An Introduction to Erlang

From behind the trenches...

GOTO Amsterdam
Amsterdam, May 25th 2012

Francesco Cesarini
Founder, Technical Director

@FrancescoC
francesco@erlang-solutions.com
So Here I Am....
Telecom Applications: Issues

Complex
No down time
Scalable
Maintainable
Distributed

VS

Time to Market
The Ancestors

Languages like SmallTalk, Ada, Modula or Chill

Functional languages like ML or Miranda

Logical languages like Prolog
Erlang Highlights

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

- Functional programming language
- High abstraction level
- Pattern matching
- Concise readable programs
Erlang Highlights: Factorial

Factorial using Recursion

**Definition**

\[
\begin{align*}
\text{n!} &= \begin{cases} 
1 & \text{n = 0} \\
\text{n*(n-1)!} & \text{n} \geq 1 
\end{cases}
\end{align*}
\]

**Implementation**

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

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

Eshell V5.0.1 (abort with ^G)
1> c(ex1).
{ok,ex1}
2> ex1:factorial(6).
720
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"
Erlang Highlights: High-level Constructs

Parsing a TCP packet using the Bit Syntax

\[
<< \text{SourcePort:16}, \text{DestinationPort:16}, \text{SequenceNumber:32},
\text{AckNumber:32}, \text{DataOffset:4}, \_\text{Reserved:4}, \text{Flags:8},
\text{WindowSize:16}, \text{Checksum:16}, \text{UrgentPointer:16},
\text{Payload/binary>>} = \text{Segment},
\]

\[
\text{OptSize} = (\text{DataOffset} - 5) \times 32,
<< \text{Options:OptSize, Message/binary >>} = \text{Payload},
<< \text{CWR:1, ECE:1, URG:1, ACK:1, PSH:1,}
\text{RST:1, SYN:1, FIN:1>>} = <<\text{Flags:8>>},
\]

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

\text{etc...}
Erlang Highlights

Declarative
Concurrent
Robust
Distributed
Hot code loading
Multicore Support
OTP

Either transparent or explicit concurrency
Light-weight processes
Highly scalable
Erlang Highlights: Concurrency

Creating a new process using spawn

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

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

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

Source: Joe Armstrong SICS

© 2012 - Erlang Solutions Ltd.
Processes communicate by asynchronous message passing

Pid ! {data,12,13}

receive
  {start} -> ........
  {stop}  -> ........
  {data,X,Y}  -> ........
end
Erlang Highlights: Concurrency

Message passing times

Source: Joe Armstrong, SICS

© 2012 - Erlang Solutions Ltd.
Erlang Highlights

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

Simple and consistent error recovery
Supervision hierarchies
"Program for the correct case"
Cooperating processes may be linked together using

```
using
spawn_link(...,...,...)
or
link(Pid)
```
Erlang Highlights: Robustness

When a process terminates, an exit signal is sent to all linked processes

... and the termination is propagated
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

“Supervisors”

“Workers”
Erlang Highlights

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

Explicit or transparent distribution
Network-aware runtime system
Erlang Highlights: Distribution

B ! Msg

C ! Msg

Erlang Run-Time System

network
Simple Remote Procedure Call

```erlang
{rex, Node} ! {self(), {apply, M, F, A}},
receive
    {rex, Node, What} -> What
end.
```

```erlang
loop() ->
    receive
        {From, {apply, M, F, A}} ->
            Answer = apply(M, F, A),
            From ! {rex, node(), Answer}
        loop();
        _Other -> loop()
    end.
```
Erlang Highlights

Declarative
Concurrent
Robust
Distributed
Hot code loading
Multicore Support
OTP

Easily change code in a running system
Enables non-stop operation
Simplifies testing
Erlang Highlights: Hot Code Swap

Version 2
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.
Ericsson’s strategy with SMP

Make it work -> measure -> optimize

Hide the problems and awareness of SMP from the programmer
Programmed in the normal style using processes for encapsulation and parallelisation
Multicore Erlang
## Telephony Gateway Controller

