Thinking in a Highly Concurrent, Mostly-functional Language

Chicago Erlang

Chicago, September 22nd 2014

Francesco Cesarini
Founder & Technical Director

@francescoC
francesco@erlang-solutions.com

Erlang Training and Consulting Ltd

Thinking in a Highly Concurrent, Mostly-functional Language

QCON London, March 12th, 2009

Francesco Cesarini

francesco@erlang-consulting.com

```
counter_loop(Count) ->
  receive
      increment ->
        counter_loop(Count + 1);
      {count, To} ->
        To ! {count, Count},
        counter_loop(Count)
   end.
```

Erlang

© 2014 – Erlang Solutions Ltd

After you’ve opened the top of your head, reached in and turned your brain inside out, this starts to look like a natural way to count integers. And Erlang does require some fairly serious mental readjustment.

However… having spent some time playing with this, I tell you...

Tim Bray, Director of Web Technologies – Sun Microsystems

© 2014 – Erlang Solutions Ltd

… If somebody came to me and wanted to pay me a lot of money to build a large scale message handling system that really had to be up all the time, could never afford to go down for years at the time, I would unhesitatingly choose Erlang to build it in.

Tim Bray, Director of Web Technologies – Sun Microsystems

© 2014 – Erlang Solutions Ltd

Syntax

© 2014 – Erlang Solutions Ltd

Concurrency

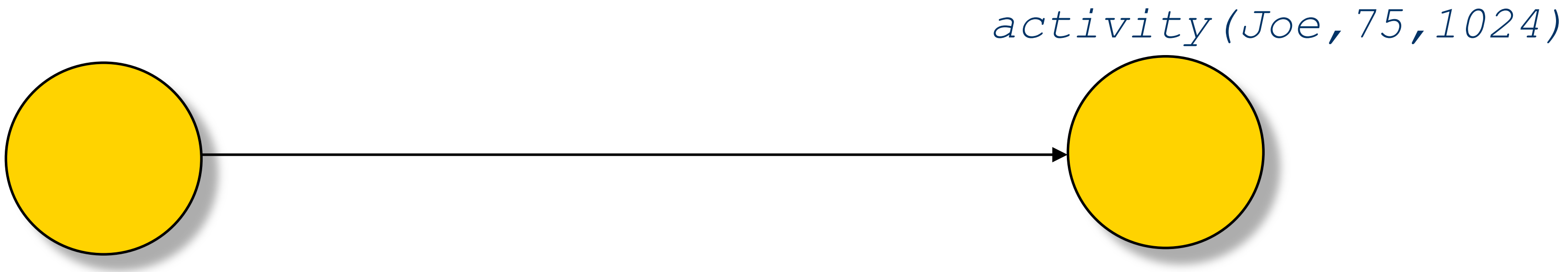
© 2014 – Erlang Solutions Ltd

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])
```

© 2014 – Erlang Solutions Ltd

Erlang Highlights: Concurrency

Processes communicate by asynchronous message passing

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

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

© 2014 – Erlang Solutions Ltd

Products: AXD301 Switch - 1996

A Telephony-Class, scalable (10 – 160 GBps) ATM switch

Designed from scratch in less than 3 years

AXD 301 Success factors:

- Competent organisation and people
- Efficient process
- Excellent technology (e.g. Erlang/OTP)

© 2014 – Erlang Solutions Ltd

Products: AXD301 Switch - 1996

Erlang: ca 1.5 million lines of code

- Nearly all the complex control logic
- Operation & Maintenance
