THE ROAD TO 2 MILLION WEB SOCKET CONNECTIONS WITH PHOENIX

BLOG POST

http://www.phoenixframework.org/blog/the-road-to-2-million-websocket-connections

GARY RENNIE

Gazler @TheGazler

TALK FORMAT

- Why?
- How? (Beware: contains XML)
- Results
- Future

WHY?

- WhatsApp used as a reference for all Phoenix presentations
- Lots of Phoenix HTTP benchmarks, 0 for persistent connections

https://blog.whatsapp.com/196/1-million-is-so-2011
https://github.com/mroth/phoenix-showdown
http://www.littlelines.com/blog/2014/07/08/elixir-vs-ruby-showdown-phoenix-vs-rails/

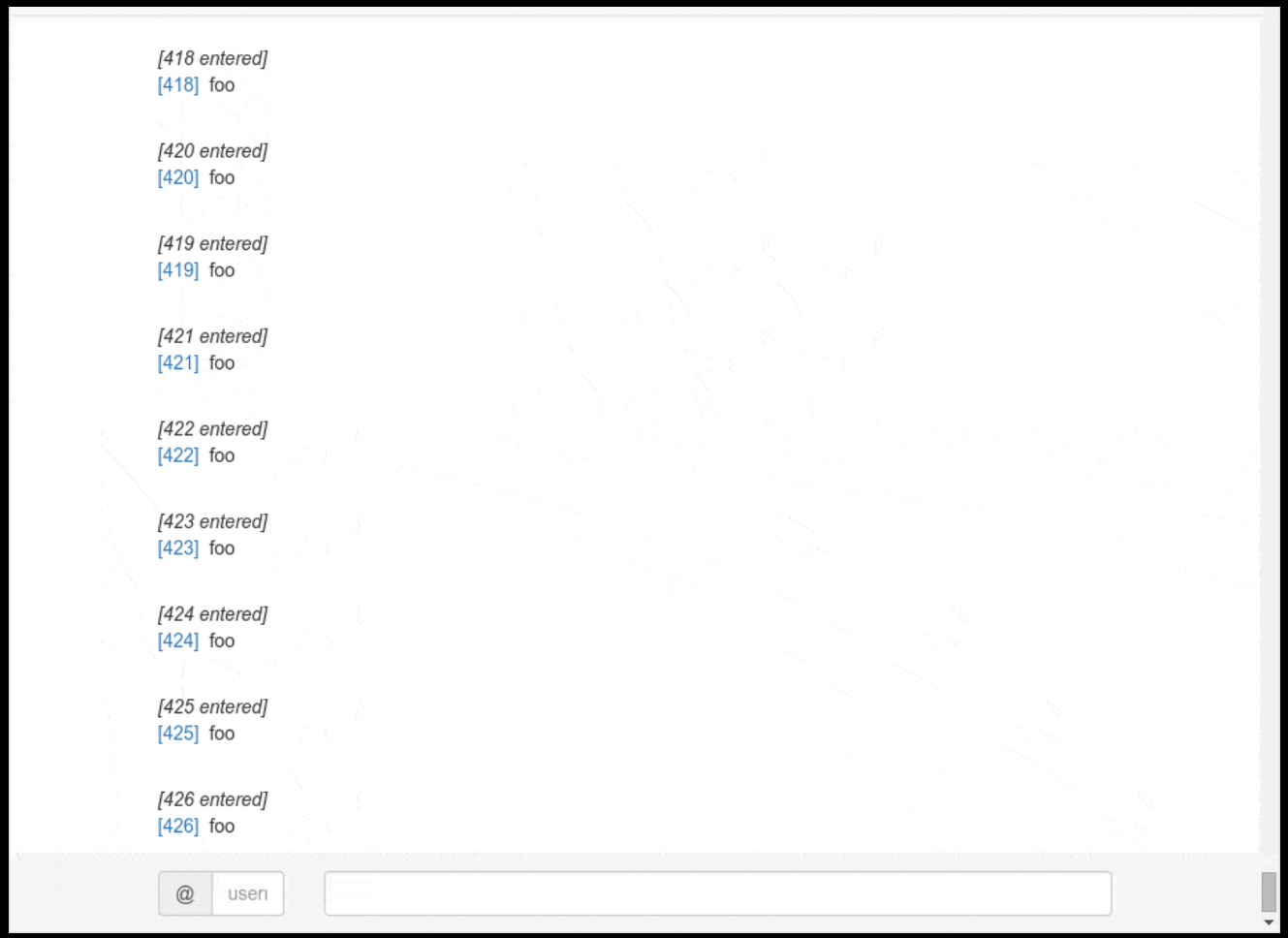
WHAT MAKES BENCHMARKING WEB SOCKETS DIFFERENT?

- Uses Web Socket protocol
- Connections have to stay open
- Can't recycle connections
- Increased memory usage
- Hard limit on connections per machine (~64k however adding additional ip addresses/using different ports can fix this)

CHAT APPLICATION

https://github.com/chrismccord/phoenix_chat_example

```
diff --git a/web/channels/room_channel.ex b/web/channels/room_channel.ex
index bc92759..66ead5c 100644
--- a/web/channels/room_channel.ex
+++ b/web/channels/room_channel.ex
@@ -14,8 +14,6 @@ defmodule Chat.RoomChannel do
   """
   def join("rooms:lobby", message, socket) do
     Process.flag(:trap_exit, true)
-    :timer.send_interval(5000, :ping)
-    send(self, {:after_join, message})

     {:ok, socket}
   end
```


[418 entered]
[418] foo
[420 entered]
[420] foo
[419 entered]
[419] foo
[421 entered]
[421] foo
[422 entered]
[422] foo
[423 entered]
[423] foo
[424 entered]
[424] foo
[425 entered]
[425] foo
[426 entered]
[426] foo
@
usern

TSUNG

- An open-source multi-protocol distributed load testing tool
- Written in Erlang
- Started in 2001 as Idx-Tsunami
- Supports Web Sockets (As of version 1.5)
- Supports multiple machines
- Produces pretty charts (web dashboard as of 1.6)

https://github.com/processone/tsung

TSUNG CONFIG

```
<?xml version="1.0"?>
<!DOCTYPE tsung SYSTEM "/user/share/tsung/tsung-1.0.dtd">
<tsung loglevel="debug" version="1.0">
  <clients></clients>
  <servers></servers>
  <load></load>
  <sessions></sessions>
</tsung>
```

- Machine running with the config is the controller
- Other machines are clients (the controller can also be a client)

TSUNG CLIENTS

```
<clients>
  <client host="phoenix1" weight="1" cpu="4" use_controller_vm="false" maxusers="60000" />
  <client host="phoenix2" weight="2" cpu="4" use_controller_vm="false" maxusers="60000">
    <ip value="10.9.195.12"></ip>
    <ip value="10.9.195.13"></ip>
  </client>
  <client host="phoenix3" weight="1" cpu="4" use_controller_vm="false" maxusers="60000" />
</clients>
```

- One client per machine
- Keep maxusers below machine limit
- Can set a weight on the machines (load ratio)
- Can set multiple virtual IP addresses (we didn't use this)
- use_controller_vm="false" - don't share Erlang VM

TSUNG SERVERS

```
<servers>
  <server host="server1" port="4000" type="tcp" weight="4"></server>
  <server host="server2" port="4000" type="tcp" weight="1"></server>
</servers>
```

- Can set multiple servers (we only used 1)
- Can set a weight on the machines (load ratio)
- Can set a connection type (tcp, ssl, udp, websocket)
- We actually used the tcp type instead of websocket as websocket didn't stay open

TSUNG LOAD

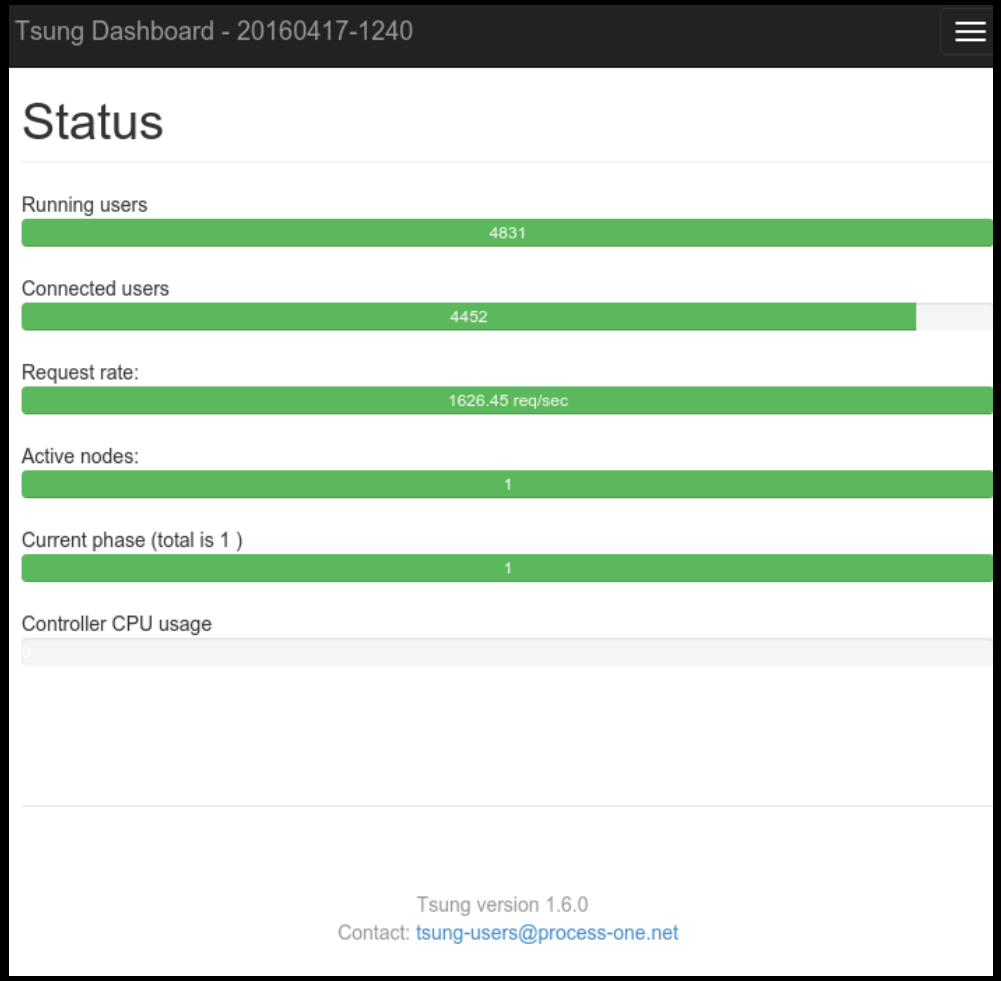
```
<load duration="1" unit="hour">
  <arrivalphase phase="1" duration="10000" unit="second">
    <users maxnumber="100000" arrivalrate="1000" unit="second" />
  </arrivalphase>
</load>
```

