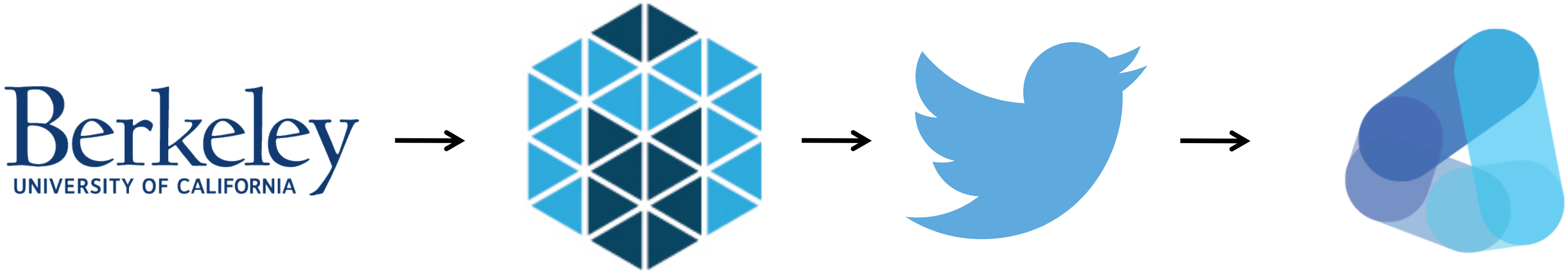
Building/Running Distributed Systems with Apache Mesos

Philly ETE

April 8, 2015

Benjamin Hindman – @benh

$ whoami



2007 - 2012

2009 -

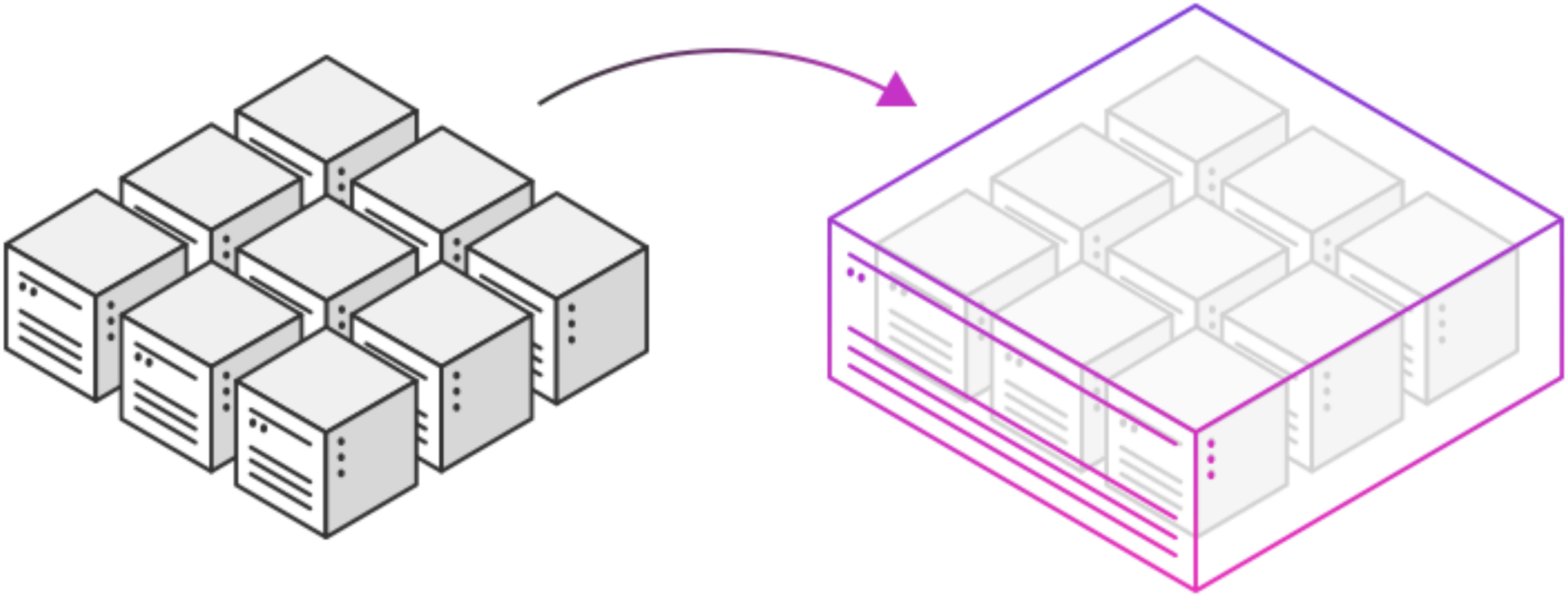
2010 - 2014

my other computer

**my other computer
is a datacenter**

my other computer is a datacenter*

* collection of physical and/or virtual machines



how should we run applications on the datacenter computer?

how do we program applications for the datacenter computer?

what are
datacenter applications?

agenda

① what are datacenter applications?

② how should we run datacenter applications?

③ how should we program datacenter applications?

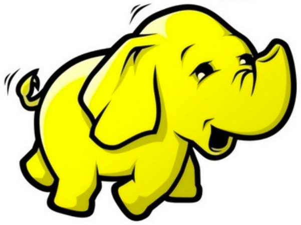
agenda

① what are datacenter applications?

② how should we run datacenter applications?

③ how should we program datacenter applications?

distributed systems!



kafka

cassandra

HDFS

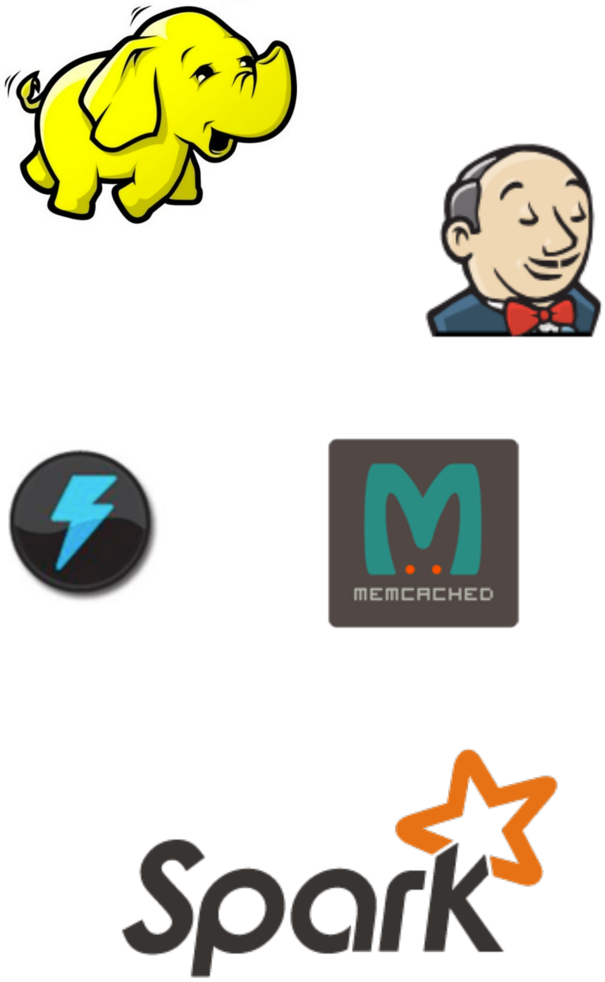
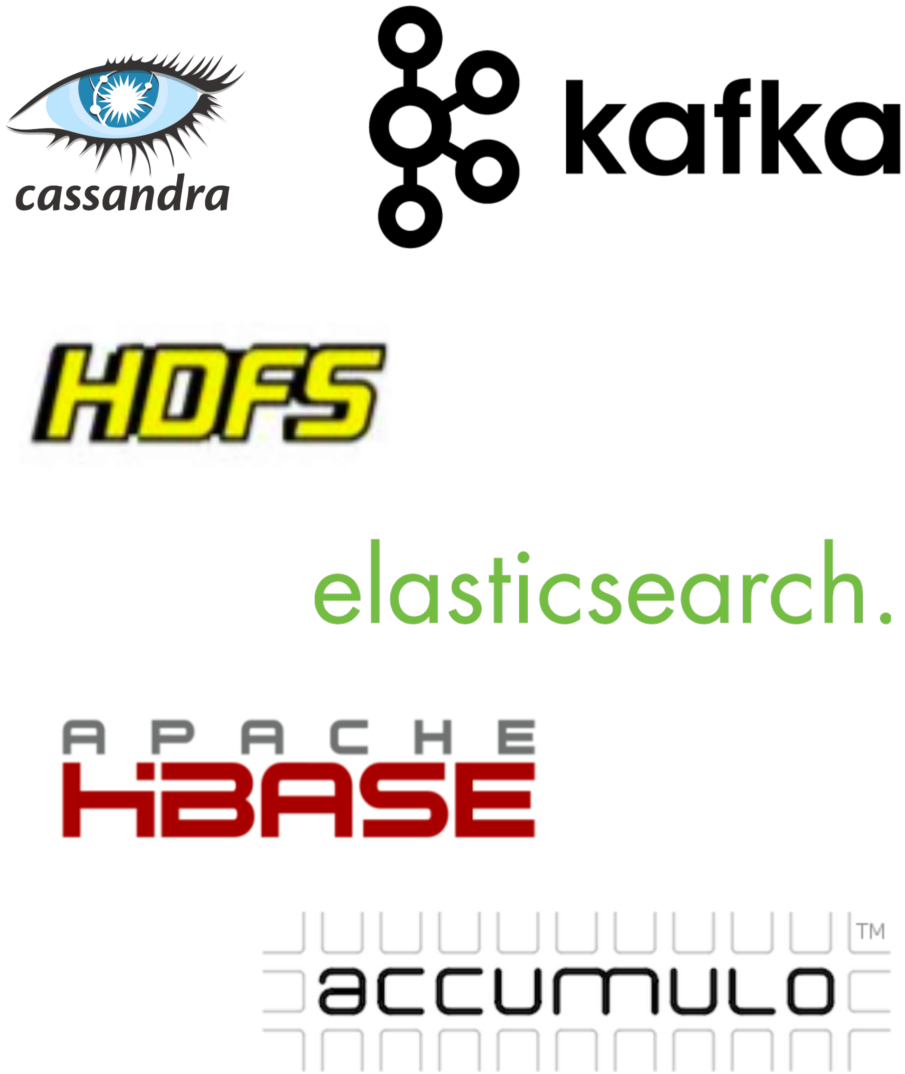
elasticsearch.

M
MEMCACHED

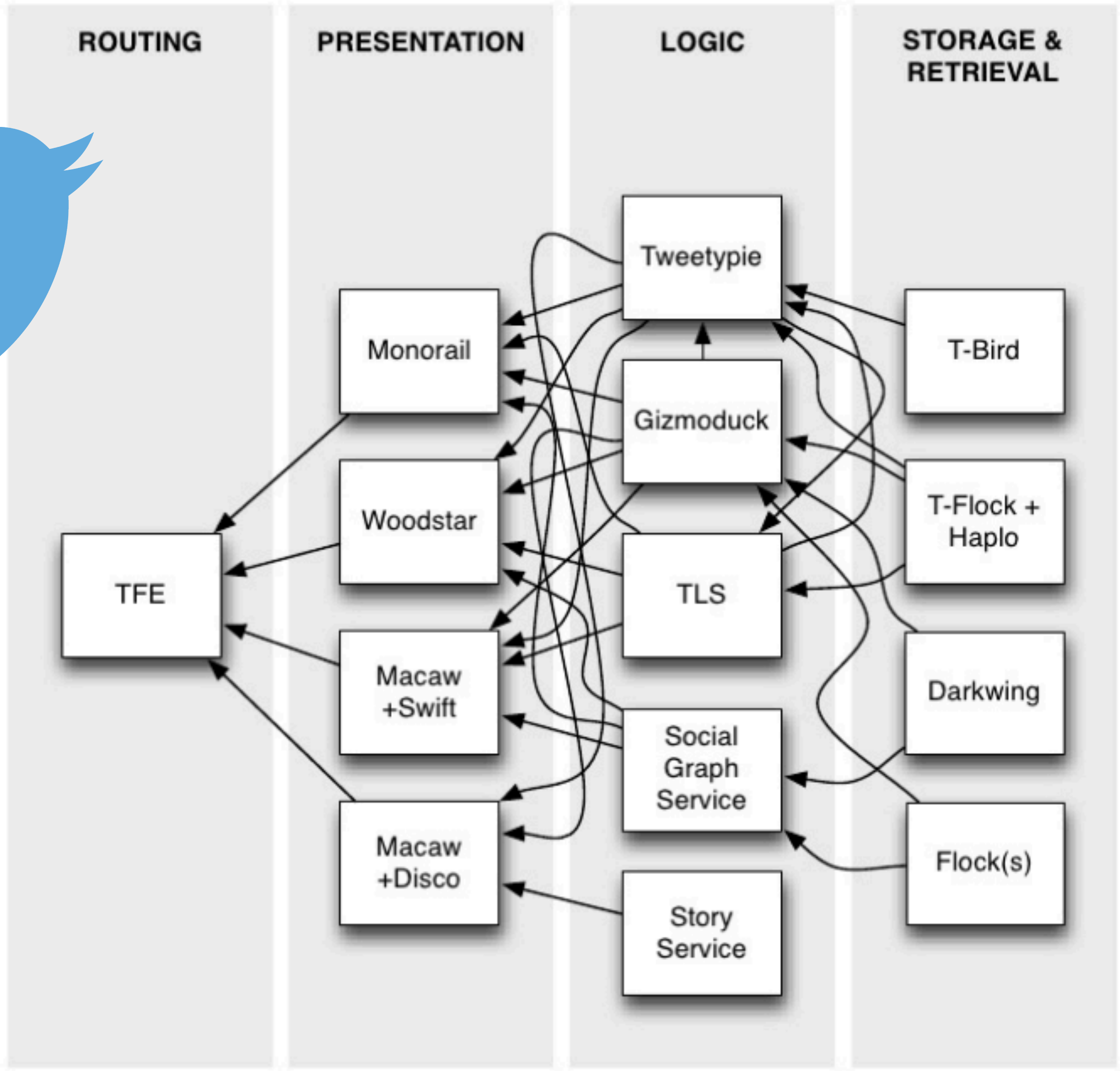
APACHE
HBASE

Spark

accumulo
TM

stateless	stateful
 	 cassandra kafka HDFS elasticsearch. APACHE HBASE accumulo TM

other distributed systems?