- Web server and runtime HTML/ JavaScript generation

C/C++: ca 500k lines of code

- Third party software
- Low-level protocol drivers
- Device drivers

Java: ca 13k lines of code

- Operator GUI applets

© 2014 – Erlang Solutions Ltd

Concurrency Modeling

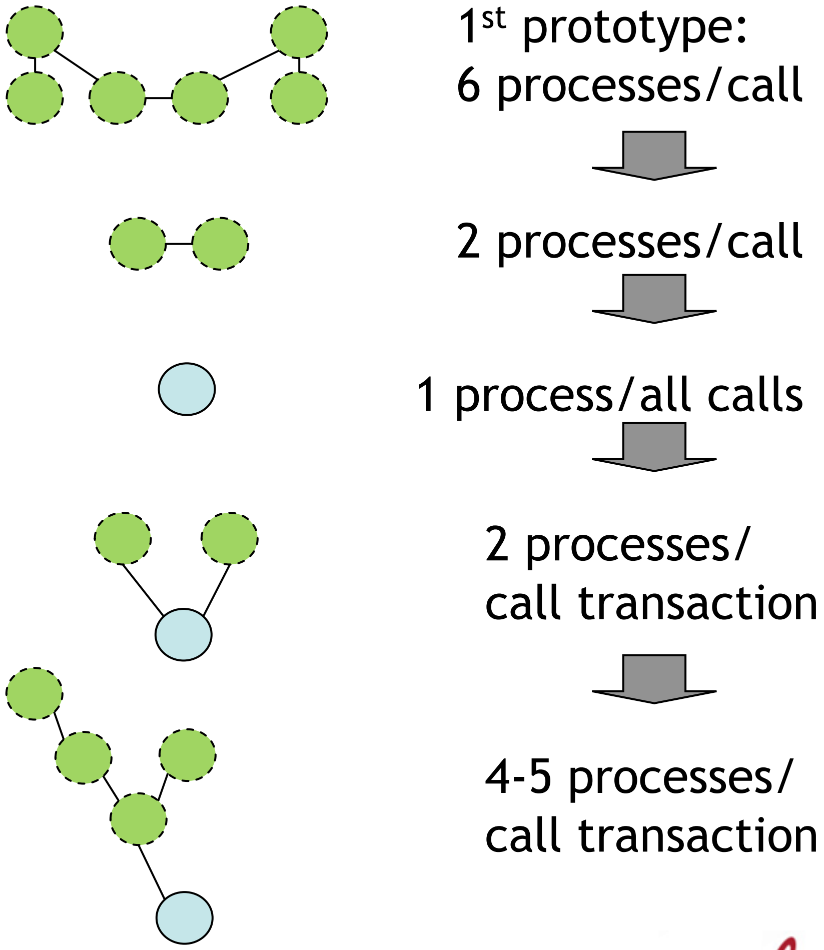
Model for the natural concurrency in your problem

In the old days, processes were a critical resource

- Rationing processes led to complex and unmanageable code

Nowadays, processes are very cheap: if you need a process – create one!

Example: AXD301 process model



1st prototype:
6 processes/call

2 processes/call

1 process/all calls

2 processes/
call transaction

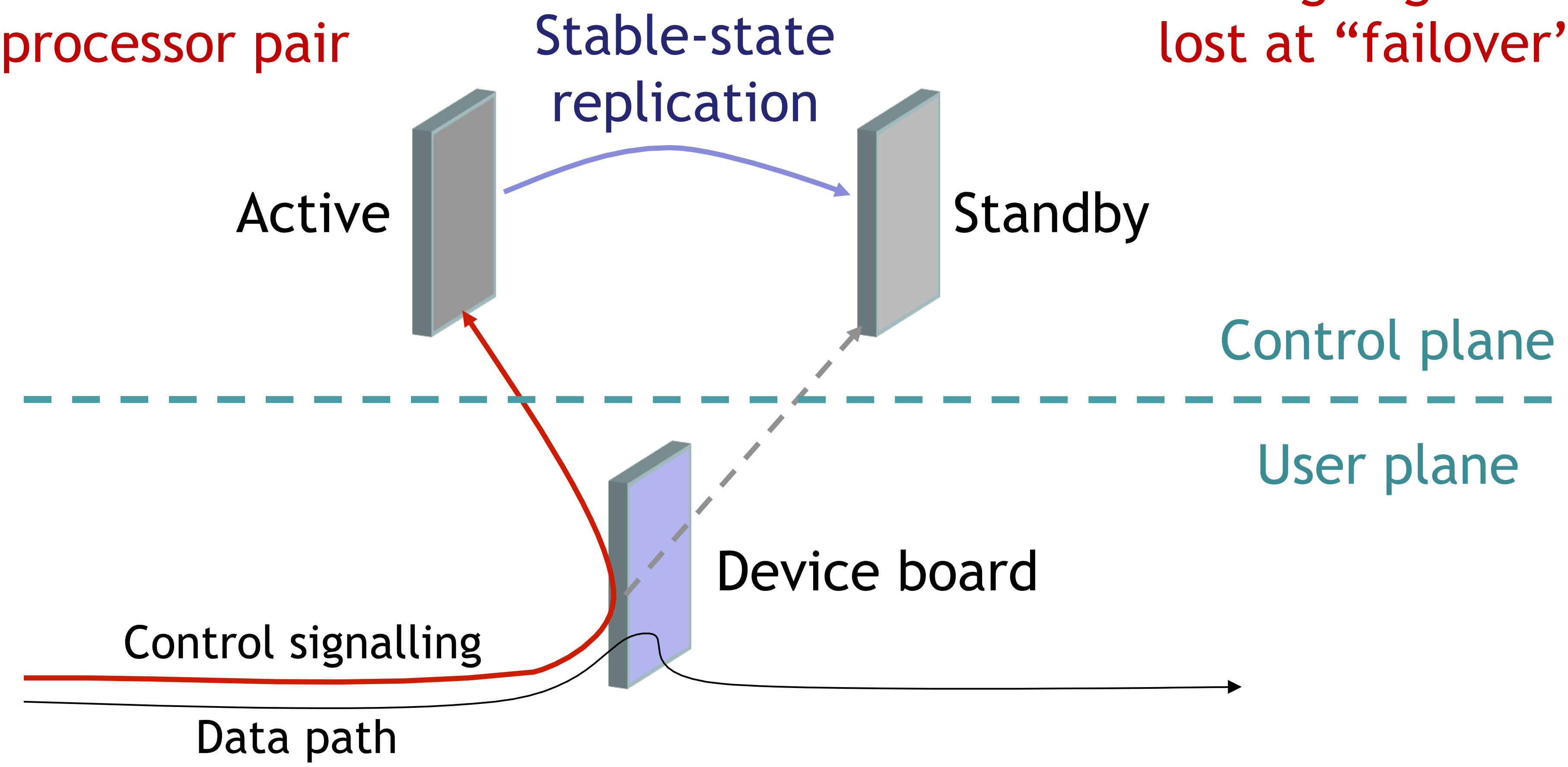
4-5 processes/
call transaction

© 2014 – Erlang Solutions Ltd

1+1 Redundancy – Good ol’ Telecoms

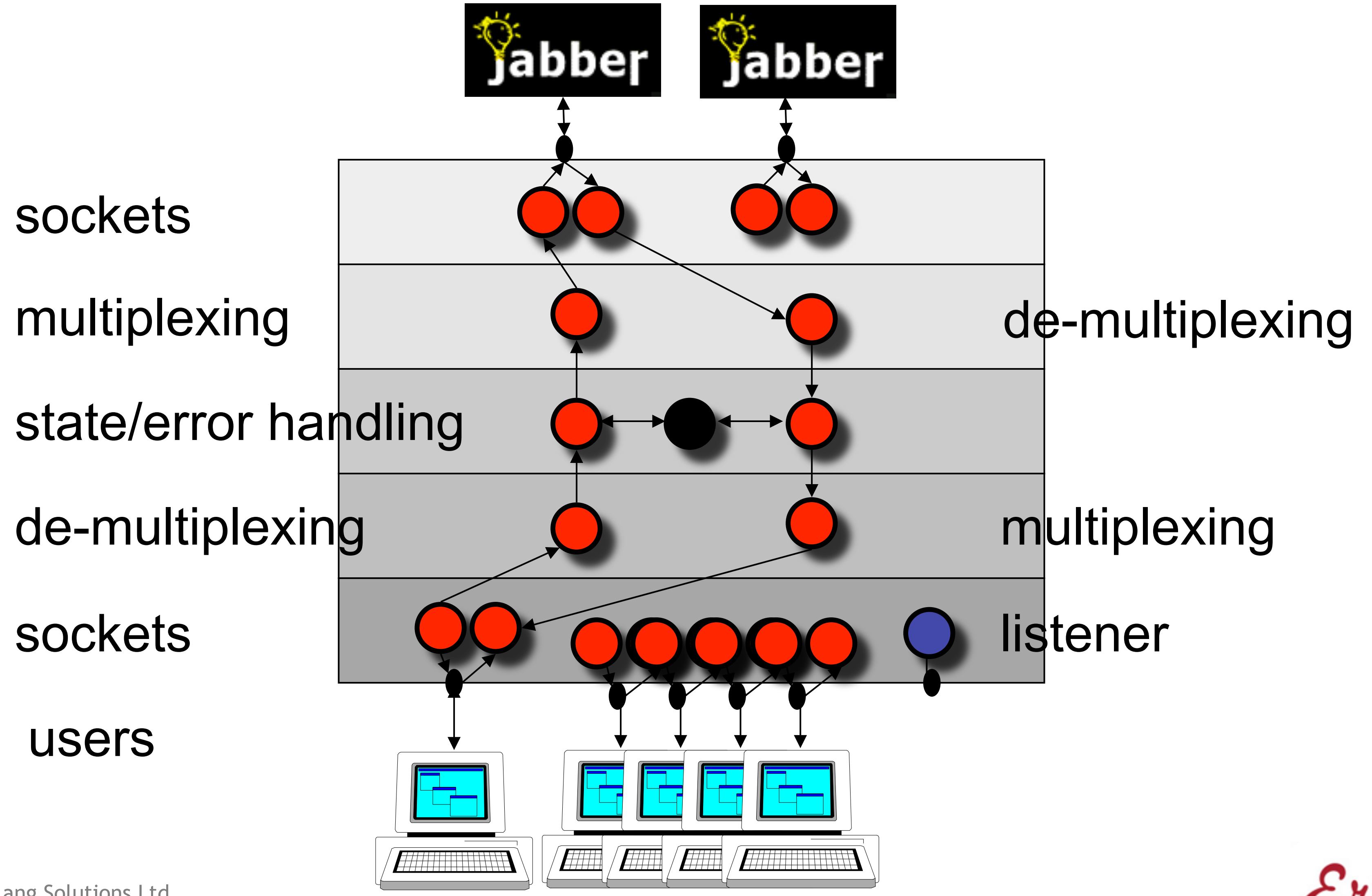
~ 35 000 calls
per processor pair

No ongoing sessions
lost at “failover”



© 2014 – Erlang Solutions Ltd

First IM Proxy Prototype - 2000



sockets

multiplexing

de-multiplexing

state/error handling

de-multiplexing

multiplexing