- Can set multiple phases (we used 1)
- Phases have a duration
- Total load can have a duration (max just under 50 days!)
- Phases can be looped
- We set duration high and manually terminated
- Arrival rate is the number of connections arriving

TSUNG SESSIONS

```
<sessions>
  <session name="websocket" probability="100" type="ts_websocket">
    <request><websocket type="connect" path="/socket/websocket"></websocket></request>
    <request subst="true">
      <websocket type="message">{"topic":"rooms:lobby", "event":"phx_join", "payload":
      {"user":"%%ts_user_server:get_unique_id%%"}, "ref":"1"}</websocket>
    </request>
    <for var="i" from="1" to="1000" incr="1"><thinktime value="30"/></for>
  </session>
</sessions>
```

- How users interact with the application (we connect and wait)
- Can use Erlang terms (%%ts_user_server:get_unique_id%%)
- Can use different request types (websocket, http)
- Different probabilities per session


Tsung Dashboard - 20160417-1240
Status
Running users
4831
Connected users
4452
Request rate:
1626.45 req/sec
Active nodes:
1
Current phase (total is 1)
1
Controller CPU usage
Tsung version 1.6.0
Contact: tsung-users@process-one.net

Tsung Dashboard - 20140429-1339

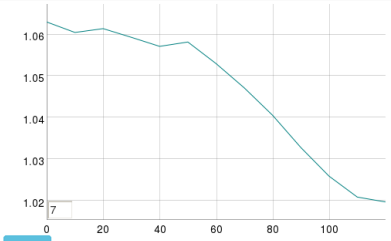
Status Reports Graphs Logs

- Main statistics
- Transactions
- Network Throughput
- Counters
- Server monitoring
- HTTP status
- Response times
- Throughput graphs
- Simultaneous Users
- Server monitoring
- HTTP status

20140429-1342: Report and graphs generated in 0.26 sec ×

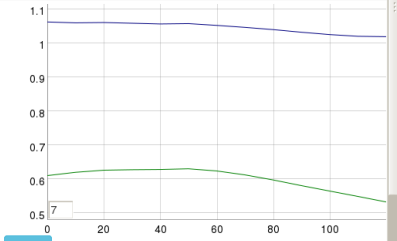
Response Time

Transactions



Info »

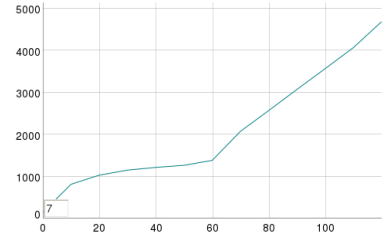
Requests and connection establishment



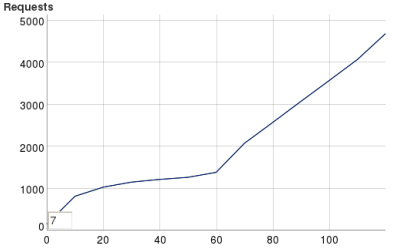
Info »

Throughput

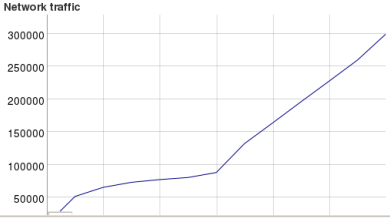
Transactions



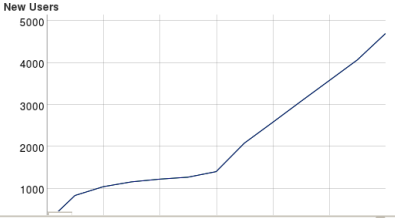
Requests



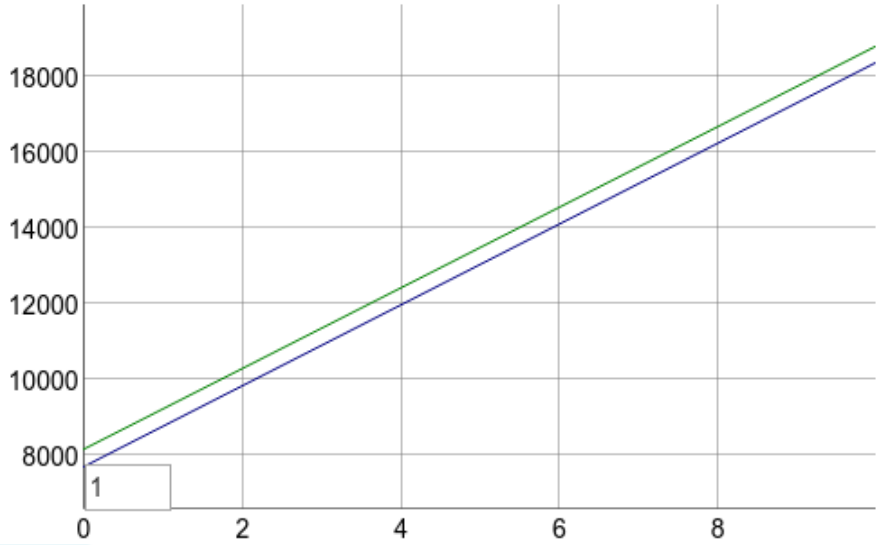
Network traffic



New Users



Simultaneous Users



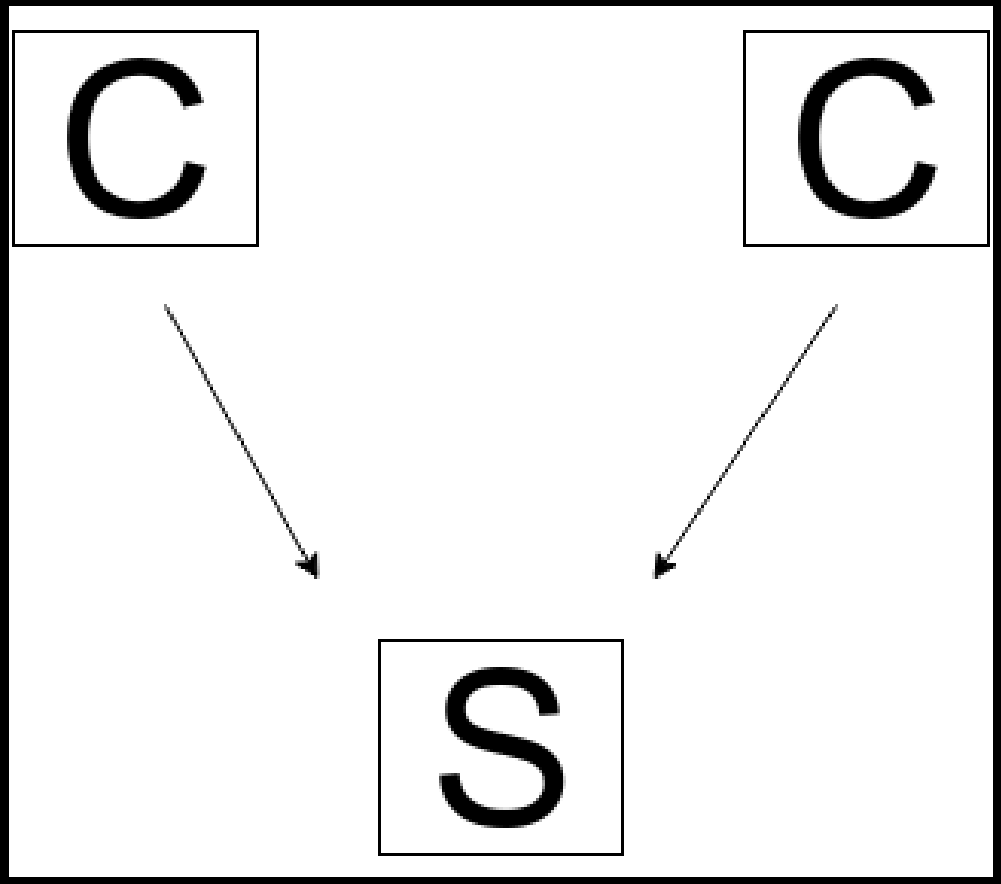
Info »

Simultaneous Users

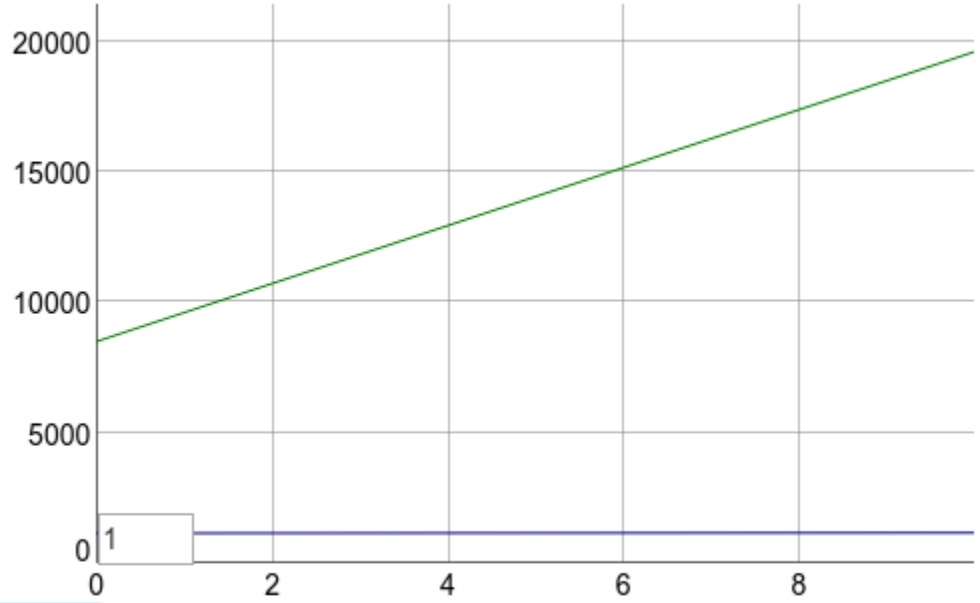
THE FIRST RUN OF TSUNG

```
tsung -f config.xml start
```

- 1 server Rackspace I/O v1
- 15GB RAM, 4 cores
- 2 clients as above


C
C
S

Simultaneous Users



Info »