ROUTING

PRESENTATION

LOGIC

STORAGE & RETRIEVAL

TFE

Monorail

Woodstar

Macaw +Swift

Macaw +Disco

Tweetypie

Gizmoduck

TLS

Social Graph Service

Story Service

T-Bird

T-Flock + Haplo

Darkwing

Flock(s)

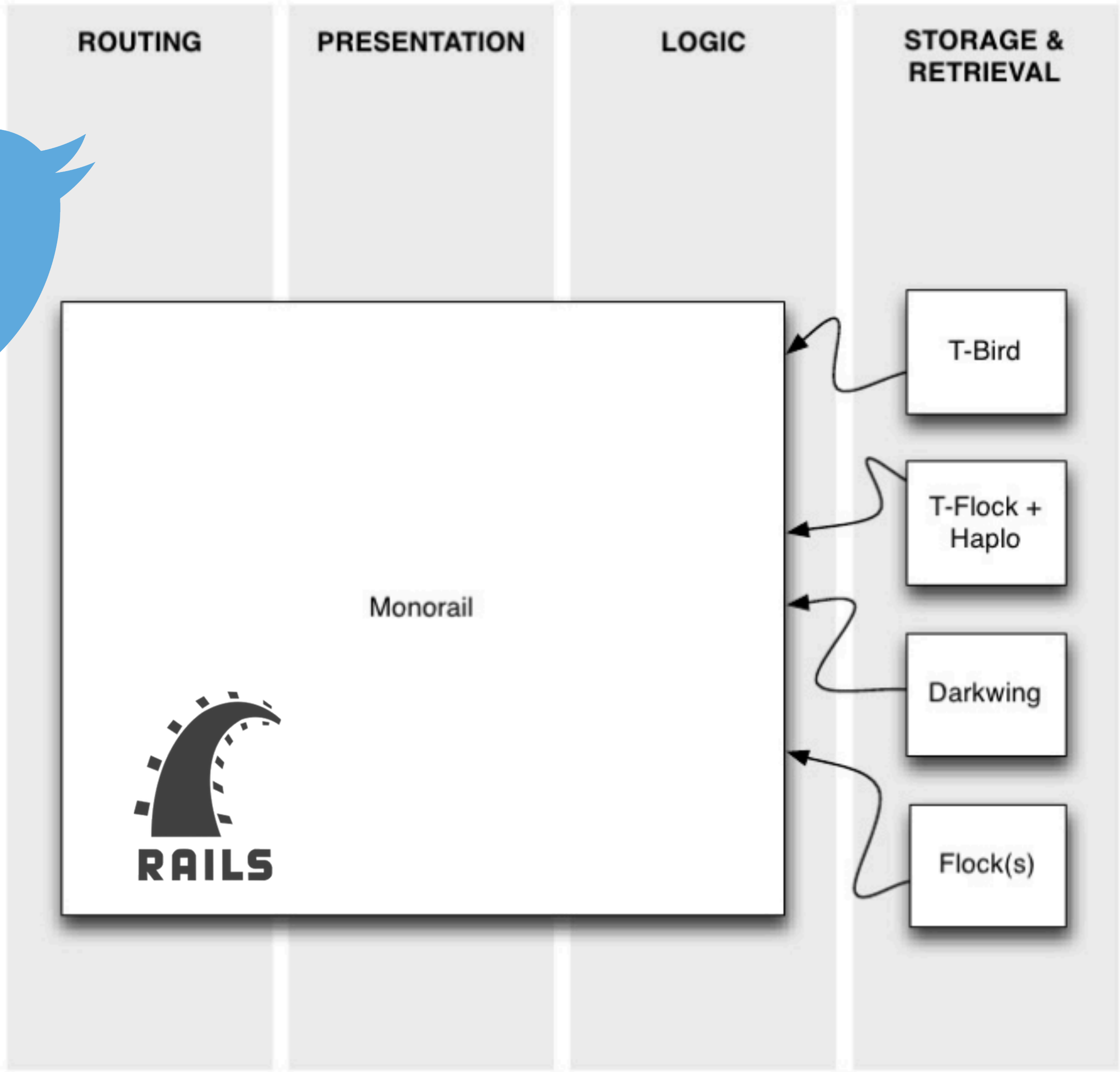
(micro)services

① do one thing and do it well (UNIX)

② compose!

③ build/commit in isolation, test in isolation, deploy in isolation (with easy rollback)

④ captures organizational structure (many teams working in parallel)



ROUTING

PRESENTATION

LOGIC

STORAGE & RETRIEVAL

Monorail

RAILS

T-Bird

T-Flock + Haplo

Darkwing

Flock(s)

There‘s Just No Getting Around It: You‘re Building a Distributed System

acmqueue

Association for Computing Machinery

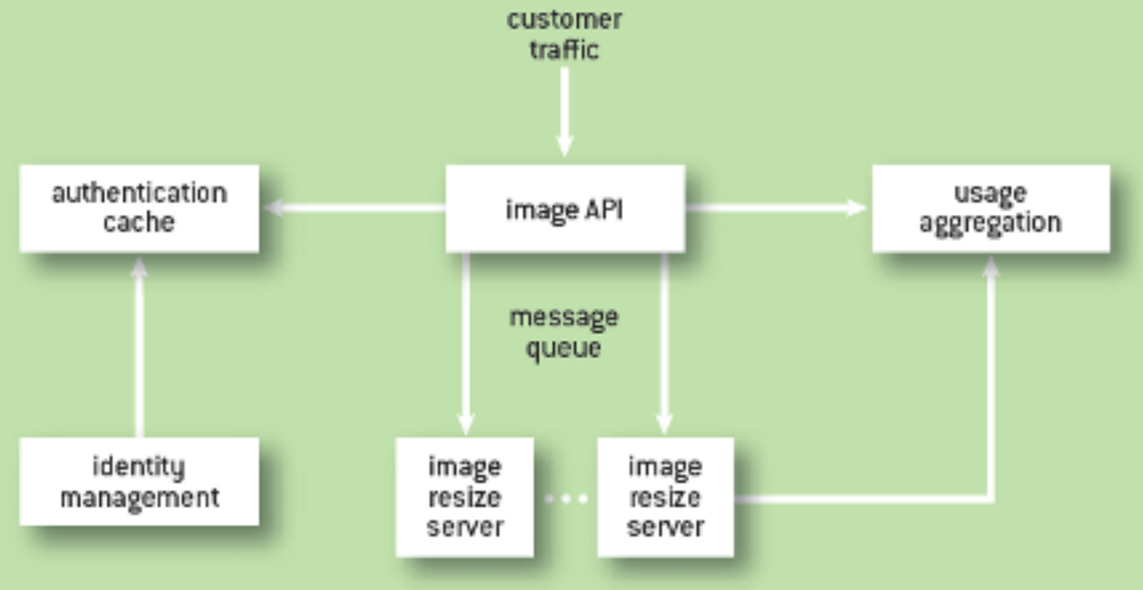
There's Just No Getting Around It: You're Building a Distributed System

by Mark Cavage | May 3, 2013

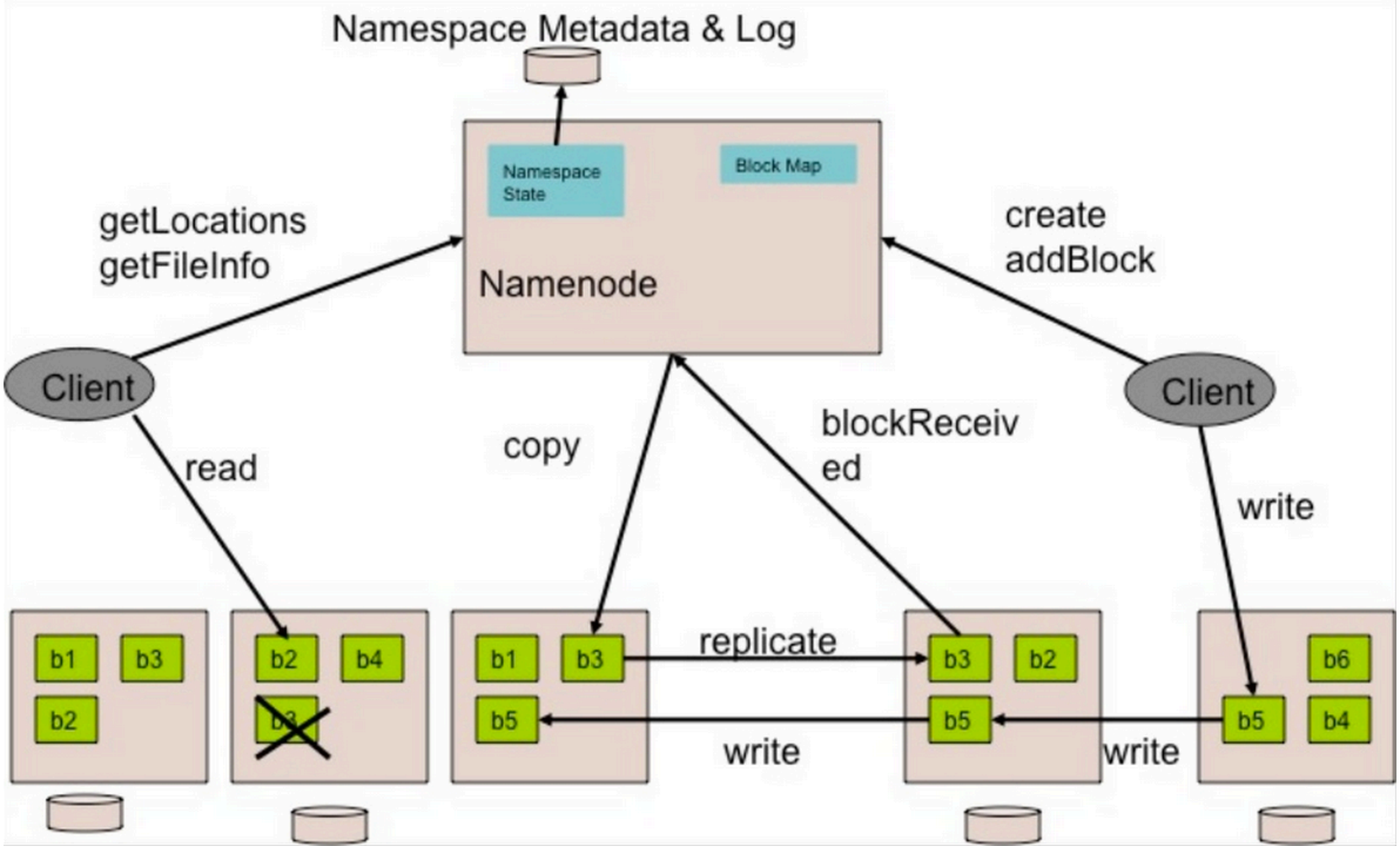
https://queue.acm.org/detail.cfm?id=2482856

FIGURE 1

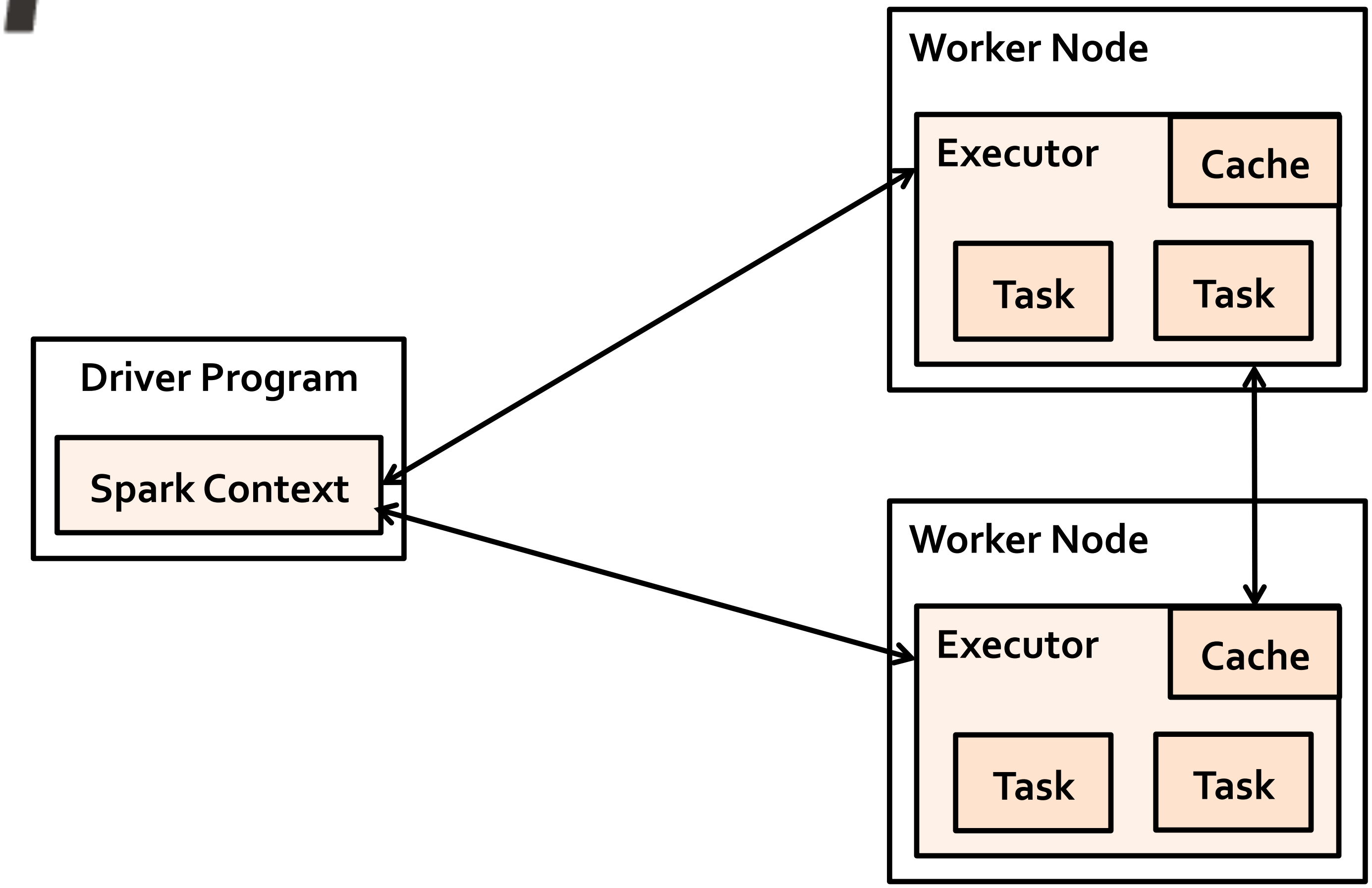
The Distributed Services of an Image Resize Service



HDFS


Namespace Metadata & Log
Namespace State
Block Map
Namenode
getLocations
getFileInfo
create
addBlock
Client
Client
read
copy
blockReceiv
ed
write
b1
b3
b2
b2
b4
b3
b1
b3
b5
replicate
b3
b2
b5
b6
b5
b4
write
write

Spark



Driver Program

Spark Context

Worker Node

Executor

Cache

Task

Task

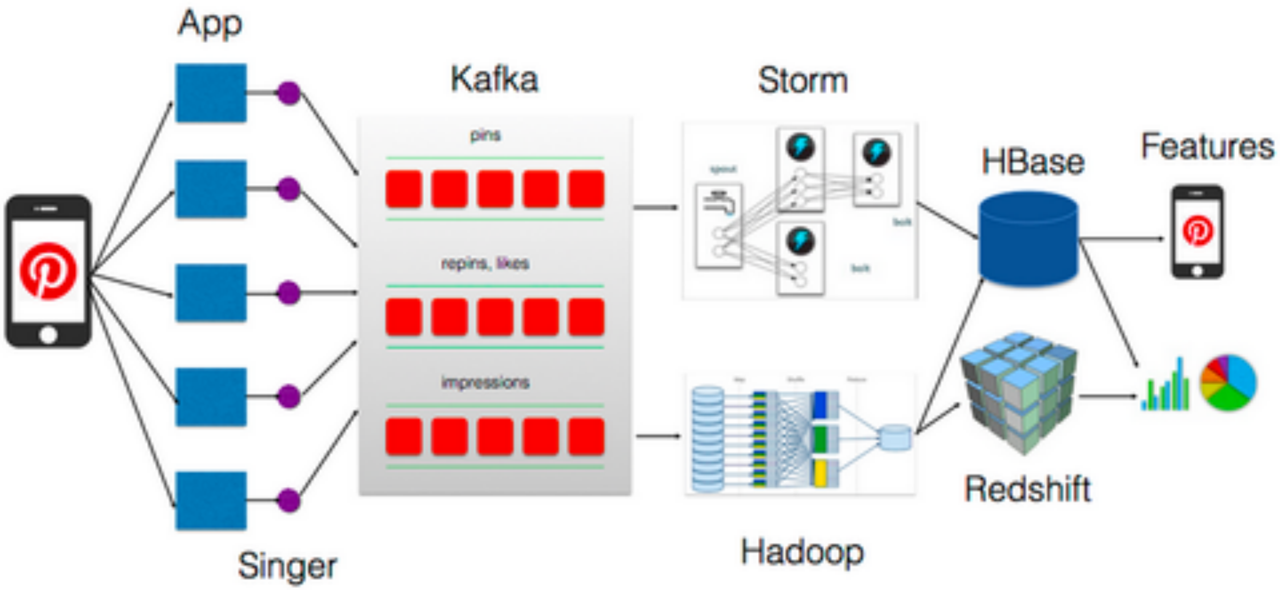
Worker Node

Executor

Cache

Task

Task



Data Architecture overview

agenda

① what are datacenter applications?

