Deploying  at scale

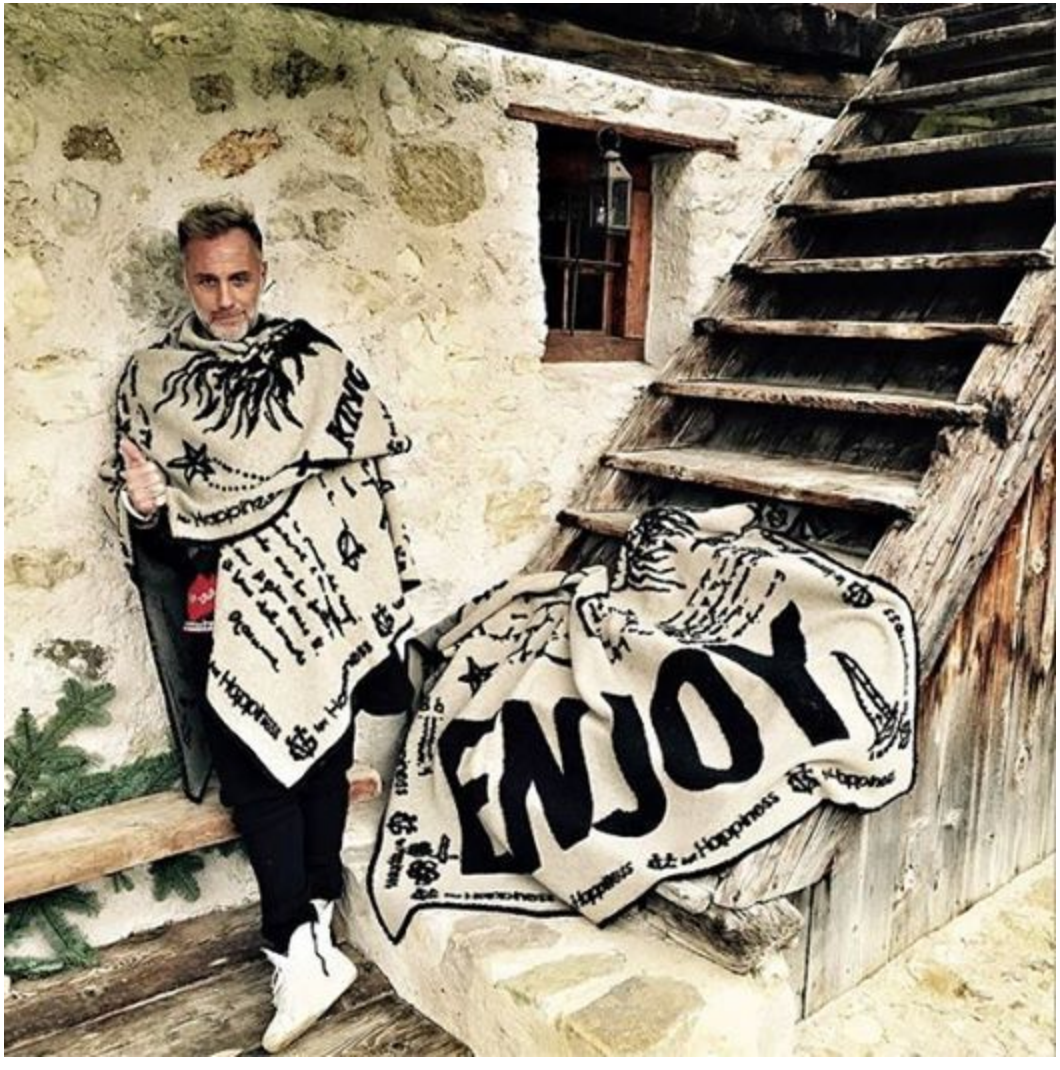
A developer journey

@lucamaraschi

With the “soft hand” touch of @matteocollina

DISCLAIMER

Nobody in this story, and no outfit or corporation, is based upon an actual person or outfit in the real world. The tale we are going to tell is completely fictional, but based upon the combined experiences of the speakers in building Node.js applications.


ENJOY

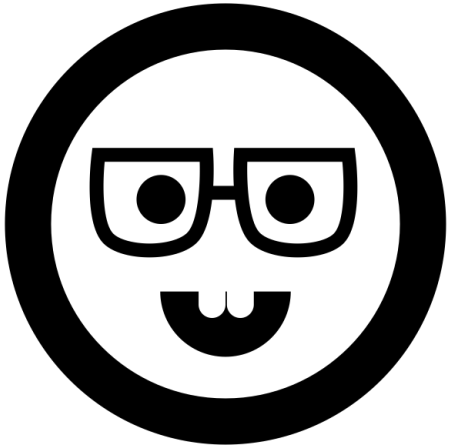
Created by Julia Holmberg
from Noun Project

Meet bob

He is a **.LAVA** developer

He writes applications on the **L2EE** stack, using **SQLobster** as a database.

Created by Rutmer Zijlstra
from Noun Project



Created by Julia Holmberg
from Noun Project

Bob reads:

nearForm

ABOUT US SERVICES OPEN SOURCE EVENTS NODECRUNCH CAREERS CONTACT

NodeCrunch

Welcome to NodeCrunch, the nearForm blog. This is where we share information of all kinds, from Node.js news to new tool releases to our latest thinking on developing company values. We love to chat, so do post your comments below.

WHY NODE.JS IS BECOMING THE GO-TO TECHNOLOGY IN THE ENTERPRISE

By: Cian Ó Maidín

March 10, 2014 Feature articles, Node.js community Tagged Mail Online, node.js growth, PayPal, Voxer, WallMart, Yammer 65 Comments

RECENT POSTS

Node v6.0.0 released today (April 26th)

Announcing microservices day London 2016

Announcing NodeConf London 2016


Such speed
So enterprise
Wow JS

Created by Julia Holmberg
from Noun Project

Bob decides:

Let's rewrite my HUGE application in node.js

Using the MEAN stack

mongoDB®

Express

amazon web services

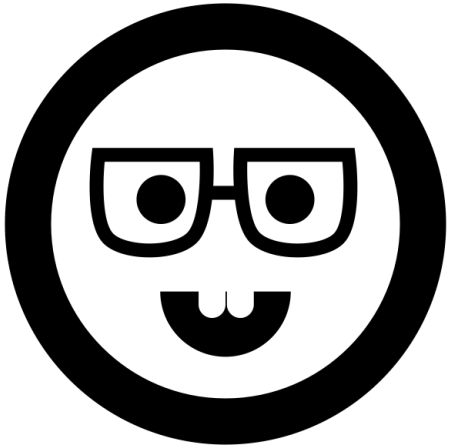
ANGULARJS by Google

mongoose

node JS®


.js

How do I deploy my Javascript monolith?



Created by Julia Holmberg
from Noun Project

I read StackOverflow!

SSH into the virtual machine

- snowflake servers
- git clone
- Forever or PM2!

No Integration tests, but 100% code coverage

The new product has huge success

Bob is happy

Created by Julia Holmberg
from Noun Project

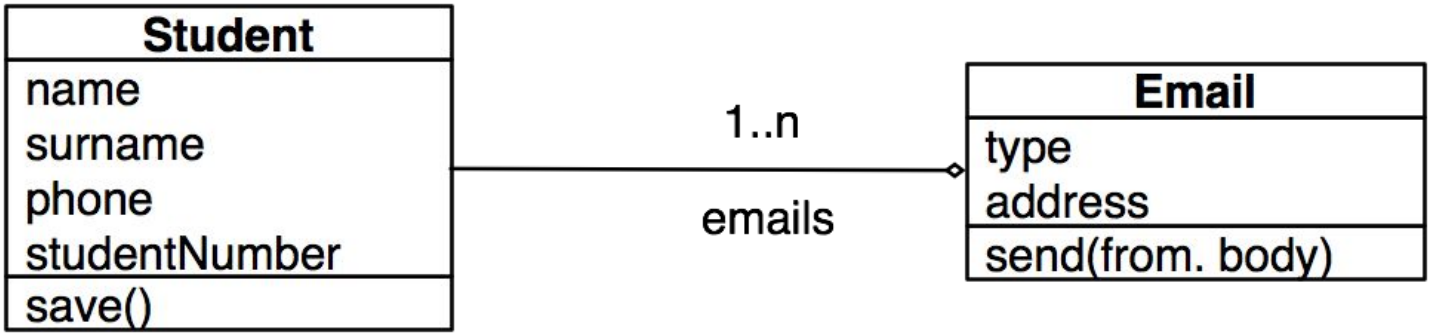
And the performance fall

Bob is SAAD

Created by Julia Holmberg
from Noun Project


Such speed
So enterprise
Wow JS

But my schema is relational...



