition by Facebook

WhatsApp

Who is using Erlang?

Telecom Applications

Complex

No down time

Scalable

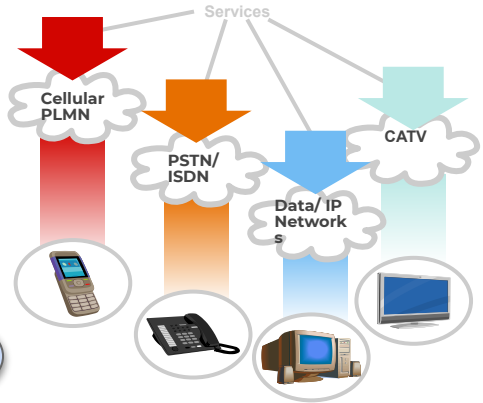
Maintainable

Distributed

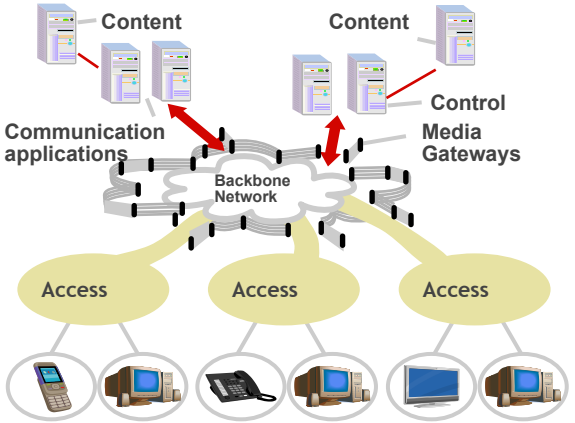
vs

Time to Market

Past
Single-service networks



Present
Multiservice networks/client server



Erlang SOLUTIONS

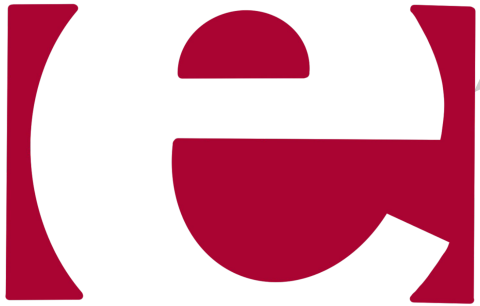
www.erlang-solutions.com

The Ancestors

Functional languages like ML or Miranda

Languages like SmallTalk,

Ada, Modula or Chill



ERLANG

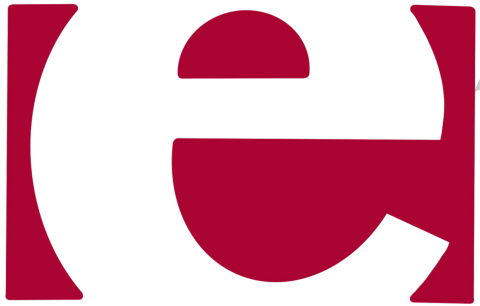
Logical languages like Prolog

The Ancestors

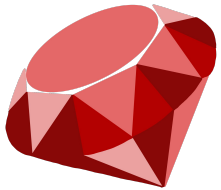
Functional languages like ML or Miranda

Languages like SmallTalk, Ada, Modula or Chill

Logical languages like Prolog



ERLANG

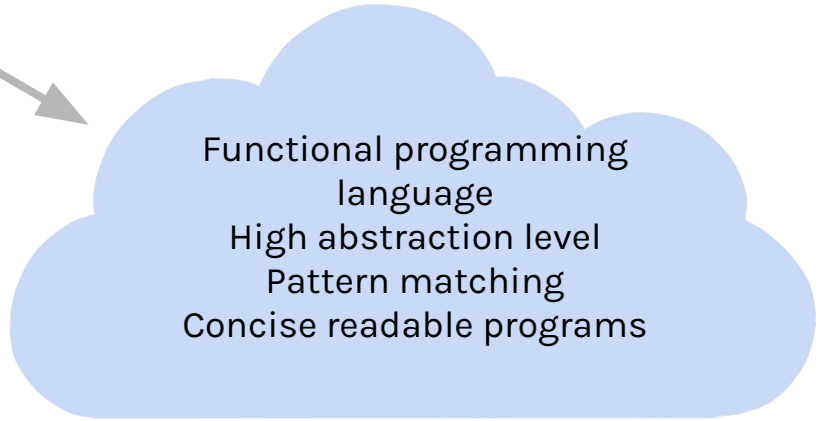


RUBY

ELIXIR

Erlang Highlights

- Declarative
- Concurrent
- Robust
- Distributed
- Hot code loading
- Multicore Support
- OTP



