

# SCALING ERLANG WEB APPLICATIONS

100 TO 100K USERS AT ONE WEB SERVER

Fernando Benavides (*@elbrujohalcon*)

Inaka Labs

March 20, 2012



Inaka Networks presents... El Brujo Halcón in... Scaling Erlang  
(Based on a true story)



Inaka Networks presents... El Brujo Halcón in... Scaling Erlang  
(Based on a true story)



# Inaka Networks presents... El Brujo Halcón in... Scaling Erlang

(Based on a true story)



Inaka Networks presents... El Brujo Halcón in... Scaling Erlang  
(Based on a true story)



A not so long time ago in a country far far away...



Hey! Boca is playing at the Bombonera now! Ok, let's watch it! I can't, I'm at *the office* ... We need an app for that!



... So we created MatchStream ...





Ok, then... We know there will be hundreds of thousands of users, right? We need the system to **scale** Of course! We should use Erlang!



Ok, then... We know there will be hundreds of thousands of users, right? We need the system to **scale** Of course! We should use Erlang!



Several days after that... MatchStream is ready! TODO:  
INSERT SYSTEM DESCRIPTION HERE



Several days after that... MatchStream is ready! TODO:  
INSERT SYSTEM DESCRIPTION HERE



Boca plays again today, let's try it out with this game! Wait, we can't handle more than 1000 users?! WTF?!?! And they can only connect four at a time???



Boca plays again today, let's try it out with this game! Wait, we can't handle more than 1000 users?! WTF?!?! And they can only connect four at a time???



Boca plays again today, let's try it out with this game! Wait, we can't handle more than 1000 users?! WTF?!?! And they can only connect four at a time???



Lesson Learned: Just using Erlang is not enough to make your system scale





So... What did we do?



We made sure the system was working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system



We made sure the system was working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system



We made sure the system was working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system



We made sure the system was working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system



1024 users / 4 at a time



The system is fine, let's tune up the server where it's installed

So, we checked the kernel variables and system limits for

- Concurrent TCP connections
- Open files limit
- TCP backlog size
- Swap memory allocation
- Erlang VM process limit



The system is fine, let's tune up the server where it's installed  
So, we checked the kernel variables and system limits for

- Concurrent TCP connections
- Open files limit
- TCP backlog size
- TCP memory allocation
- Erlang VM process limit





The system is fine, let's tune up the server where it's installed  
So, we checked the kernel variables and system limits for

- Concurrent TCP connections
- Open files limit
- TCP backlog size
- TCP memory allocation
- Erlang VM process limit



The system is fine, let's tune up the server where it's installed  
So, we checked the kernel variables and system limits for

- Concurrent TCP connections
- Open files limit
- TCP backlog size
- TCP memory allocation
- Erlang VM process limit



The system is fine, let's tune up the server where it's installed  
So, we checked the kernel variables and system limits for

- Concurrent TCP connections
- Open files limit
- TCP backlog size
- TCP memory allocation
- Erlang VM process limit



The system is fine, let's tune up the server where it's installed  
So, we checked the kernel variables and system limits for

- Concurrent TCP connections
- Open files limit
- TCP backlog size
- TCP memory allocation
- Erlang VM process limit



4096 users / 4 at a time



I've got a friend that may help us, he has a bag with several tips and tricks for us... MacGyver



I've got a friend that may help us, he has a bag with several tips and tricks for us... MacGyver



# STEP 3

## CONNECTION TWEAKS

### BACKLOG

- Allow more concurrent connections
- Remember HTTP *runs on* TCP

### CONNECTIONS

- Don't use just one of them
- Check inbound and outbound connections





# STEP 3

## CONNECTION TWEAKS

### BACKLOG

- Allow more concurrent connections
- Remember HTTP *runs on* TCP

### CONNECTIONS

- Don't use just one of them
- Check inbound and outbound connections



TODO users / TODO at a time



## SUP\_HANDLER

- Don't use it
- Monitor the processes instead

## LONG DELIVERY QUEUES

- Use *repeaters*



## SUP\_HANDLER

- Don't use it
- Monitor the processes instead

## LONG DELIVERY QUEUES

- Use *repeaters*



TODO users / TODO at a time



## CALL TIMEOUTS

Remember `gen_server:reply/2`

## MEMORY FOOTPRINT

Remember `hibernate`

## LONG INIT/1

Use 0 timeout



## CALL TIMEOUTS

Remember `gen_server:reply/2`

## MEMORY FOOTPRINT

Remember `hibernate`

## LONG INIT/1

Use 0 timeout



## CALL TIMEOUTS

Remember `gen_server:reply/2`

## MEMORY FOOTPRINT

Remember `hibernate`

## LONG INIT/1

Use 0 timeout





TODO users / TODO at a time



- Sometimes `simple_one_for_one` supervisors get **overburdened** because they have too many children
- Try a supervisor hierarchy with several managers below the main supervisor
- Turn `supervisor:start_child/2` calls into something like

```
supervisor:start_child(  
  list_to_atom("module-name_" ++  
               integer_to_list(random:uniform(#ofSupervisors))).
```



TODO users / TODO at a time



# STEP 3

## OTHER PROCESSES

### TIMERS

- Don't use the `timer` module
- Use `erlang:send_after`

### LOGGING

- Don't log too much
- Use a good logging system

### REGISTRATION

- Sometimes it's better to register processes instead of keeping track of their pids manually
- You can always register processes **both** locally and globally



### TIMERS

- Don't use the `timer` module
- Use `erlang:send_after`

### LOGGING

- Don't log too much
- Use a good logging system

### REGISTRATION

- Sometimes it's better to register processes instead of keeping track of their pids manually
- You can always register processes **both** locally and globally



### TIMERS

- Don't use the `timer` module
- Use `erlang:send_after`

### LOGGING

- Don't log too much
- Use a good logging system

### REGISTRATION

- Sometimes it's better to register processes instead of keeping track of their pids manually
- You can always register processes **both** locally and globally



64000 users / 8000 at a time



TODO: Img of what the system looks like at this point





## STEP 4

Well, let's add some nodes to it!



# STEP 4

## ADDING NODES

Again, it's not as easy as just starting the app in another Erlang node We needed to find the best topology, we considered using:

- connected nodes
- independent nodes

We had to decide which processes needed to communicate and how and of course, test the whole system again



# STEP 4

## ADDING NODES

Again, it's not as easy as just starting the app in another Erlang node. We needed to find the best topology, we considered using:

- connected nodes
- independent nodes

We had to decide which processes needed to communicate and how and of course, test the whole system again



# STEP 4

## ADDING NODES

Again, it's not as easy as just starting the app in another Erlang node. We needed to find the best topology, we considered using:

- connected nodes
- independent nodes

We had to decide which processes needed to communicate and how and of course, test the whole system again



# STEP 4

## ADDING NODES

Again, it's not as easy as just starting the app in another Erlang node. We needed to find the best topology, we considered using:

- connected nodes
- independent nodes

We had to decide which processes needed to communicate and how and of course, test the whole system again



25000 users per node / 8000 per computer at a time with 4  
nodes on the same computer... 100K users / 8000 at a time



25000 users per node / 8000 per computer at a time with 4  
nodes on the same computer... 100K users / 8000 at a time



25000 users per node / 8000 per computer at a time with 4  
nodes on the same computer... 100K users / 8000 at a time