One MongoDB collection for Student

One MongoDB collection for Email

mongoose

populate

Population

There are no joins in MongoDB but sometimes we still want references to documents in other collections. This is where population comes in.

Population is the process of automatically replacing the specified paths in the document with document(s) from other collection(s). We may populate a single document, multiple documents, plain object, multiple plain objects, or all objects returned from a query. Let's look at some examples.

```
var mongoose = require('mongoose')
  , Schema = mongoose.Schema

var personSchema = Schema({
  _id     : Number,
  name    : String,
  age     : Number,
  stories : [{ type: Schema.Types.ObjectId, ref: 'Story' }]
});

var storySchema = Schema({
  _creator : { type: Number, ref: 'Person' },
  title    : String,
  fans     : [{ type: Number, ref: 'Person' }]
});

var Story  = mongoose.model('Story', storySchema);
var Person = mongoose.model('Person', personSchema);
```

Facts

MongoDB has no concept of relationship

We need relationships in our application

Let's build a JOIN engine inside our application!

Taken from the Mongoose website

Population

So far we haven't done anything much different. We've merely created a `Person` and a `Story`. Now let's take a look at populating our story's `_creator` using the query builder:

```
Story
.findOne({ title: 'Once upon a timex.' })
.populate('_creator')
.exec(function (err, story) {
  if (err) return handleError(err);
  console.log('The creator is %s', story._creator.name);
  // prints "The creator is Aaron"
});
```

Let's do TWO queries to Mongo

If we used find() instead of findOne(), we would have done 1+N queries

Facts

MongoDB has no concept of relationship

We need relationships in our application

Let's build a JOIN engine inside our application!

Taken from the Mongoose website

NoSQL is hard!

```
{
  "name": "Matteo",
  "surname": "Collina",
  "phone": "+39 347123456",
  "Description": "call me 24/7!!!"
  "emails": [{
    "type": "university",
    "address":
"matteo.collina@cepu.org"
  }]
```

- Relational vs. NoSQL
- It is not Node fault!
- Deduplication is accepted
- Don’t shoot in your own foot

The fixed product has huge success

Bob is happy again

Created by Julia Holmberg
from Noun Project

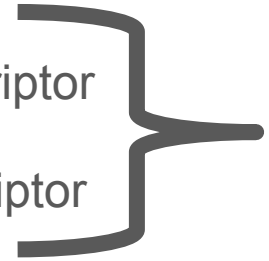
...but my servers stop responding!!!!


Such speed
So enterprise
Wow JS

Exhausted file descriptors

Every incoming HTTP request is a file descriptor

Every outgoing HTTP request is a file descriptor



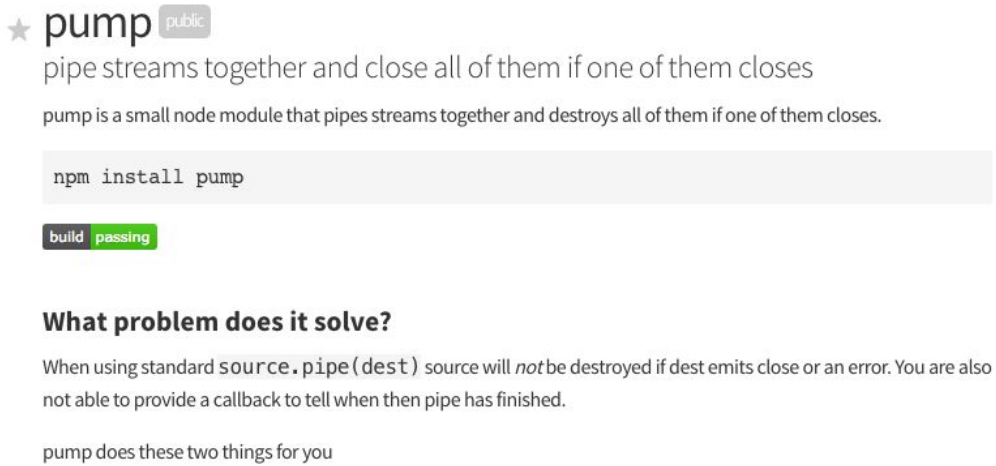
Fixed amount

1. You receive an HTTP request call to /faulty
2. In faulty, you call service A on /doSomething
3. You **pipe** the result of /doSomething to the HTTP response
4. An error happens, and **you do not close the HTTP response**

Exhausted file descriptors: solution

Replace `a.pipe(b)`

With `pump(a, b)`



★ pump public

pipe streams together and close all of them if one of them closes

pump is a small node module that pipes streams together and destroys all of them if one of them closes.

```
npm install pump
```

build passing

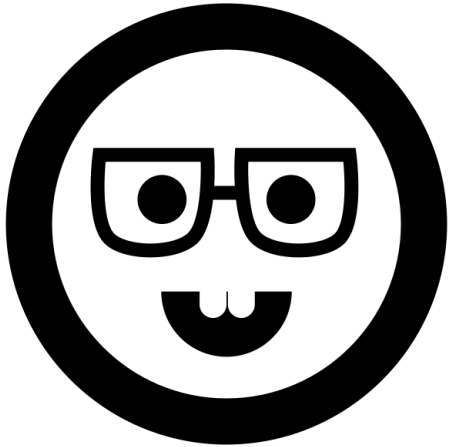
What problem does it solve?

When using standard `source.pipe(dest)` source will *not* be destroyed if dest emits close or an error. You are also not able to provide a callback to tell when then pipe has finished.

pump does these two things for you

We are back online!

Bob is happy again again



Created by Julia Holmberg
from Noun Project

But the Business (always) wants more...

- Scaling the development team from 1 (Bob) to 5
- Delivering more features in less time
- 1000 unit tests!
- Serve more customers!