Simultaneous Users

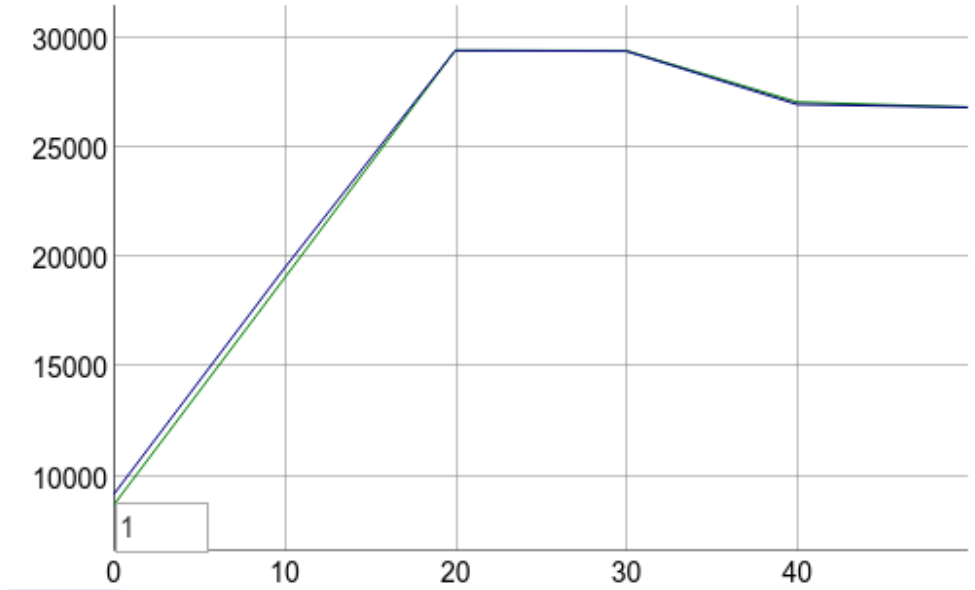

WTF?!?!

SERVER CONFIG

```
ulimit -n 2000000
```

Make sure it is set after restarts!

Simultaneous Users



Info »

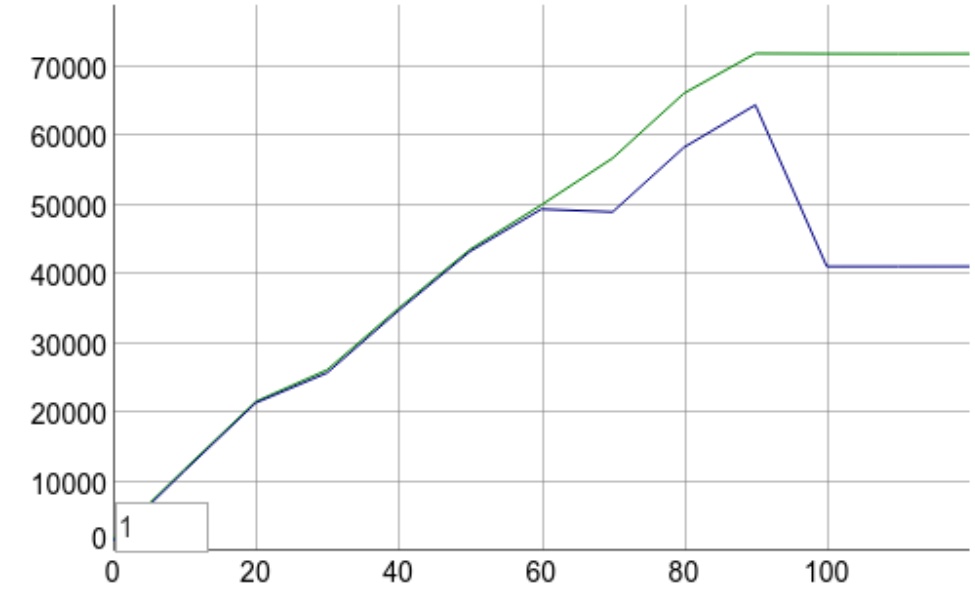
Simultaneous Users

FIRST OPTIMIZATION

```
061c69b Only rely on ETS tables inside local (5 months ago, José Valim)

 lib/phoenix/pubsub/local.ex | 83 +++++++++++++++-----------------------------------
 1 file changed, 14 insertions(+), 69 deletions(-)
```

- Remove local HashDict of PIDs
- Use topics ETS tables to find subscribers (PIDs)

Simultaneous Users

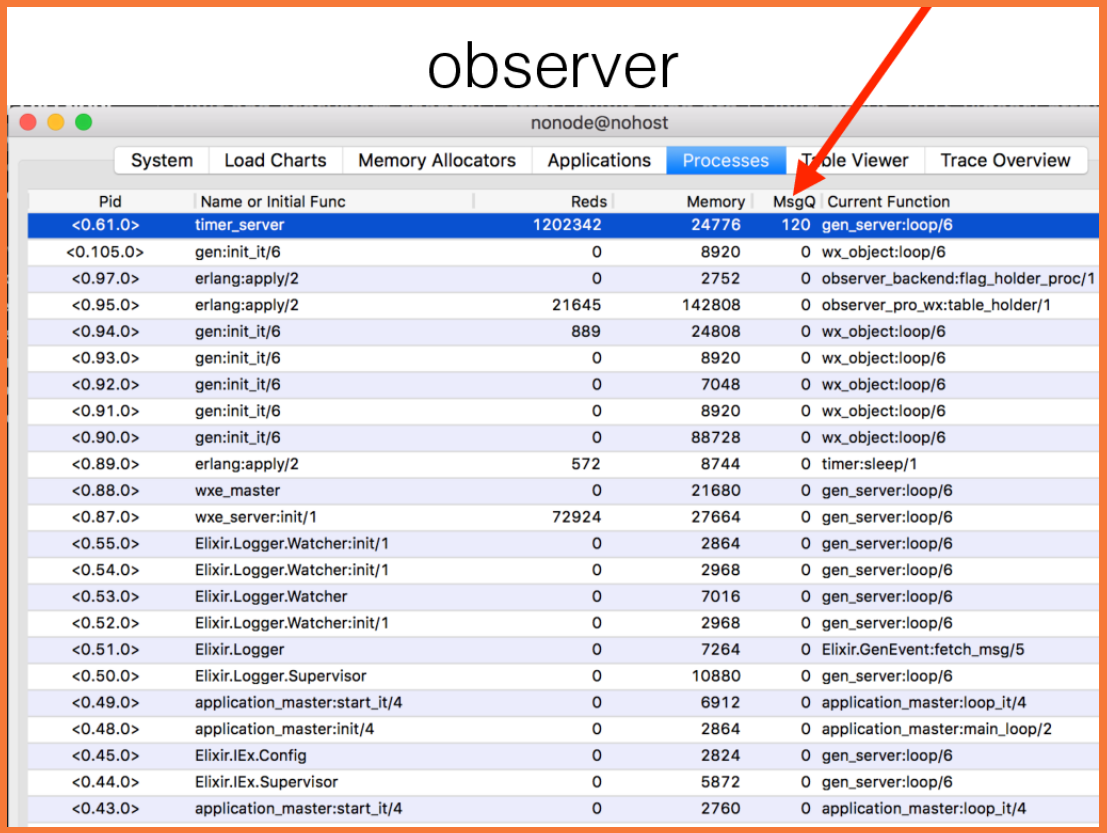


Info »

Simultaneous Users

USING OBSERVER

observer



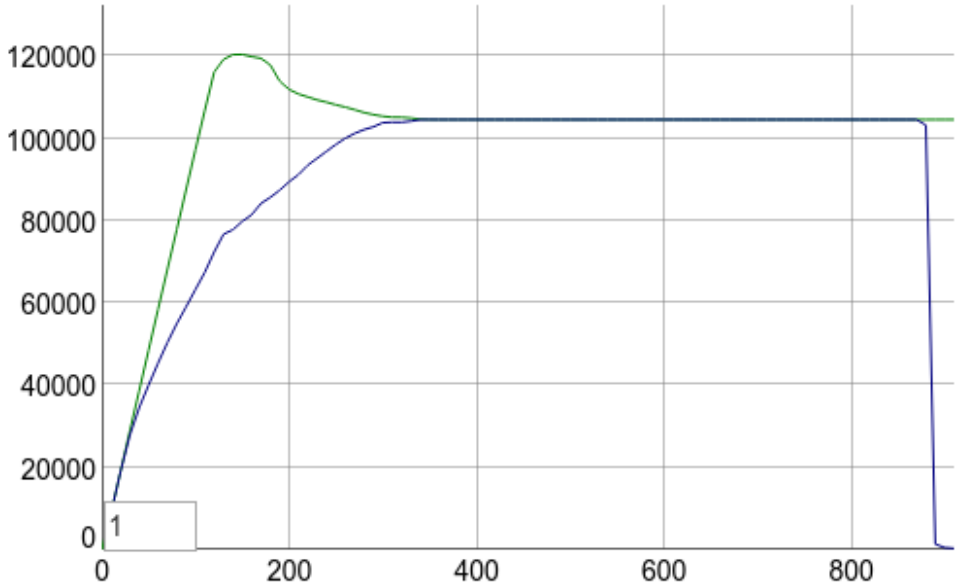
Pid	Name or Initial Func	Reds	Memory	MsgQ	Current Function
<0.61.0>	timer_server	1202342	24776	120	gen_server:loop/6
<0.105.0>	gen:init_it/6	0	8920	0	wx_object:loop/6
<0.97.0>	erlang:apply/2	0	2752	0	observer_backend:flag_holder_proc/1
<0.95.0>	erlang:apply/2	21645	142808	0	observer_pro_wx:table_holder/1
<0.94.0>	gen:init_it/6	889	24808	0	wx_object:loop/6
<0.93.0>	gen:init_it/6	0	8920	0	wx_object:loop/6
<0.92.0>	gen:init_it/6	0	7048	0	wx_object:loop/6
<0.91.0>	gen:init_it/6	0	8920	0	wx_object:loop/6
<0.90.0>	gen:init_it/6	0	88728	0	wx_object:loop/6
<0.89.0>	erlang:apply/2	572	8744	0	timer:sleep/1
<0.88.0>	wxe_master	0	21680	0	gen_server:loop/6
<0.87.0>	wxe_server:init/1	72924	27664	0	gen_server:loop/6
<0.55.0>	Elixir.Logger.Watcher:init/1	0	2864	0	gen_server:loop/6
<0.54.0>	Elixir.Logger.Watcher:init/1	0	2968	0	gen_server:loop/6
<0.53.0>	Elixir.Logger.Watcher	0	7016	0	gen_server:loop/6
<0.52.0>	Elixir.Logger.Watcher:init/1	0	2968	0	gen_server:loop/6
<0.51.0>	Elixir.Logger	0	7264	0	Elixir.GenEvent:fetch_msg/5
<0.50.0>	Elixir.Logger.Supervisor	0	10880	0	gen_server:loop/6
<0.49.0>	application_master:start_it/4	0	6912	0	application_master:loop_it/4
<0.48.0>	application_master:init/4	0	2864	0	application_master:main_loop/2
<0.45.0>	Elixir.IEx.Config	0	2824	0	gen_server:loop/6
<0.44.0>	Elixir.IEx.Supervisor	0	5872	0	gen_server:loop/6
<0.43.0>	application_master:start_it/4	0	2760	0	application_master:loop_it/4