② how should we run datacenter applications?

③ how should we program datacenter applications?

considerations

① configuration/package management

② deployment

③ service discovery

considerations

① configuration/package management

② deployment

③ service discovery

④ monitoring

considerations

① configuration/package management

② deployment

③ service discovery

④ monitoring

ops

considerations

① configuration/package management

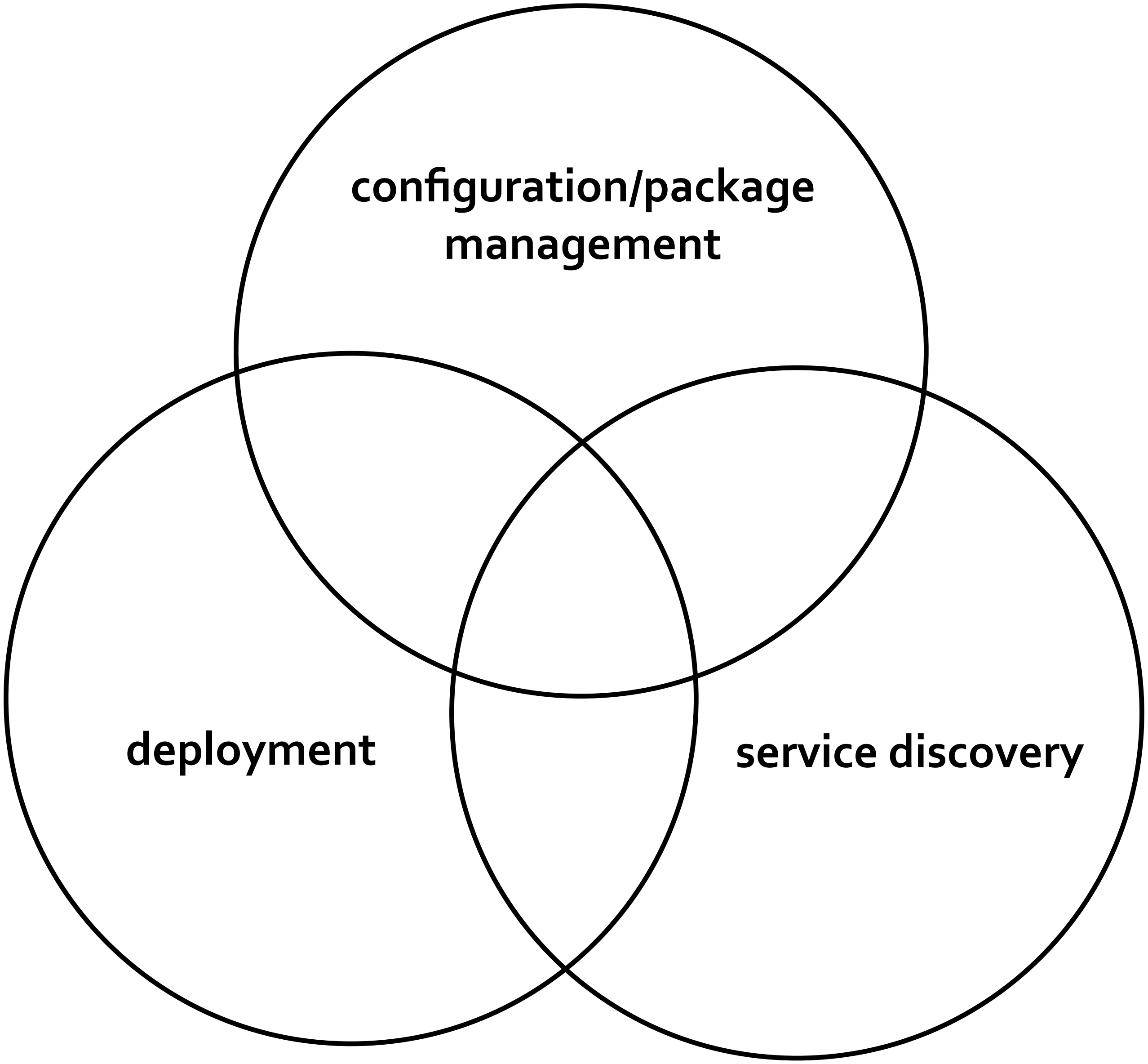
② deployment

③ service discovery

④ monitoring

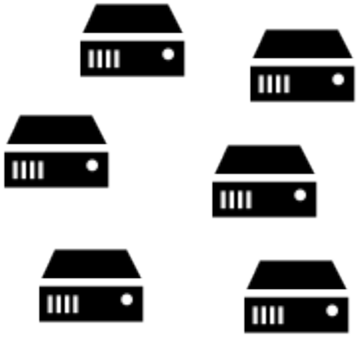
developers

ops

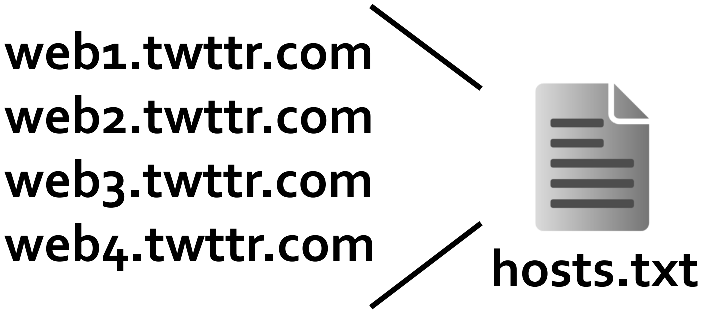

configuration/package management
deployment
service discovery

configuration/package management

"what/how do things get installed?"



(10's of machines)



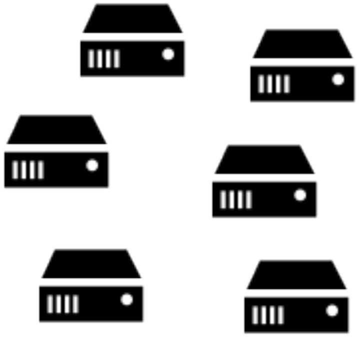
web1.twttr.com
web2.twttr.com
web3.twttr.com
web4.twttr.com

hosts.txt

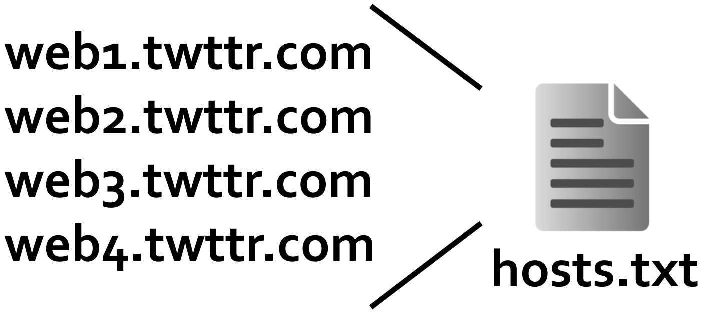
```
$ ssh host ./configure && make install
```

configuration/package management

“what/how do things get installed?”



(10’s of machines)



web1.twttr.com
web2.twttr.com
web3.twttr.com
web4.twttr.com

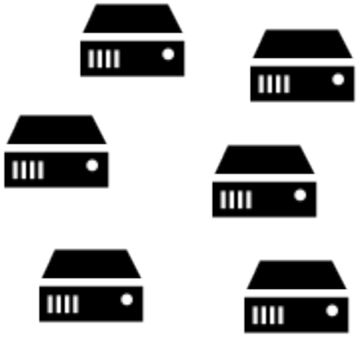
hosts.txt

```
$ ssh host rpm -ivh pkg-x.y.z.rpm
```

deployment

"what should run where?"

"how should it be started/stopped?"



(10's of machines)

web1.twttr.com
web2.twttr.com
web3.twttr.com
web4.twttr.com

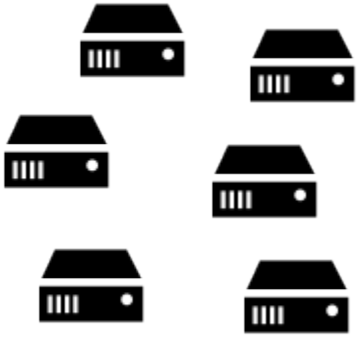
hosts.txt

```
$ ssh host nohup myapp
```

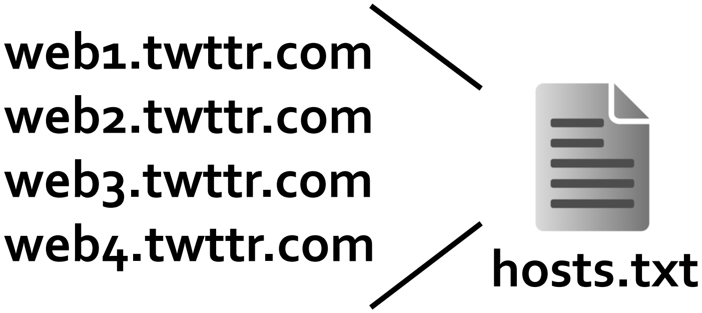
deployment

"what should run where?"

"how should it be started/stopped?"



(10's of machines)



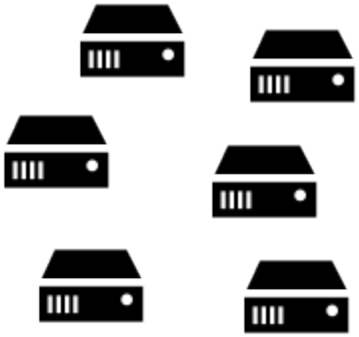
web1.twttr.com
web2.twttr.com
web3.twttr.com
web4.twttr.com

hosts.txt

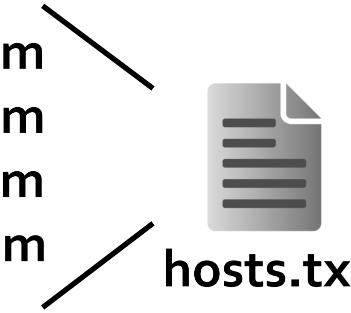
```
$ ssh host monit start myapp
```

deployment

"what should run where?"
"how should it be started/stopped?"



(10's of machines)



web1.twttr.com
web2.twttr.com
web3.twttr.com
web4.twttr.com

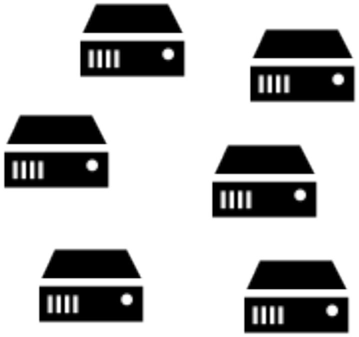
hosts.txt

```
$ scp myapp host
$ ssh host monit myapp
```

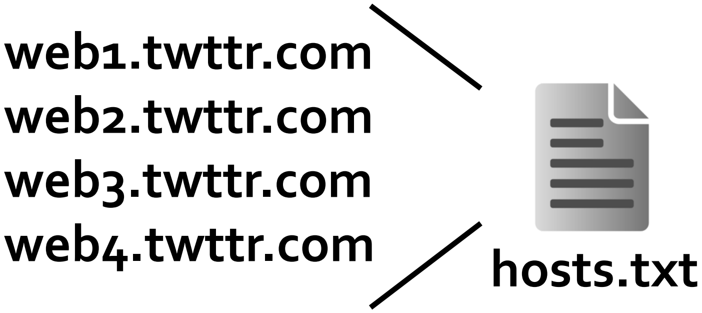
deployment

“what should run where?”

“how should it be started/stopped?”



(10’s of machines)



web1.twttr.com
web2.twttr.com
web3.twttr.com
web4.twttr.com

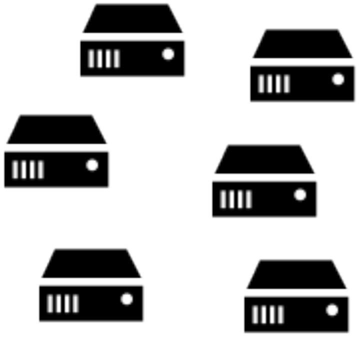
hosts.txt

git

```
$ ssh host git pull && \
   monit myapp
```

service discovery

“how should apps find each other?”