Such speed
So enterprise
Wow JS

The new team does not deliver

Bob is SAAD

Created by Julia Holmberg
from Noun Project

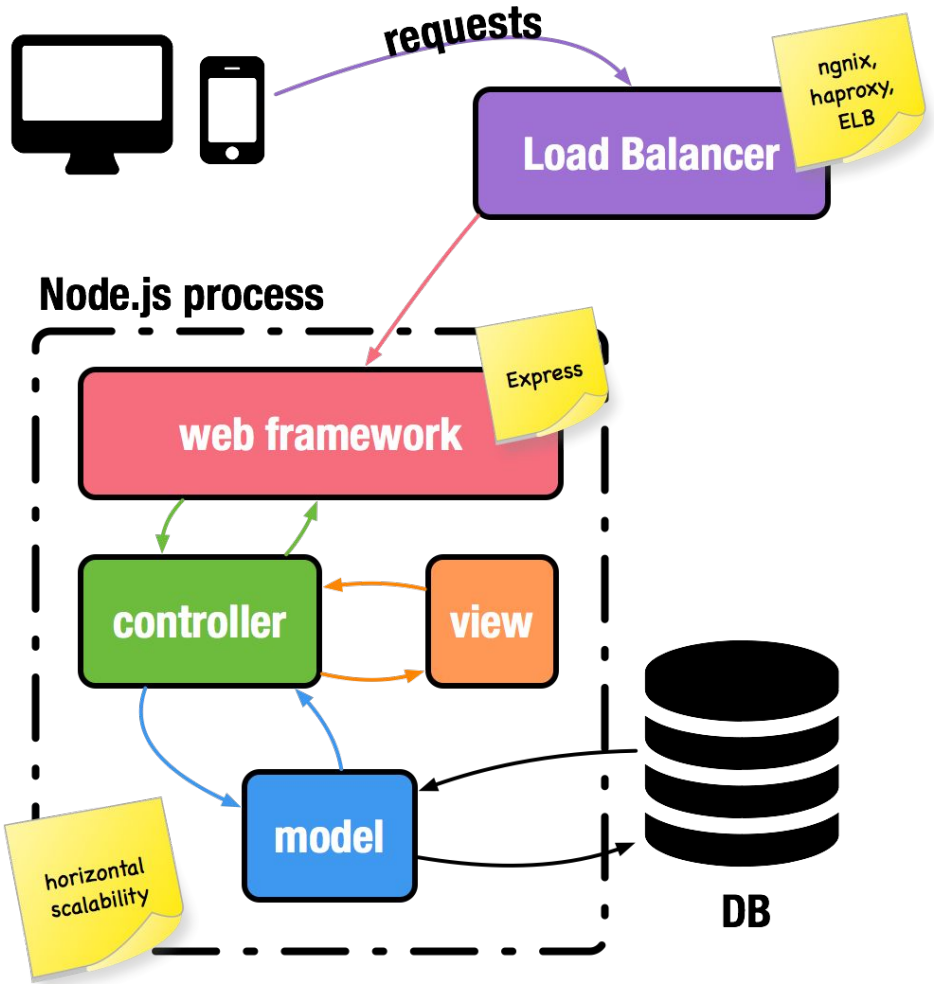
MVC architecture

Really monolithic

Adding a feature means changing all “layers” in an application

Does not scale with the team size

Equivalent VMs need to be dimensioned to handle the most demanding feature



Don’t push anything till tomorrow..

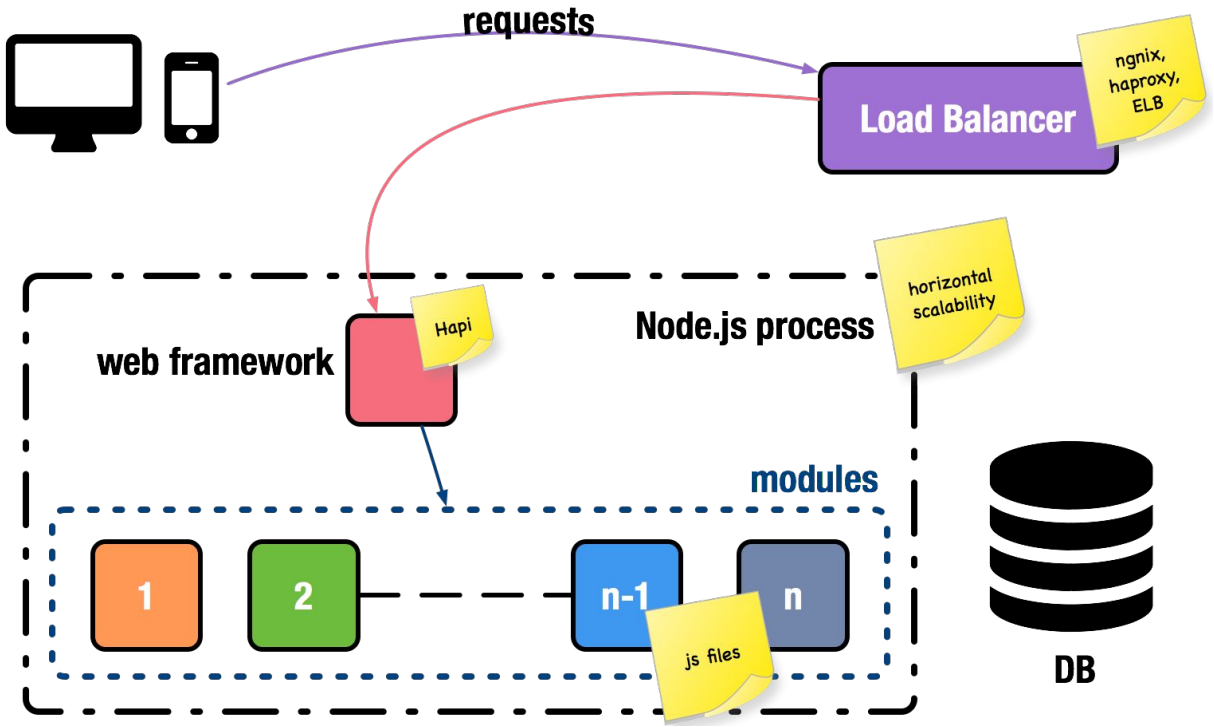
Developing with modules

Really monolithic

~~Adding a feature means changing all "layers" in an application~~

Does ~~not~~ scale with the team size *up to a certain point*

Equivalent VMs need to be dimensioned to handle the most demanding feature



Productivity goes up again!

Bob is happy again again again

Created by Julia Holmberg from Noun Project

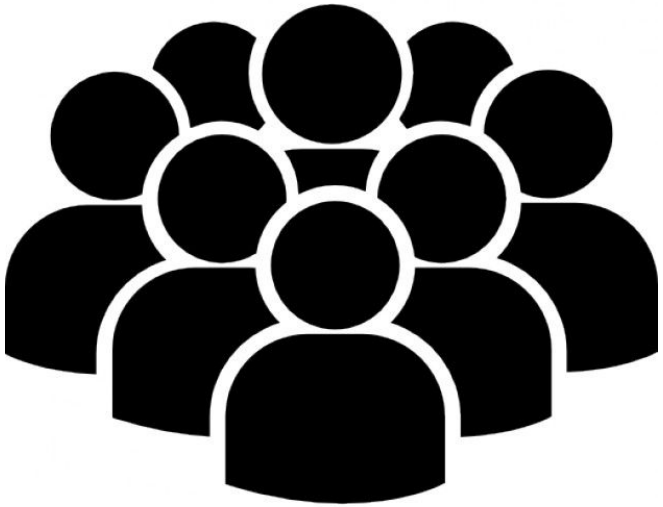
But the Business (always) wants more...

- Scaling the development team from 5 to 20
- Delivering **even** more features in less time
- 10000 unit tests!
- Serve more customers!

The NEW new team does not deliver

Bob is
DESPERATE

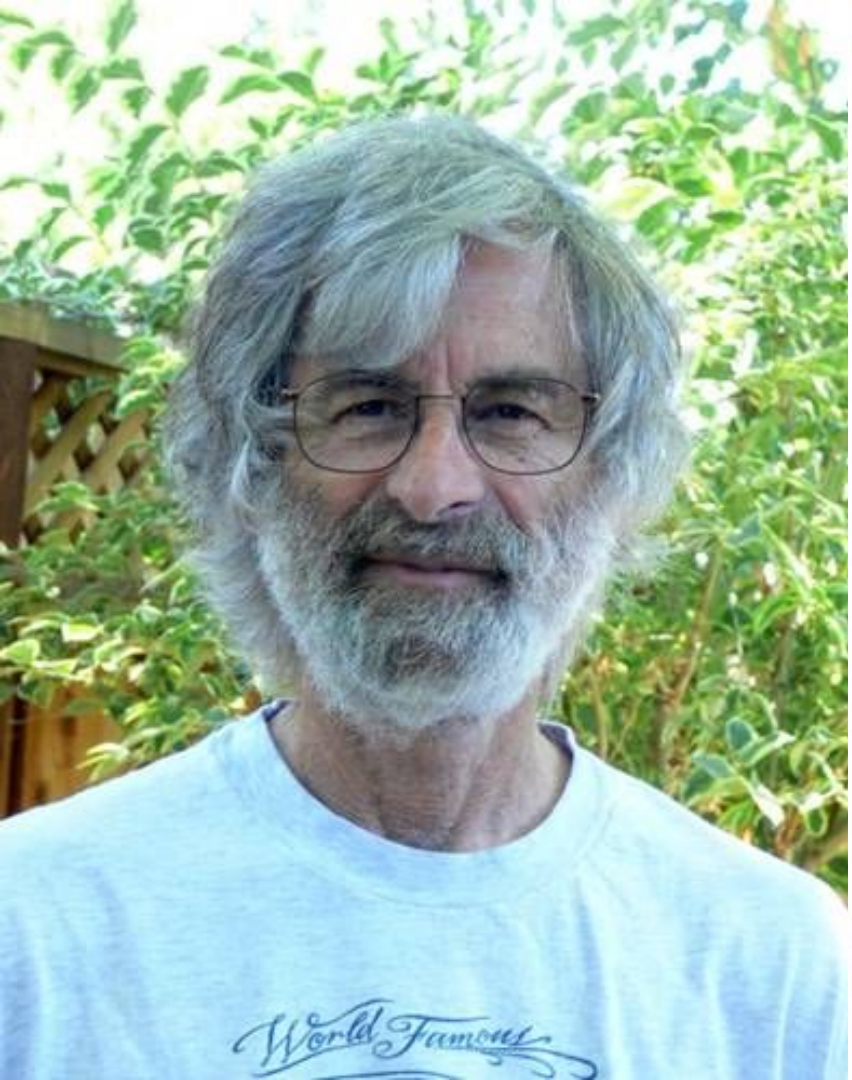
Created by Julia Holmberg
from Noun Project