SECOND OPTIMIZATION

```
7b252f4 Remove uneeded heartbeat since cowboy handles timeouts (5 months ago, Chris McCord)

lib/phoenix/transports/websocket.ex         | 30 ++----------------------------
test/phoenix/integration/websocket_test.exs | 11 ++---------
test/phoenix/socket_test.exs                |  2 +-
3 files changed, 5 insertions(+), 38 deletions(-)
```

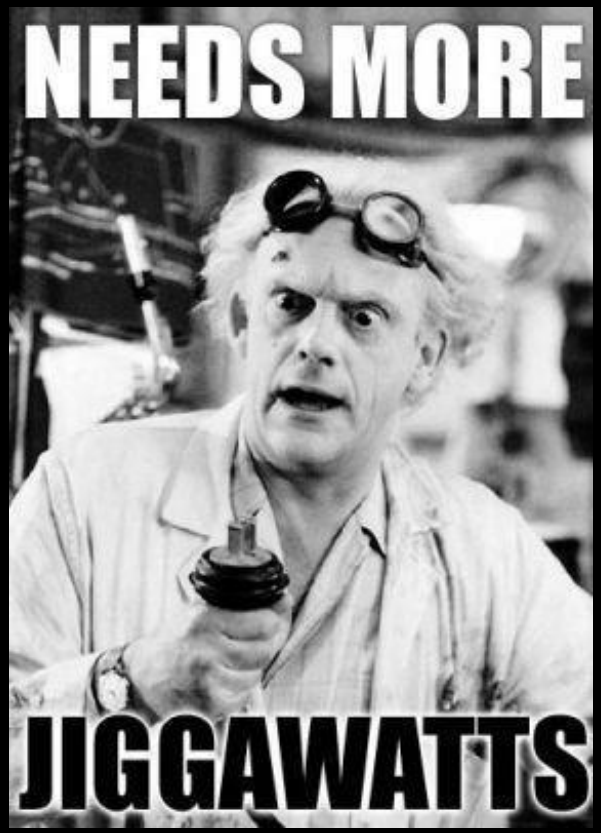
- :timer.send_after is expensive
- Remove 30s heartbeat
- Cowboy handles heartbeat

Simultaneous Users



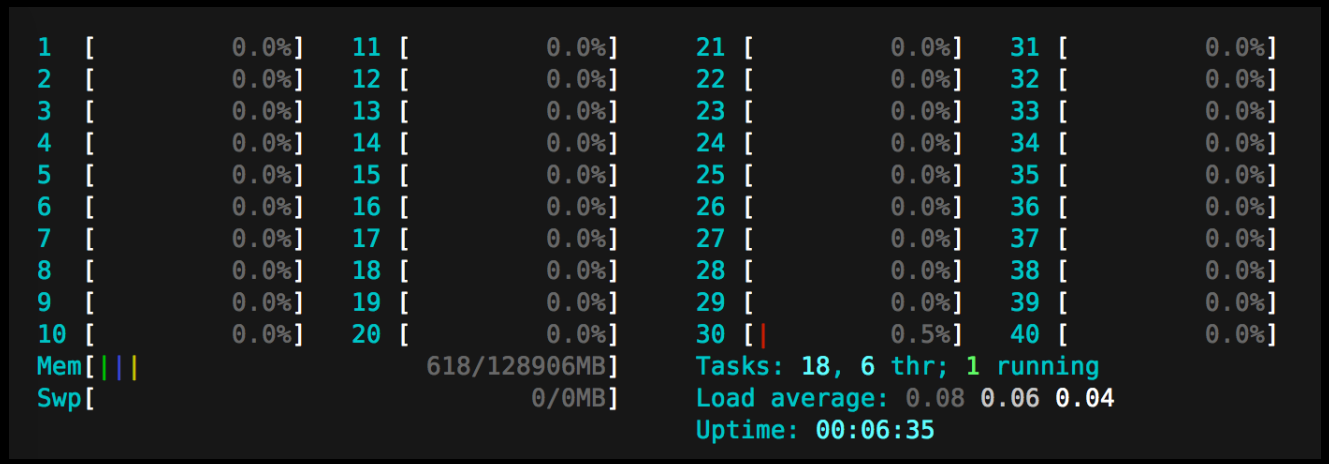
Info »