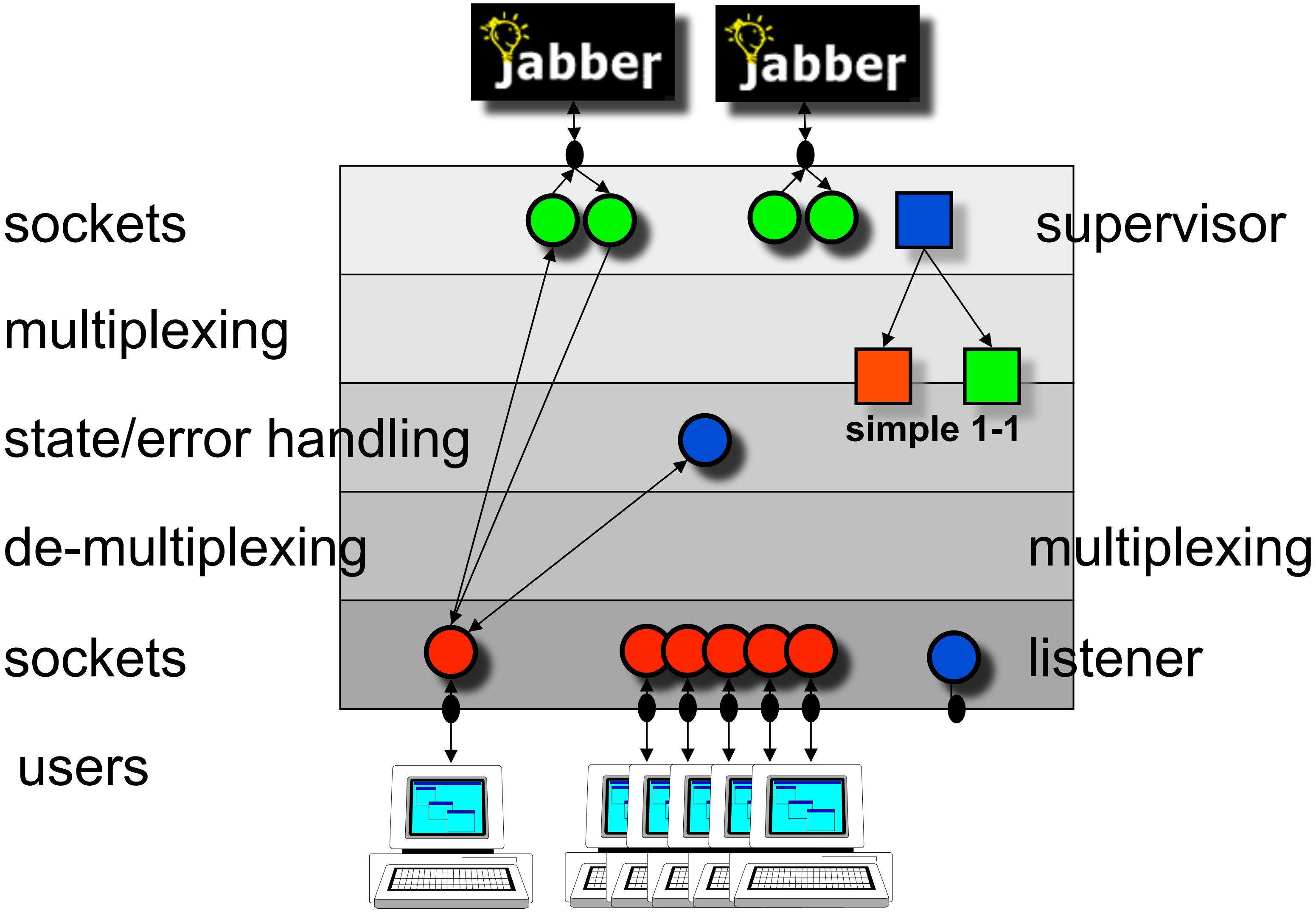
sockets

listener

users

© 2014 – Erlang Solutions Ltd

First IM Proxy Prototype - 2000



© 2014 – Erlang Solutions Ltd

Products: EjabberD IM Server - 2002

A distributed XMPP server

Started as an Open Source Project by *Alexey Shchepin*

Commercially Supported by Process-One (Paris)

- 40% of the XMPP IM market
- Used as a transport layer
- Managed 30,000 users / node

process one
INSTANT MESSAGES CREATIVE BUSINESSES

© 2014 – Erlang Solutions Ltd

Products: EjabberD IM Server - 2002

A distributed XMPP server

Started as an Open Source Project by *Alexey Shchepin*

Commercially Supported by Process-One (Paris)

- 40% of the XMPP IM market
- Used as a transport layer
- 2008, Managed 30,000 users / node

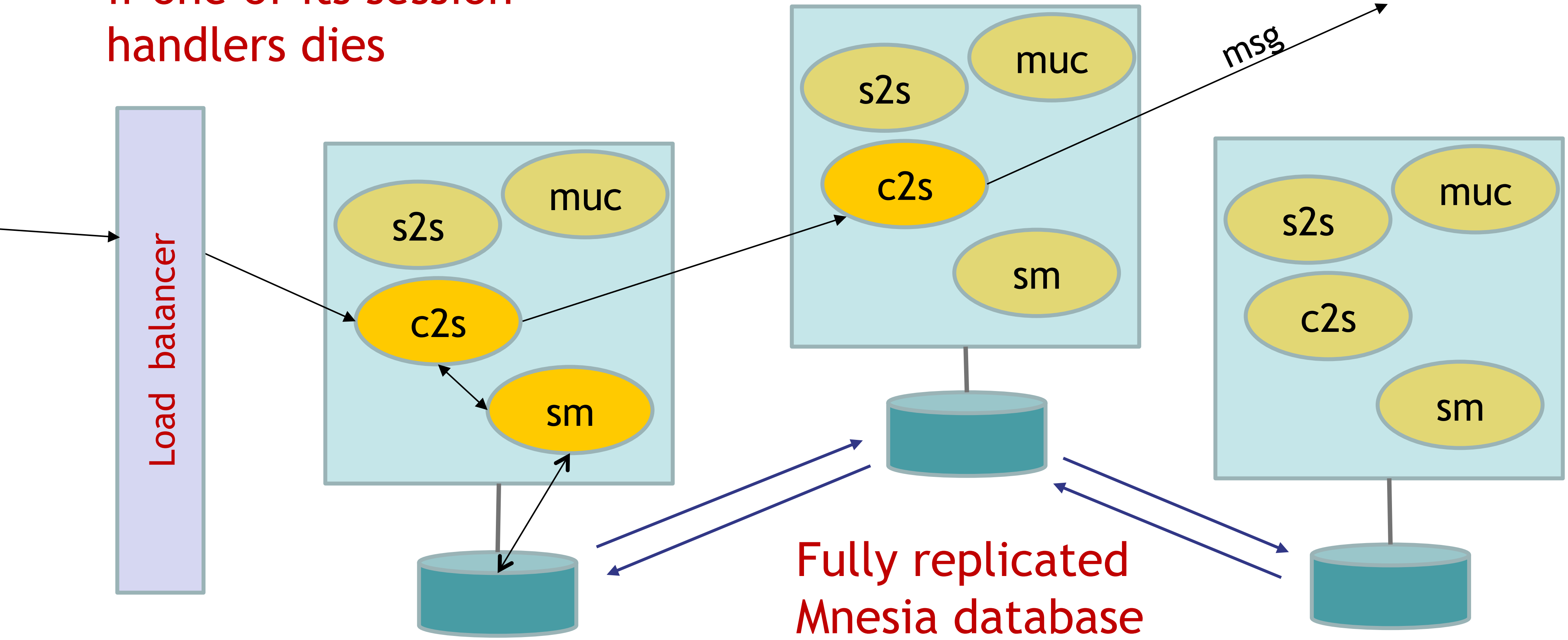
MongooseIM is a fork and rewrite

- Open Source, supported by Erlang Solutions
- Used for Messaging and Device Management
- 2014, managed 1 million users / node

© 2014 – Erlang Solutions Ltd

Fully Replicated Cluster – Ejabberd 2002

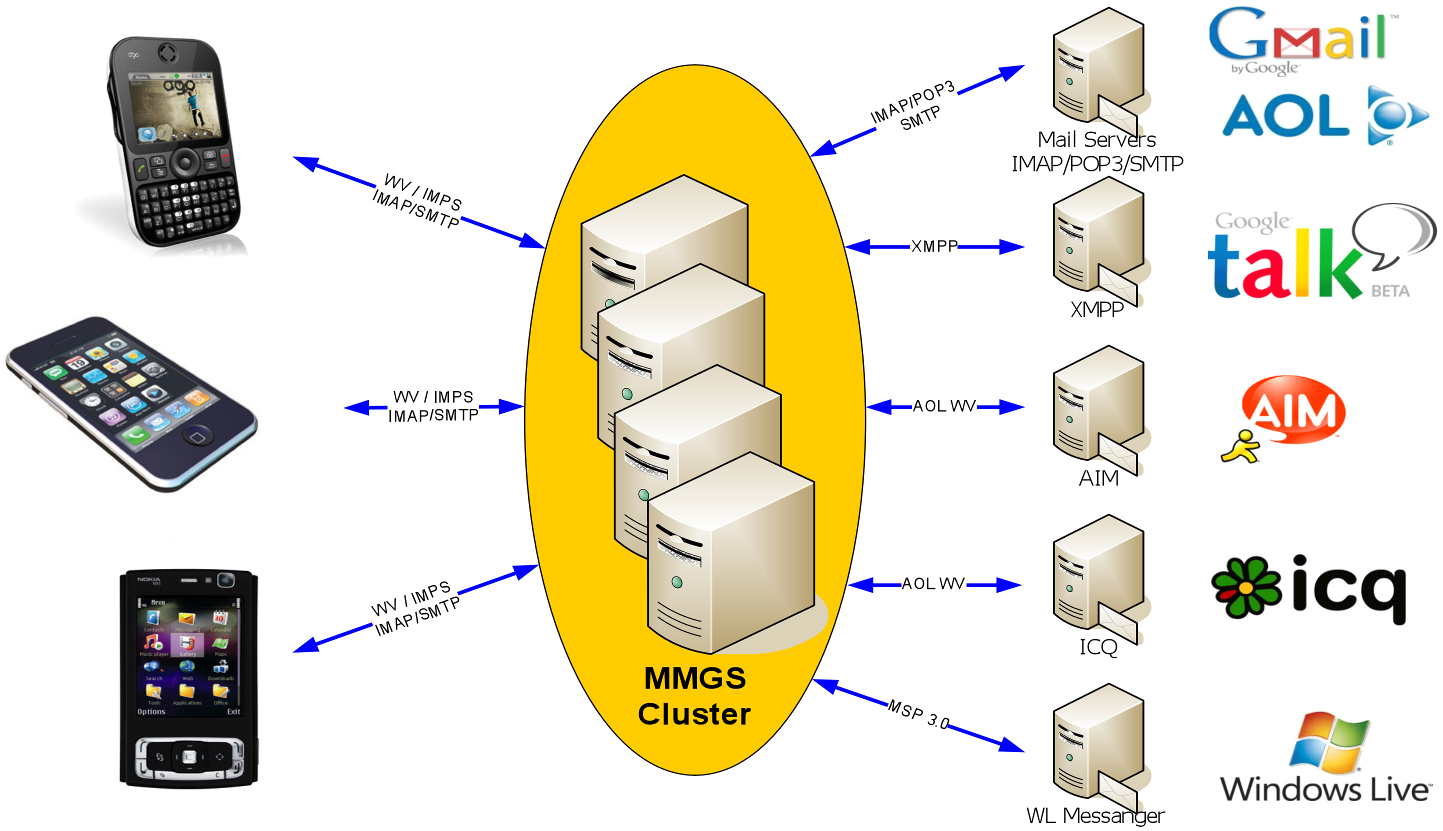