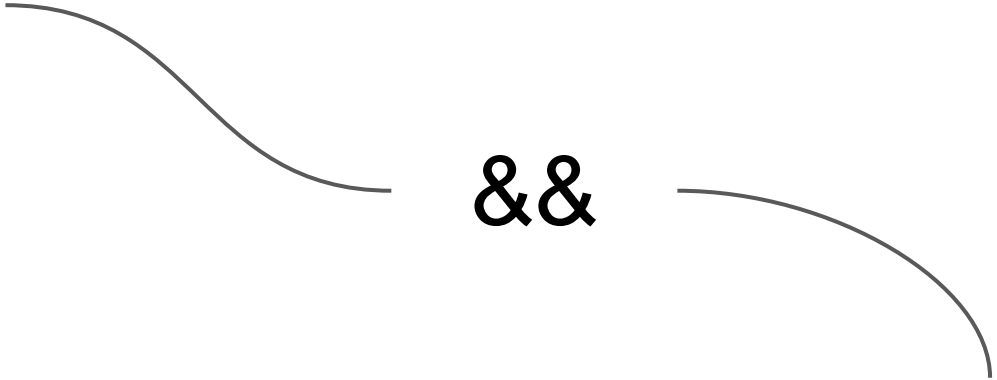
==

++

distribution


World Famous

Missing Resiliency

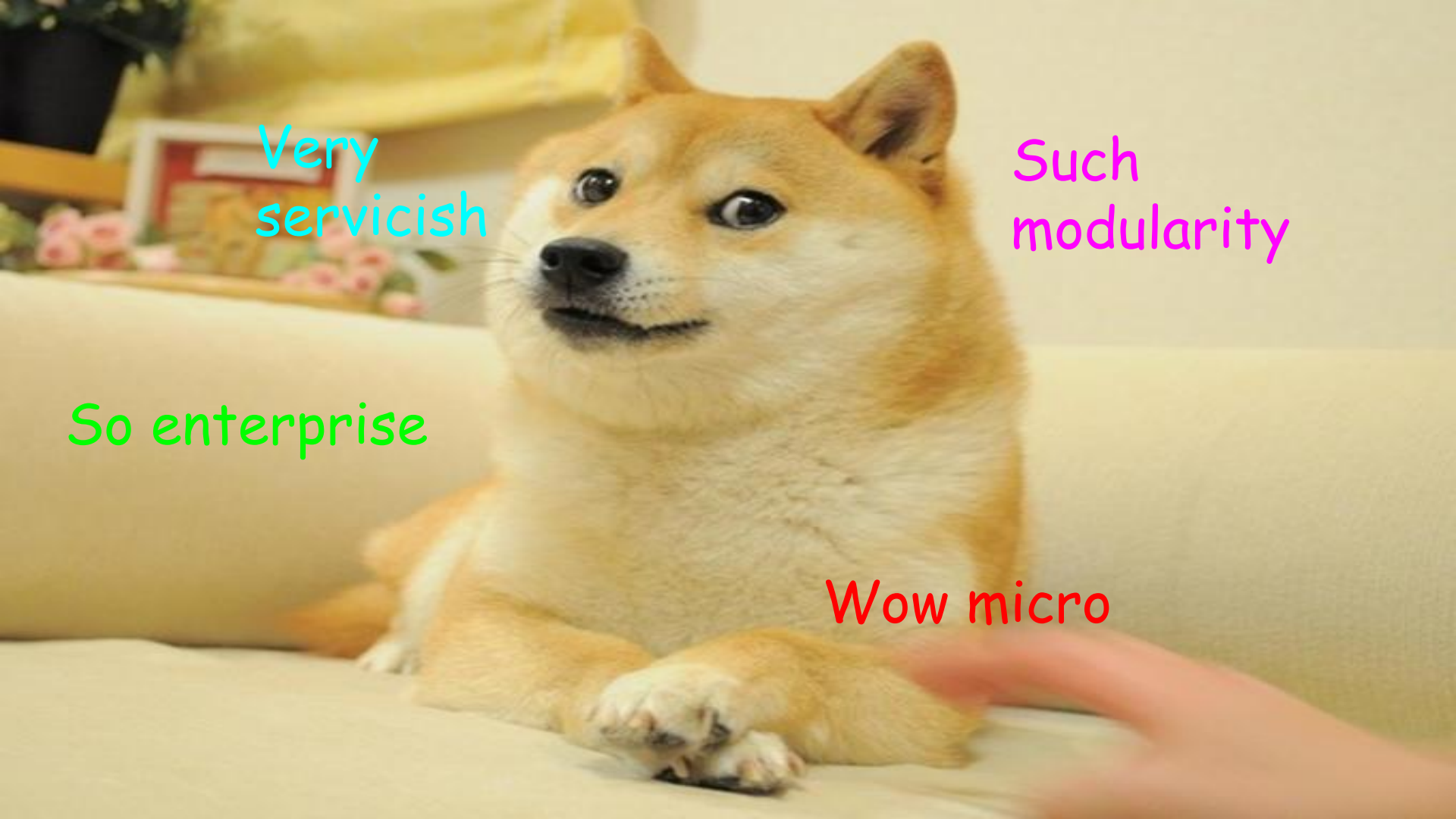


Fault tolerance

A pre-made “stack” might not be good


Let’s throw
everything away...
But .LAVA42 does it better! ;-)