(10’s of machines)

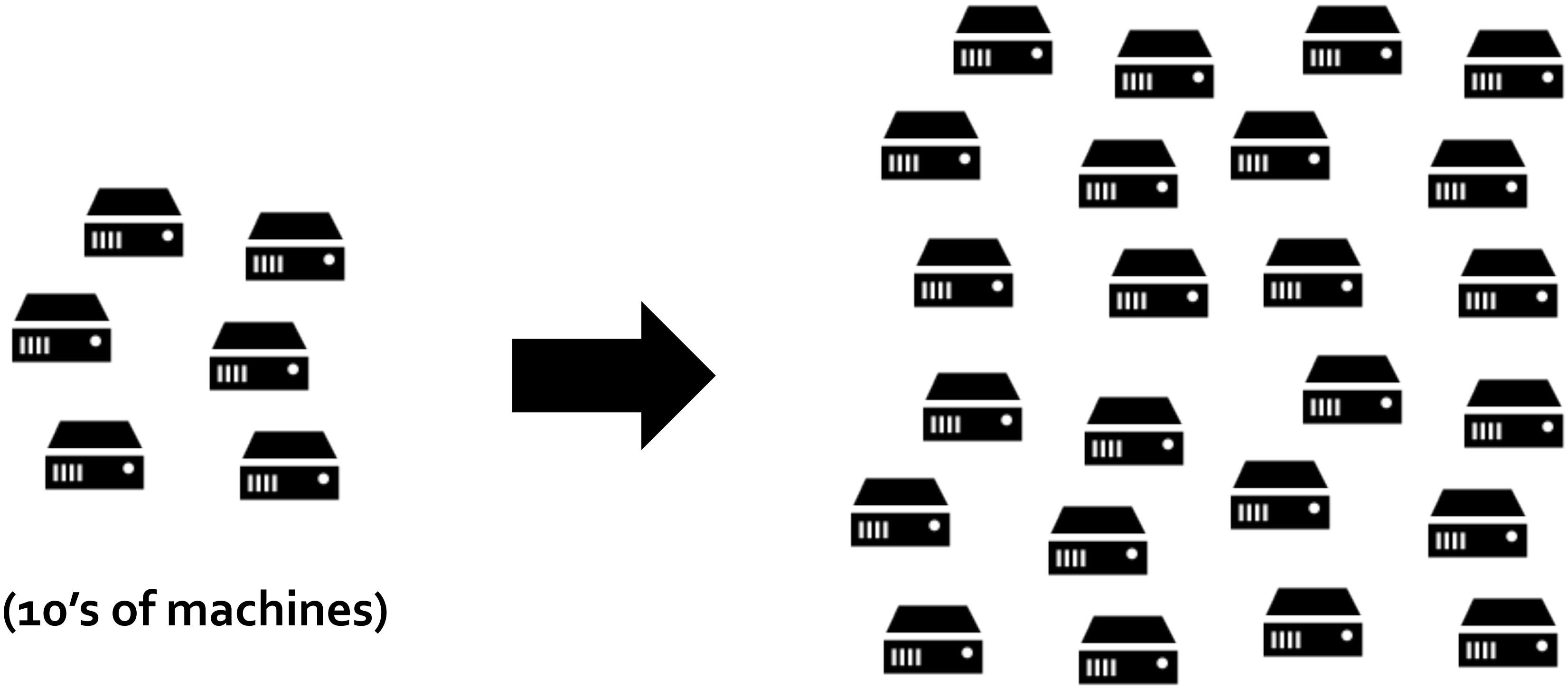
web1.twttr.com
web2.twttr.com
web3.twttr.com
web4.twttr.com

webhosts.txt

db1.twttr.com
db2.twttr.com
db3.twttr.com
db4.twttr.com

dbhosts.txt

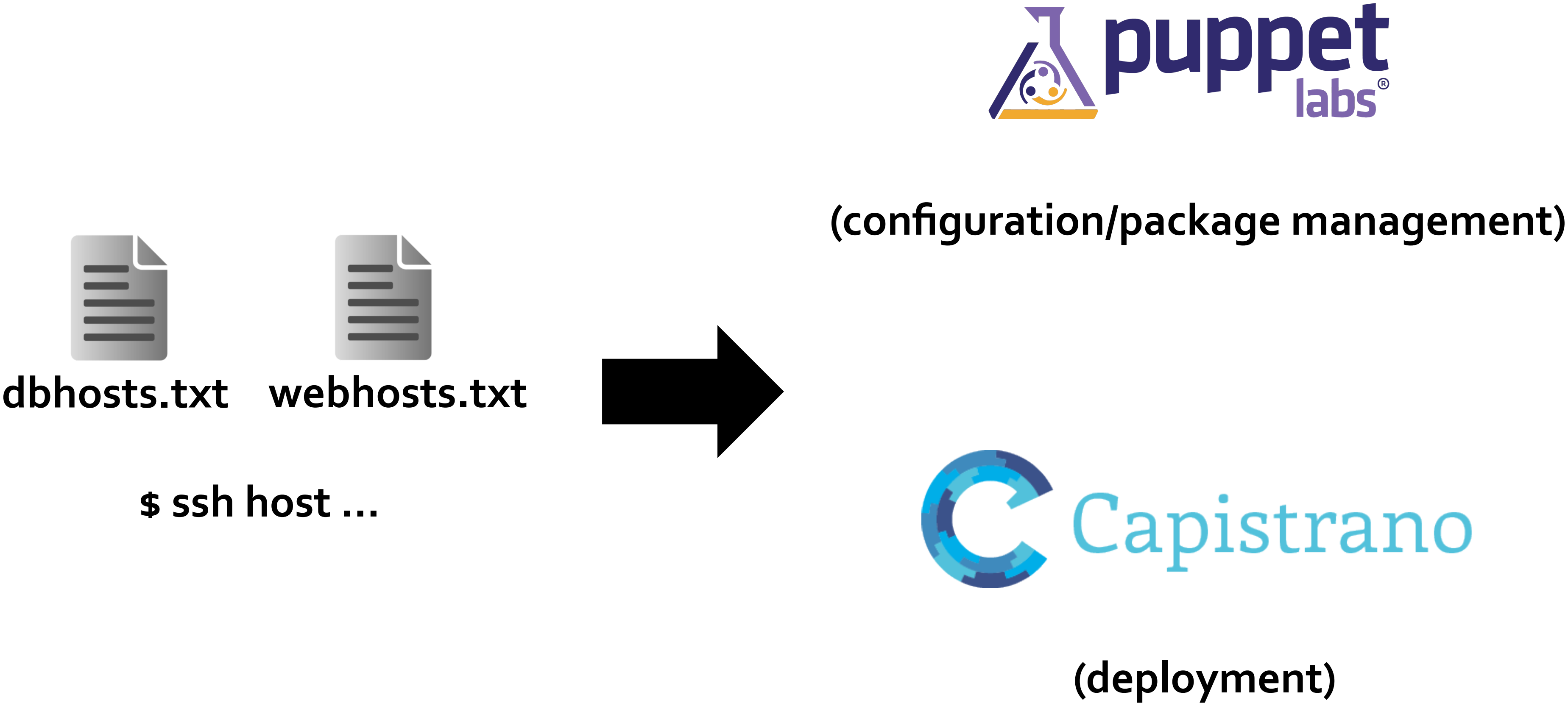
to scale,
need less moving parts,
more automation



(10’s of machines)

(100’s -> 1000’s of machines)