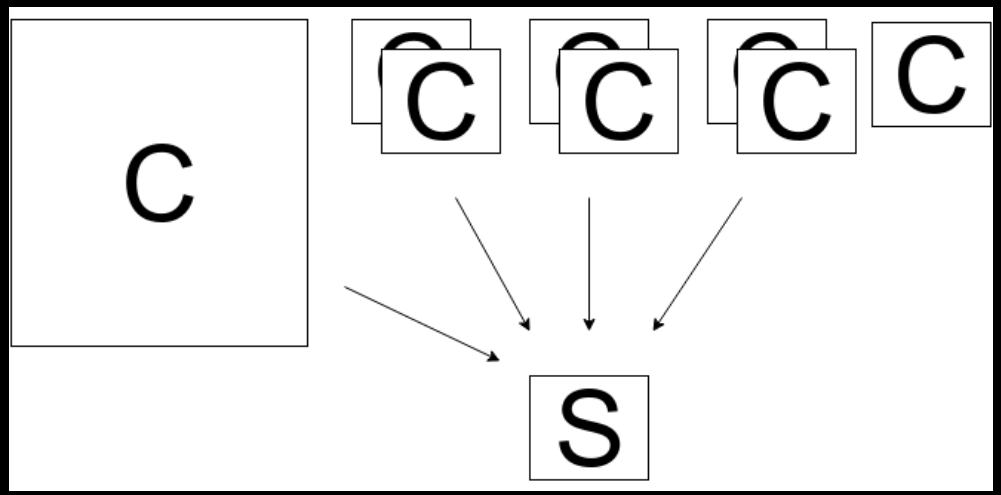
Simultaneous Users

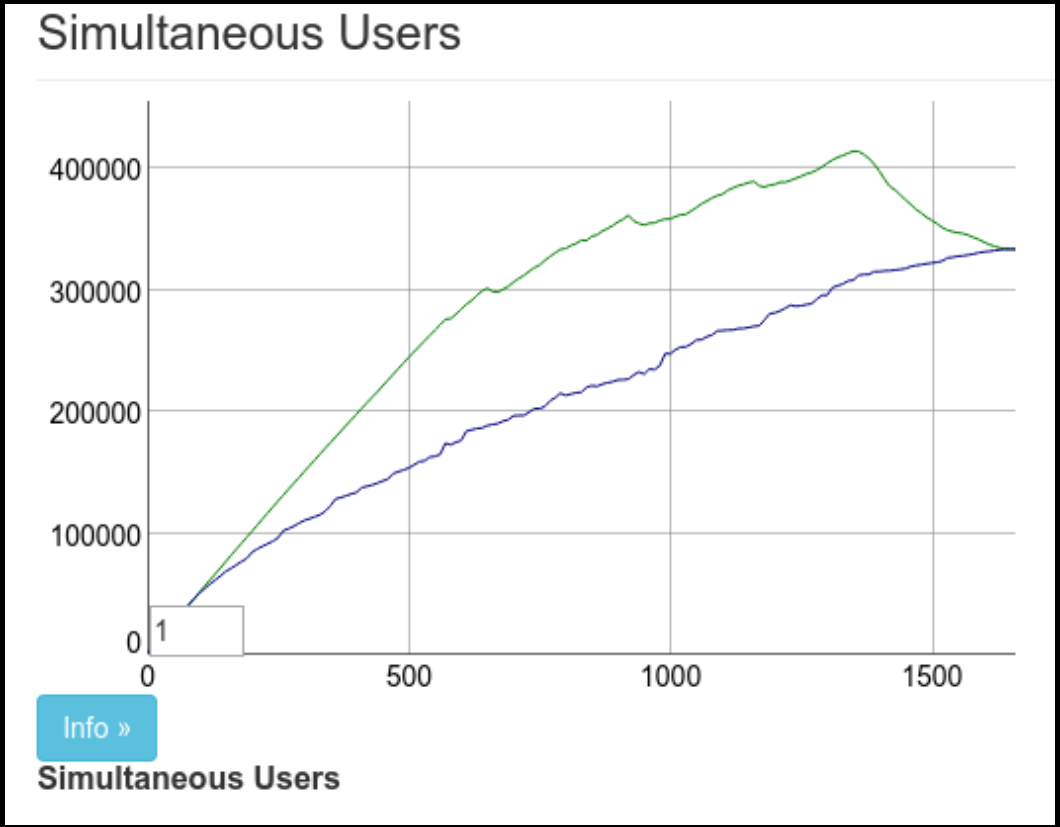

NEEDS MORE
JIGGAWATTS

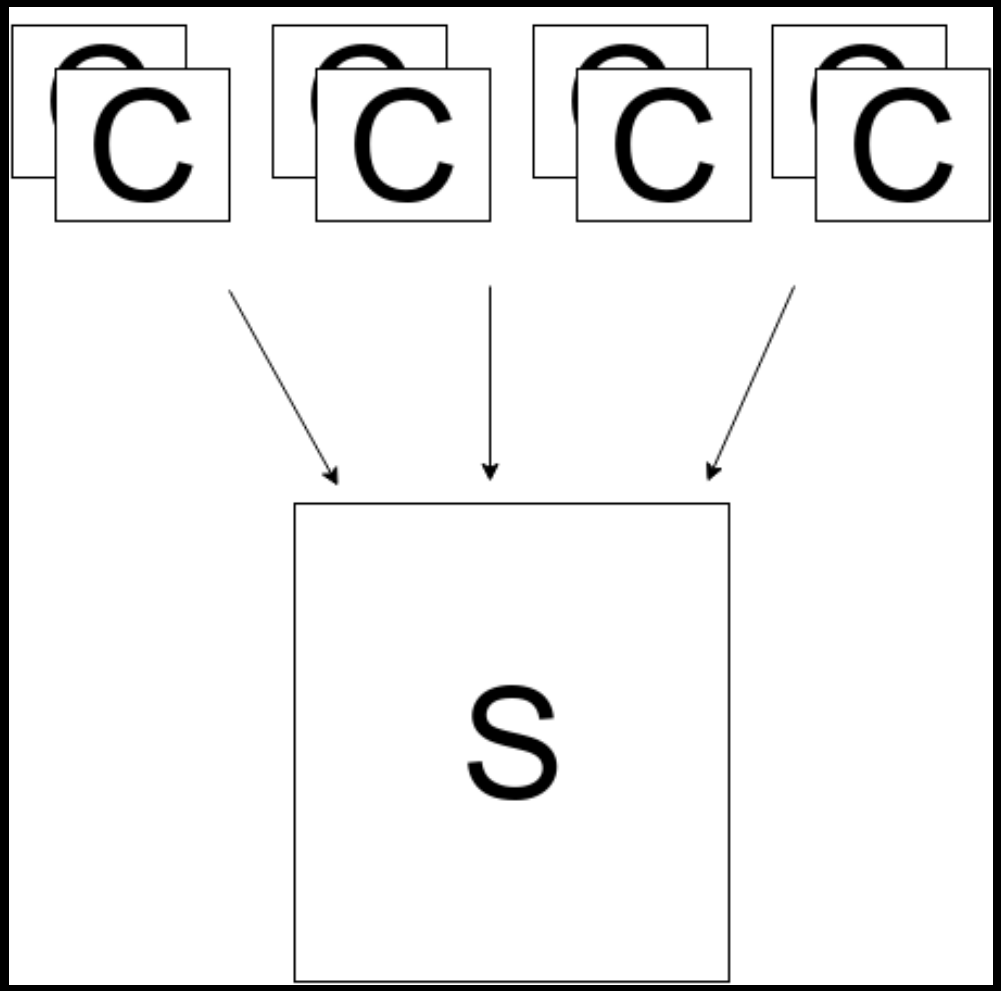
MACHINE SPECS

- 1 server Rackspace I/O v1 - 15GB RAM, 4 cores
- 2 clients as above
- 1 client - Rackspace OnMetal 128GB RAM, 40 cores
- 5 client - Rackspace general purpose 4GB RAM, 4 cores

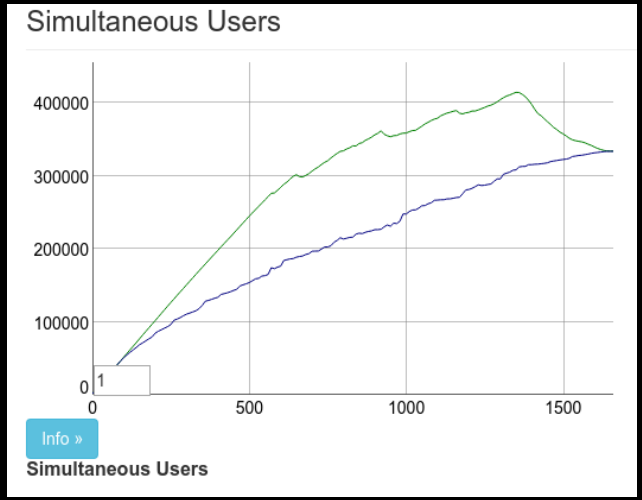

1 [0.0%] 11 [0.0%] 21 [0.0%] 31 [0.0%]
2 [0.0%] 12 [0.0%] 22 [0.0%] 32 [0.0%]
3 [0.0%] 13 [0.0%] 23 [0.0%] 33 [0.0%]
4 [0.0%] 14 [0.0%] 24 [0.0%] 34 [0.0%]
5 [0.0%] 15 [0.0%] 25 [0.0%] 35 [0.0%]
6 [0.0%] 16 [0.0%] 26 [0.0%] 36 [0.0%]
7 [0.0%] 17 [0.0%] 27 [0.0%] 37 [0.0%]
8 [0.0%] 18 [0.0%] 28 [0.0%] 38 [0.0%]
9 [0.0%] 19 [0.0%] 29 [0.0%] 39 [0.0%]
10 [0.0%] 20 [0.0%] 30 [0.5%] 40 [0.0%]
Mem[||| 618/128906MB]
Swp[0/0MB]
Tasks: 18, 6 thr; 1 running
Load average: 0.08 0.06 0.04
Uptime: 00:06:35


C
C
C
C
C
C
S

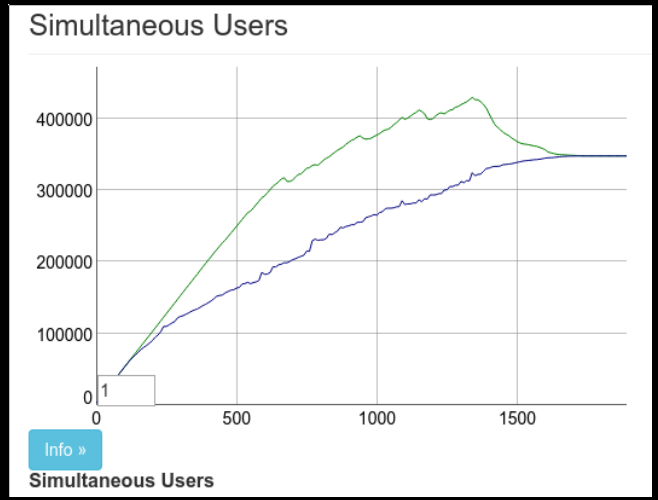

Simultaneous Users
400000
300000
200000
100000
0
1
0
500
1000
1500
Info »
Simultaneous Users