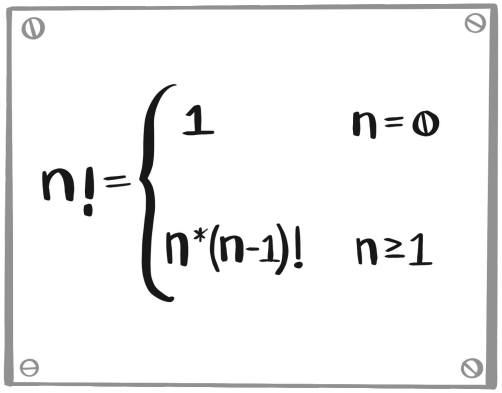
Erlang SOLUTIONS

www.erlang-solutions.com

Erlang Highlights: Factorial

Factorial using Recursion

Definition




```
Eshell V5.0.1  (abort with ^G)
1> c(ex1).
{ok,ex1}
2> ex1:factorial(6).
720
```

Implementation

```
-module(ex1).
-export([factorial/1]).

factorial(0) ->
   1;
factorial(N) when N >= 1 ->
   N * factorial(N-1).
```

Erlang SOLUTIONS

www.erlang-solutions.com

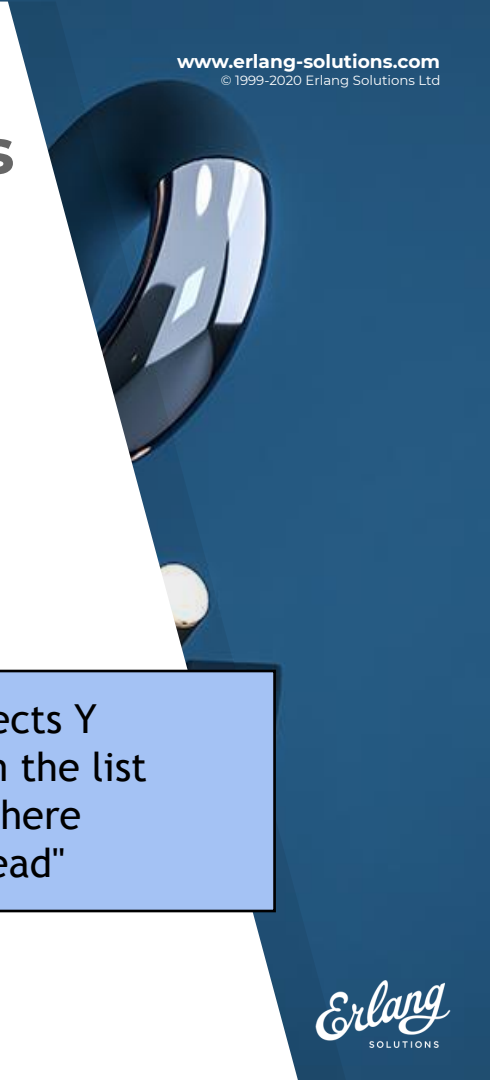
Erlang Highlights: High-level Constructs

QuickSort using List Comprehensions

```
module(ex2).
-export([qsort/1]).

qsort([Head|Tail]) ->
  First = qsort([X || X <- Tail, X =< Head]),
  Last  = qsort([Y || Y <- Tail, Y > Head]),
  First ++ [Head] ++ Last;
qsort([]) ->
[].
```

"all objects Y taken from the list Tail, where Y > Head"

```
Eshell V5.0.1  (abort with ^G)
1> c(ex2).
{ok,ex2}
2> ex2:qsort([7,5,3,8,1]).
[1,3,5,7,8]
```



Erlang Highlights: High-level Constructs

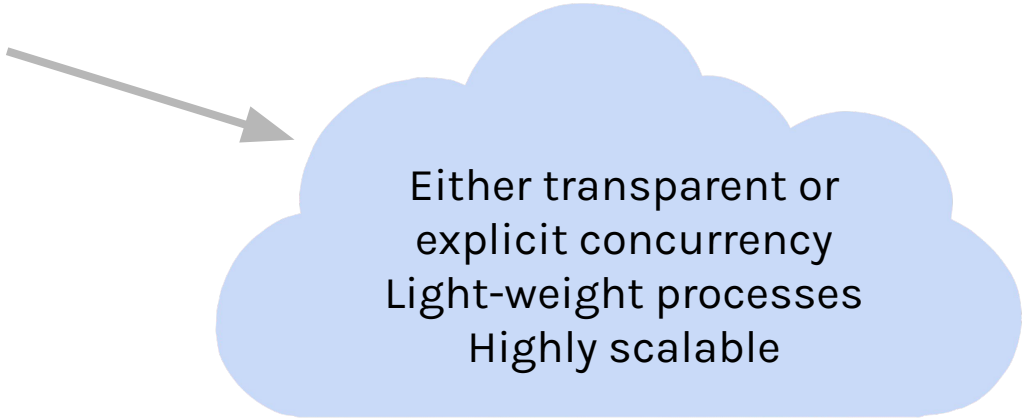
Parsing a TCP packet using the Bit Syntax

```
<< SourcePort:16, DestinationPort:16, SequenceNumber:32,
   AckNumber:32, DataOffset:4, _Reserved:4, Flags:8,
   WindowSize:16, Checksum:16, UrgentPointer:16,
   Payload/binary>> = Segment,

OptSize = (DataOffset - 5)*32,
<< Options:OptSize, Message/binary >> = Payload,
<< CWR:1, ECE:1, URG:1, ACK:1, PSH:1,
   RST:1, SYN:1, FIN:1>> = <<Flags:8>>,

%% Can now process the Message according to the
%% Options (if any) and the flags CWR, ..., FIN
```

etc…

Erlang Highlights

- Declarative
- Concurrent
- Robust
- Distributed
- Hot code loading
- Multicore Support
- OTP