Client must re-connect if one of its session handlers dies



Fully replicated Mnesia database

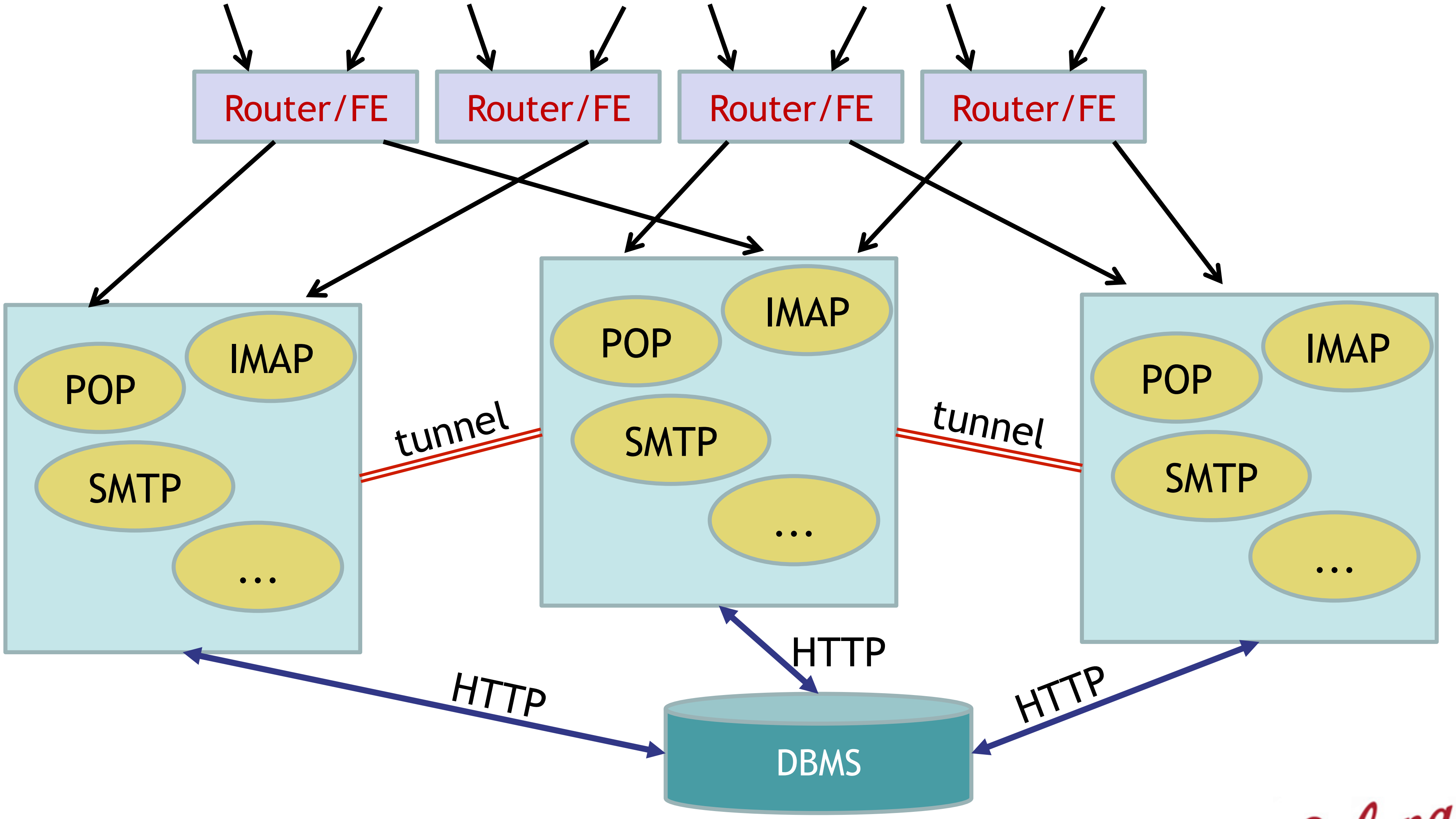
© 2014 – Erlang Solutions Ltd

Share-nothing Architecture – Messaging Gateway



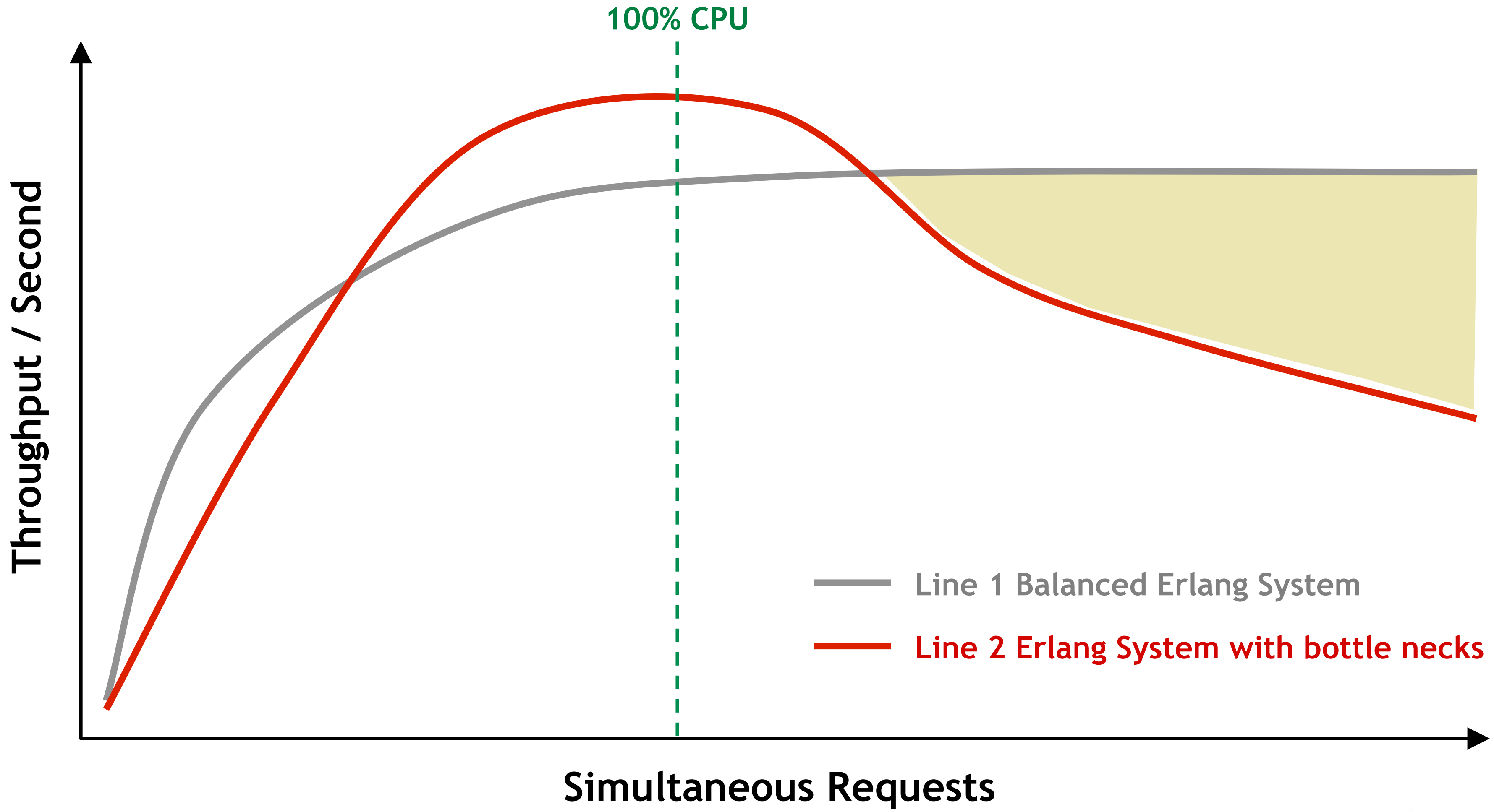
© 2014 – Erlang Solutions Ltd

Share-nothing Architecture – Messaging Gateway



© 2014 – Erlang Solutions Ltd

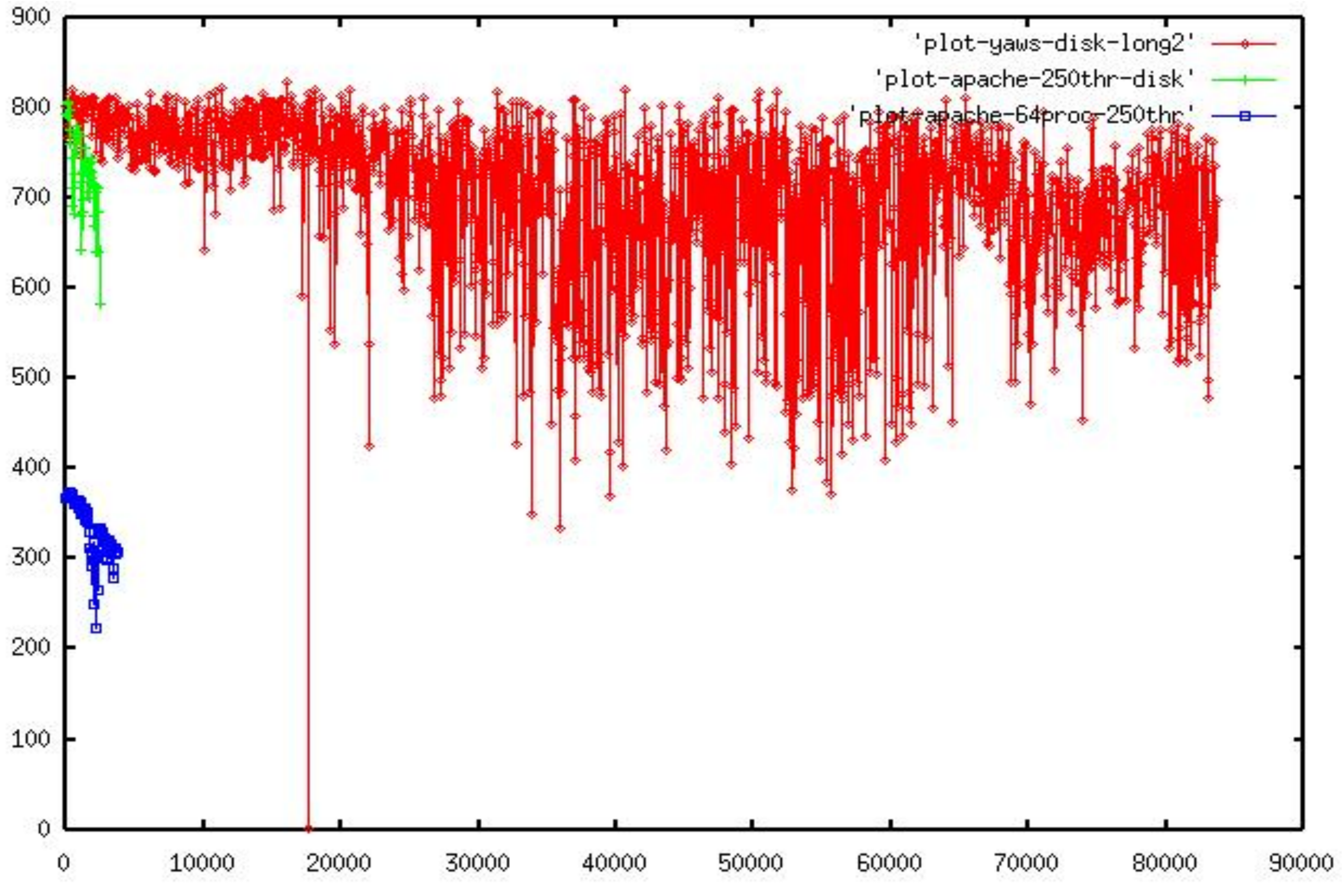
Erlang Concurrency Under Stress – Pre-SMP



© 2014 – Erlang Solutions Ltd

Erlang Concurrency Under Stress – Pre-SMP

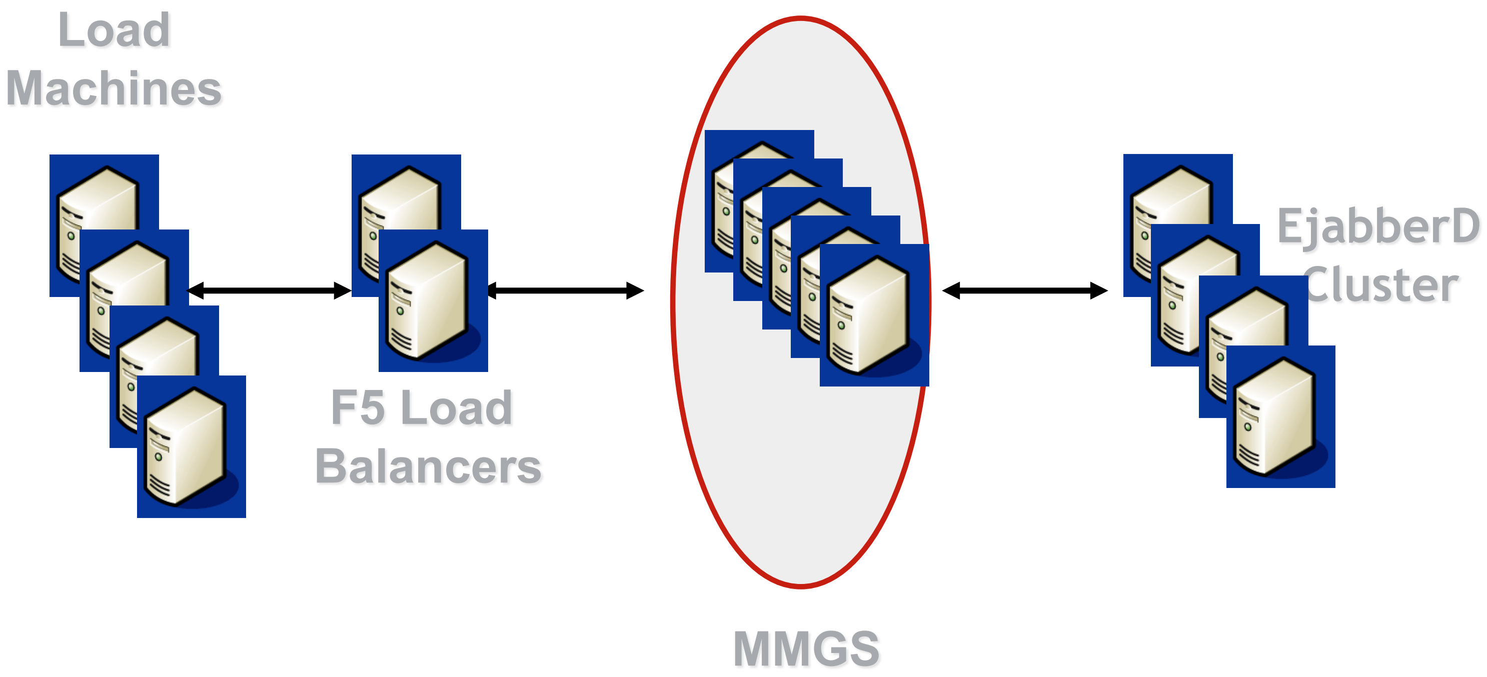
YAWS Throughput
(KBytes/second)



Simultaneous Requests

© 2014 – Erlang Solutions Ltd

Erlang Concurrency Under Stress – Post-SMP



© 2014 – Erlang Solutions Ltd

Stress Tests With SMP

I/O Starvation

TCP/IP Congestion

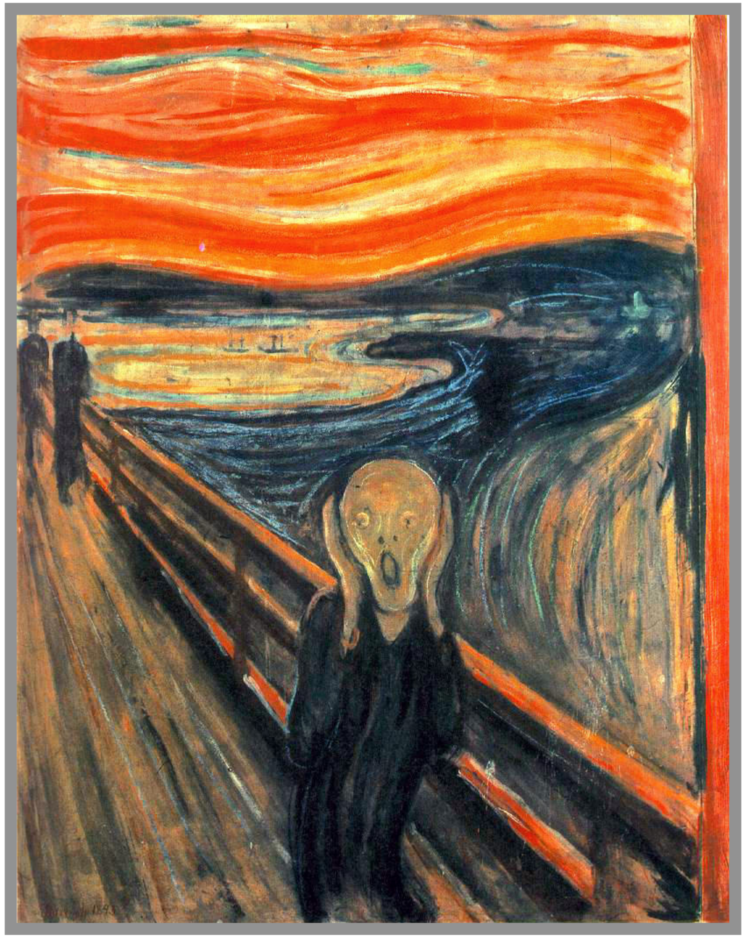
Memory Spikes

Timeout Fine-tuning

OS Limitations

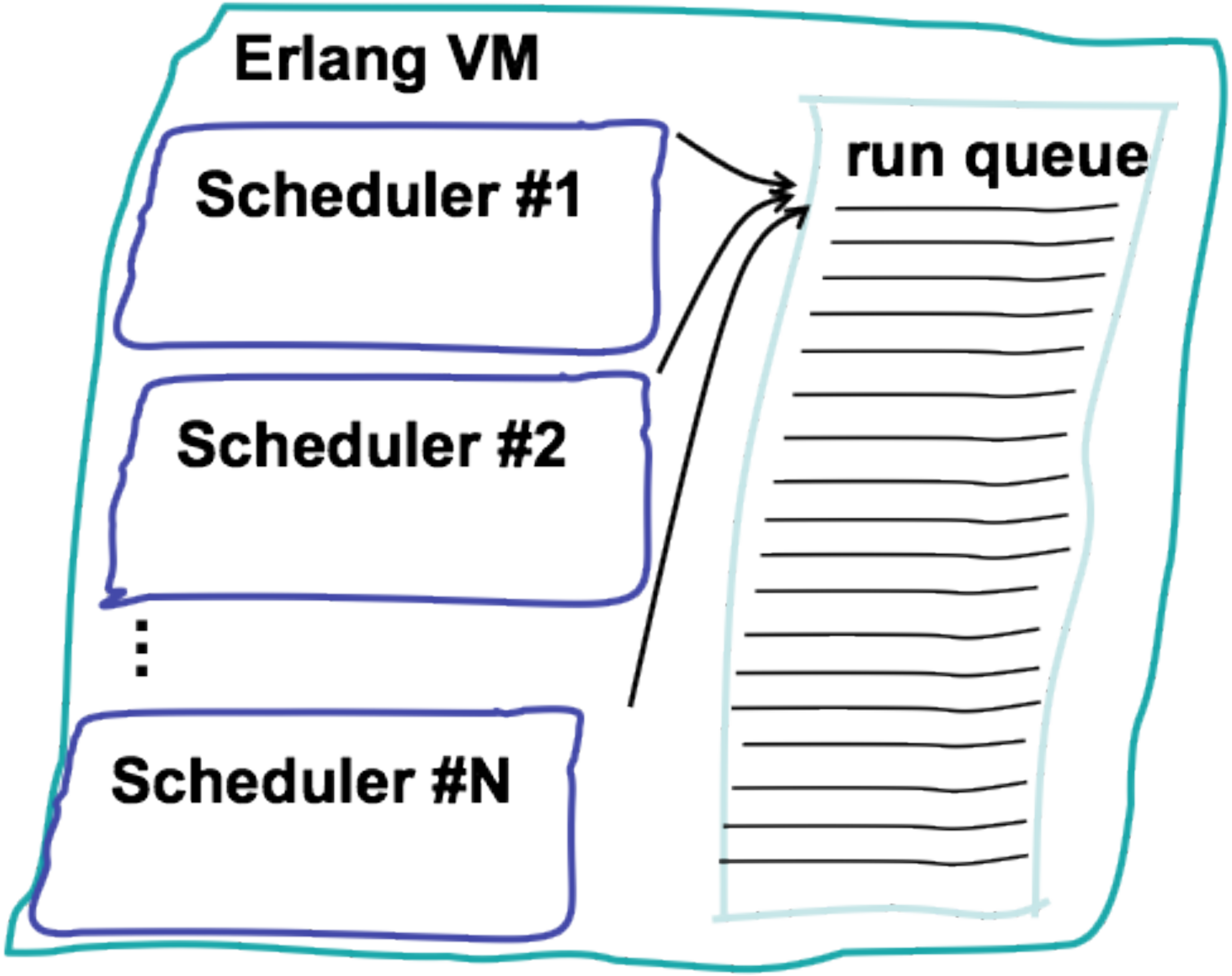
ERTS Configuration Flags

Shut down Audit Logs



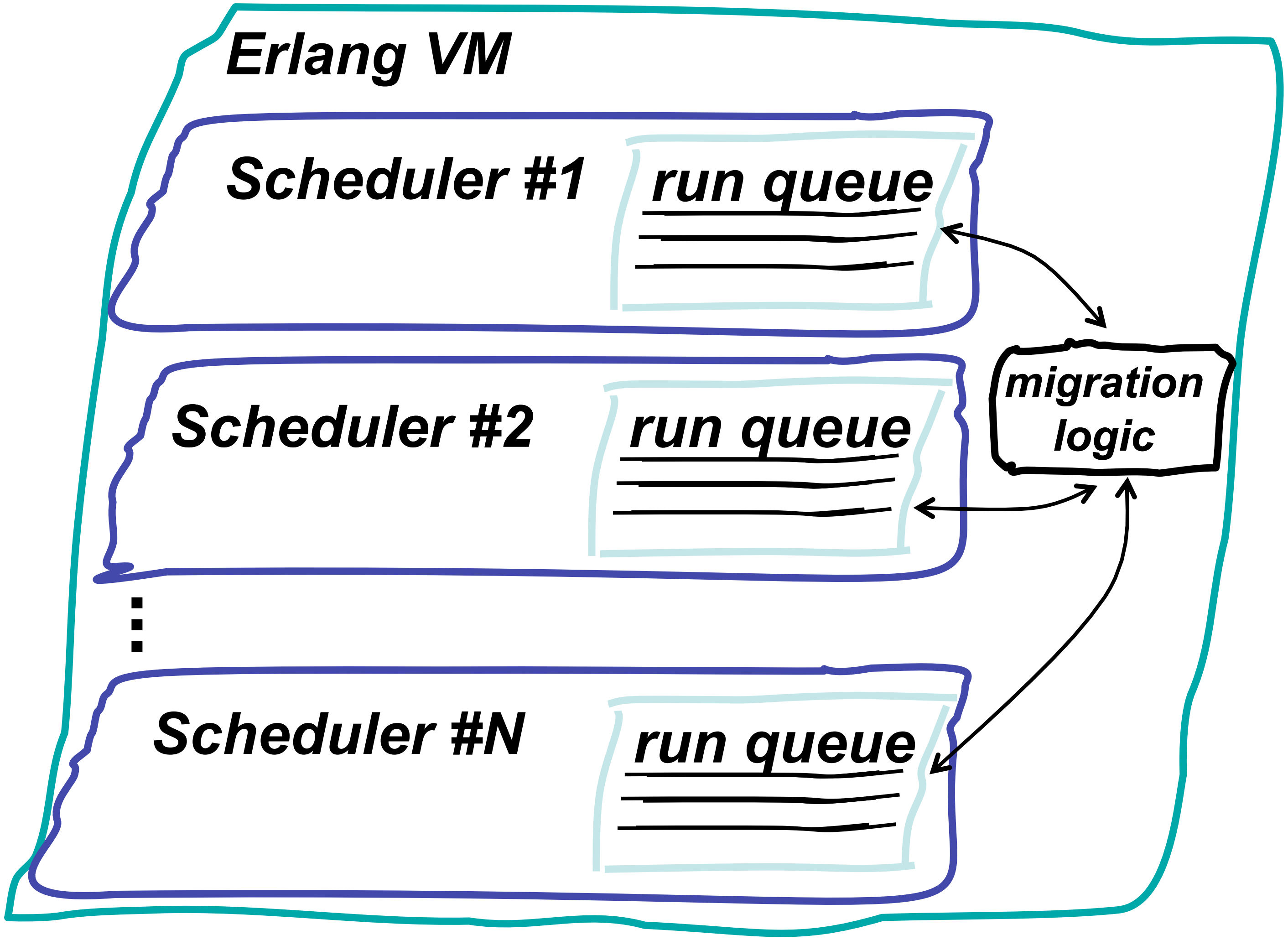
© 2014 – Erlang Solutions Ltd

SMP bottlenecks – pre 2008