Hipsters use
microservices


Very servicish
Such modularity
So enterprise
Wow micro

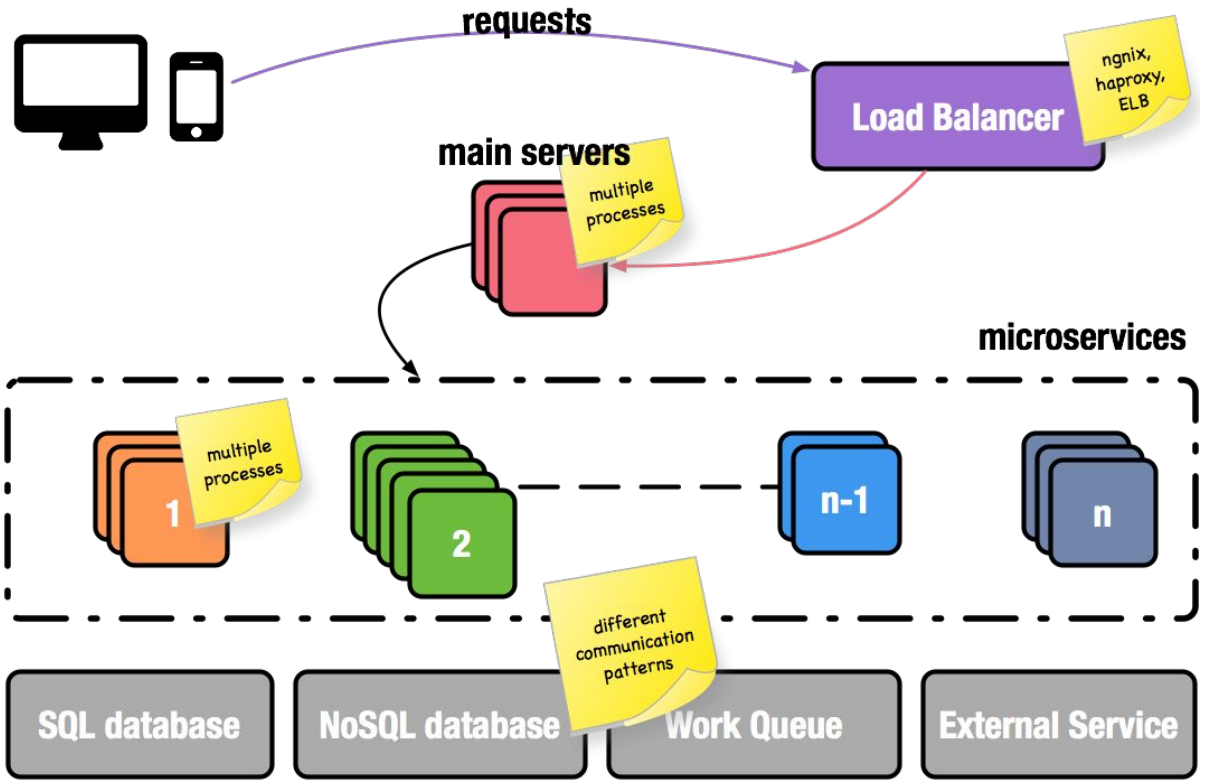
Microservices architecture

~~Really monolithic~~

~~Adding a feature means changing all "layers" in an application~~

~~Does not scale with the team size *up to a certain point*~~

~~Equivalent VMs need to be dimensioned to handle the most demanding feature~~

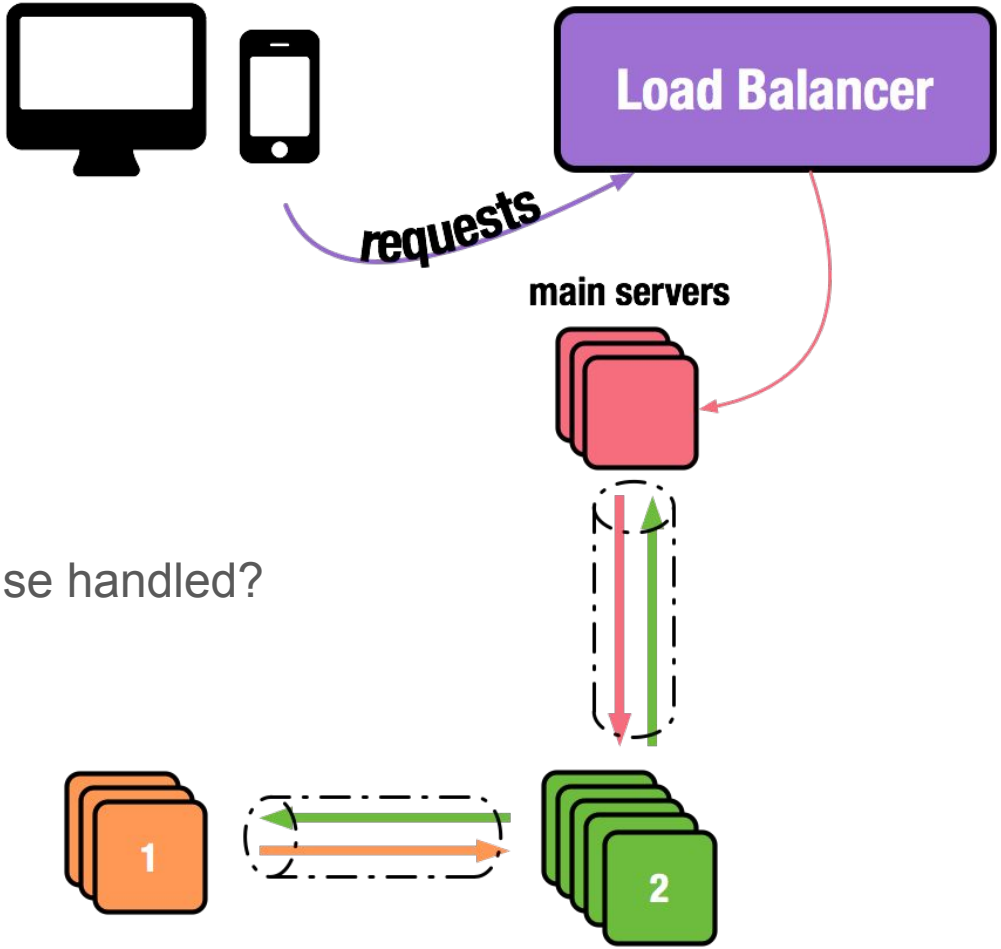



```
bash-3.2$
```

Queue in between

How we discover the queues?

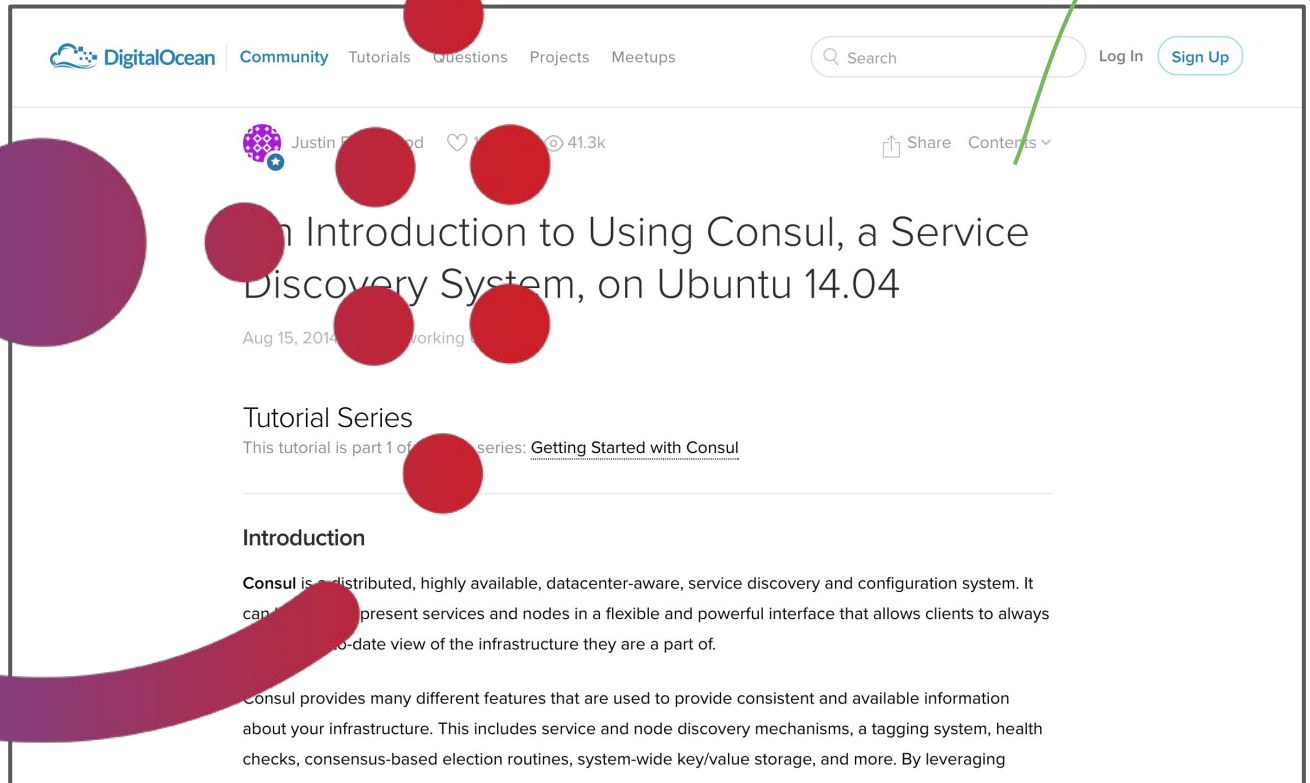
How are the back channels / response handled?