Erlang Highlights: Concurrency

Creating a new process using spawn

```
-module(ex3).
-export([activity/3]).

activity(Name,Pos,Size) ->
    ..........
```

```
activity(Joe,75,1024)
```

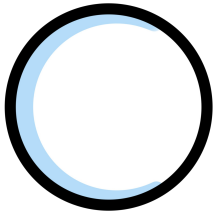
```
Pid = spawn(ex3,activity,[Joe,75,1024])
```

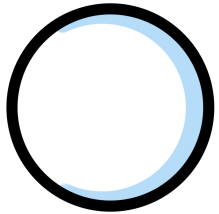
Erlang SOLUTIONS

www.erlang-solutions.com

Erlang Highlights: Concurrency

Processes communicate by asynchronous message passing





```
Pid ! {data,12,13}
```

```
receive
    {start} -> ……
    {stop} -> ……
    {data,X,Y} -> ……
end
```

Erlang SOLUTIONS

www.erlang-solutions.com

Erlang Highlights

Declarative

Concurrent

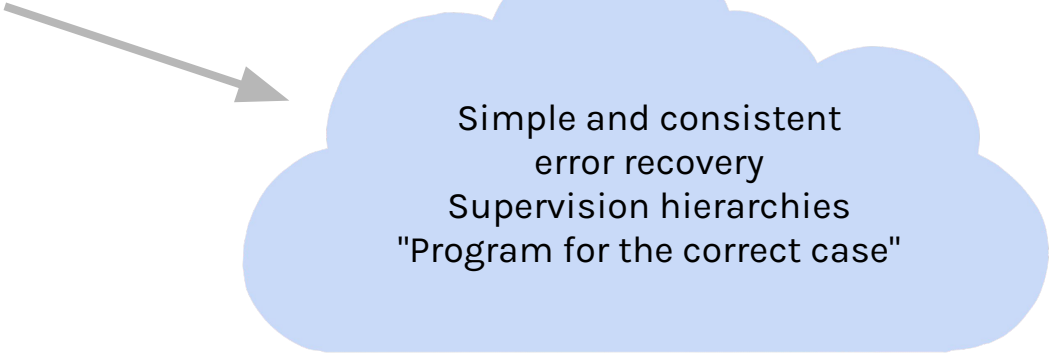
Robust

Distributed

Hot code loading

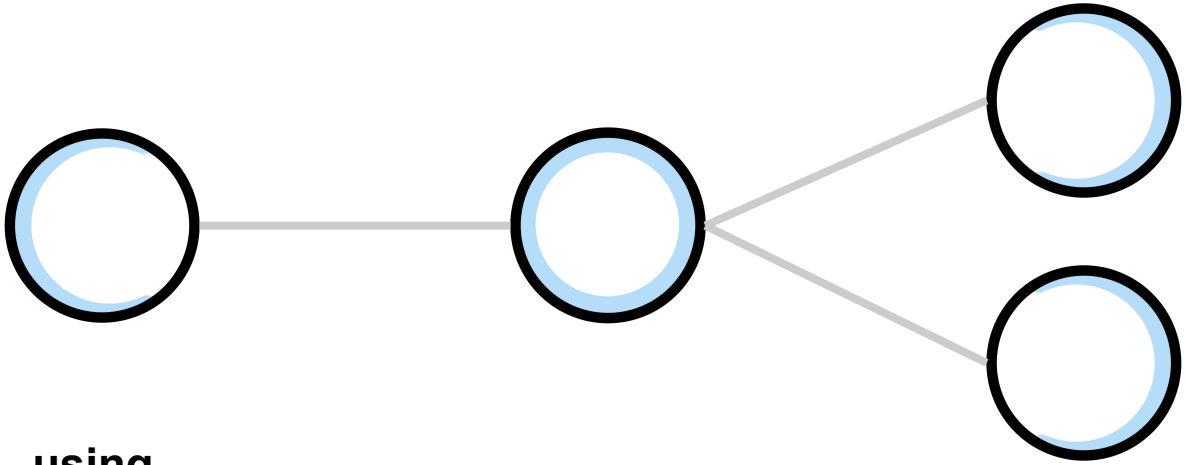
Multicore Support

OTP



Erlang Highlights: Robustness

Cooperating processes may be linked together