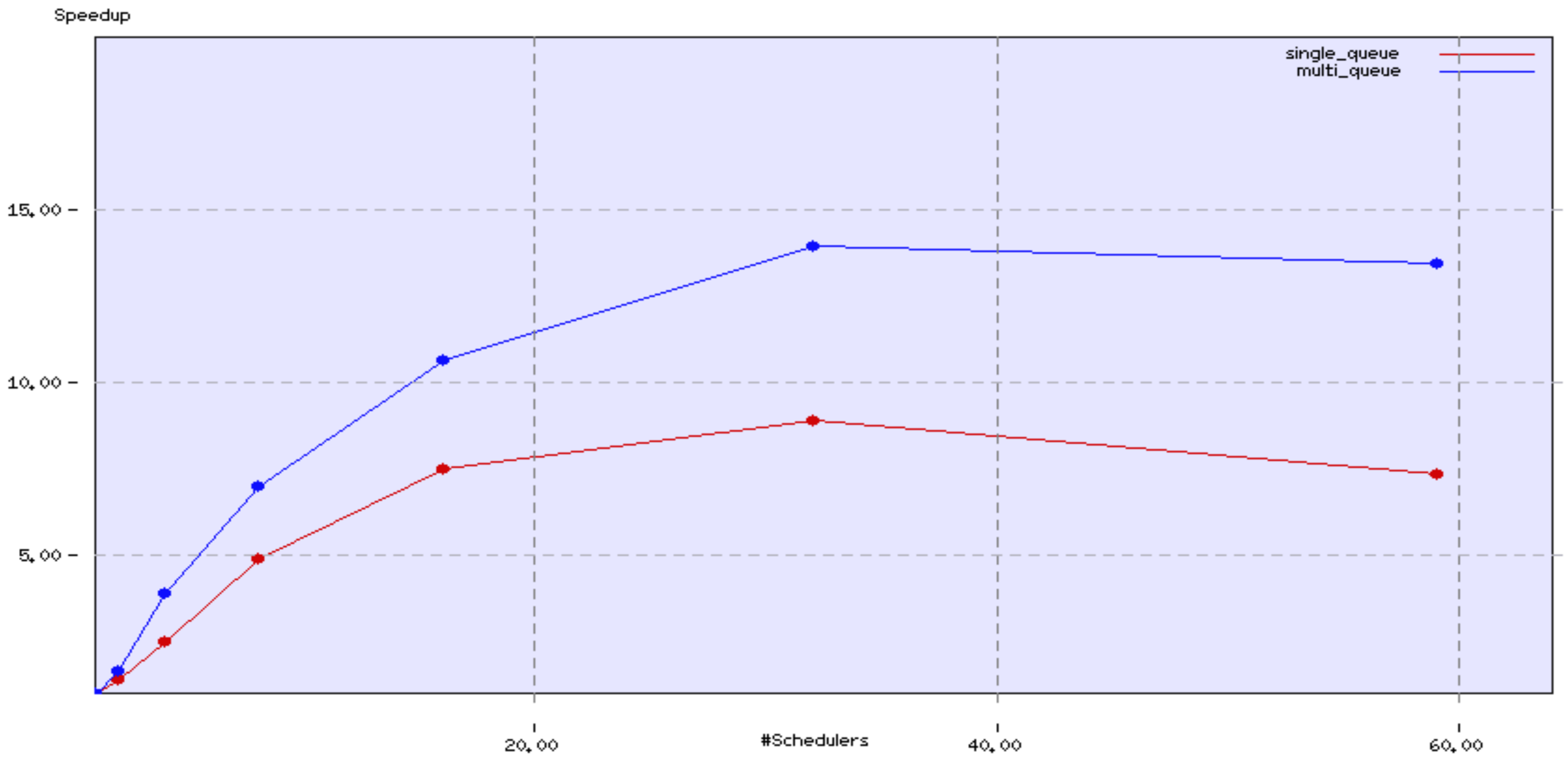
© 2014 – Erlang Solutions Ltd

SMP bottlenecks – post 2008



© 2014 – Erlang Solutions Ltd

Big Bang Benchmark – post 2008

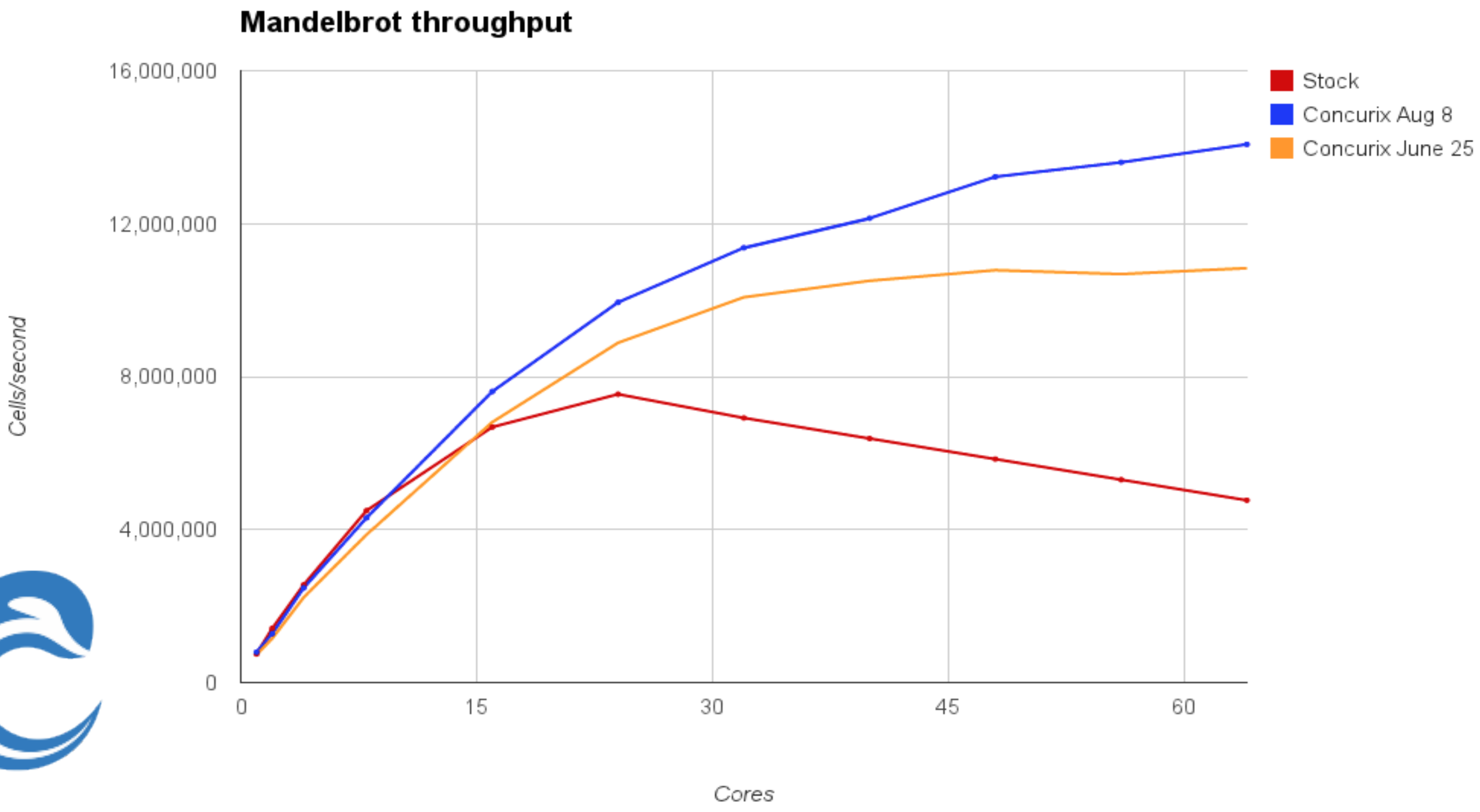


Red: Single Queue, Blue: Multple Run Queue on a Tilera TilePro64 (64 cores)

© 2014 – Erlang Solutions Ltd

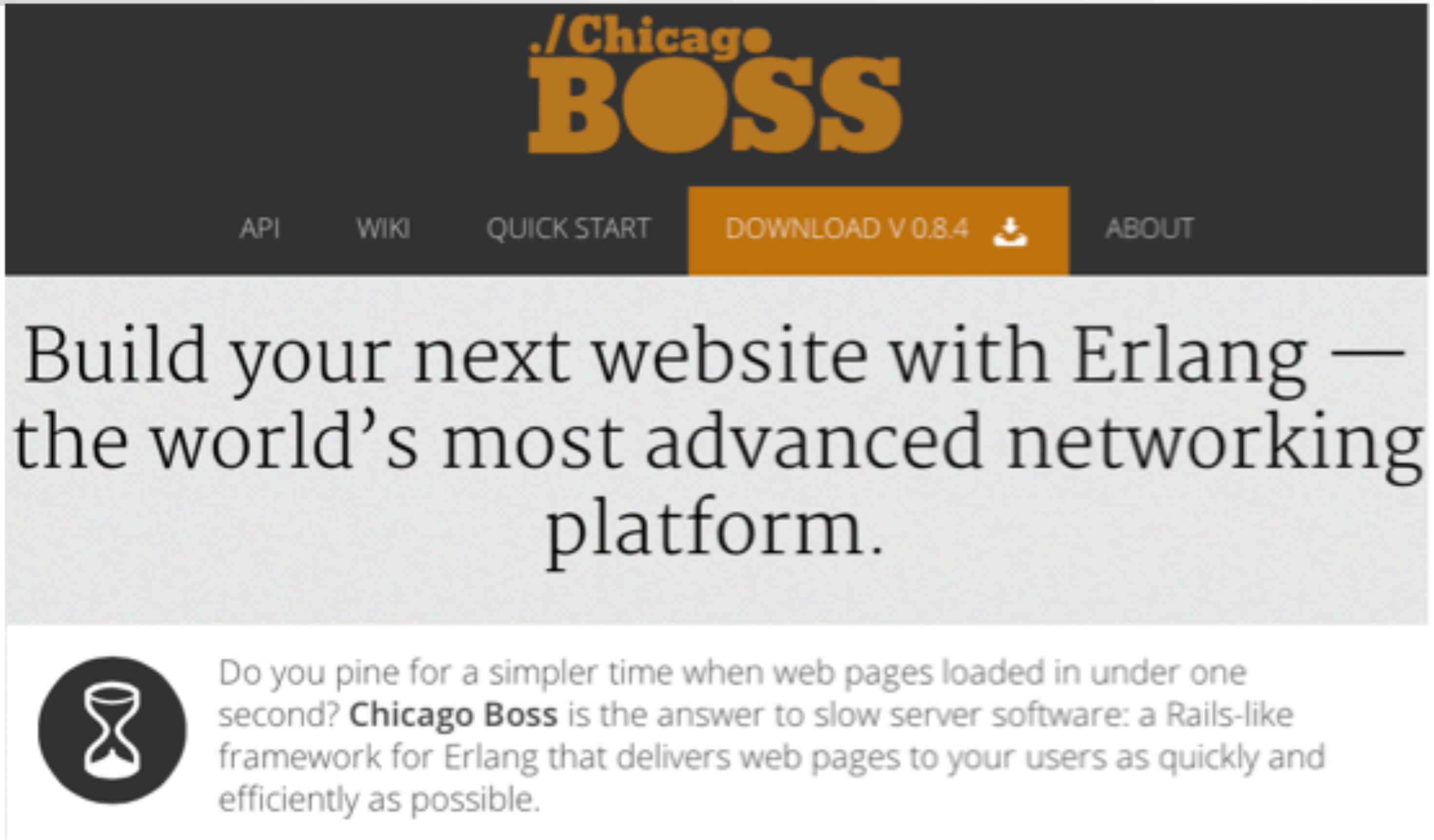
Mandelbrot- 2013

➢



Now for the Bottlenecks

-

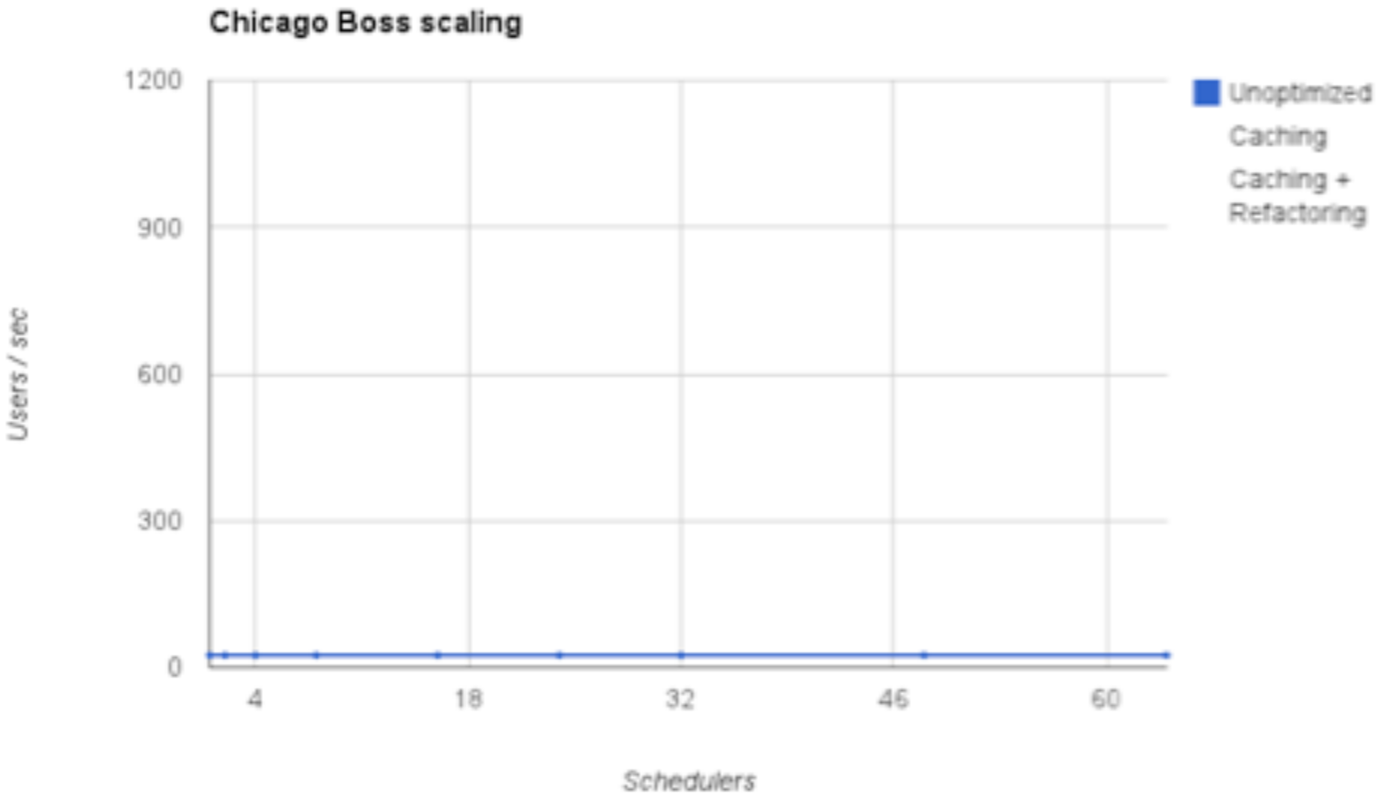


Build your next website with Erlang — the world's most advanced networking platform.

Do you pine for a simpler time when web pages loaded in under one second? **Chicago Boss** is the answer to slow server software: a Rails-like framework for Erlang that delivers web pages to your users as quickly and efficiently as possible.

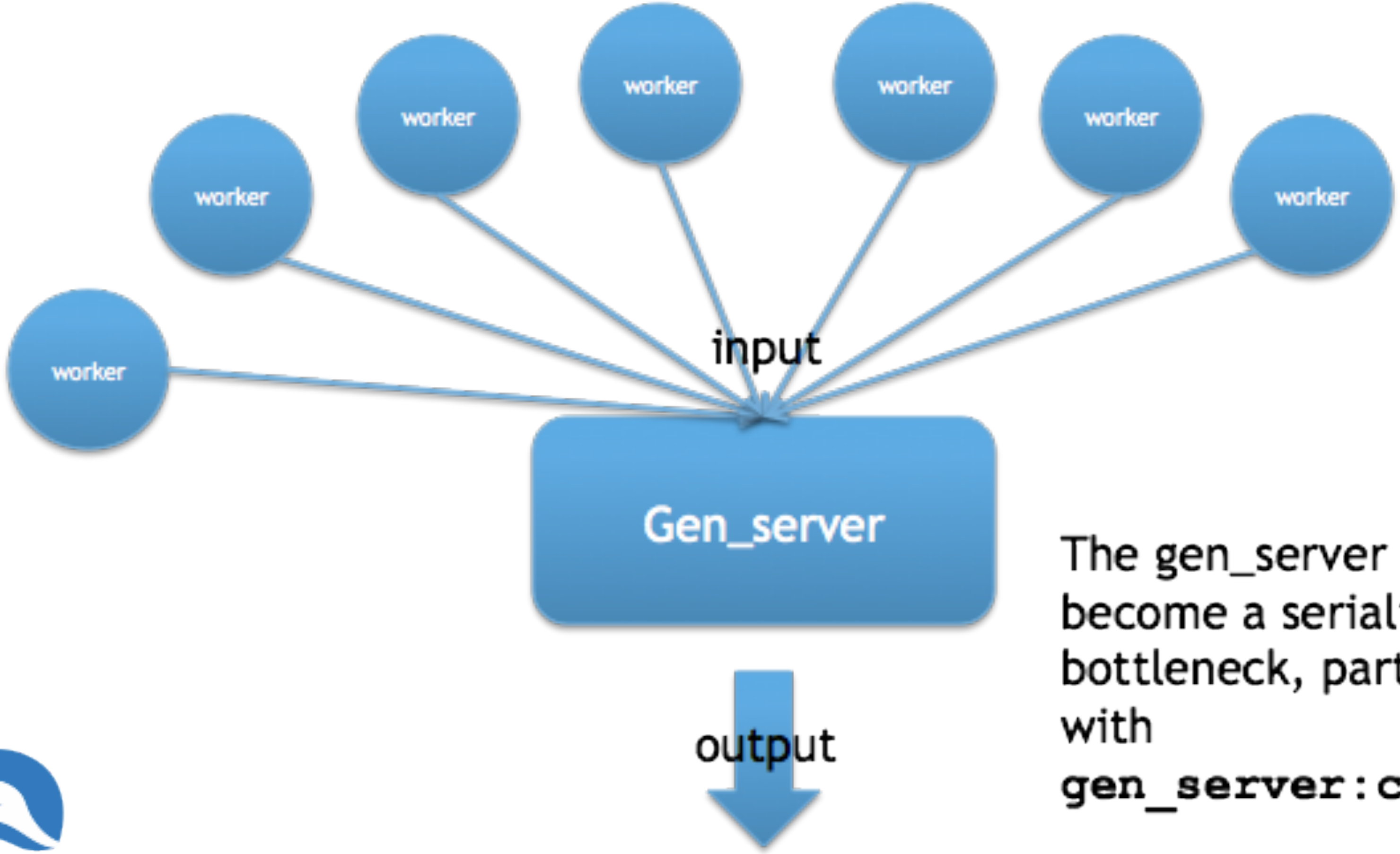
© 2014 – Erlang Solutions Ltd

Now for the Bottlenecks

- 

© 2014 – Erlang Solutions Ltd

Now for the Bottlenecks

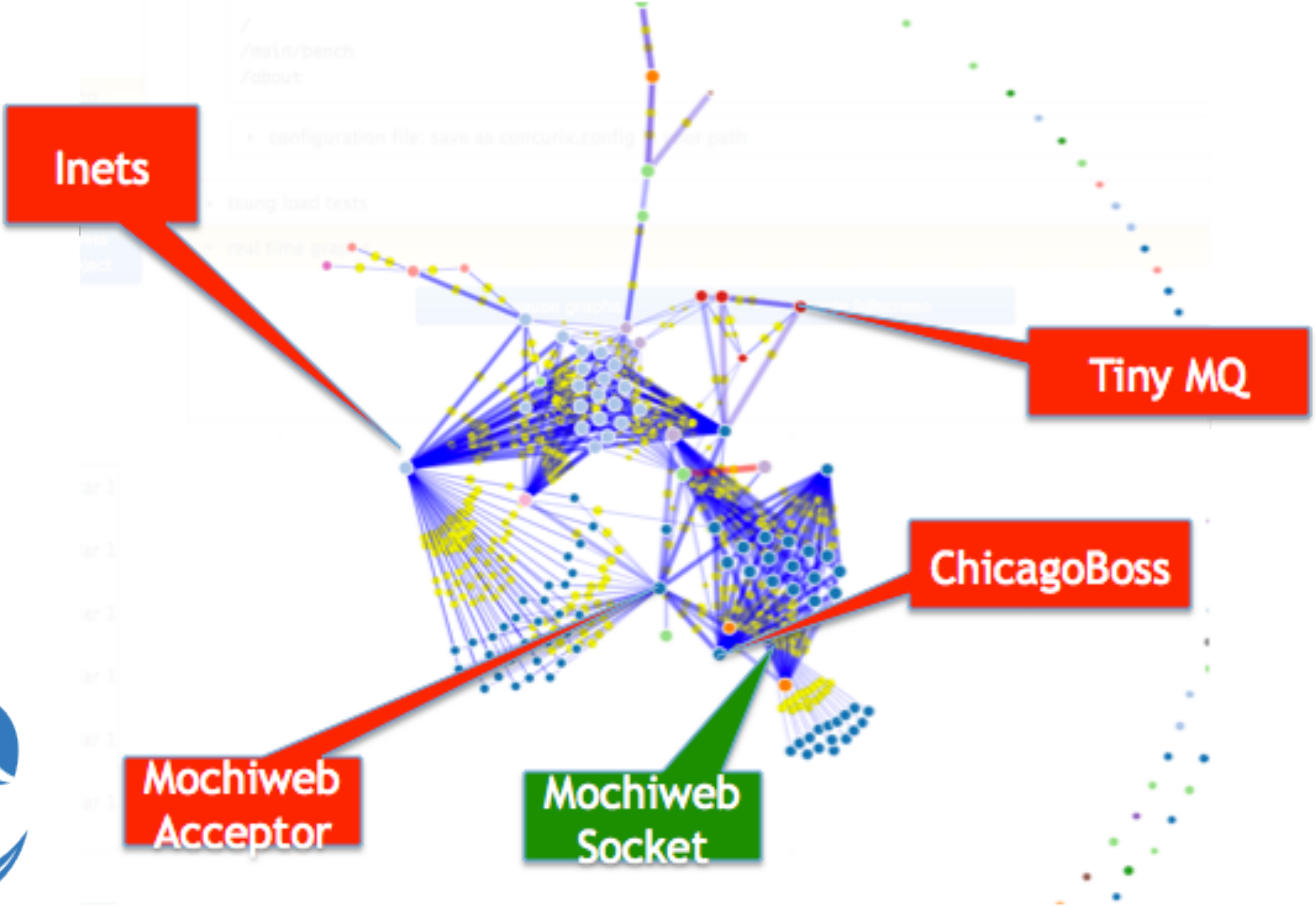


The gen_server can become a serialization bottleneck, particularly with **`gen_server:call(…)`**

© 2014 – Erlang Solutions Ltd