<table>
<thead>
<tr>
<th>Traffic scenario</th>
<th>IS/GCP 1slot/board</th>
<th>IS/GEP Dual core One core running 2slots/board</th>
<th>IS/GEP Dual core Two cores running 2slots/board</th>
<th>AXD CPB5</th>
<th>AXD CPB6</th>
</tr>
</thead>
<tbody>
<tr>
<td>POTS-POTS / AGW</td>
<td>X call/sec</td>
<td>2.3X call/sec One core used</td>
<td><strong>4.3X call/sec</strong> OTP R11_3 beta +patches</td>
<td>0.4X call/sec</td>
<td>2.1X call/sec</td>
</tr>
<tr>
<td>ISUP-ISUP /Inter MGW</td>
<td>3.6X call/sec</td>
<td>7.7X call/sec One core used</td>
<td><strong>13X call/sec</strong> OTP R11_3 beta +patches</td>
<td>5.55X call/sec</td>
<td>7.6X call/sec</td>
</tr>
<tr>
<td>ISUP-ISUP /Intra MGW</td>
<td>5.5X call/sec</td>
<td></td>
<td><strong>26X call/sec</strong></td>
<td>3.17X call/sec</td>
<td>14X call/sec</td>
</tr>
</tbody>
</table>
Tilera “Tile64”

Chatty

500 processes created

Each process randomly sends messages and receives a response from all other processes
Multicore Benchmark - Big Bang

Big_bang benchmark, 500 processes chatting
- Bound schedulers
- Default (unbound)

(No Tilera-specific optimizations!)

Erlang/OTP R13B on Tilera Pro 64-core

© 2012 - Erlang Solutions Ltd.
Erlang Highlights

Declarative
Concurrent
Robust
Distributed
Hot code loading
Multicore Support

Provides the design patterns, libraries and tools to develop distributed fault tolerant systems.
OTP Middleware

Applications & Libraries

System Design Principles

© 2012 - Erlang Solutions Ltd.
Erlang Highlights

Declarative
Concurrent
Robust
Distributed
Hot code loading
Multicore Support
OTP
I wrote my Erlang system in 4 weeks!
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 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 99.9999999999 availability!
“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
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!

Non Software related issues
- Power Outages
- Network Failures, Firewall Configurations
- Hardware Faults
Who is using Erlang?
Erlang: It’s Happening!

CouchDB
Distributed Robust document database

Riak
Distributed, partition tolerant and scalable database

YAWS
Yet Another Web Server

RabbitMQ
High performance enterprise messaging

Ejabberd
XMPP instant messaging server
erlang.org site usage (Unique visits, 30 days)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Country</th>
<th>Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USA</td>
<td>19,483</td>
</tr>
<tr>
<td>2</td>
<td>Sweden</td>
<td>8,465</td>
</tr>
<tr>
<td>3</td>
<td>United Kingdom</td>
<td>4,984</td>
</tr>
<tr>
<td>4</td>
<td>China</td>
<td>4,897</td>
</tr>
<tr>
<td>5</td>
<td>Germany</td>
<td>4,745</td>
</tr>
<tr>
<td>6</td>
<td>Russia</td>
<td>4,465</td>
</tr>
<tr>
<td>7</td>
<td>France</td>
<td>2,722</td>
</tr>
<tr>
<td>8</td>
<td>Canada</td>
<td>2,647</td>
</tr>
<tr>
<td>9</td>
<td>India</td>
<td>2,299</td>
</tr>
<tr>
<td>10</td>
<td>Japan</td>
<td>2,261</td>
</tr>
</tbody>
</table>
Books
More Information

Programming Erlang
- Software for a Concurrent World
- by Joe Armstrong

Erlang Programming
- A Concurrent Approach to Software Development
- by Francesco Cesarini & Simon Thompson

Erlang and OTP in Action
- Large-scale software design with OTP
- by Richard Carlsson, Martin Logan & Eric Merrit
Questions
Thank You!

@FrancescoC on Twitter or francesco@erlang-solutions.com