using

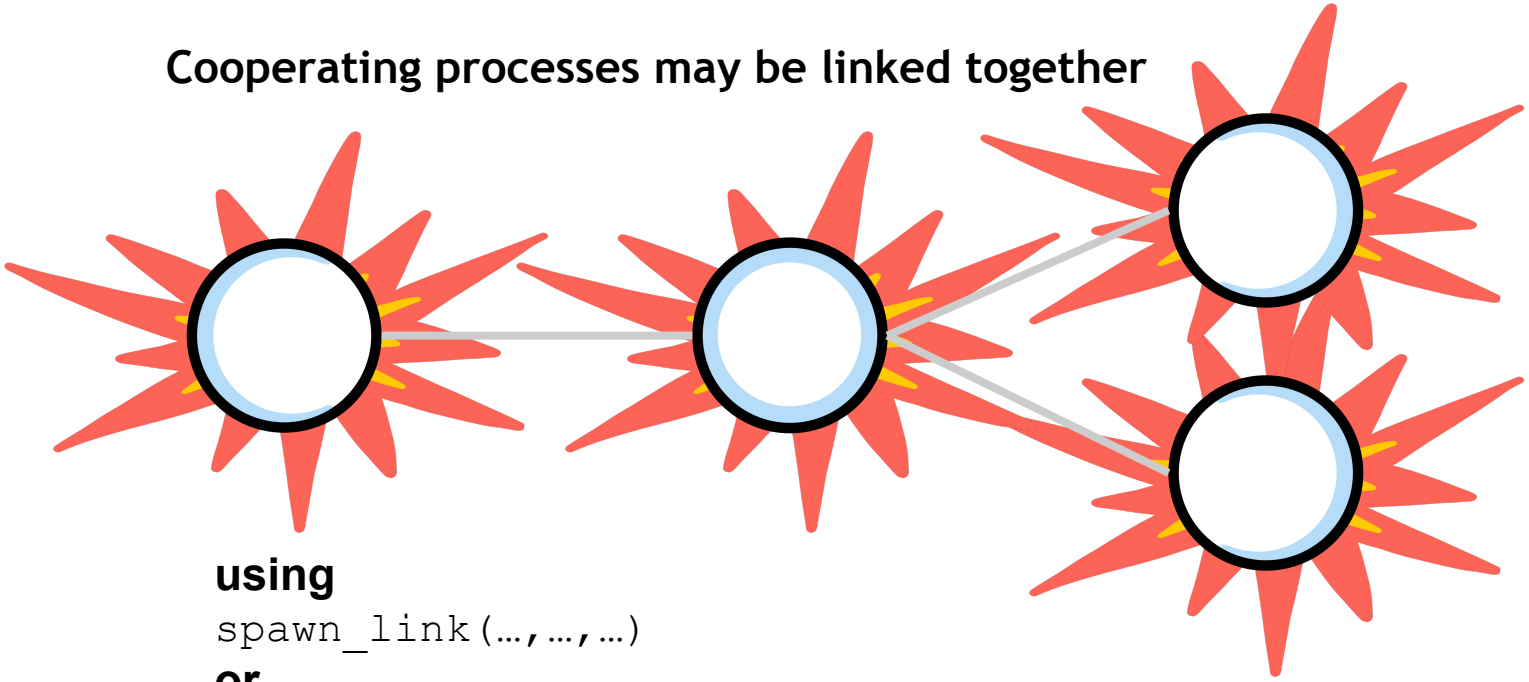
`spawn_link(…,…,…)`

or

`link(Pid)`

Erlang Highlights: Robustness

Cooperating processes may be linked together



using

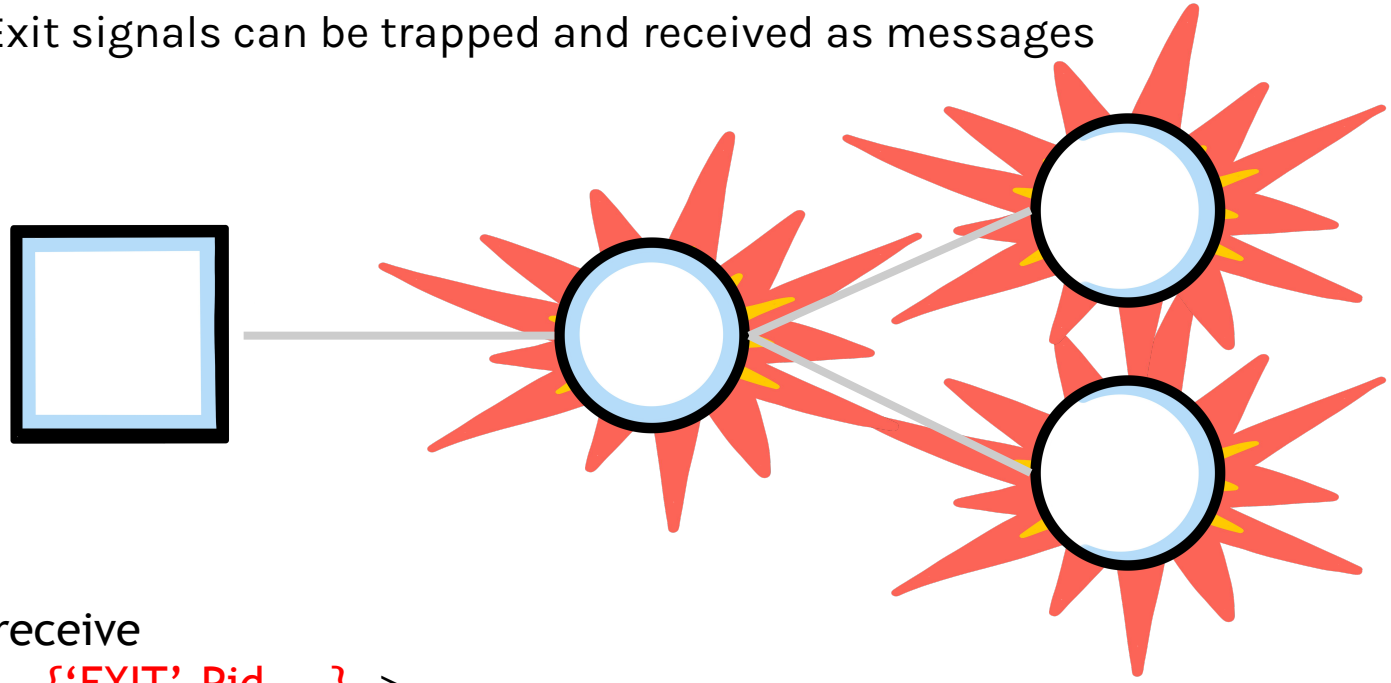
```
spawn_link(…,…,…)
```

or

```
link(Pid)
```

Erlang Highlights: Robustness

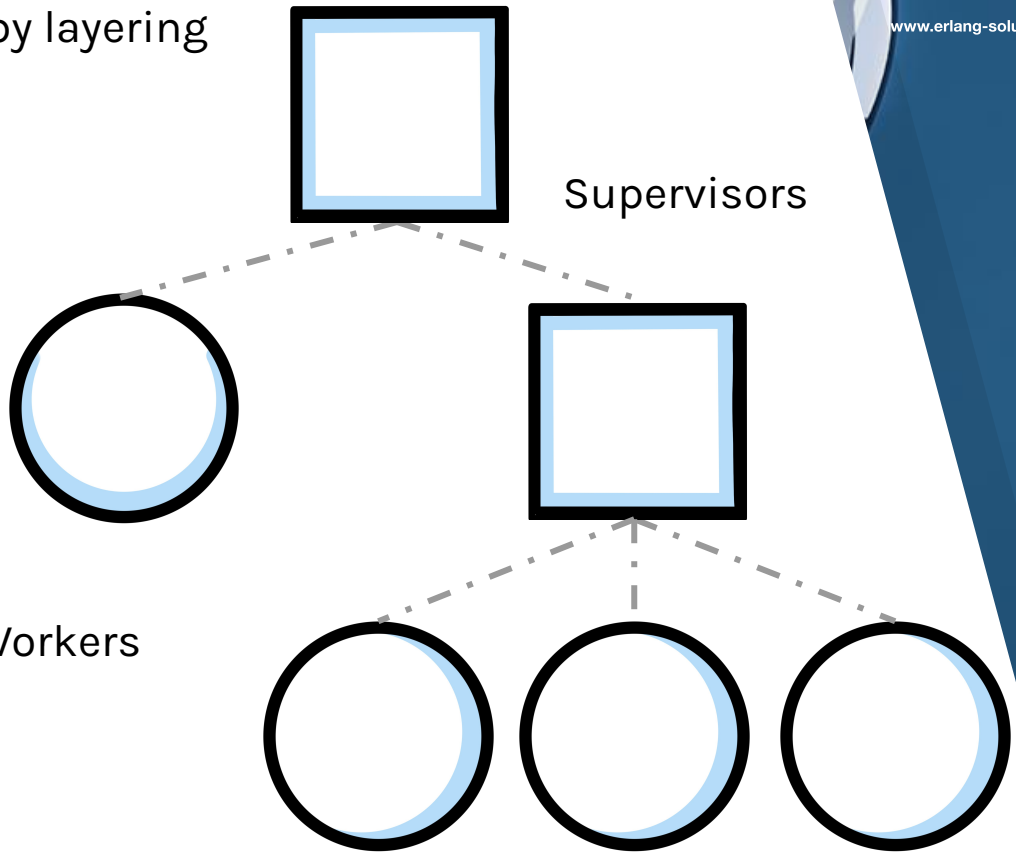
Exit signals can be trapped and received as messages



```
receive
    {‘EXIT’,Pid,...} -> ...
end
```