Twitter, circa 2010

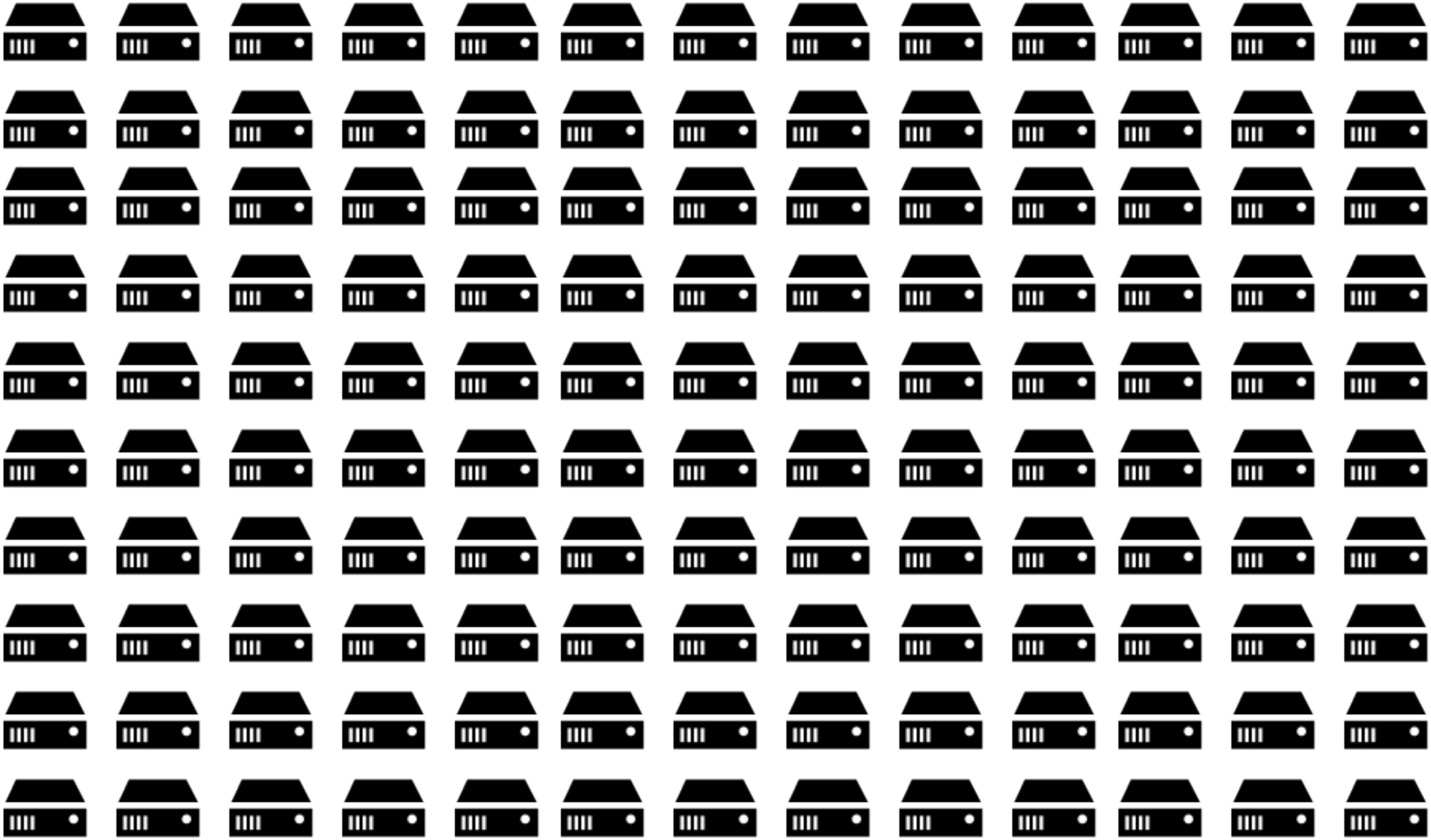


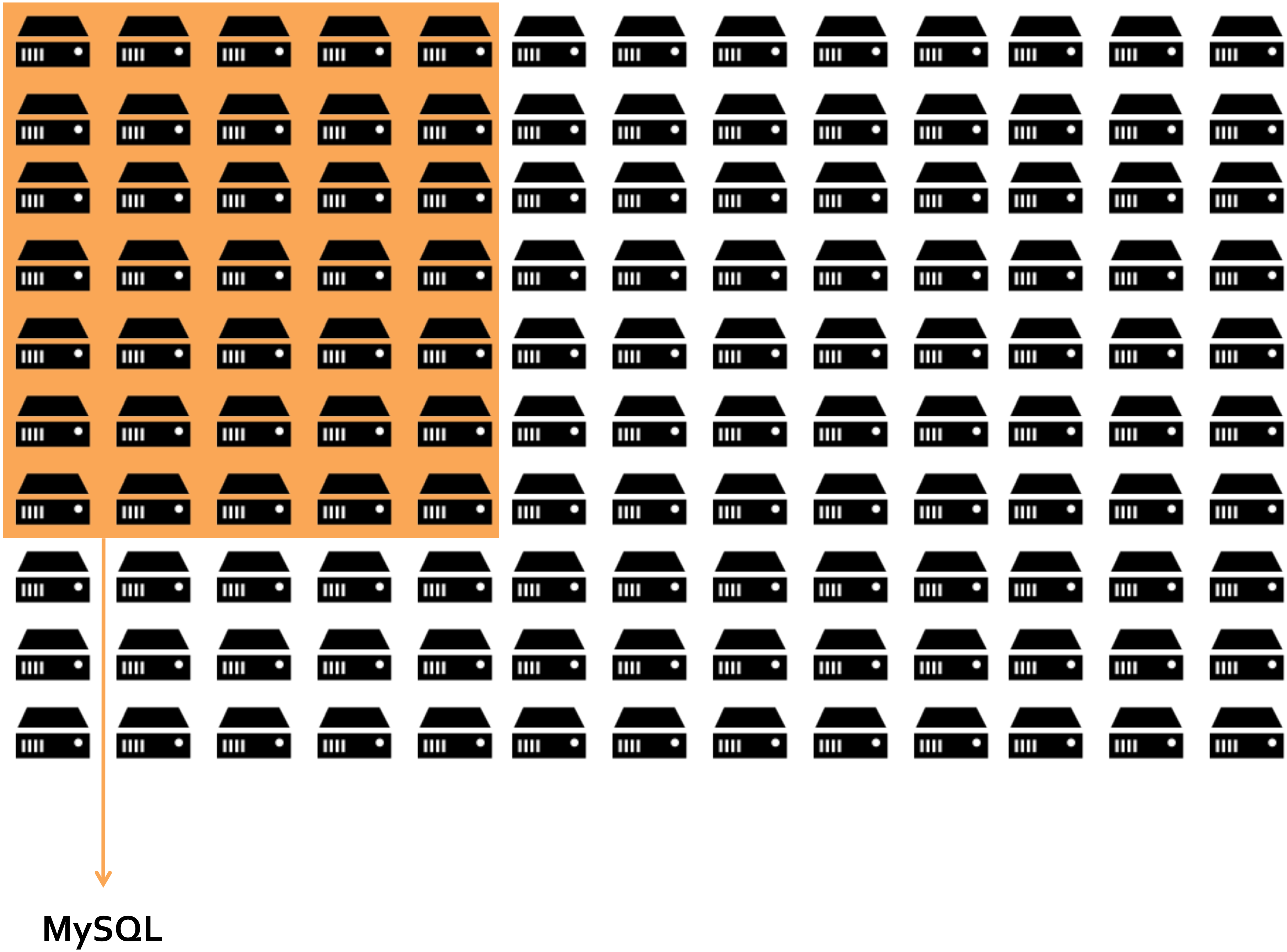
dbhosts.txt webhosts.txt

$ ssh host …

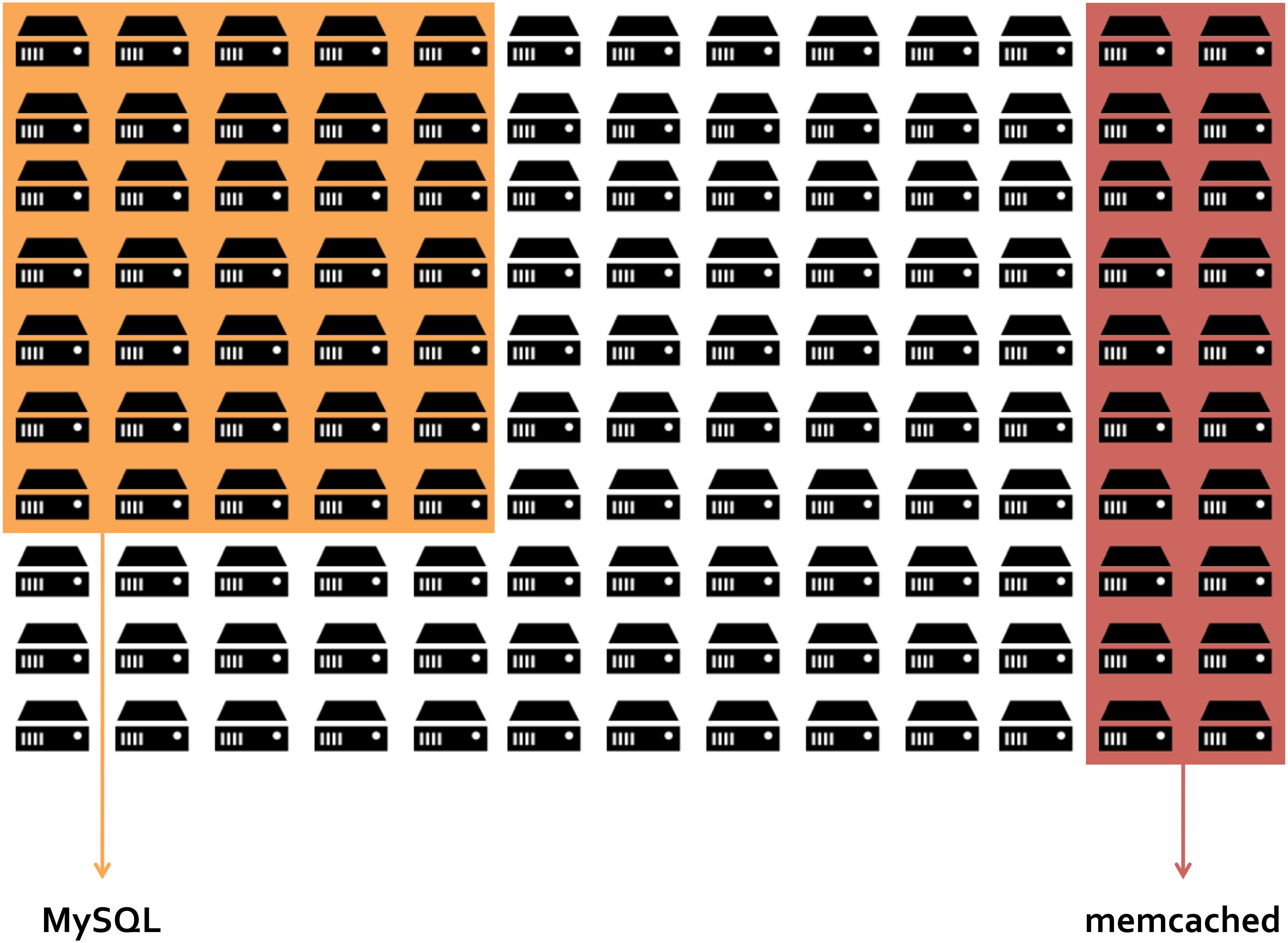
(configuration/package management)

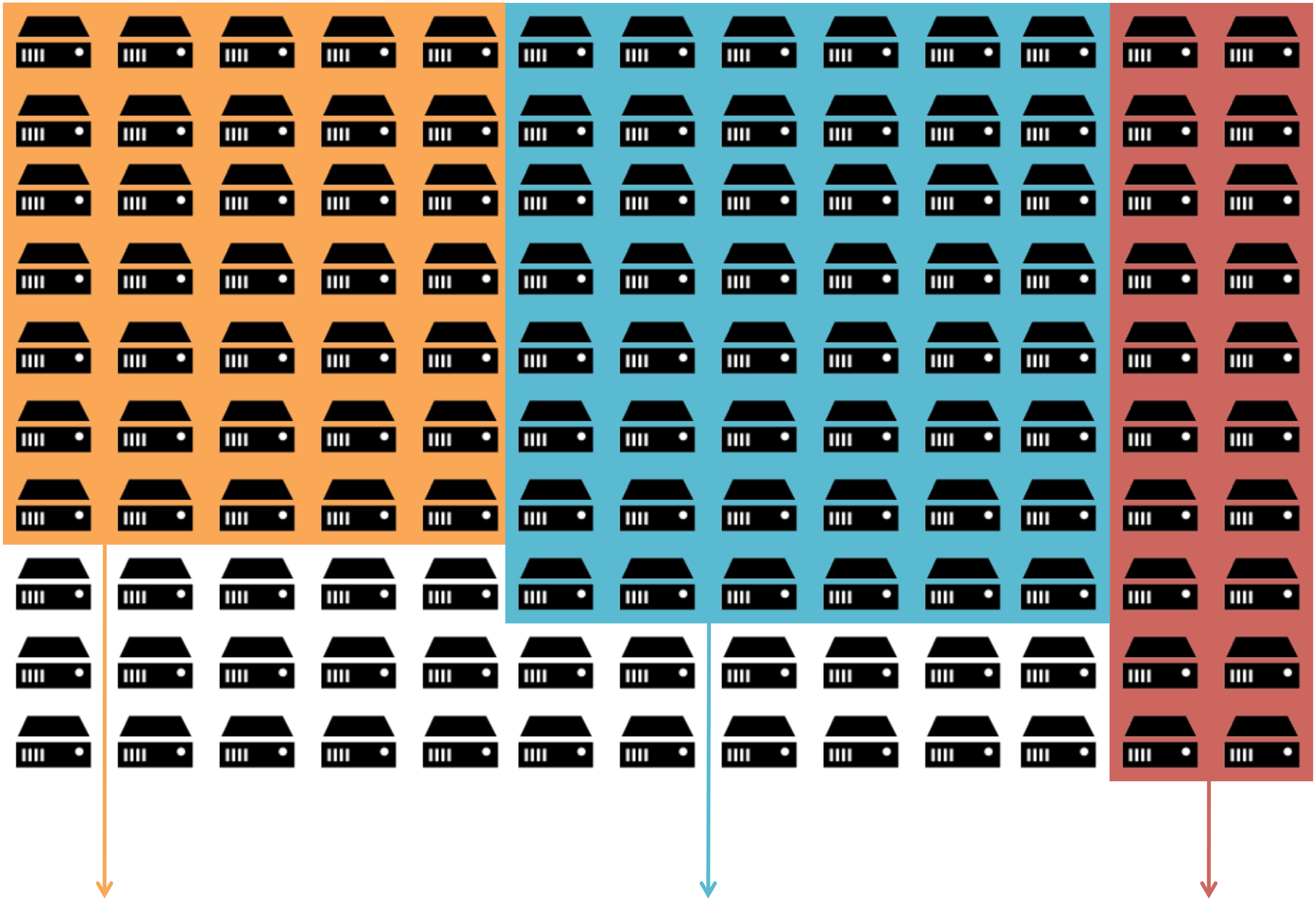
(deployment)



