Scaling Erlang Web Applications

100 to 100K users at one web server

Fernando Benavides (*@elbrujohalcon*)

Inaka Labs

March 20, 2012

INAKA NETWORKS

presents . . .

INAKA NETWORKS

presents ...

El Brujo Halcón

in . . .

El Brujo Halcón

in . . .

Scaling Erlang

Based on a true story

SCALING ERLANG

Based on a true story

A *not so* long time ago

A *not so* long time ago

in a country far far away. . .

A Friend

Hey! Let's watch the *superclásico*!!!

Brujo

I can't, I'm at *the office*

A Friend

. . .

Brujo

We need an app for that!

A FRIEND

Hey! Let's watch the *superclásico*!!!

BRUJO

I can't, I'm at *the office*

A FRIEND

...

BRUJO

We need an app for that!

A FRIEND

Hey! Let’s watch the *superclásico*!!!

BRUJO

I can’t, I’m at *the office*

A FRIEND

...

BRUJO

We need an app for that!

A FRIEND

Hey! Let's watch the *superclásico*!!!

BRUJO

I can't, I'm at *the office*

A FRIEND

...

BRUJO

We need an app for that!

BRUJO

Let's call it MATCHSTREAM

A FRIEND

Ok, then. . . We know there will be hundreds of thousands of users, right?
We need the system to **scale**

BRUJO

Of course! We should use Erlang!

BRUJO

Let's call it MATCHSTREAM

A FRIEND

Ok, then... We know there will be hundreds of thousands of users, right?
We need the system to **scale**

BRUJO

Of course! We should use Erlang!

BRUJO

Let’s call it MATCHSTREAM

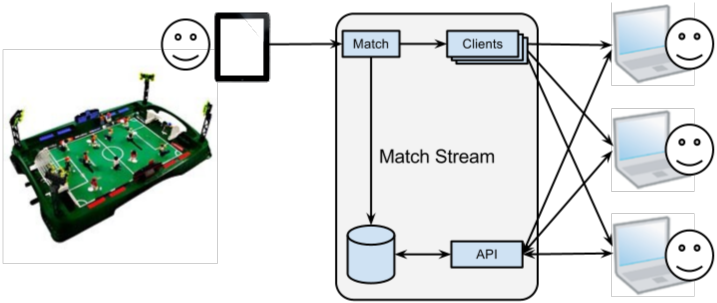
A FRIEND

Ok, then. . . We know there will be hundreds of thousands of users, right?
We need the system to **scale**

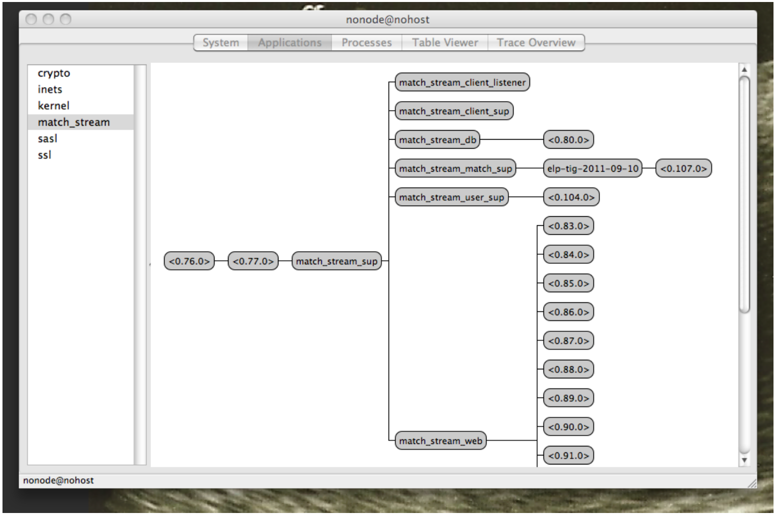
BRUJO

Of course! We should use Erlang!

