Hi, I’m Nathan

I go by @myobie online

on almost every network

There will be Q&A

I design distributed applications from UI to data storage

I built this presentation using choo and phoenix

choo.io

This talk is interactive

If you have cellular data then maybe let others use the wifi

elixir.myobie.com

You can rate how you feel about this talk anytime using the slider

The average rating will frequently update and will appear on top of the slides

You can also submit an emoji that will show up in the bottom right corner

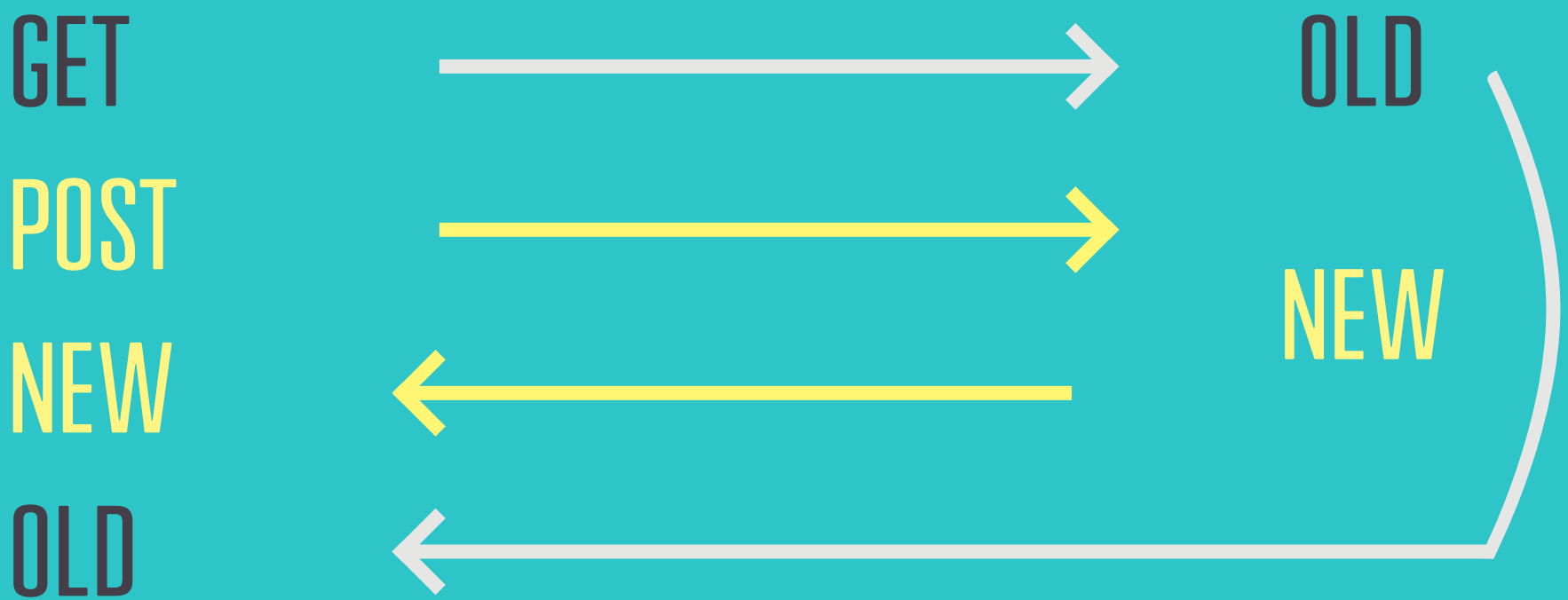
Have fun

Distributed applications are hard

Network partitions
Concurrency
Locking (Linearization)
Stale data

by Ted Parker
http://ted-parker.com/portfolio/surfs-dead/

Network latency can cause stale reads from consistent systems


GET
POST
NEW
OLD
OLD
NEW

Happens
all the time

Scaling is hard

Load balancing
Parrallelism
Databases
AUTOVACUUM & VACUUM

Managing databases is still hard

Why build a distributed system in the first place?