Now for the Bottlenecks

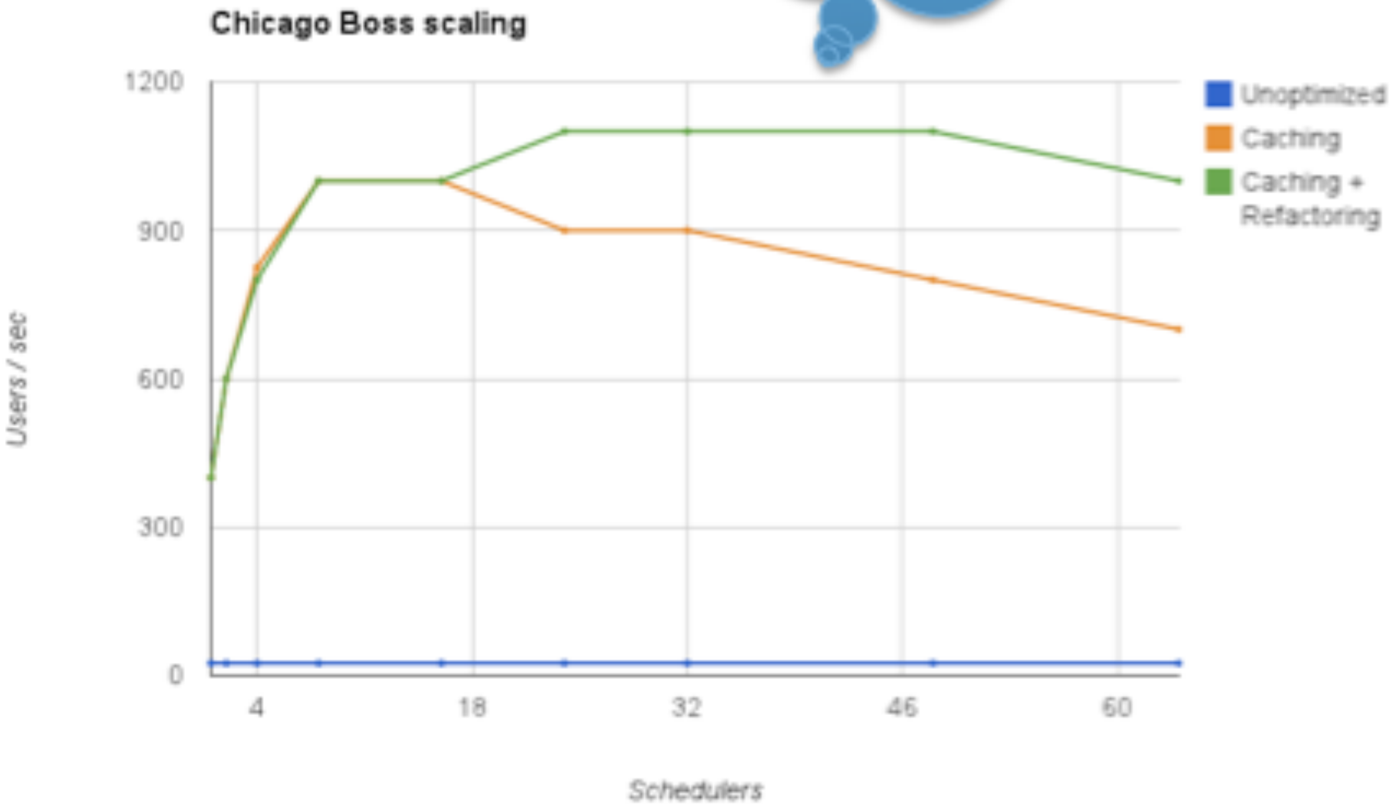
www.concurix.com



© 2014 – Erlang Solutions Ltd

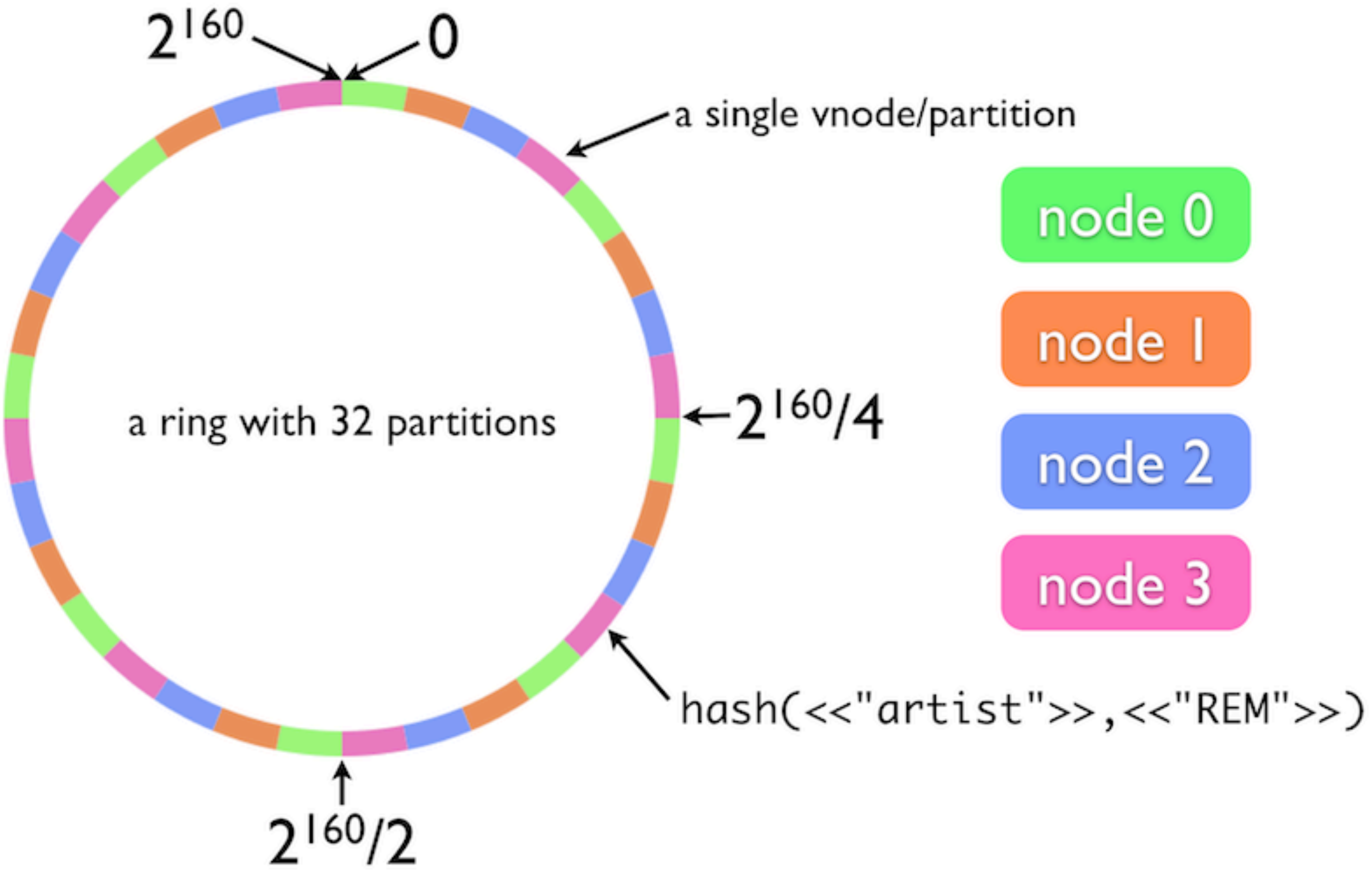
Now for the Bottlenecks

Tons of headroom still!



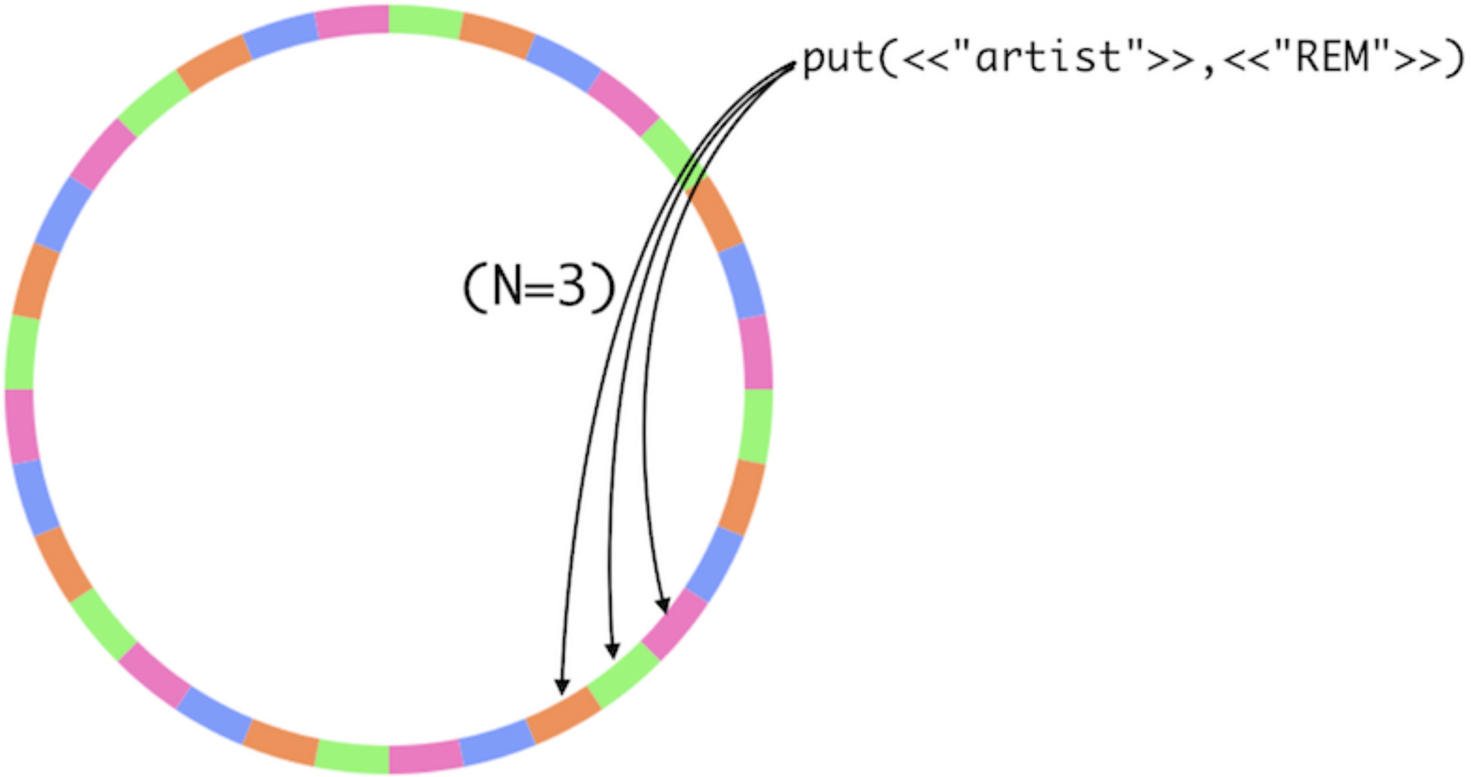
© 2014 – Erlang Solutions Ltd

Riak and other scalable architectures



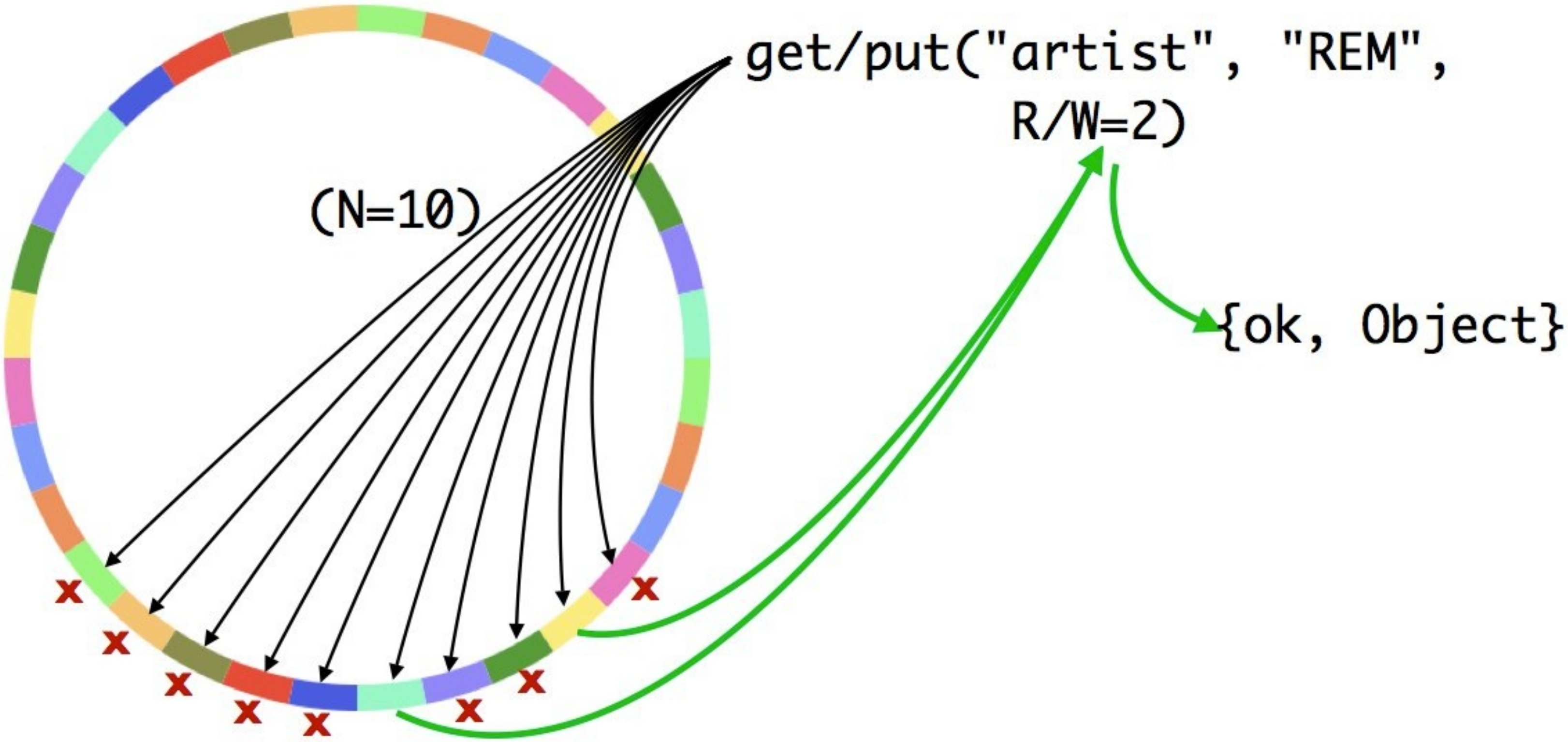
© 2014 – Erlang Solutions Ltd

N/R/W Values



© 2011 Erlang Solutions Ltd
© 2011 Erlang Solutions Ltd

N/R/W Values

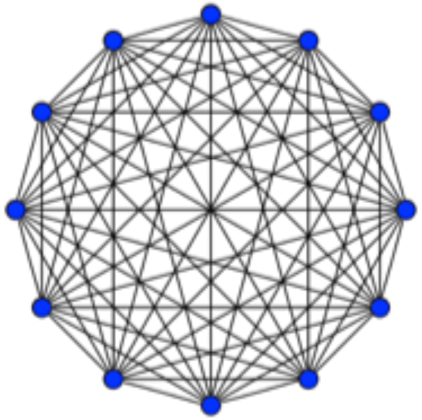


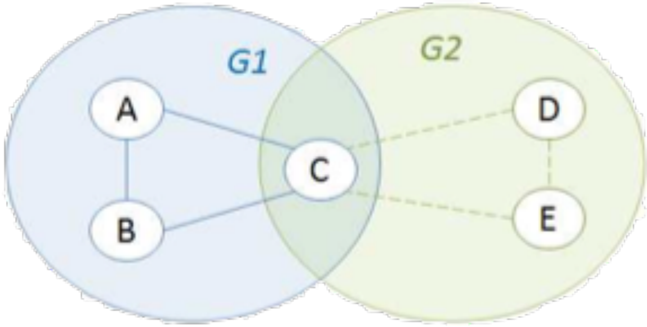
© 2011 Erlang Solutions Ltd

© 2011 Erlang Solutions Ltd

Clusters and SD Erlang

- Two major issues
  - Fully connected clusters
  - Explicit process placement
- Scalable Distributed (SD) Erlang
  - Nodes grouping
  - Non-transitive connections
  - Implicit process placement
  - Part of the standard Erlang/OTP package
- New concepts introduced
  - Locality, Affinity and Distance





© 2014 – Erlang Solutions Ltd

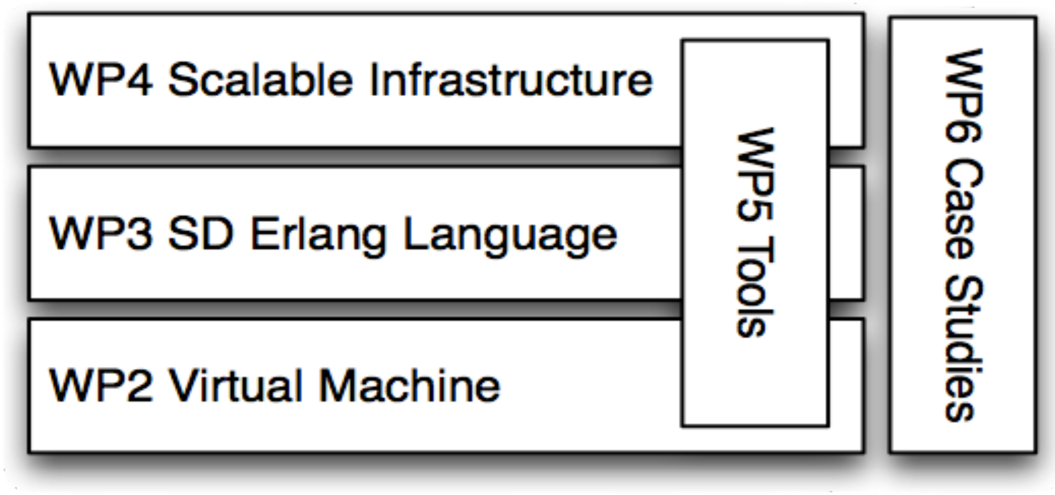
Release Statement of Aims