C
C
C
C
S

15GB - 4 cores

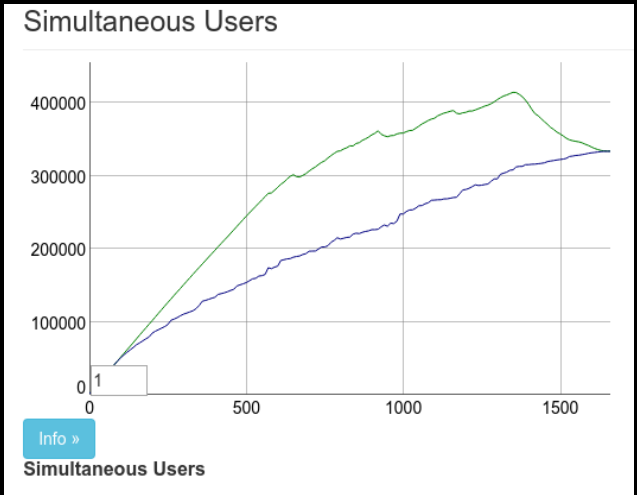


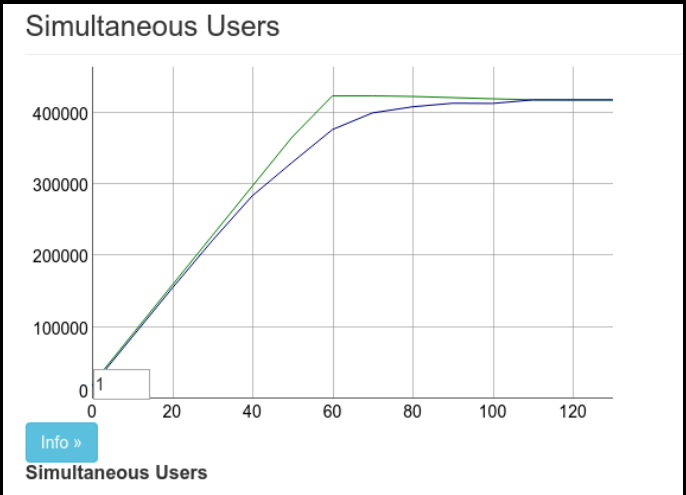
128GB - 40 cores



THIRD OPTIMIZATION

```
bad6b11 Change PG2 ETS from bag to duplicate_bag to improve insert performance
for elements with same key (5 months ago, Gabi Zuniga)

lib/phoenix/pubsub/local.ex | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```

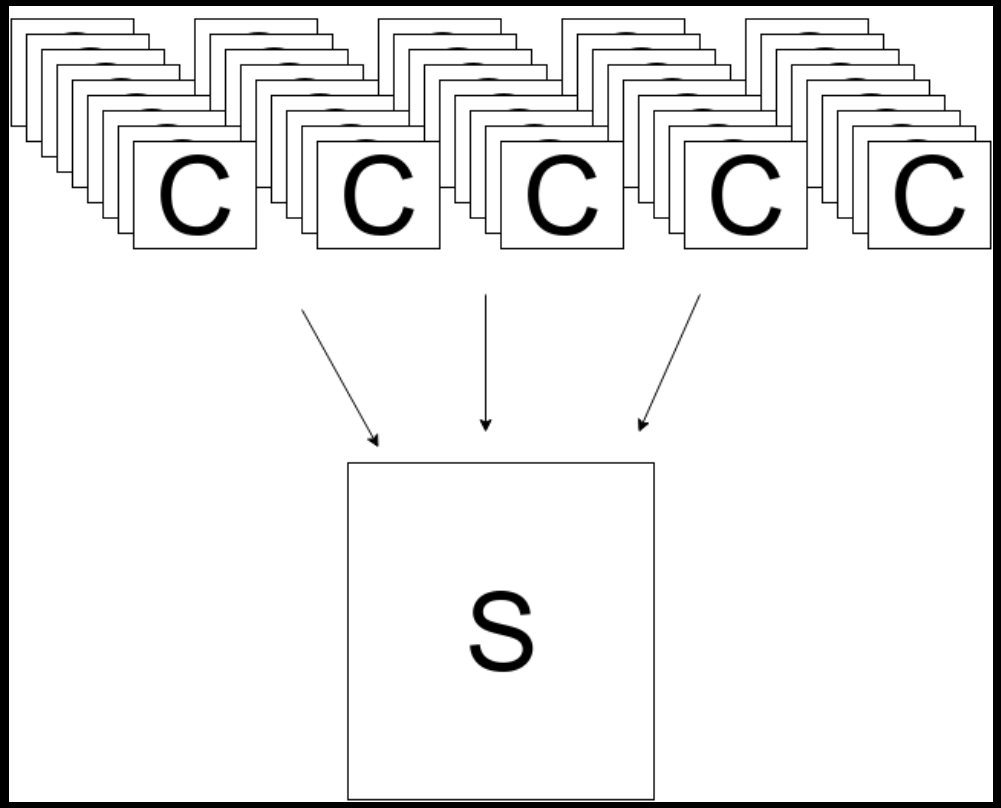

Simultaneous Users
400000
300000
200000
100000
0
1
0
500
1000
1500
Info »
Simultaneous Users

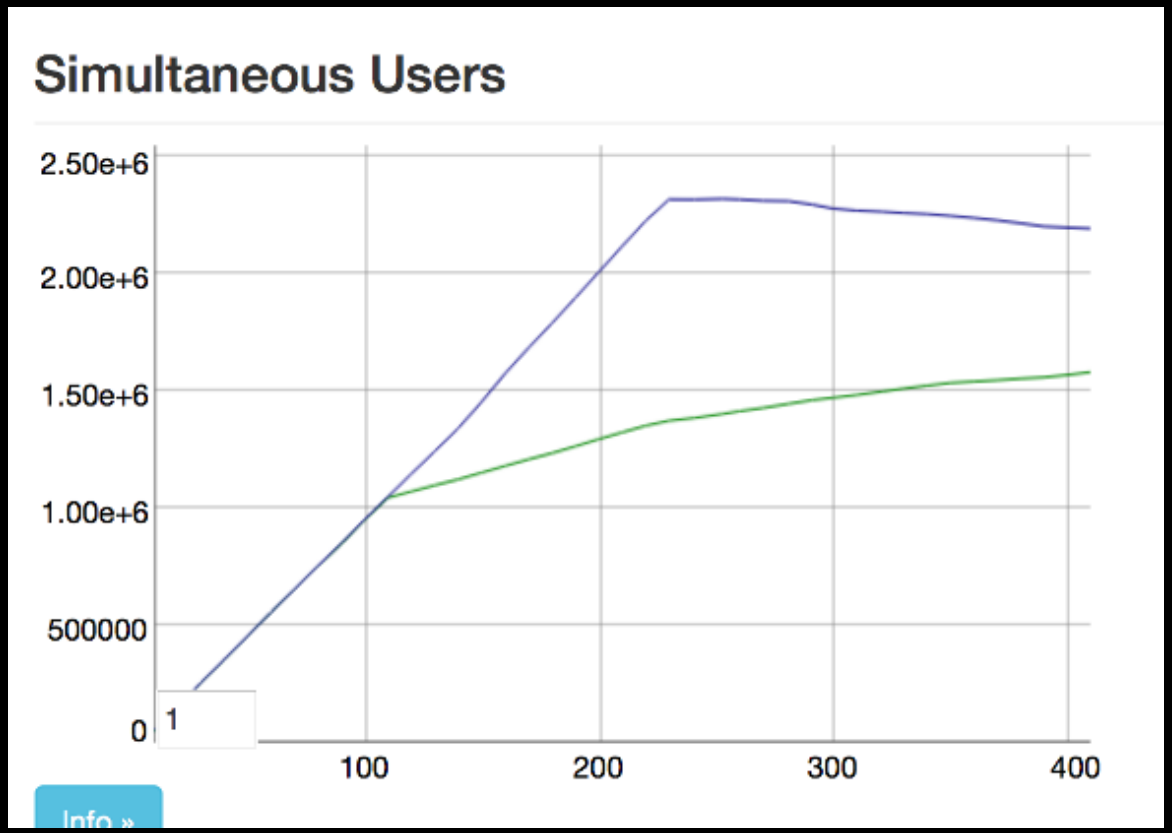

Simultaneous Users
400000
300000
200000
100000
0
1
0
20
40
60
80
100
120
Info »
Simultaneous Users

THIRD OPTIMIZATION

```
-    ^local = :ets.new(local, [:bag, :named_table, :public,
+    ^local = :ets.new(local, [:duplicate_bag, :named_table, :public,
```

- Know your ETS types!
- Duplicates don't matter since each subscriber unique
- 10x arrival rate increase


C
C
C
C
C
S


Simultaneous Users
2.50e+6
2.00e+6
1.50e+6
1.00e+6
500000
0
1
100
200
300
400
Info »

PROBLEMS

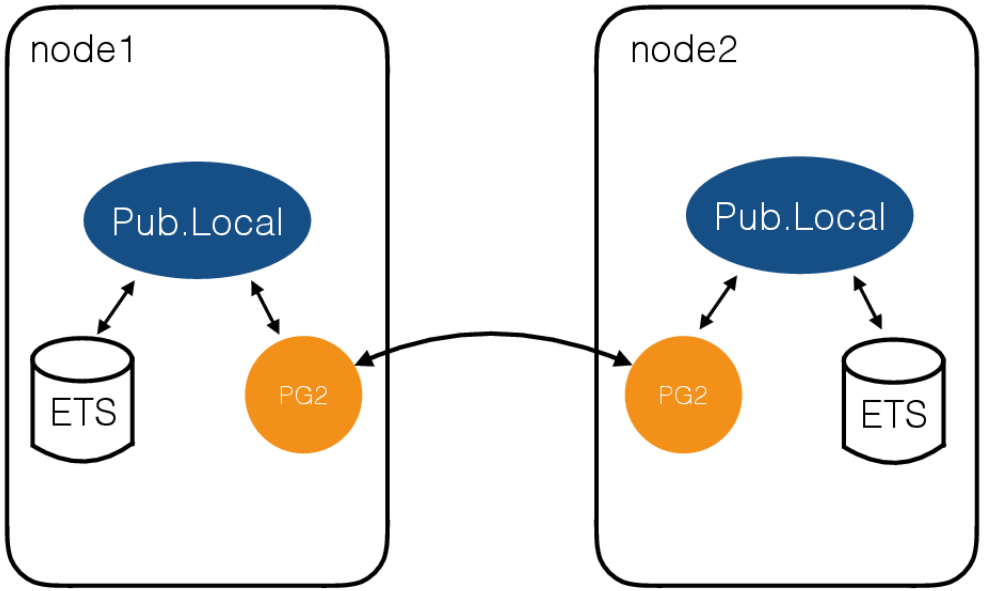
- All low hanging fruit gone (as far as we know!)
- Client crash was resulting in 60k down messages
- Server was timing out on connections
- Broadcasts taking 5 seconds to send to all users
- We tried parallelizing broadcasts but still got timeouts

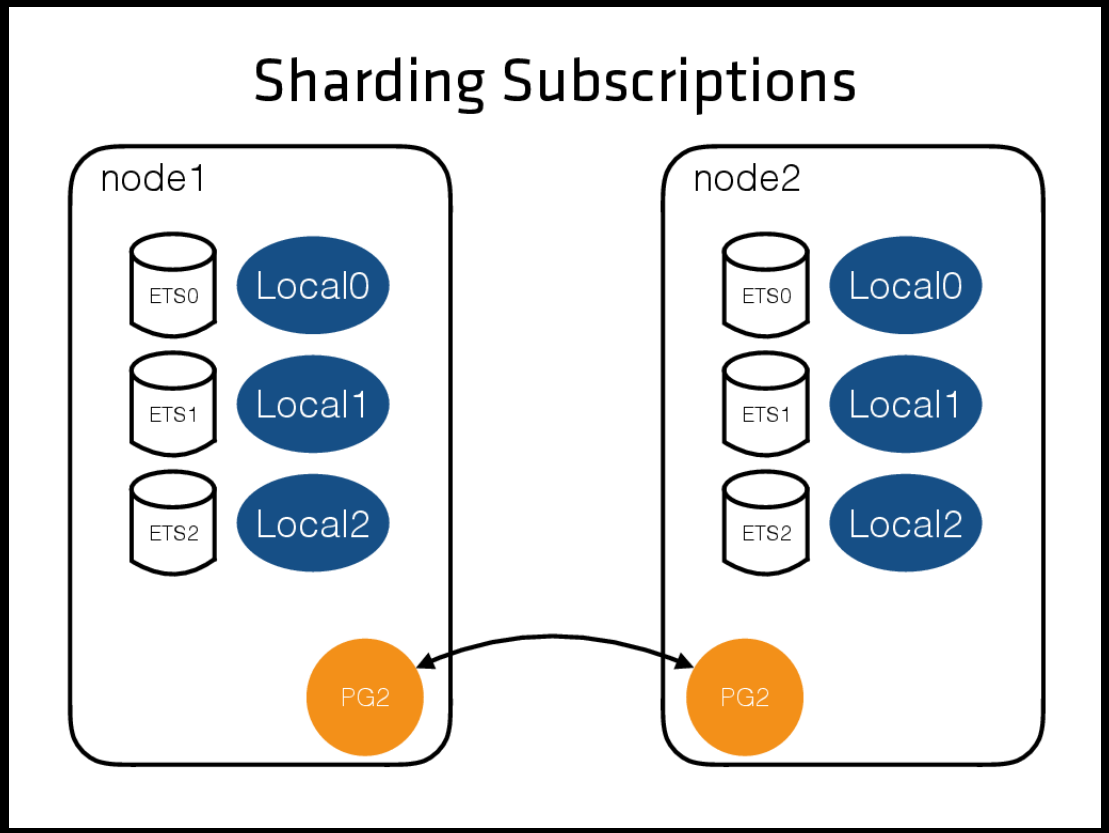
FOURTH OPTIMIZATION

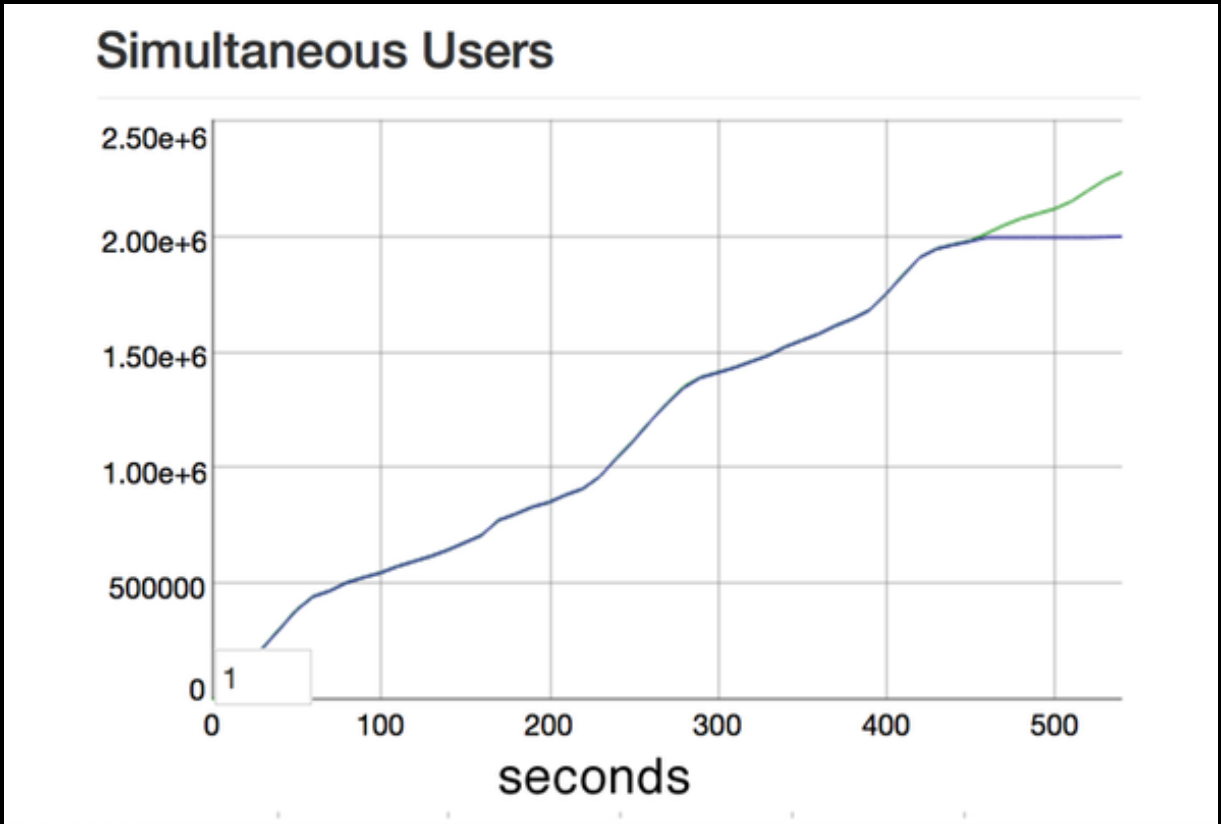
```
8eb9dfa Add random local pool and sharded subscribers (5 months ago, Chris McCord)