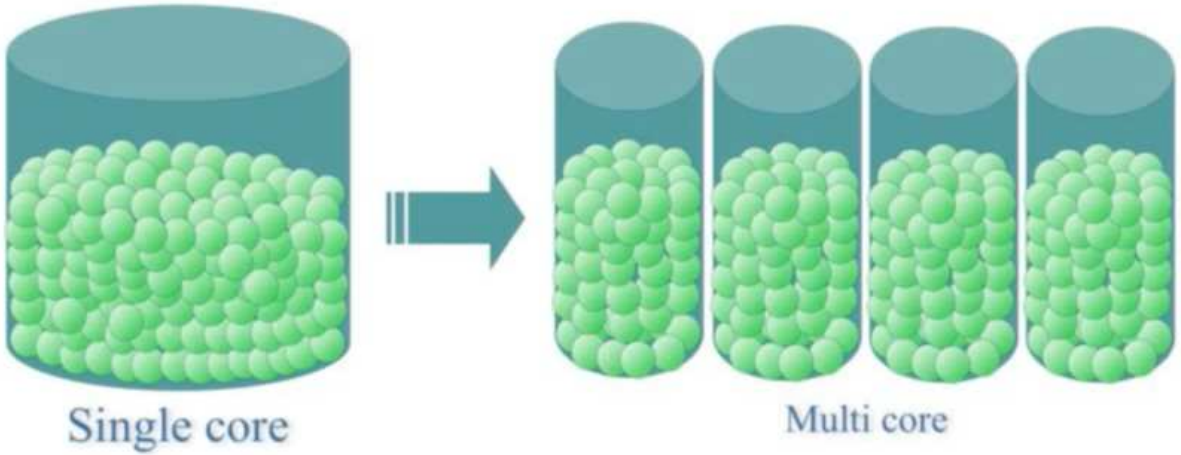
Horizontal scaling
Support “billions” of users
Offline
Bad networks

I watched a talk by
Joe Armstrong recently

https://youtu.be/bo5WL5IQAd0

But packing sand into boxes is easy

Source
Erlang factory 2010
San Francisco
Patrik Nyblom



granular = scalable

Crazy Idea

Having a central database makes some things easier

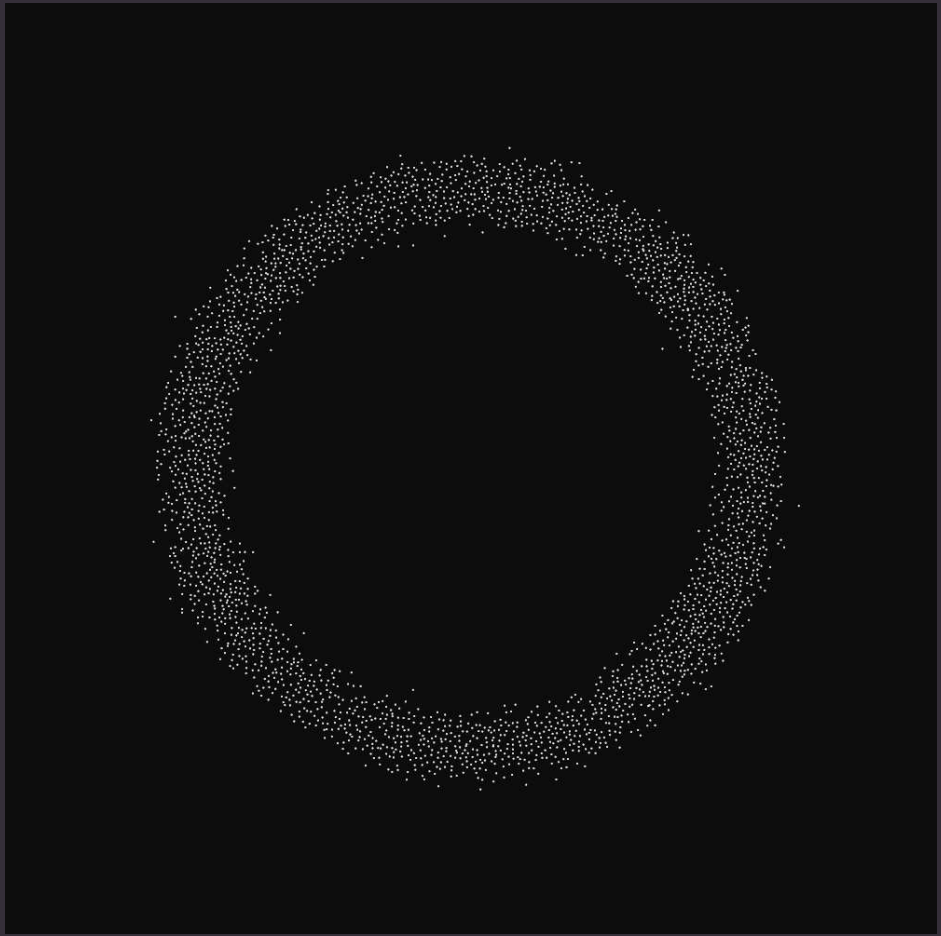
However, a single database is hard to scale