> “To scale the radical concurrency-oriented programming paradigm to build reliable general-purpose software, such as server-based systems, on massively parallel machines (10^5 cores).”

© 2014 – Erlang Solutions Ltd

Release

> *“Limitations exist on all levels. You would not want an Erlang VM to run with 10^5 schedulers.”*



WP4 Scalable Infrastructure

WP3 SD Erlang Language

WP2 Virtual Machine

WP5 Tools

WP6 Case Studies

© 2014 – Erlang Solutions Ltd

Release

Push the responsibility for scalability from the programmer to the VM

Analyze performance and scalability

Identify bottlenecks and prioritize changes and extensions

Tackle well-known scalability issues

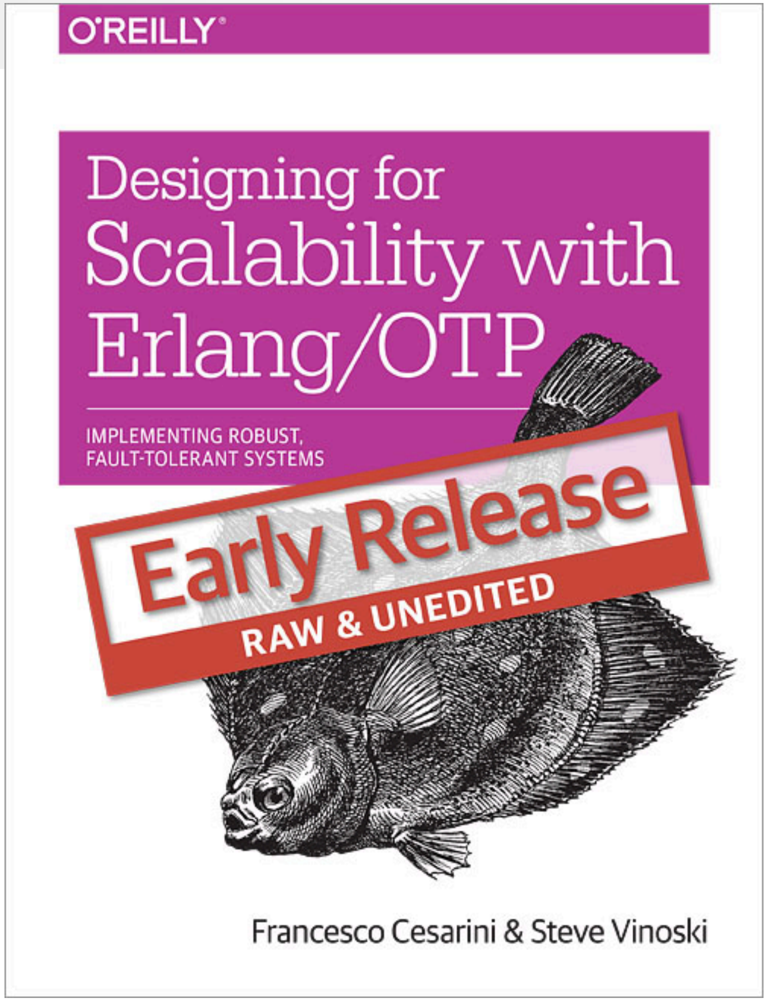
Ets tables (shared global data structure)

Message passing, copying and frequently communicating processes

© 2014 – Erlang Solutions Ltd

Thank You!

@francescoc
francesco@erlang-solutions.com



O'REILLY

Designing for Scalability with Erlang/OTP

IMPLEMENTING ROBUST, FAULT-TOLERANT SYSTEMS

Early Release
RAW & UNEDITED

Francesco Cesarini & Steve Vinoski

© 2014 – Erlang Solutions Ltd