A while later. . .


Match
Clients
Match Stream
API

MatchStream
Architecture



TODO: Take this picture with client(s) connected

BRUJO

Boca plays again today, let's try our system out with this game!
What can **possibly** go wrong?

USER 1

Wow! MATCHSTREAM is awesome!

. . .

USER 100

Hey! this system is a total crap! It doesn't even let me connect to it!

BRUJO

WTF?! The system doesn't scale!!

A FRIEND

Didn't you use Erlang?

Brujo
: Boca plays again today, let's try our system out with this game!
What can **possibly** go wrong?

User 1
: Wow! MatchStream is awesome!

. . .

User 100
: Hey! this system is a total crap! It doesn't even let me connect to it!

Brujo
: WTF?! The system doesn't scale!!

A Friend
: Didn't you use Erlang?

BRUJO

Boca plays again today, let's try our system out with this game!
What can **possibly** go wrong?

USER 1

Wow! MATCHSTREAM is awesome!

. . .

USER 100

Hey! this system is a total crap! It doesn't even let me connect to it!

BRUJO

WTF?! The system doesn't scale!!

A FRIEND

Didn't you use Erlang?

BRUJO
: Boca plays again today, let's try our system out with this game!
What can **possibly** go wrong?

USER 1
: Wow! MATCHSTREAM is awesome!

. . .

USER 100
: Hey! this system is a total crap! It doesn't even let me connect to it!

BRUJO
: WTF?! The system doesn't scale!!

A FRIEND
: Didn't you use Erlang?

BRUJO

Boca plays again today, let’s try our system out with this game!
What can **possibly** go wrong?

USER 1

Wow! MATCHSTREAM is awesome!

...

USER 100

Hey! this system is a total crap! It doesn’t even let me connect to it!

BRUJO

WTF?! The system doesn’t scale!!

A FRIEND

Didn’t you use Erlang?

BRUJO

Boca plays again today, let's try our system out with this game!
What can **possibly** go wrong?

USER 1

Wow! MATCHSTREAM is awesome!

. . .

USER 100

Hey! this system is a total crap! It doesn't even let me connect to it!

BRUJO

WTF?! The system doesn't scale!!

A FRIEND

Didn't you use Erlang?

Lesson Learned

Just using Erlang is not enough to make your system scale

So, we made it scale. . .

Step 1

First of all we wanted to be sure that the system was actually working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system
- We found its initial scale limits

Step 1

First of all we wanted to be sure that the system was actually working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system
- We found its initial scale limits

Step 1

First of all we wanted to be sure that the system was actually working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system
- We found its initial scale limits

Step 1

First of all we wanted to be sure that the system was actually working.

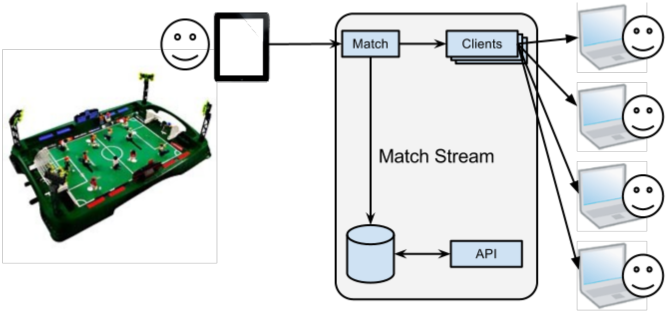
- We built a simulator
- We improved the logging mechanisms
- We tested the system
- We found its initial scale limits

Step 1

First of all we wanted to be sure that the system was actually working.