Erlang Highlights: Robustness

Robust systems can be built by layering



Erlang Highlights

Declarative

Concurrent

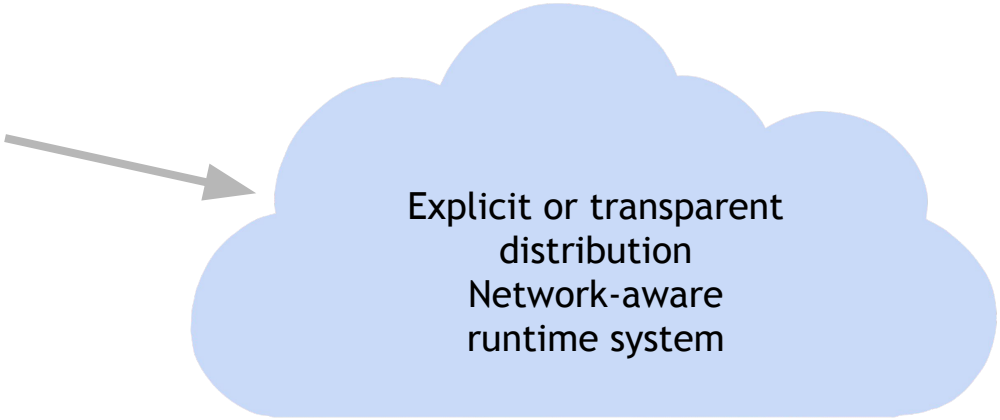
Robust

Distributed

Hot code loading

Multicore Support

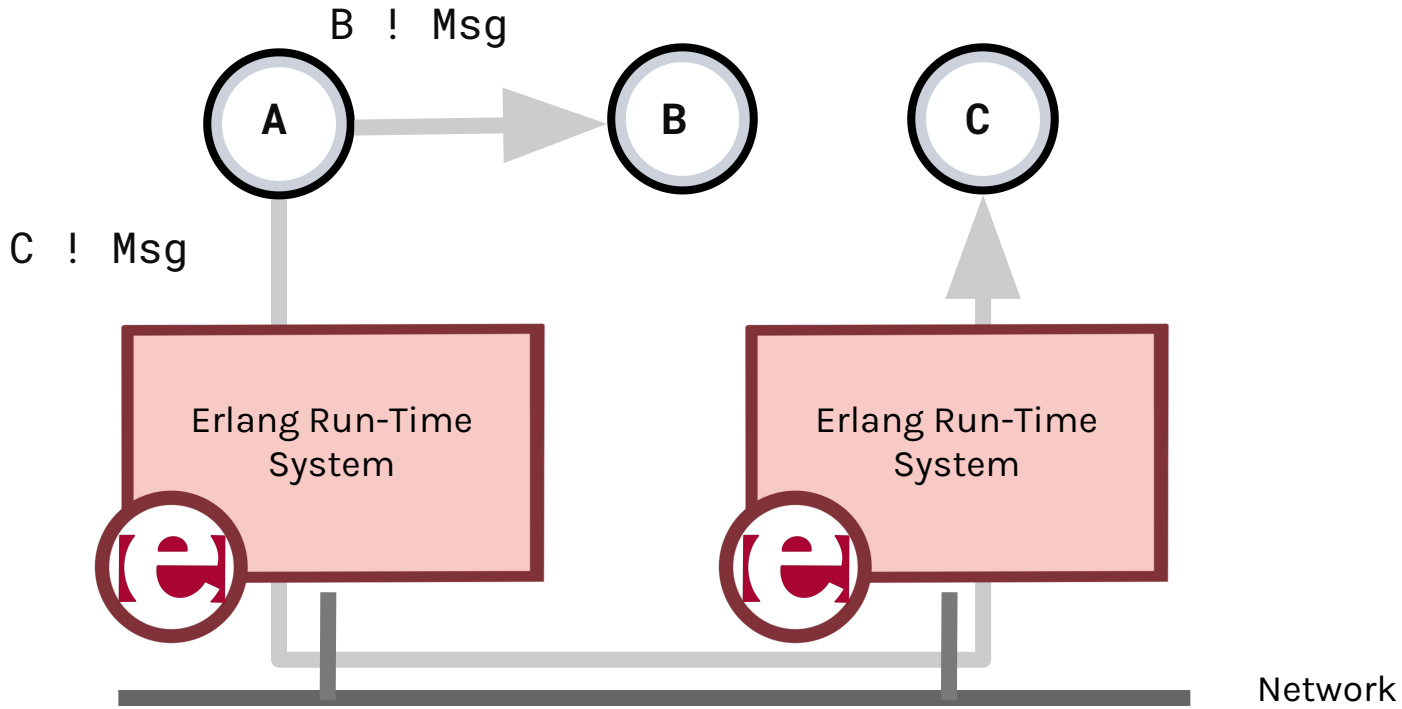
OTP



Erlang SOLUTIONS

www.erlang-solutions.com

Erlang Highlights: Distribution



Erlang Highlights

Declarative

Concurrent

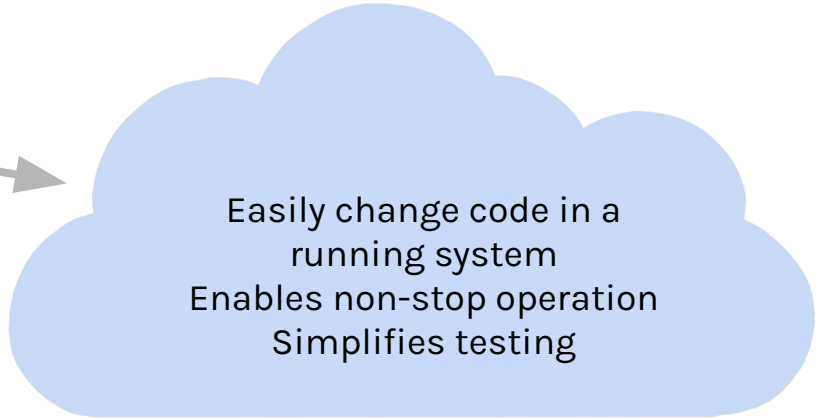
Robust

Distributed

Hot code loading

Multicore Support

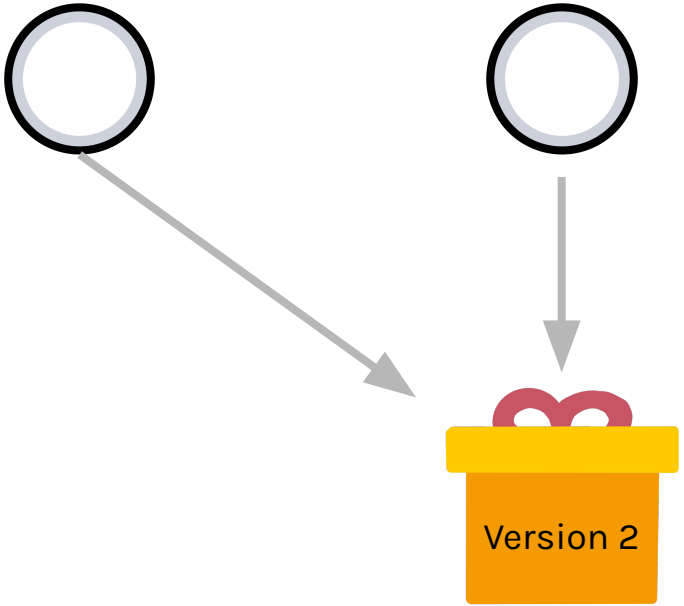
OTP



Erlang SOLUTIONS

www.erlang-solutions.com

Erlang Highlights: Hot Code Swap