What if every user had their own database? Stored on S3 as a file?!

by Charis Tsevis
https://flic.kr/p/8hUVRV

Time

Human time cannot be trusted in a distributed system



by Anders Hoff
https://twitter.com/inconvergent

Any working distributed system is an implementation of how that organization has understood time

Actors

by Nagano Toyokazu
https://www.flickr.com/photos/toyokazu

At Wunderlist
we used actors to wrap
each websocket
connection

A great way to contain state and linearize access

Wait a second?!

Elixir processes

What if every user had their own process?!

To trust a process as the authority for a user ...

We need to know there will be only one per user

If we were to accidentally have two "myobie" processes, then we have a split brain

:global

Not consistent

Are you sure?

If any name clashes are discovered, function Resolve is called.

http://erlang.org/doc/man/global.html

:pg2

Not consistent

Are you sure?

pg2 replicates all name lookup information in a way that doesn't require consistency …

http://erlang.org/pipermail/erlang-questions/2012-June/067220.html

:gproc

Same

Are you sure?

> *While gproc has been tested very thoroughly … its reliance on gen_leader is problematic.*

https://christophermeiklejohn.com/erlang/2013/06/05/erlang-gproc-failure-semantics.html

Does gen_leader have problems?

I don't know.

Elixir.Registry

Single vm

Consistent

It's "easy" to be consistent in one process on one vm

If that one vm stops ...

... then our entire registry is gone 💨

How do we make 100% certain we never accidentally boot two vms?

Zookeeper

Consistent
Pretty available
Partition tolerant

Also: includes recipes

https://zookeeper.apache.org/doc/current/recipes.html

Recipe: distributed locks

Major 🔑

by Meghan Roberts
https://www.flickr.com/photos/88009602@N05/8696887207

Battle-tested system for getting time under control

We can know if one thing happens before another

Highlander

github.com/myobie/highlander

There can be only one

90's TV show reference

“Objects” persist their state to S3 during update

“Objects” read from S3 during init

“Objects” respond to calls using their in-memory state like GenServers

“Objects” teardown after inactivity

Process registry stored in zookeeper

```
GenServer :via
```

```
GenServer.call(
  {:via, MyRegistry, {:user, user_id}},
  :do_stuff
)
```

```ruby
defmodule MyRegistry do
  def send(name, message) do
  end

  def whereis_name(name) do
  end

  def register_name(name) do
  end

  def unregister_name(name) do
  end
end
```

Zookeeper is a simple key value store. The key will be the object's id and the value the node name.

```elixir
defmodule MyRegistry do
  def whereis_name(name) do
    case ZK.get_node_name(name) do
      # ...
    end
  end
end
```

```elixir
defmodule MyRegistry do
  def register_name(name) do
    case ZK.create_znode(name) do
      # ...
    end
  end
end
```

```elixir
defmodule ZK do
  def create_znode(name) do
    {:ok, pid} = ZNode.start_link(name)

    if ZNode.first?(pid) do
      {:ok, pid}
    else
      :ok = ZNode.delete(pid)
      {:error, :already_exists}
    end
  end
end
```

```elixir
defmodule ZNode do
  def init(name) do
    {:ok, path} =
      Zookeeper.Client.create(:zk,
        prefix(name, UUID.uuid4(:hex)),
        to_string(Node.self),
        makepath: true,
        create_mode: :ephemeral_sequential)
    {:ok, %{path: path, name: name}}
  end
end
```

```
defmodule ZNode do
  def init(name) do
    {:ok, path} =
      Zookeeper.Client.create(:zk,
        prefix(name, UUID.uuid4(:hex)),
        to_string(Node.self),
        makepath: true,
        create_mode: :ephemeral_sequential)
    {:ok, %{path: path, name: name}}
  end
end
```