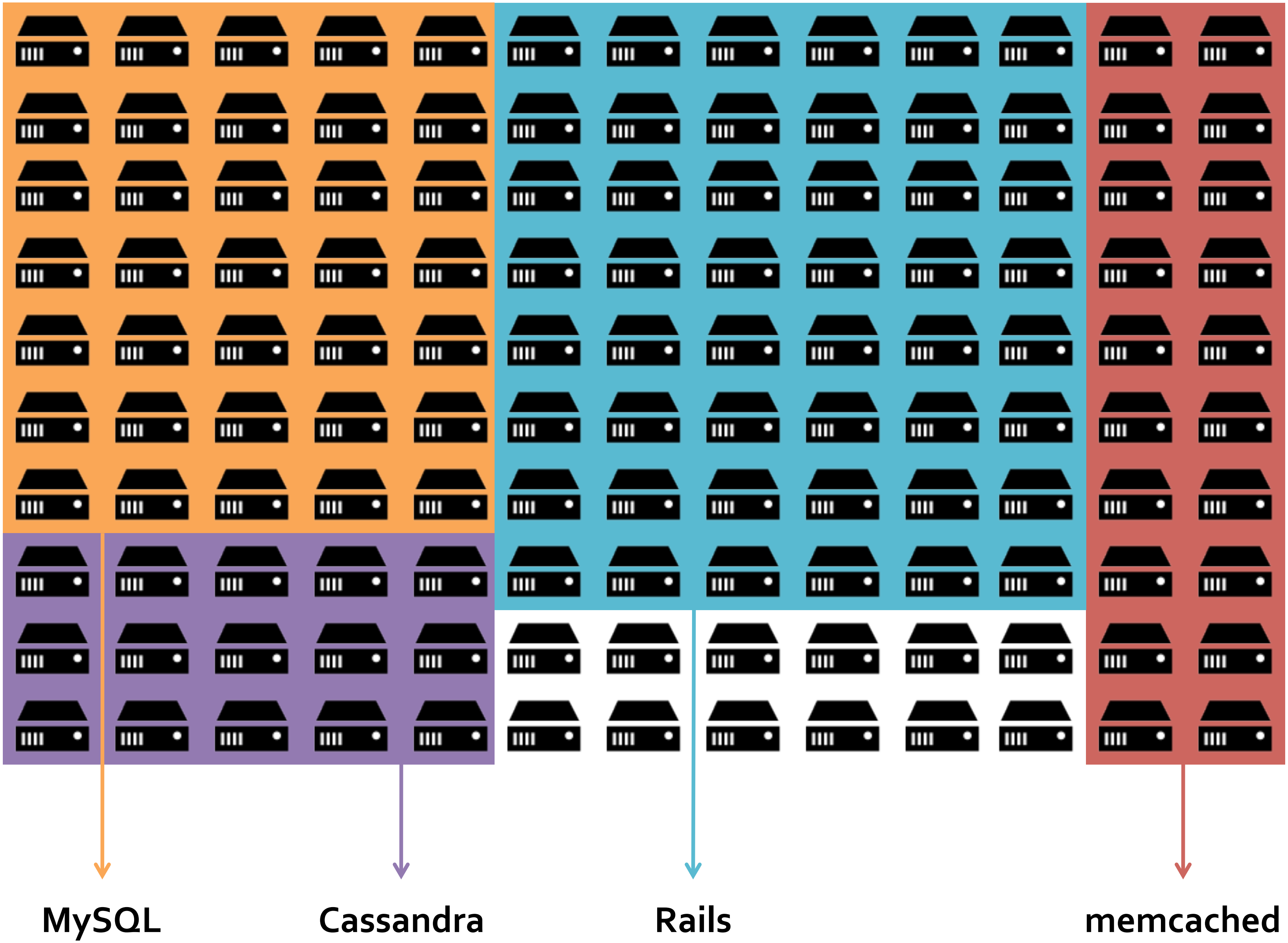


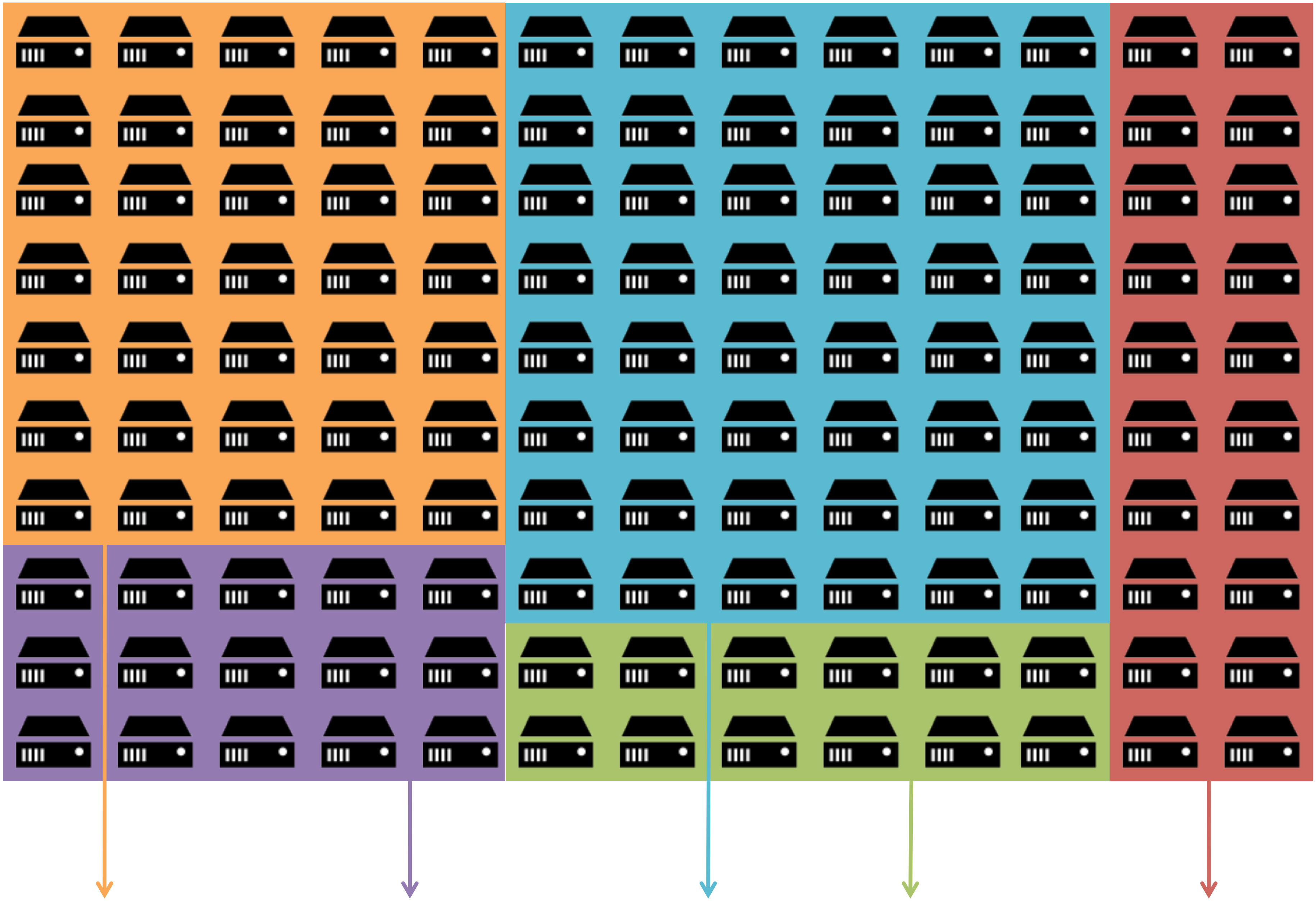
MySQL


MySQL
memcached



MySQL
Rails
memcached


MySQL
Cassandra
Rails
memcached



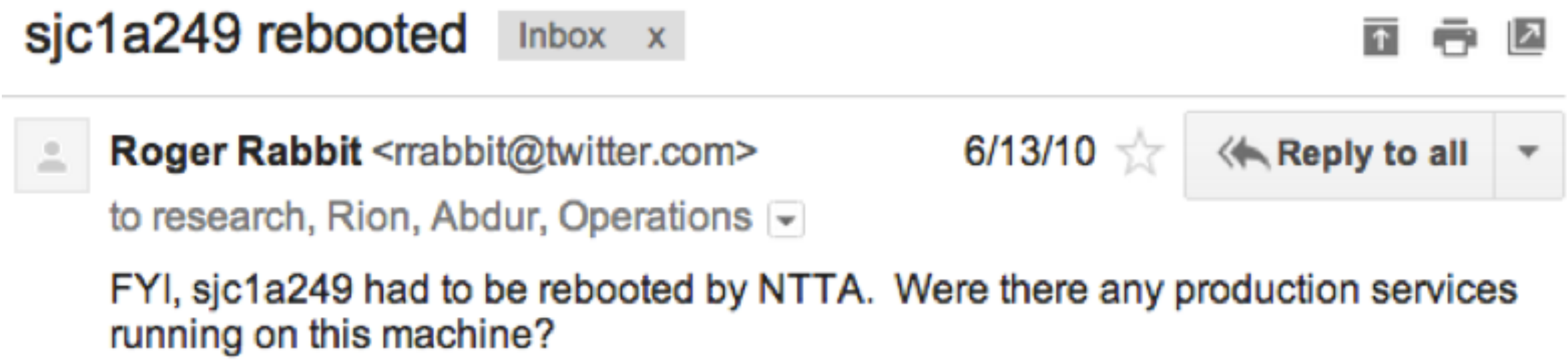
MySQL
Cassandra
Rails
Hadoop
memcached

challenges

challenges

① failures

failures

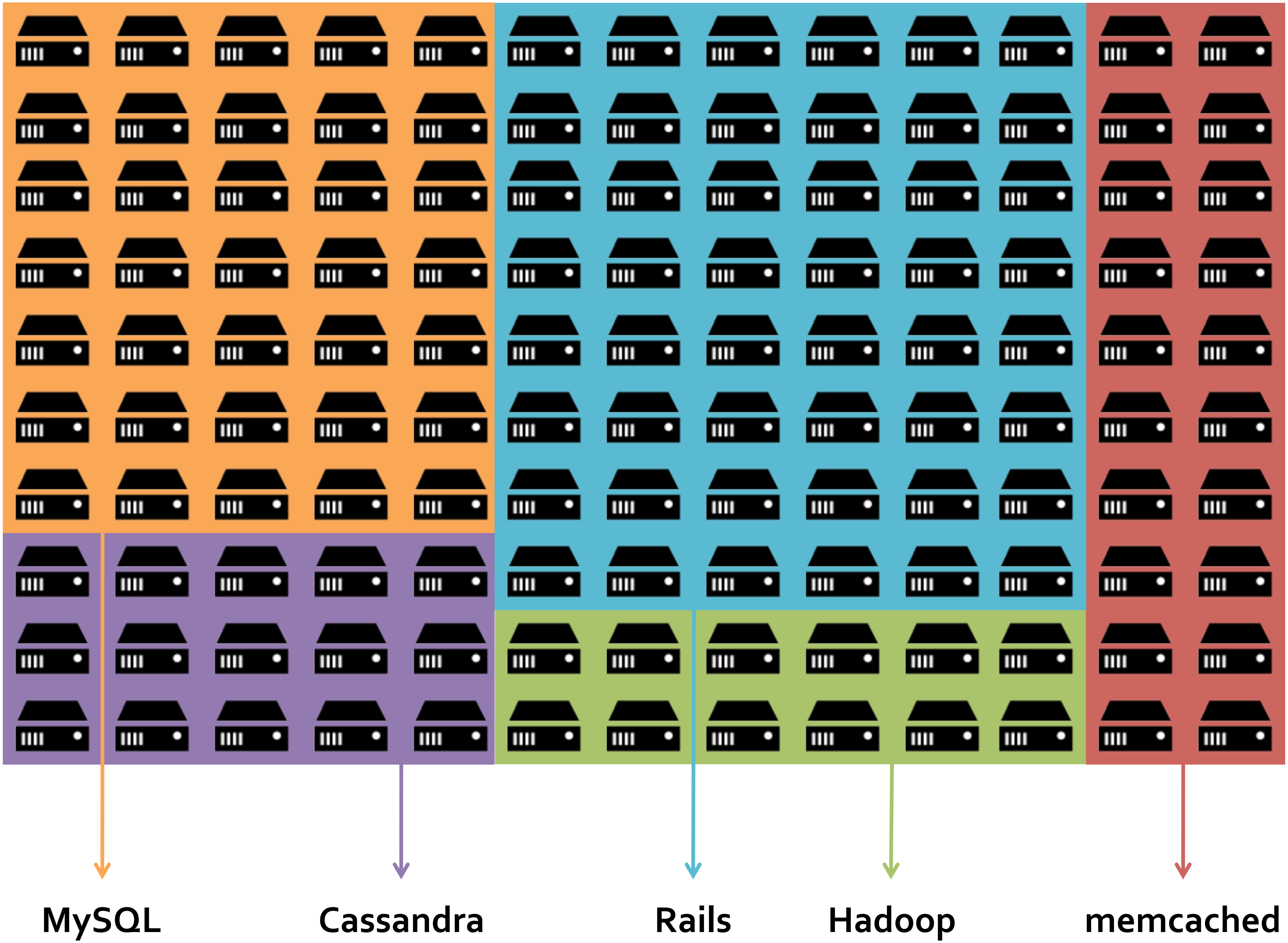


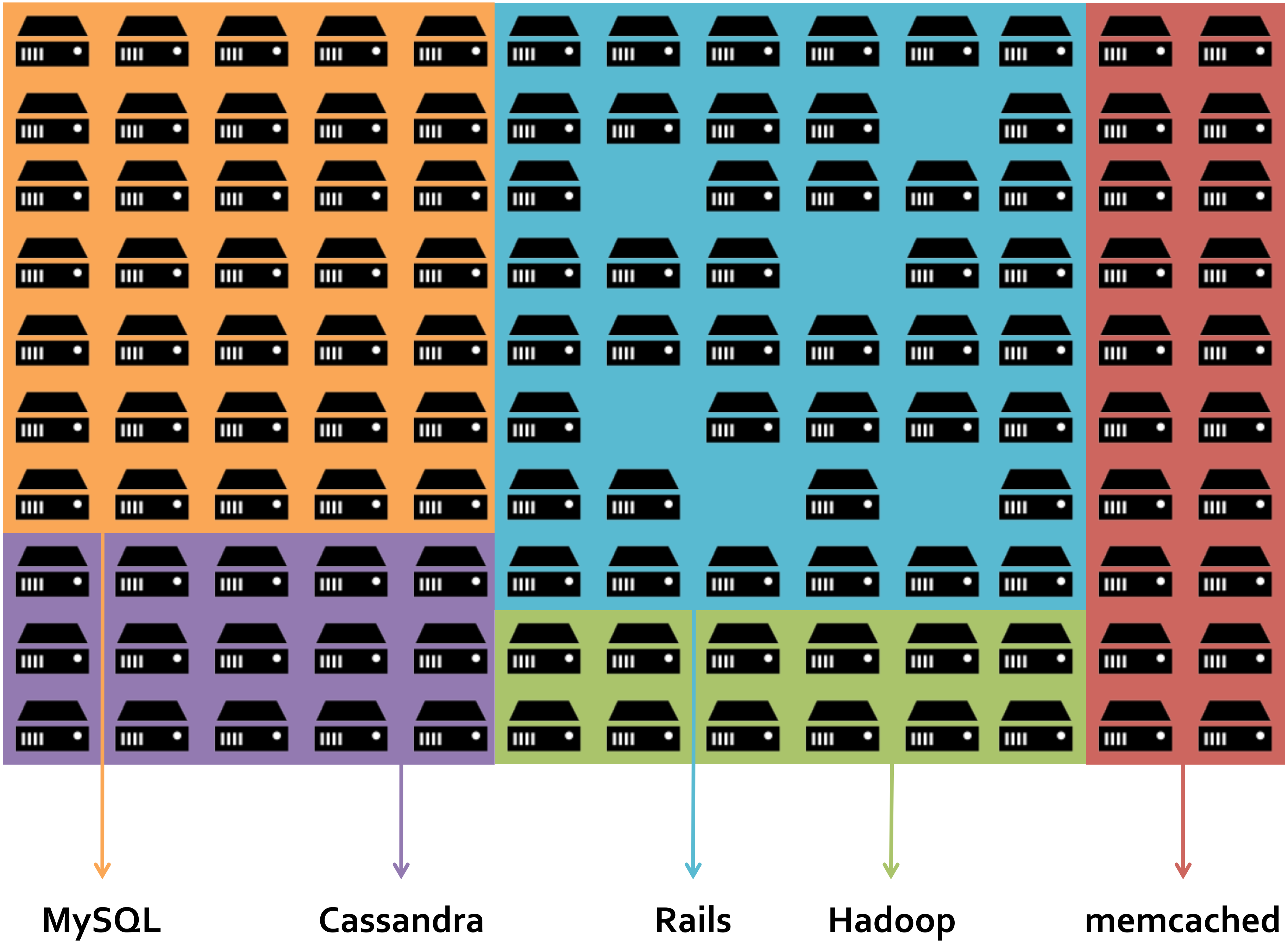
sjc1a249 rebooted

Roger Rabbit <rrabbit@twitter.com> 6/13/10

to research, Rion, Abdur, Operations

FYI, sjc1a249 had to be rebooted by NTTA. Were there any production services running on this machine?


MySQL
Cassandra
Rails
Hadoop
memcached


MySQL
Cassandra
Rails
Hadoop
memcached

types of failure: fault domains

machine
(disk, memory, CPU, etc)

rack
(switch, PDU)

datacenter

challenges

② maintenance

(aka “planned failures”)

maintenance

① upgrading software (i.e., the kernel)