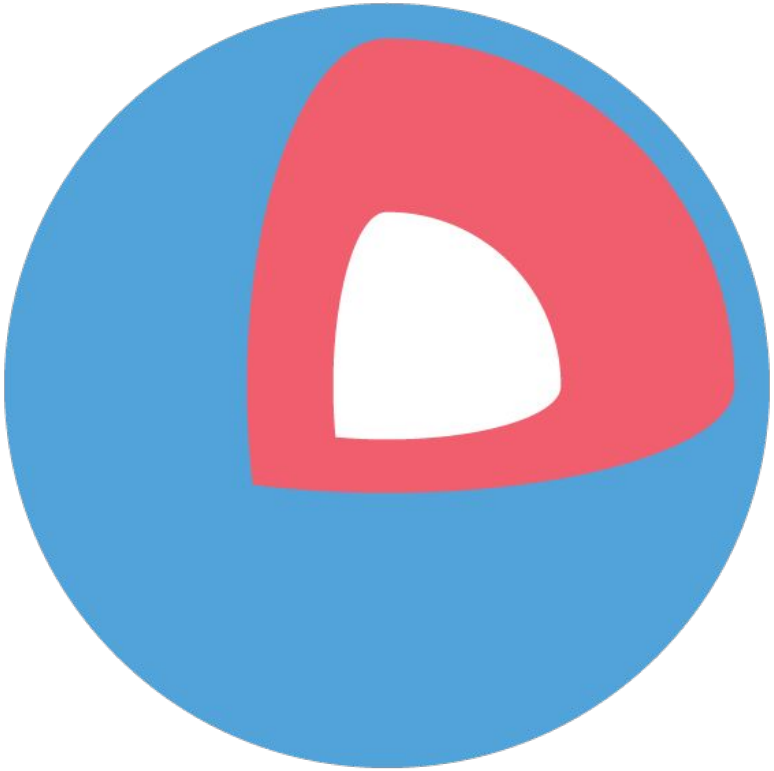

The answer to
life
universe,
and everything
Spiral Galaxy NGC 3370
HUBBLESITE.org


~~The answer to life universe, and everything~~
The answer to everything is **service discovery**
Spiral Galaxy NGC 3370
HUBBLESITE.org

Discovery with DNS: Meet Consul



Etcd… but CoreOS



CoreOS

SWIM: *S*calable *W*eakly-consistent *I*nfection-style Process Group *M*embership Protocol

Abhinandan Das, Indranil Gupta, Ashish Motivala*
Dept. of Computer Science, Cornell University
Ithaca NY 14853 USA
{asdas,gupta,ashish}@cs.cornell.edu

Abstract

Several distributed peer-to-peer applications require weakly-consistent knowledge of process group membership information at all participating processes. SWIM is a generic software module that offers this service for large-scale process groups. The SWIM effort is motivated by the unscalability of traditional heart-beating protocols, which either impose network loads that grow quadratically with group size, or compromise response times or false positive frequency w.r.t. detecting process crashes. This paper reports on the design, implementation and performance of the SWIM sub-system on a large cluster of commodity PCs.

Unlike traditional heartbeating protocols, SWIM separates the failure detection and membership update dissemination functionalities of the membership protocol. Processes are monitored through an efficient peer-to-peer peri-

1. Introduction

As you swim lazily through the milieu,
The secrets of the world will infect you.

Several large-scale peer-to-peer distributed process groups running over the Internet rely on a distributed membership maintenance sub-system. Examples of existing middleware systems that utilize a membership protocol include reliable multicast [3, 11], and epidemic-style information dissemination [4, 8, 13]. These protocols in turn find use in applications such as distributed databases that need to reconcile recent disconnected updates [14], publish-subscribe systems, and large-scale peer-to-peer systems[15]. The performance of other emerging applications such as large-scale cooperative gaming, and other collaborative distributed applications, depends critically on the reliability and scalability of the membership maintenance protocol used within.

Briefly, a membership protocol provides each process

Scalable

Weakly consistent

Infection style

Membership protocol

Advantages

- Weakly consistent vs. strongly consistent
- Bornt for distributed systems
- Failure detection over the cluster
- Information dissemination
- Equally distribution of the workload


Very distributed
Such consistency
So infectious
Wow failure detection

Common implementations

Hashicorp SERF

If you are into GO

mrhooray/swim-js

If you are into JS ;-)

But a new cluster node is added...

How to rebalance the cluster topology?


ReactionGIFS.me

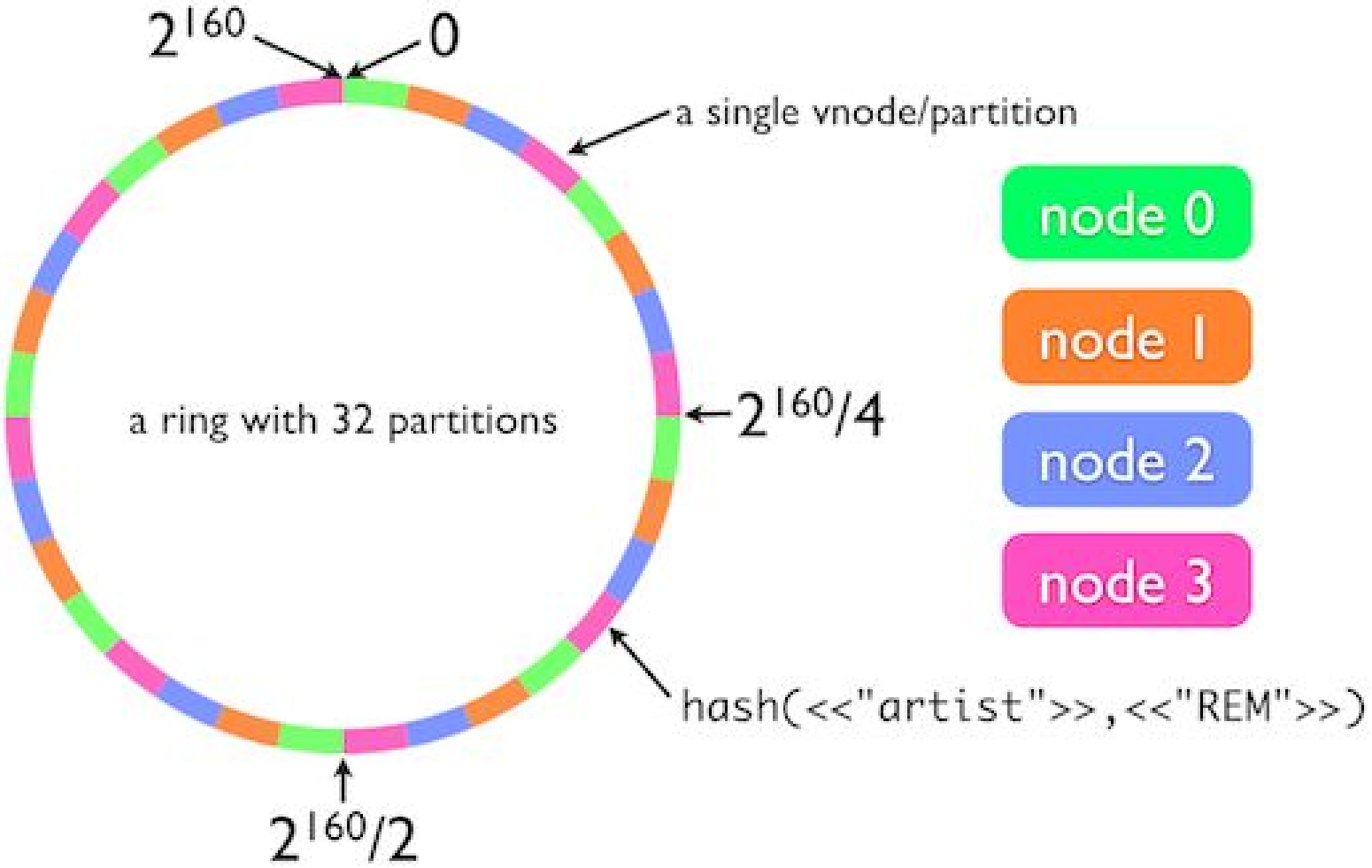
Consistent hashring

PROBLEM

How can I find a server over an unstable network?