Zookeeper will keep time in order

```
opts = [host,
  [stop_on_disconnect: true, name: :zk]]

children = [
  worker(Zookeeper.Client, opts, []),
  worker(Registry.Server, [], []),
  worker(Registry.NodeCycleServer, [], []),
  supervisor(Object.Supervisor, [], [])
]

supervise children, strategy: :rest_for_one
```

OTP will keep the processes running

Macros

```
defmodule Todo do
  use Highlander.Object, type: :todo

  defobject title: "",
    completed: false,
    color: :blue
end
```

```
id = UUID.uuid4()

{:ok, todo} = Todo.get id
```

```
assert todo.completed == false
assert todo.title == ""
assert todo.color == :blue
```

```
todo = %{todo | title: "Hello"}

{:ok, _} = Todo.put id, todo
```

```
# any amount of time later

{:ok, todo} = Todo.get id

assert todo.title == "Hello"
```

Model everything as a process

```elixir
defmodule User do
  use Highlander.Object, type: :user

  defobject name: "",
    email: ""
end
```

What about shared objects?

```
defmodule List do
  use Highlander.Object, type: :list

  defobject title: "",
    todos: []
end

defmodule Todo do
  defstruct title: "",
    completed: false,
    color: :blue
end
```

Like an app where you can share a todo list with other people...

User ⟷ List ⟷ User

```
defmodule User do
  use Highlander.Object, type: :user

  defobject name: "",
    email: "",
    known_lists: []
end
```

```
defmodule List do
  use Highlander.Object, type: :list

  defobject title: "",
    todos: []
    allowed_users: []
end
```

```
defmodule List do
  def handle_update(:insert_todo, list,
                    todo, user) do
    if Enum.any?(list.allowed_users,
      fn id -> id == user.id end) do
        %{todos: [todo | state.todos]}
    end
  end
end
```

One list’s activity doesn’t slow down another

Aside: multi-vm testing

I copied how phoenix does it

```elixir
def spawn do
  :net_kernel.start([:"primary@127.0.0.1"])
  :erl_boot_server.start([])
  allow_boot to_char_list("127.0.0.1")

  Application.get_env(:highlander,
    :spawn_nodes, [])
  |> Enum.map(&Task.async(fn ->
    {:ok, _node} = spawn_node(&1)
  end))
  |> Enum.map(&Task.await(&1, 30_000))
end
```

```
defp spawn_node(node_host) do
  {:ok, node} = :slave.start(
    to_char_list("127.0.0.1"),
    node_name(node_host),
    inet_loader_args())

  add_code_paths(node)
  transfer_configuration(node)
  ensure_applications_started(node)
  {:ok, node}
end
```

test/support/cluster.ex

```
$ epmd -daemon
```

Scaling

Like pouring sand...

How many locks can a small zookeeper cluster handle?

I don’t know.

Sharded zk

Sharded registry keyspaces

Halo

halowaypoint.com

HALO GAMES NEWS ESPORTS COMMUNITY FORUMS UNIVERSE SHOP Sign In

WELCOME COLONY

Unleash the power of Colony, a new Lekgolo leader in Halo Wars 2! This enigmatic alien force commands a deadly and dangerous host of powerful new Hunter forms in concert with Banished forces on the Ark.

READ MORE BUY NOW

HALO LEGENDS NOW AVAILABLE ON NETFLIX

HALO CRATE #6 FIGURE REVEALED

HCS COMING TO DAYTONA BEACH

Orleans

https://dotnet.github.io/orleans/Documentation/Introduction.html

MSFT Service Fabric

Uses zk to setup partitions

Partitions actors and services based on the zk registry

Akka

Sure.

riak_core

Sure.

🤷🌳🤷‍♀️

Have fun.
Have crazy ideas.
Build things.

Please ask any question, even if you think it's dumb or novice or complicated or ...

Don’t make a long statement that isn’t a question.

Q&A