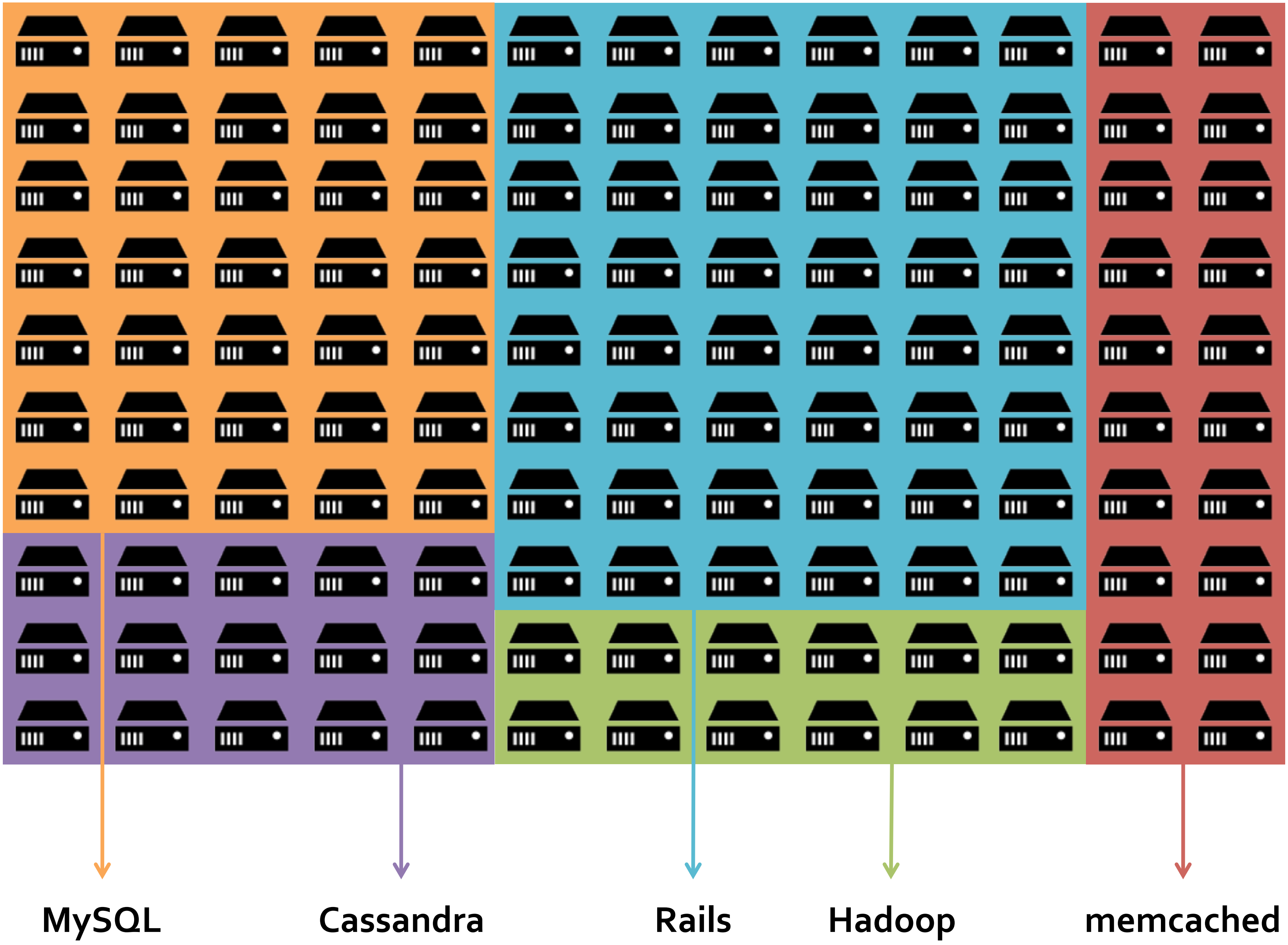
developers

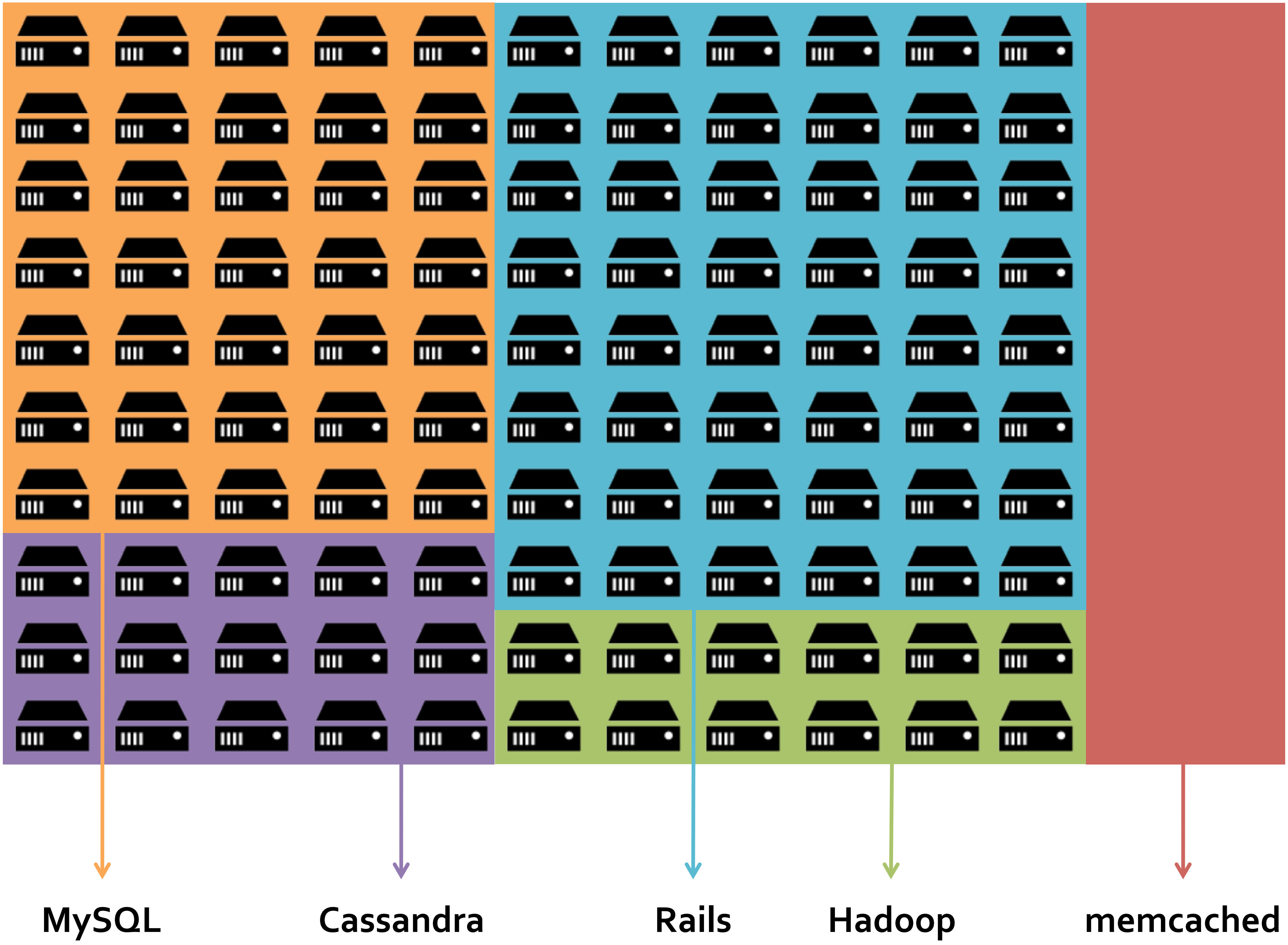
ops

maintenance

① upgrading software (i.e., the kernel)

② replacing machines, switches, PDUs, etc

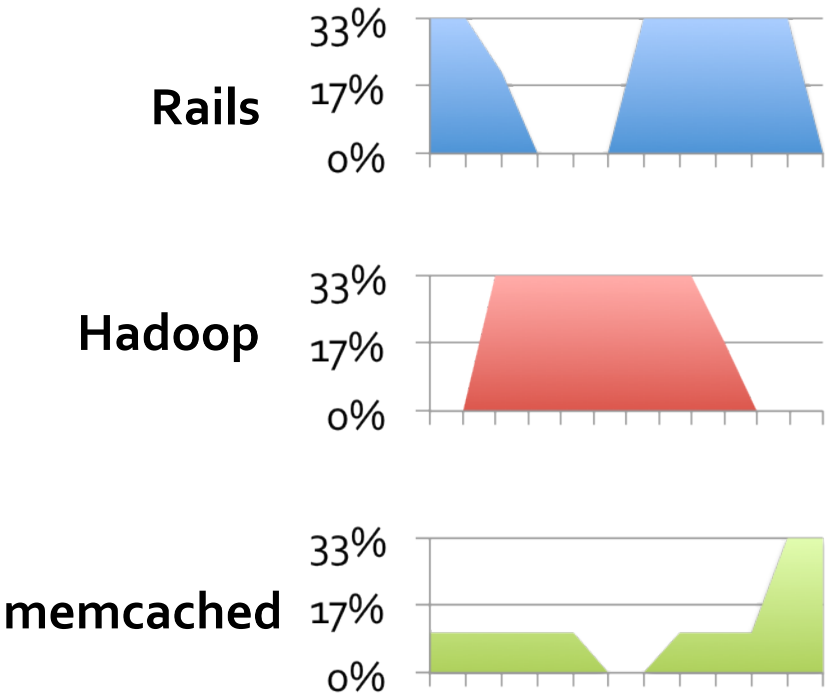

MySQL
Cassandra
Rails
Hadoop
memcached


MySQL
Cassandra
Rails
Hadoop
memcached

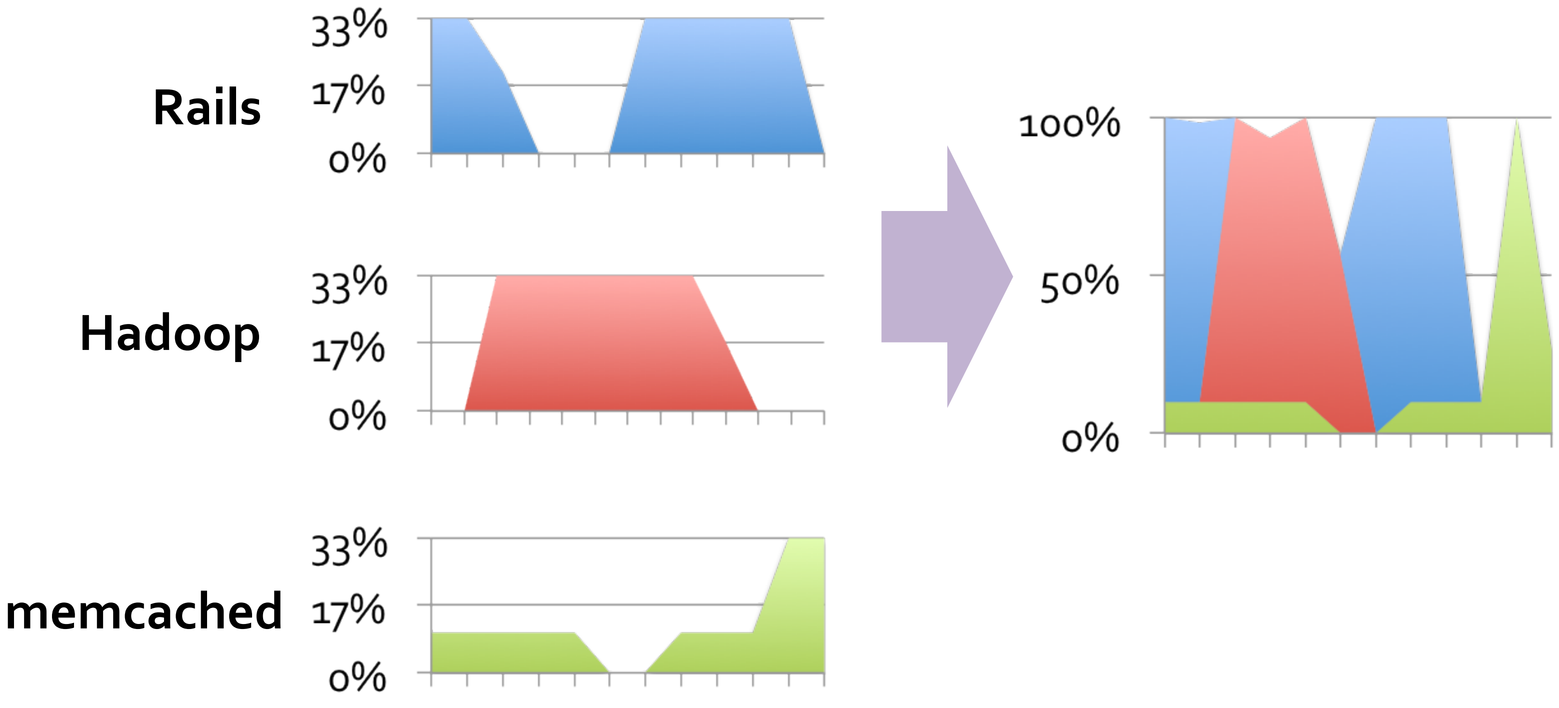
challenges

③ utilization

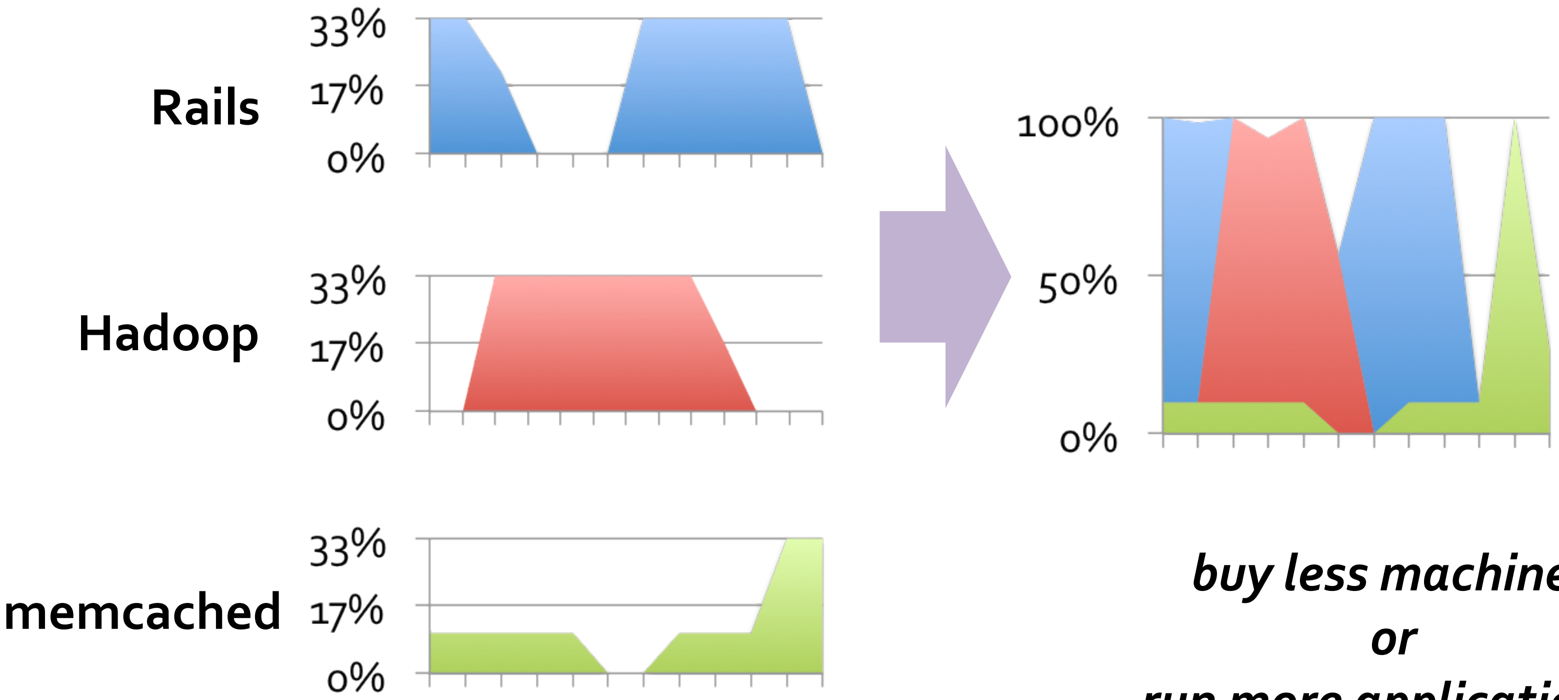
utilization



utilization



utilization



buy less machines
or
run more applications!

challenges

① failures

② maintenance

③ utilization

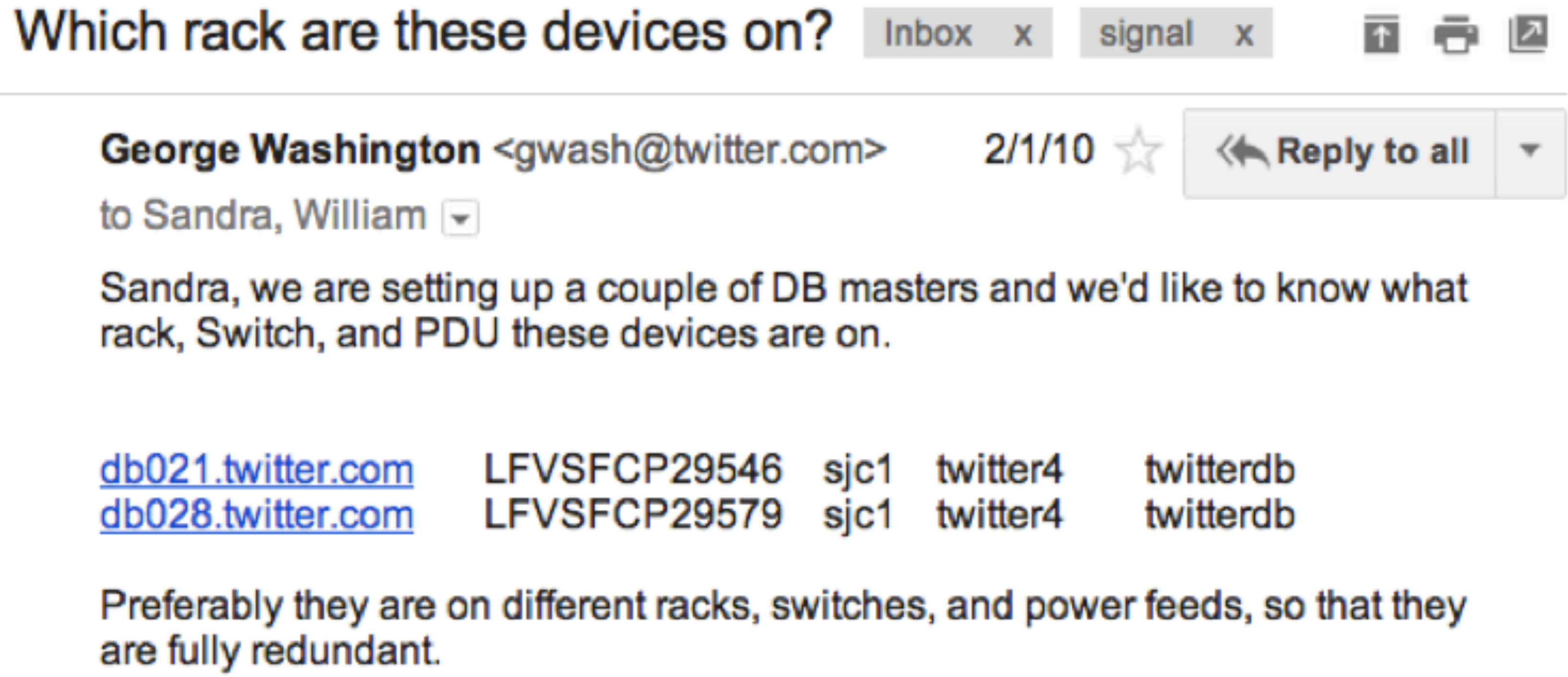
challenges

① failures

② maintenance

③ utilization

planning for failure



Which rack are these devices on?