lib/phoenix/pubsub/local.ex      | 99 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++---------------------------------------
lib/phoenix/pubsub/pg2.ex        | 15 +++++++++++-----
lib/phoenix/pubsub/pg2_server.ex | 10 +++++-----
3 files changed, 73 insertions(+), 51 deletions(-)
```

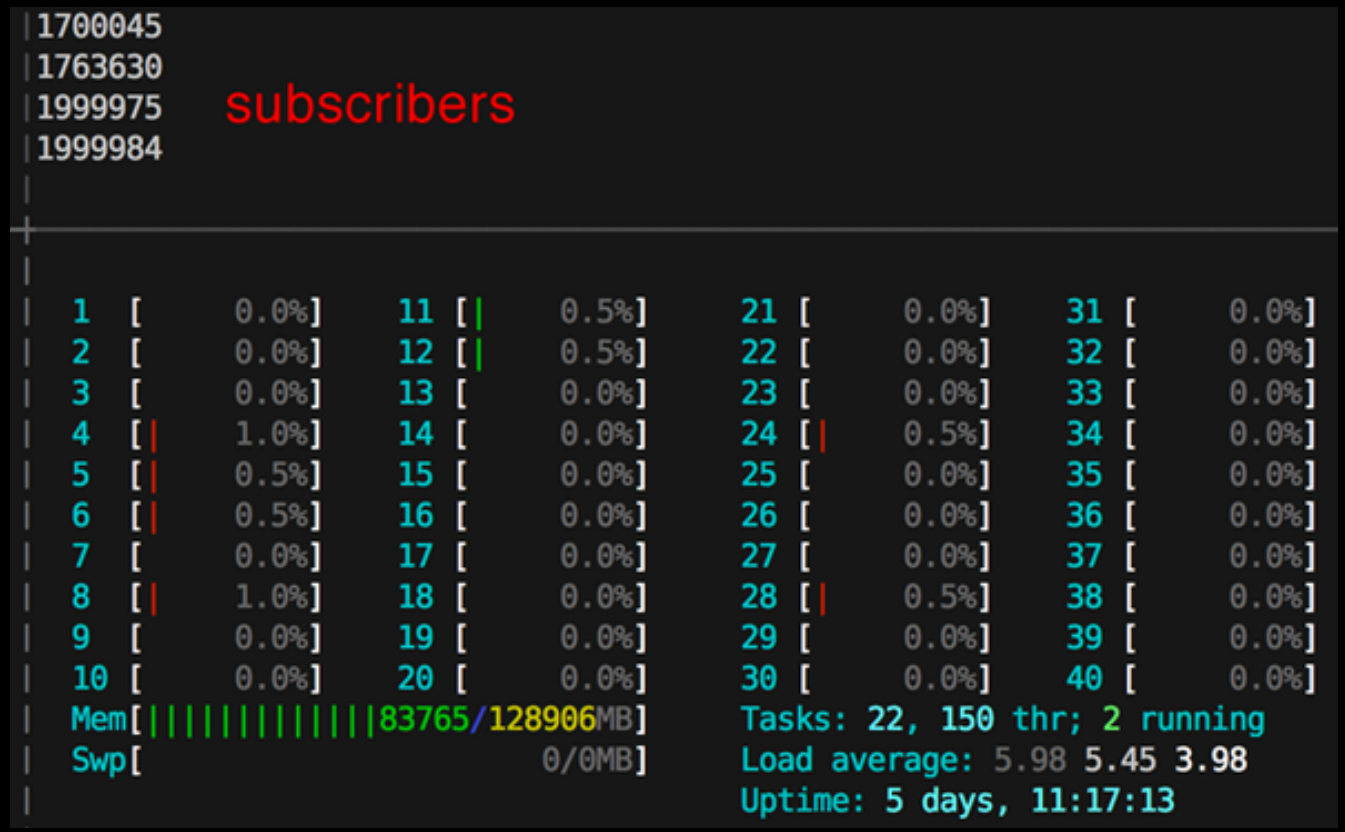
- First time we added code!
- Sharding with a pool of servers and ETS tables
- Shard based on PID
- Use :erlang.phash2(pid, shard_size)
- Configure with `pubsub: [pool_size: 40]`
- Able to maintain 1-2s broadcasts

Sharding Subscriptions




Sharding Subscriptions
node1
ETS0
Local0
ETS1
Local1
ETS2
Local2
PG2
node2
ETS0
Local0
ETS1
Local1
ETS2
Local2
PG2


Simultaneous Users
2.50e+6
2.00e+6
1.50e+6
1.00e+6
500000
0
1
0
100
200
300
400
500
seconds

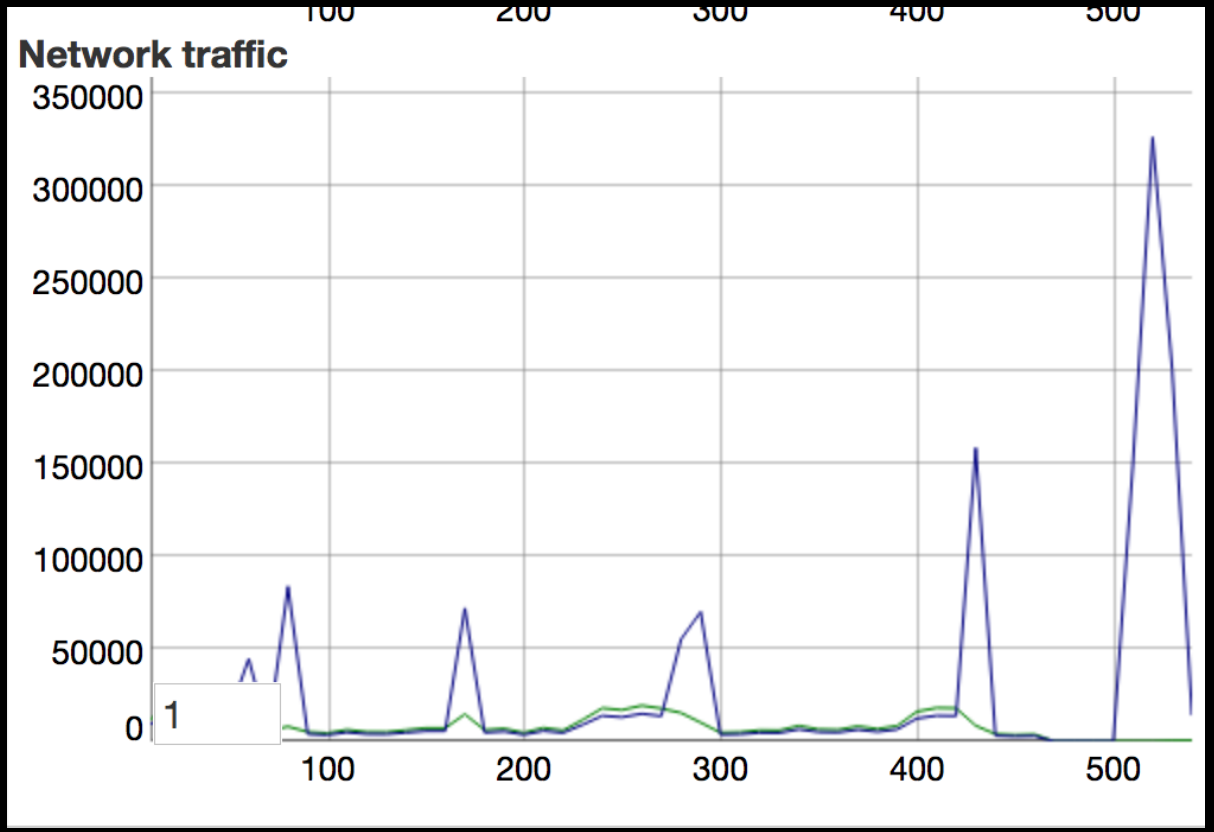

1700045
1763630
1999975
1999984
subscribers
1 [0.0%] 11 [| 0.5%] 21 [0.0%] 31 [0.0%]
2 [0.0%] 12 [| 0.5%] 22 [0.0%] 32 [0.0%]
3 [0.0%] 13 [0.0%] 23 [0.0%] 33 [0.0%]
4 [| 1.0%] 14 [0.0%] 24 [| 0.5%] 34 [0.0%]
5 [| 0.5%] 15 [0.0%] 25 [0.0%] 35 [0.0%]
6 [| 0.5%] 16 [0.0%] 26 [0.0%] 36 [0.0%]
7 [0.0%] 17 [0.0%] 27 [0.0%] 37 [0.0%]
8 [| 1.0%] 18 [0.0%] 28 [| 0.5%] 38 [0.0%]
9 [0.0%] 19 [0.0%] 29 [0.0%] 39 [0.0%]
10 [0.0%] 20 [0.0%] 30 [0.0%] 40 [0.0%]
Mem[||||||||||||83765/128906MB]
Swp[0/0MB]
Tasks: 22, 150 thr; 2 running
Load average: 5.98 5.45 3.98
Uptime: 5 days, 11:17:13

HOW TO OPTIMIZE

- Be José Valim
- Use the tools available to you (observer)
- Isolate the bottlenecks
- Know your data types (ETS)
- Use a pool if a process is bottlenecked

SO DO THE CONNECTIONS DO ANYTHING?

- We have a chat room with 2 million people


100
200
300
400
500
Network traffic
350000
300000
250000
200000
150000
100000
50000
0
1
100
200
300
400
500

Erlang (programming language)

From Wikipedia, the free encyclopedia

This article is about the programming language. For other things named "Erlang", see Erlang (disambiguation).

Erlang (/ˈɜːrlæŋ/ *ER-lang*) is a general-purpose, concurrent, garbage-collected programming language and runtime system. The sequential subset of Erlang is almost a functional language (excluding certain built-in functions), with eager evaluation, single assignment, and dynamic typing. It was originally designed by Ericsson to support distributed, fault-tolerant, soft real-time, highly available, non-stop applications. It supports hot swapping, thus code can be changed without stopping a system.[3]

While threads require external library support in most languages, Erlang provides language-level features for creating and managing processes with the aim of simplifying concurrent programming. Though all concurrency is explicit in Erlang, processes communicate using message passing instead of shared variables, which removes the need for explicit locks (a locking scheme is still used internally by the VM[4]).

The first version was developed by Joe Armstrong, Robert Virding and Mike Williams in 1986.[5] It was originally a proprietary language within Ericsson, but was released as open source in 1998. Erlang, along with OTP, a collection of middleware and libraries in Erlang, are now supported and maintained by the OTP product unit at Ericsson and widely referred to as Erlang/OTP.

Contents [hide]

Erlang

Paradigm	multi-paradigm: concurrent, functional
Designed by	Joe Armstrong, Robert Virding and Mike Williams
Developer	Ericsson
First appeared	1986; 30 years ago
Stable release	18.3[1] / March 16, 2016; 15 days ago
Typing discipline	dynamic, strong
License	Apache License 2.0 (since OTP 18.0) Erlang Public License 1.1 (earlier releases)
Filename extensions	.erl .hrl
Website	www.erlang.org
Major implementations	
Erlang	
Influenced by	
Prolog, Smalltalk, PLEX[2]	

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons

Languages
العربية
Български

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons

Languages
العربية

Elixir

From Wikipedia, the free encyclopedia

For other uses, see Elixir (disambiguation).

This article **does not cite any sources**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(December 2011)*

An **elixir** (from Arabic: *الإكسير - al-'iksīr*) is a clear, sweet-flavored liquid used for medicinal purposes, to be taken orally and intended to cure one's ills. When used as a pharmaceutical preparation, an elixir contains at least one active ingredient designed to be taken orally.

Contents [hide]

Types [edit]

Non-medicated elixirs [edit]

They are used as solvents or vehicles for the preparation of medicated elixirs: aromatic elixirs (USP), isoalcoholic elixirs (NF), or compound benzaldehyde elixirs (NF). Active ingredient dissolved in a solution that contains 15 to 50% by volume of ethyl alcohol.

Medicated elixirs [edit]

- Antihistaminic elixirs: used against allergy: chlorambhepiramine maleate elixirs (USP), diphenhydramine HCl elixirs

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons
Wikiquote

Languages

Troll

From Wikipedia, the free encyclopedia

This article is about beings from Scandinavian folklore and mythology. For the internet term, see Internet troll. For other uses, see Troll (disambiguation).

A **troll** is a supernatural being in Norse mythology and Scandinavian folklore. In origin, *troll* may have been a negative synonym for a *jötunn* (plural *jötnar*). In Old Norse sources, beings described as trolls dwell in isolated rocks, mountains, or caves, live together in small family units, and are rarely helpful to human beings.

Later, in Scandinavian folklore, trolls became beings in their own right, where they live far from human habitation, are not Christianized, and are considered dangerous to human beings. Depending on the region from which accounts of trolls stem, their appearance varies greatly; trolls may be ugly and slow-witted, or look and behave exactly like human beings, with no particularly grotesque characteristic about them.

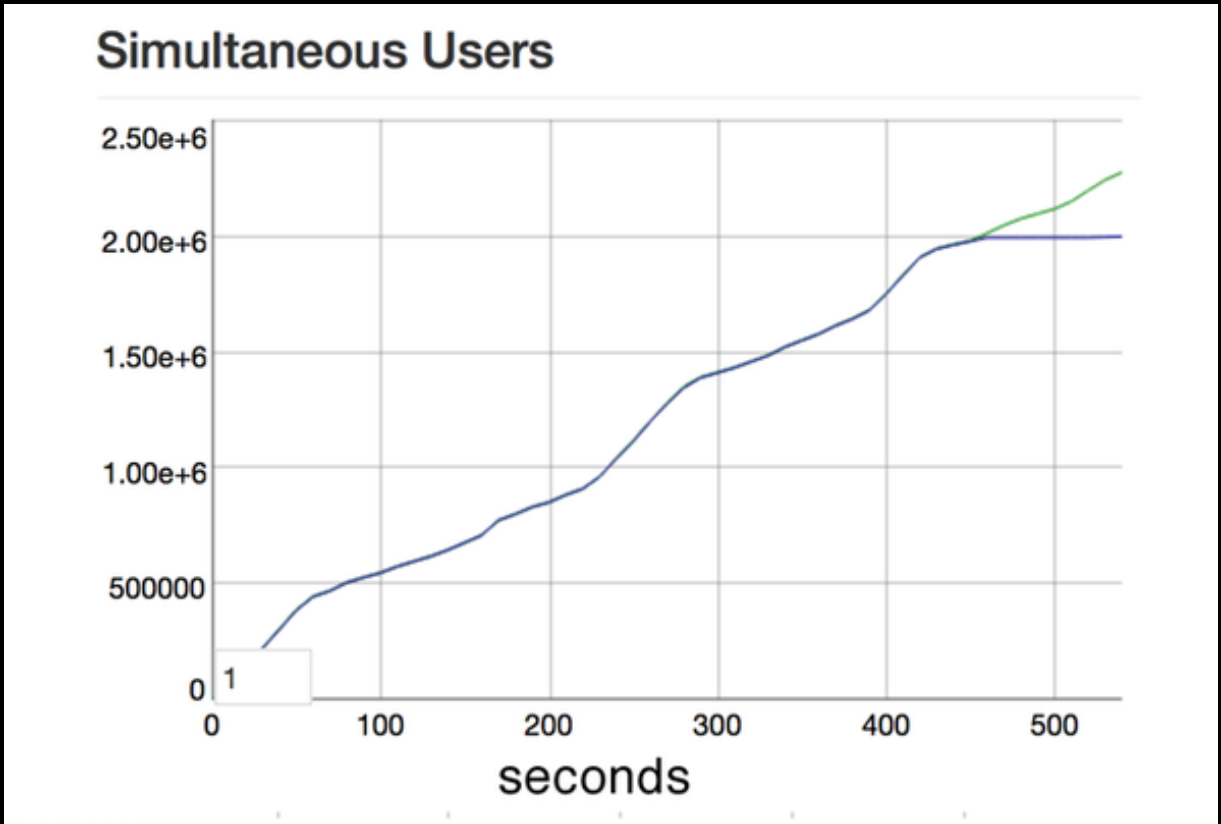
Trolls are sometimes associated with particular landmarks, which at times may be explained as formed from a troll exposed to sunlight. Trolls are depicted in a variety of media in modern popular culture.

Look at them, troll mother said. Look at my sons! You won't find more beautiful trolls on this side of the moon. (1915) by John Bauer

Contents [hide]

Norse mythology

In Norse mythology, *troll*, like *thurs*, is a term applied to *jötnar*, and are mentioned throughout the Old Norse corpus. In


Simultaneous Users
2.50e+6
2.00e+6
1.50e+6
1.00e+6
500000
0
1
0
100
200
300
400
500
seconds

TSUNG COMMON ISSUES

- Everything requires a host name (clients and controller)
- If /etc/hosts names don't match then you will get an error
- Needs to be correct for every client
- SSH keys need to be set up from the controller to each client
- ulimit needs to be set

PHOENIX CONFIGURATION

- Use production environment
- Disable logging