Erlang Highlights

Declarative

Concurrent

Robust

Distributed

Hot code loading

Multicore Support

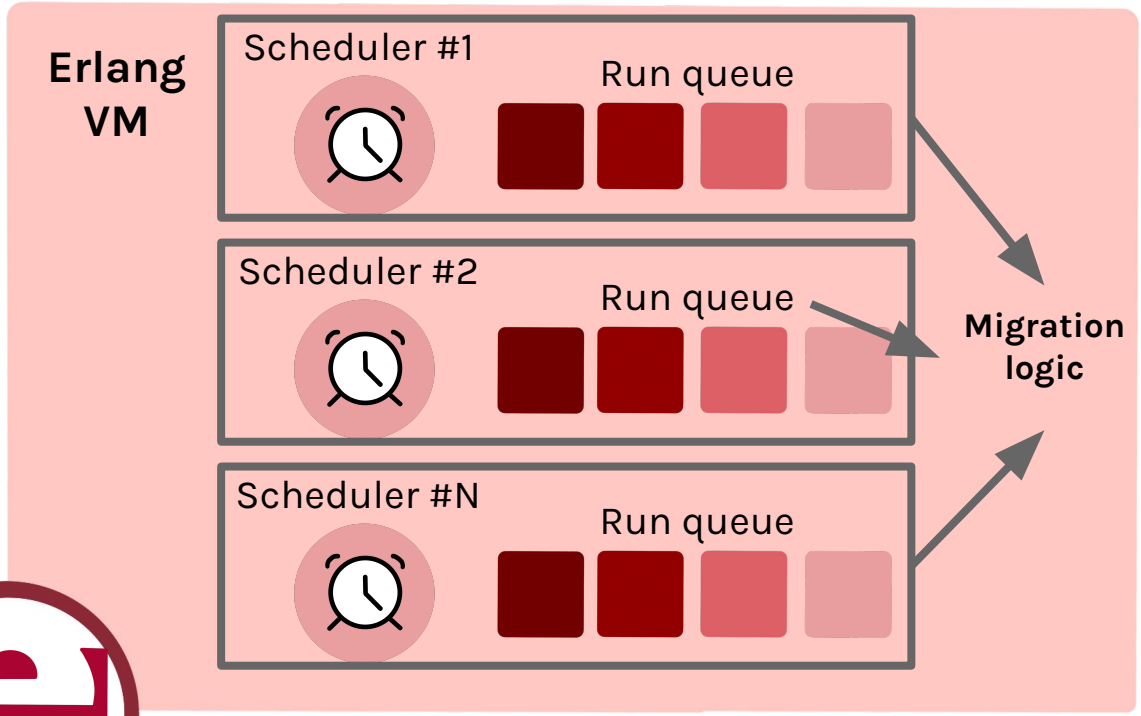
OTP

SMP support provides linear scalability out of the box thanks to its no shared memory approach to concurrency.

Erlang SOLUTIONS

www.erlang-solutions.com

Multicore Erlang



Erlang Highlights

Declarative

Concurrent

Robust

Distributed

Hot code loading

Multicore Support

OTP

Erlang SOLUTIONS

www.erlang-solutions.com

I wrote my Erlang system in 4 weeks!

Erlang SOLUTIONS

www.erlang-solutions.com

The Myths of Erlang....

Is it Documented?

Is the developer supporting it?