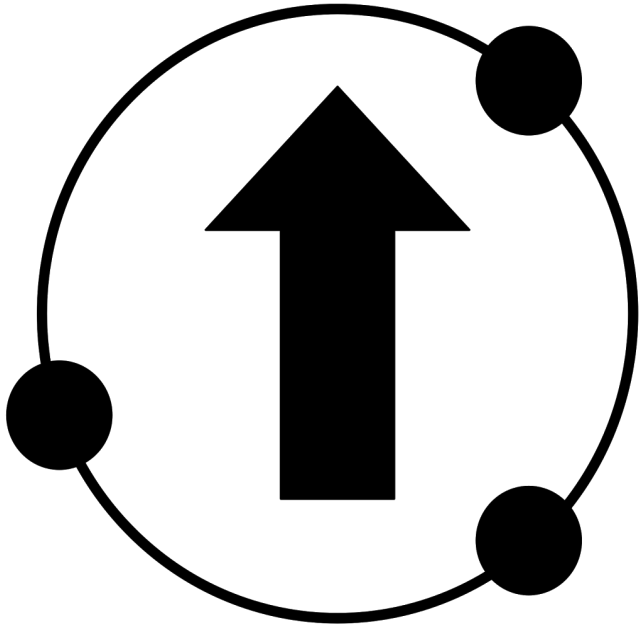
SOLUTION

Each server and keys are hashed in a circular hashspace


2^{160}
0
a single vnode/partition
node 0
node 1
node 2
node 3
a ring with 32 partitions
$2^{160}/4$
hash(<<"artist">>,<<"REM">>)
$2^{160}/2$

Advantages

- Easier to avoid hotspots
- Enable partitioning
- Predictable elasticity
- Easy replication
- Scalability and Availability
- Local state



Upring

application-level
sharding for Node.js

mcollina/upring

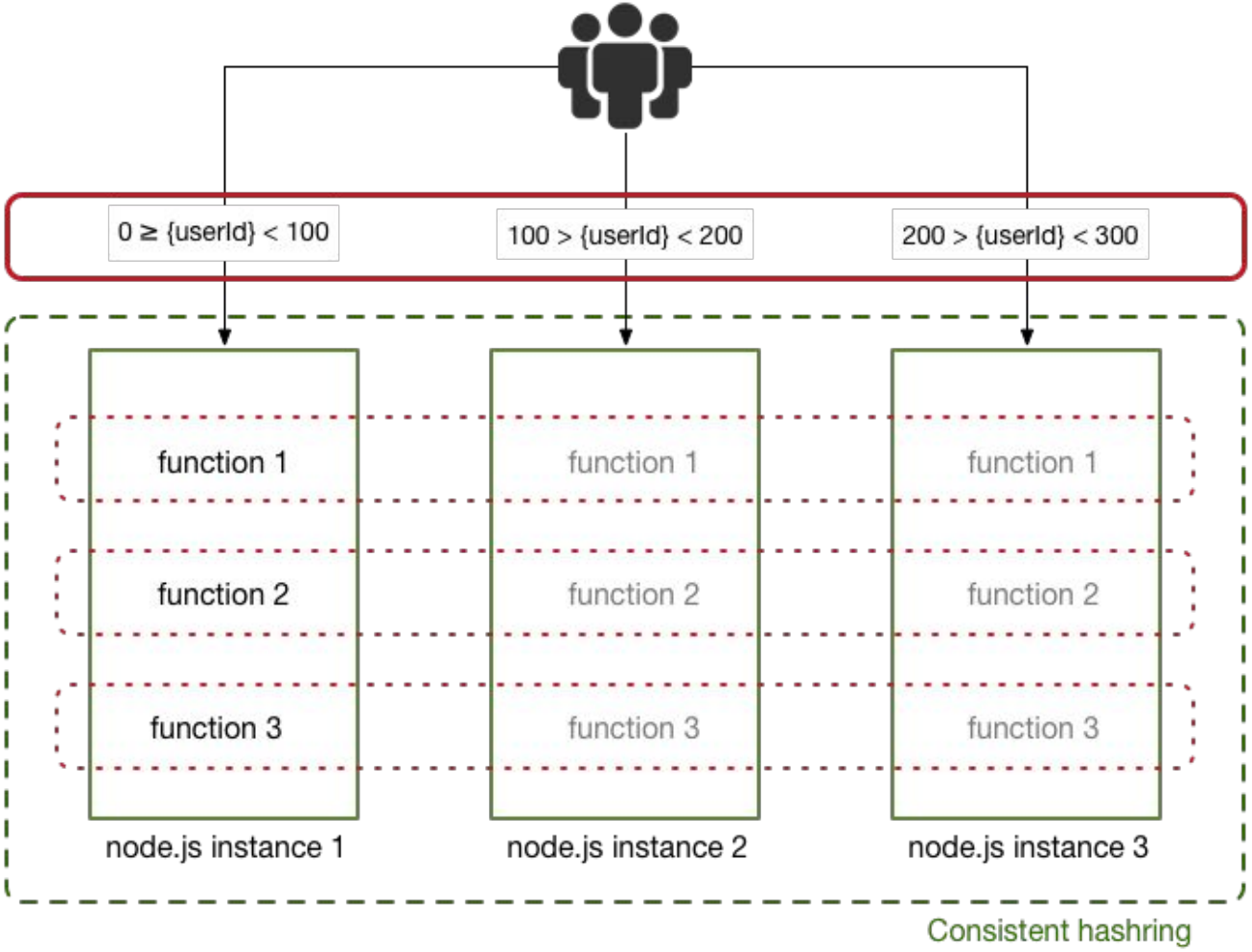
What if Bob was working for Uber? ;-)

ringpop

Scalable, fault-tolerant application-layer sharding

FORK ME ON GITHUB

Application-level sharding


0 ≥ {userId} < 100
100 > {userId} < 200
200 > {userId} < 300
function 1
function 2
function 3
function 1
function 2
function 3
function 1
function 2
function 3
node.js instance 1
node.js instance 2
node.js instance 3
Consistent hashring