```
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1569"}
16:35:15.070 [info] Replied rooms:lobby :ok
16:35:15.071 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1571"}
16:35:15.071 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1570"}
16:35:15.071 [info] Replied rooms:lobby :ok
16:35:15.071 [info] Replied rooms:lobby :ok
16:35:15.072 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1572"}
16:35:15.072 [info] Replied rooms:lobby :ok
16:35:15.074 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1573"}
16:35:15.074 [info] Replied rooms:lobby :ok
16:35:15.076 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1575"}
16:35:15.076 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1574"}
16:35:15.076 [info] Replied rooms:lobby :ok
16:35:15.076 [info] Replied rooms:lobby :ok
16:35:15.076 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1576"}
16:35:15.076 [info] Replied rooms:lobby :ok
16:35:15.078 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1577"}
16:35:15.079 [info] Replied rooms:lobby :ok
16:35:15.079 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1578"}
16:35:15.079 [info] Replied rooms:lobby :ok
16:35:15.079 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1579"}
16:35:15.080 [info] Replied rooms:lobby :ok
16:35:15.081 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1580"}
16:35:15.082 [info] Replied rooms:lobby :ok
16:35:15.082 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1581"}
16:35:15.082 [info] Replied rooms:lobby :ok
16:35:15.083 [info] JOIN rooms:lobby to Chat.RoomChannel
  Transport:  Phoenix.Transports.WebSocket
  Parameters: %{"user" => "1582"}
```

EASIEST WAY TO DO THIS AT SCALE

- Create a server, install erlang, tsung, etc.
- Create SSH key and add it to ~.ssh/.authorized_keys
- Set up own hostname (tsung-controller or something)
- Set ulimit in `/etc/security/limits`
- Create an image
- Spawn new servers from that image

WHAT NEXT?

- Benchmarking is expensive!
- multi-node
- Chat room with messages being sent periodically
- Benchmarking more "chat rooms" on a single server
- Use IP aliasing to make testing require fewer machines
- Automate the tests for each release (both websocket and HTTP)

THANKS FOR LISTENING