- We built a simulator
- We improved the logging mechanisms
- We tested the system
- We found its initial scale limits

Step 1
Results



N = 1024 / C = 4

Step 2

Once we knew the system was fine, we decided to tune up the server where it was 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

Step 2

Once we knew the system was fine, we decided to tune up the server where it was 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

Step 2

Once we knew the system was fine, we decided to tune up the server where it was 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

Step 2

Once we knew the system was fine, we decided to tune up the server where it was 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

Step 2

Once we knew the system was fine, we decided to tune up the server where it was installed. So, we checked the kernel variables and system limits for

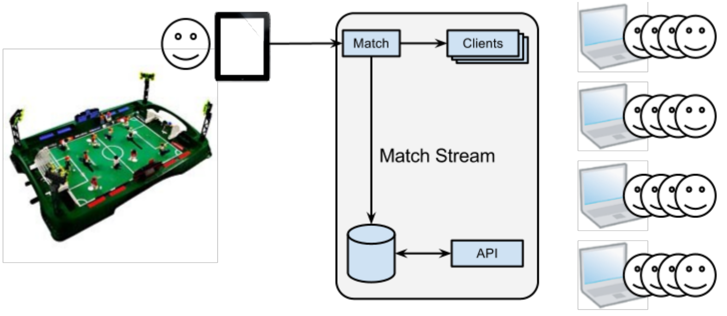
- Concurrent TCP connections
- Open files limit
- TCP backlog size
- TCP memory allocation
- Erlang VM process limit

Step 2

Once we knew the system was fine, we decided to tune up the server where it was 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

Step 2
Results



N = 4096 / C = 4

Step 3

Then we decided to start improving the different components of the system.

We called a friend to help us...

Step 3

Then we decided to start improving the different components of the system.
We called a friend to help us...

Step 3

Then we decided to start improving the different components of the system.
We called a friend to help us...

Backlog

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

Connections

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

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

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

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`

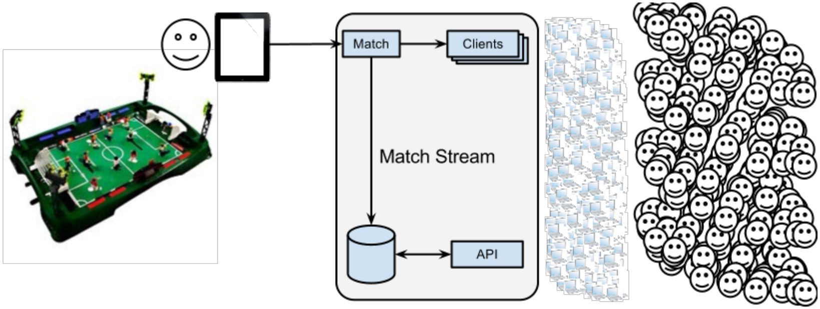
LOGGING

- 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

Step 3
Results



N = 65536 / C = 8192

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

Step 4

Well, let's add some nodes to it!

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

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

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:

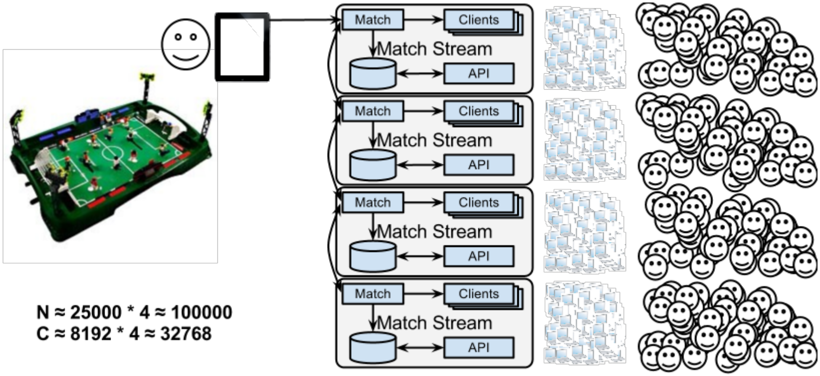
- connected nodes
- independent nodes

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

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



N ≈ 25000 * 4 ≈ 100000
C ≈ 8192 * 4 ≈ 32768