The fourth secret of Fatima

hyperbahn

Service discovery and routing for large-scale microservice operations

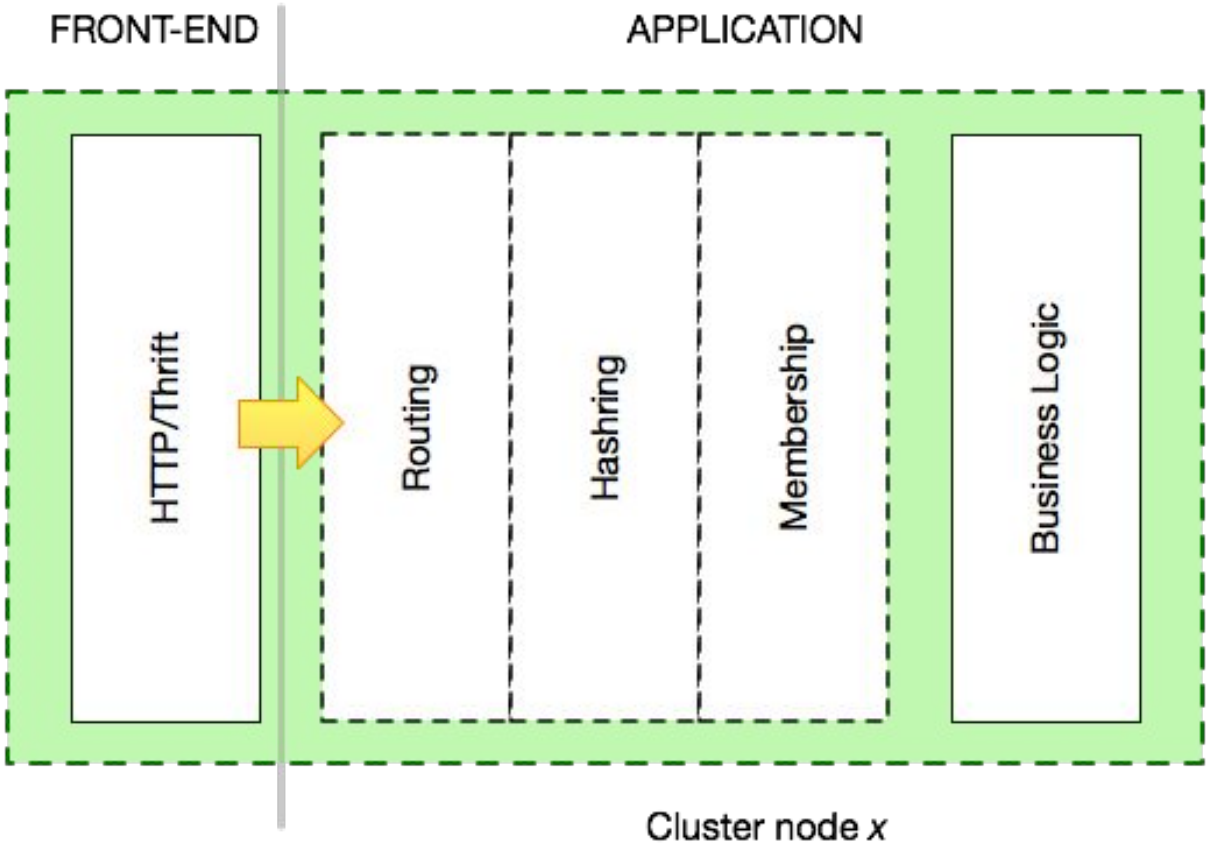
FORK ME ON GITHUB

“Hyperbahn enables service discovery and routing for large-scale systems comprised of many microservices. Distributed, fault tolerant, and highly available, it lets one service find and communicate with others simply and reliably without having to know where those services run.”

Featuring…

- Configuration Discovery
- Retries
- Load Balancing
- Rate Limiting
- Circuit Breaking
- Distributed tracing

But what about HTTP?



tchannel

Network multiplexing and framing protocol for RPC

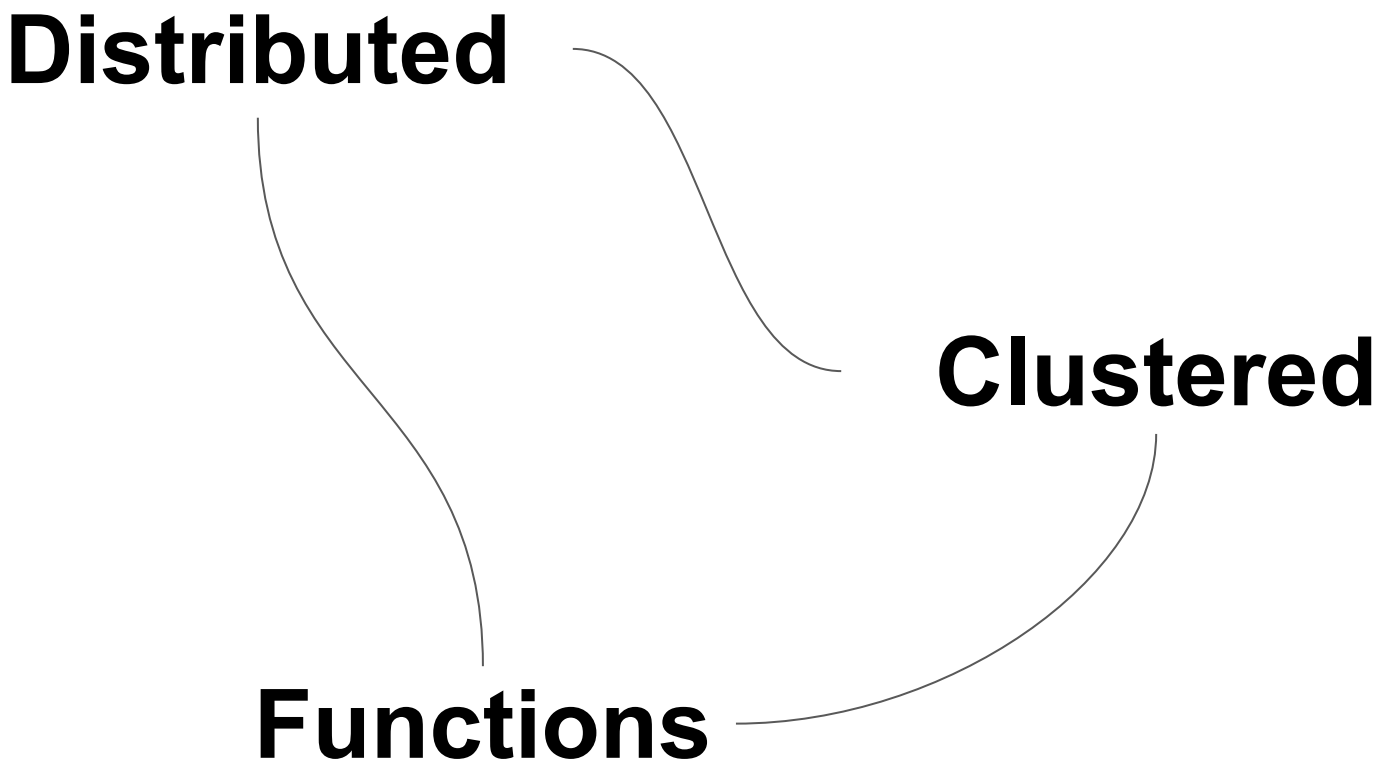
FORK ME ON GITHUB

*TChannel is a networking framing protocol used for general **RPC**, supporting out-of-order responses at extremely high performance where intermediaries can make a forwarding decision quickly*

Tracing as a first class resident...

https://github.com/openzipkin/zipkin

WHY RPC?


Distributed
Clustered
Functions

~~Operation Complexity~~

...but kills

~~DEVELOPMENT WORKFLOW~~

“SERVERLESS”

is the new Hipster


Very
reactive
Such servers
So hipster
Wow functions

“FUNCTIONAL”

is the new Hipster

“DISTRIBUTED FUNCTIONS”

are the future!

Don’t be too late!

@lucamaraschi

Matteo Collina

mcollina

In the clouds above Italy

matteo.collina@gmail.com

http://matteocollina.com

Joined on 5 Feb 2009

856 Followers **3.7k** Starred **281** Following

Organizations

Contributions | Repositories | Public activity

Follow

Popular repositories

mosca 1,111 ★
MQTT broker as a module

levelgraph 687 ★
Graph database JS style for Node.js and the B…

ascoltatori 233 ★
The pub/sub library for node backed by Redis,…

node-coap 177 ★
CoAP - Node.js style

msgpack5 174 ★
A msgpack v5 implementation for node.js, with…

Repositories contributed to

mqttjs/MQTT.js 1,377 ★
The MQTT client for Node.js and the browser

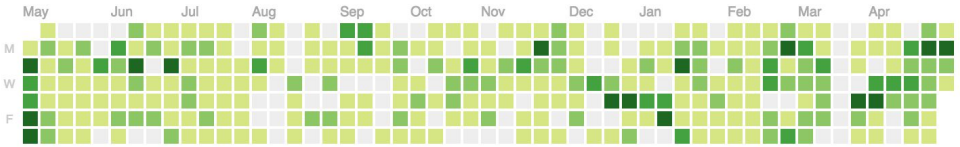
mqttjs/mqtt-packet 35 ★
Parse and generate MQTT packets like a bree…

nearform/nscale-kernel 2 ★

nodejs/node 22,823 ★
Node.js JavaScript runtime ✨🐢🚀✨

apparatus/oast 5 ★
Build containers

Public contributions



Summary of pull requests, issues opened, and commits. Learn how we count contributions.

Less More

Contributions in the last year	Longest streak	Current streak
2,150 total	28 days	17 days
May 4, 2015 – May 4, 2016	June 28 – July 25	April 18 – May 4

“I did it all by myself!”

nearForm

matteo.collina@nearform.com

luca.maraschi@nearform.com

Conclusions

- **Think** about choosing a technology
- Node.js works at **scale**
- You can **implement** any architecture with Node.js