What visibility does support staff have into what is going on?

- SNMP
- Live Tracing
- Audit Trails
- Statistics
- CLI / HTTP Interface

How much new code was actually written?

Upgrades during runtime are easy!

The Myths of Erlang....

Yes, it is easy for

- Simple patches
- Adding functionality without changing the state

Non backwards compatible changes need time

- Database schema changes
- State changes in your processes
- Upgrades in distributed environments

Test, Test, Test

- A great feature when you have the manpower!

We achieved 99.9999999 availability!

The Myths of Erlang....

"As a matter of fact, the network performance has been so reliable that **there is almost a risk that our field engineers do not learn maintenance skills**"

Bert Nilsson, Director
NGS-Programs Ericsson

Ericsson Contact, Issue 19 2002

Erlang SOLUTIONS
www.erlang-solutions.com

The Myths of Erlang....

99,999 (Five Nines) is a more like it!

- Achieved at a fraction of the effort of Java & C++

Upgrades are risky!

Reliability and Resilience need to be in your initial design!

Non Software related issues

- Power Outages
- Network Failures, Firewall Configurations
- Hardware Faults

Erlang: It's Happening!

WhatsApp

Home Download WhatsApp Web FAQ Blog Contact

WhatsApp Blog

English

1 million is so 2011

Happy 2012 everyone!

A few months ago we published a blog post that talked about our servers doing 1 million tcp connections on a single box: http://blog.whatsapp.com/?p=170

Today we have an update for those keeping score at home: we are now able to easily push our systems to over 2 million tcp connections!

```
jkb@c123$ sysctl kern.ipc.numopensockets kern.ipc.numopensockets:
2277845
```

Best part is that we are able to do it with plenty of CPU and memory to spare and do it sustainably:

```
CPU: 37.9% user,  0.0% nice, 13.6% system,  6.6% interrupt, 41.9%
idle Mem: 35G Active, 14G Inact, 18G Wired, 4K Cache, 9838M Buf,
27G Free
```

This time we also wanted to share some more technical details with you about hardware, OS and software:

WHATSAPP FOR YOUR PHONE

SHARE WHATSAPP WITH FRIENDS

2.4m Tweet Like

HELP TRANSLATE WHATSAPP

Contribute to the WhatsApp translation in your language. Let's make WhatsApp available to everyone in the world!

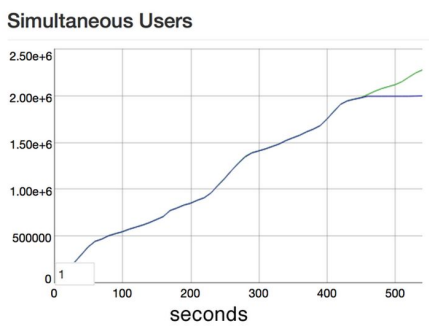
Erlang SOLUTIONS

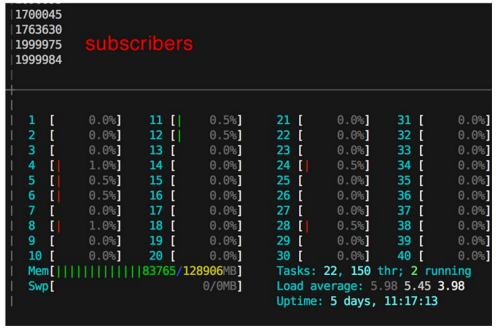
www.erlang-solutions.com

Elixir: It's Happening!

The Road to 2 Million Websocket Connections in Phoenix

By Gary Rennie · 2 months ago · v1.0.0





If you have been paying attention on Twitter recently, you have likely seen some increasing numbers regarding the number of simultaneous connections the Phoenix web framework can handle. This post documents some of the techniques used to perform the benchmarks.

HOW IT STARTED

A couple of weeks ago I was trying to benchmark the number of connections and managed to get 1k connections on my local machine. I wasn't convinced by the number so I posted in IRC to see if anyone had benchmarked Phoenix channels. It turned out they had not, but some members of the core team found the 1k number I provided suspiciously low. This was the beginning of the journey.

> Erlang is a beacon language in that it more clearly than any other language demonstrates the benefits of concurrency- oriented programming.
>
> -Simon Peyton-Jones

ERLANG/OTP MASTER CLASSES

Video master classes on

- Functional Programming
- Concurrent Programming
- OTP Behaviors

with Joe Armstrong,
Francesco Cesarini &
Simon Thompson

http://goo.gl/mhXRI7 or google
Erlang Master Classes

Erlang SOLUTIONS

ERLANG BOOKS

Programming Erlang – 2nd ed.
Software for a Concurrent World
Joe Armstrong
Learn You Some Erlang for Great Good
Fred Hebert
Erlang Programming
A Concurrent Approach to Software Development
Francesco Cesarini & Simon Thompson
Designing for Scalability with Erlang/OTP
Implementing Robust, Available and Fault Tolerant Systems
Francesco Cesarini & Steve Vinoski

Questions?

Francesco Cesarini
@francescoC
Founder & Technical Director @
Erlang Solutions

Erlang SOLUTIONS