George Washington <gwash@twitter.com> 2/1/10

to Sandra, William

Sandra, we are setting up a couple of DB masters and we'd like to know what rack, Switch, and PDU these devices are on.

db021.twitter.com	LFVSFCP29546	sjc1	twitter4	twitterdb
db028.twitter.com	LFVSFCP29579	sjc1	twitter4	twitterdb

Preferably they are on different racks, switches, and power feeds, so that they are fully redundant.

challenges

① failures

② maintenance

③ utilization

planning for utilization

intra-machine resource sharing:

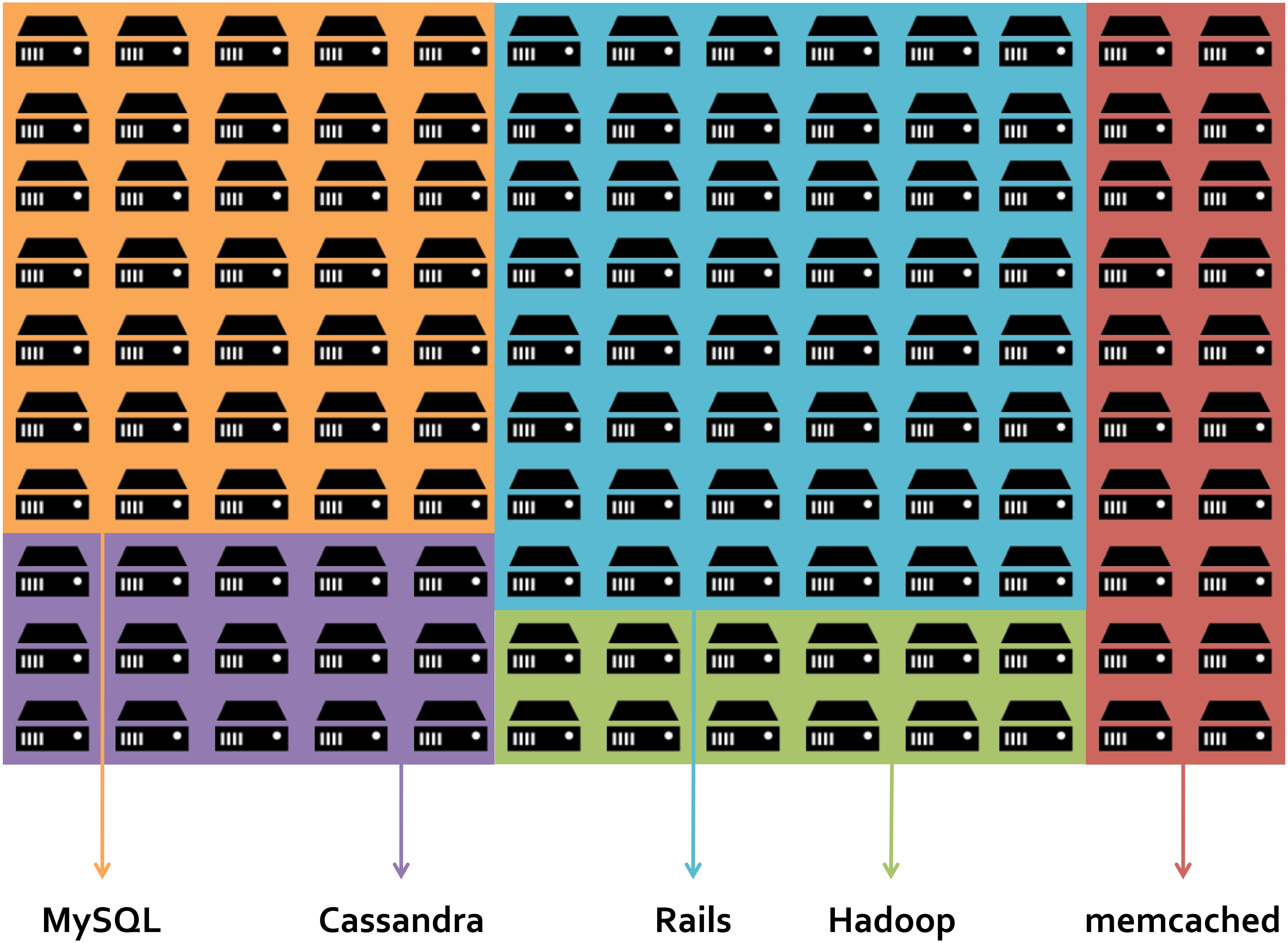
share a single machine's resources between multiple applications (multi-tenancy)

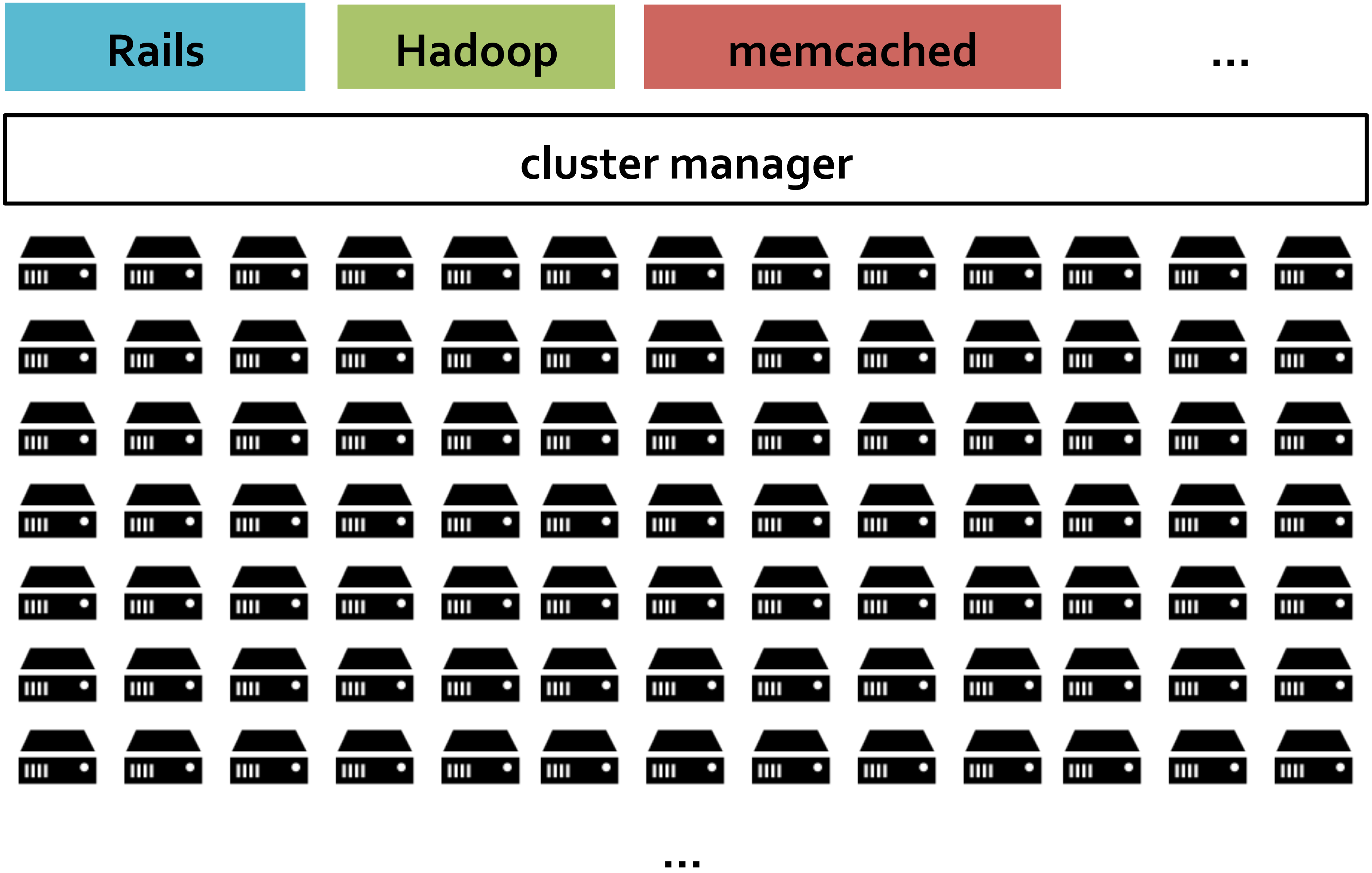
intra-datacenter resource sharing:

share multiple machine's resources between multiple applications

Twitter, circa 2010

cluster manager provides a level-of-indirection between hardware resources (machines) and applications/jobs


MySQL
Cassandra
Rails
Hadoop
memcached



Rails

Hadoop

memcached

…

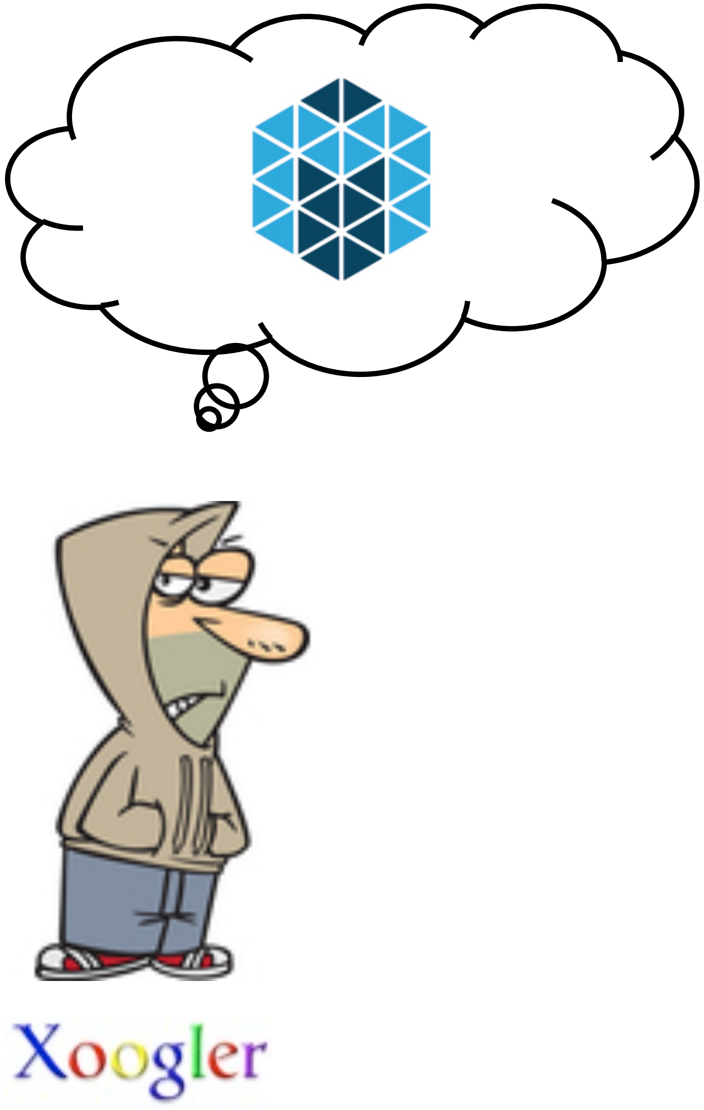
cluster manager

…

Twitter, circa 2010

Twitter, circa 2010

Twitter, circa 2010



Berkeley
UNIVERSITY OF CALIFORNIA

Apache Mesos is a modern *general purpose* cluster manager (i.e., not just focused on batch scheduling)

cluster management

academia

industry

different software

academia

- MPI (Message Passing Interface)

industry

- Apache (mod_perl, mod_php)
- web services (Java, Ruby, ...)

different scale (at first)

academia

- 100’s of machines

industry

- 10’s of machines

cluster management

academia

- PBS (Portable Batch System)
- TORQUE
- SGE (Sun Grid Engine)

cluster managers

industry

- ssh
- Puppet/Chef
- Capistrano/Ansible

different scale (converging)

academia

- 100’s of machines

industry

- 10’s of machines

1,000’s of machines

cluster management

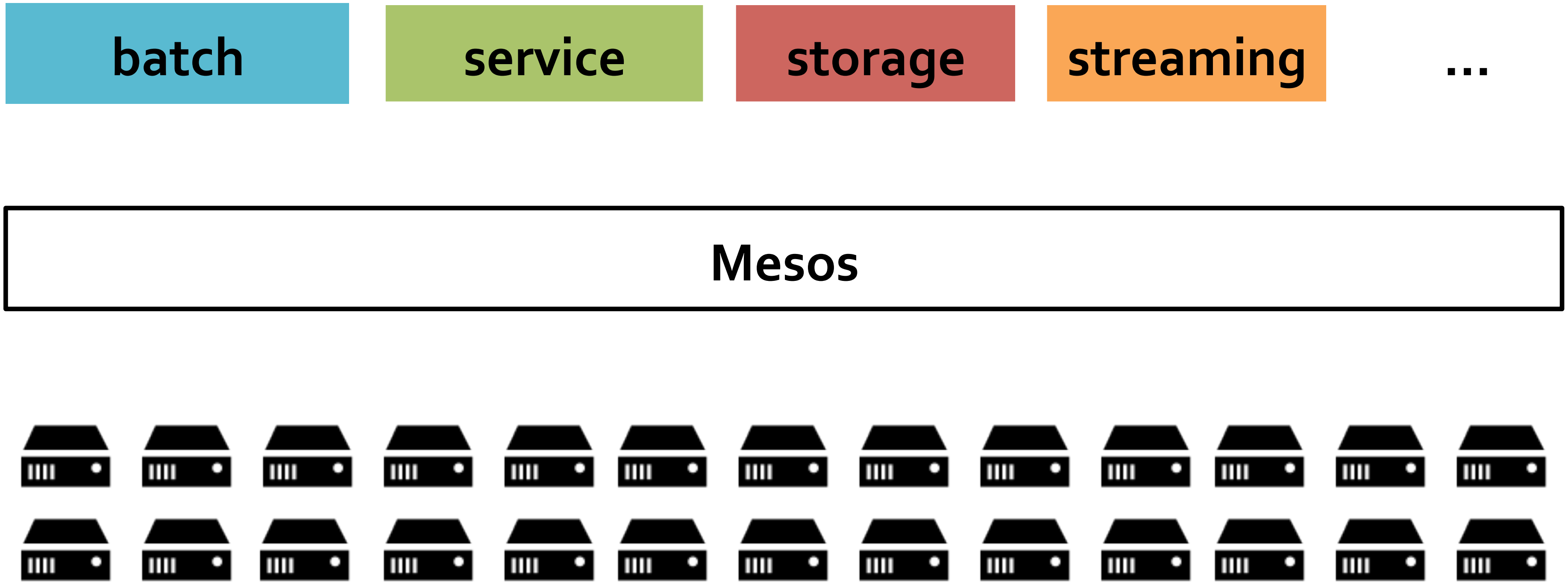
academia

- PBS (Portable Batch System)
- TORQUE
- SGE (Sun Grid Engine)

batch computation!

industry

- ssh
- Puppet/Chef
- Capistrano/Ansible



batch

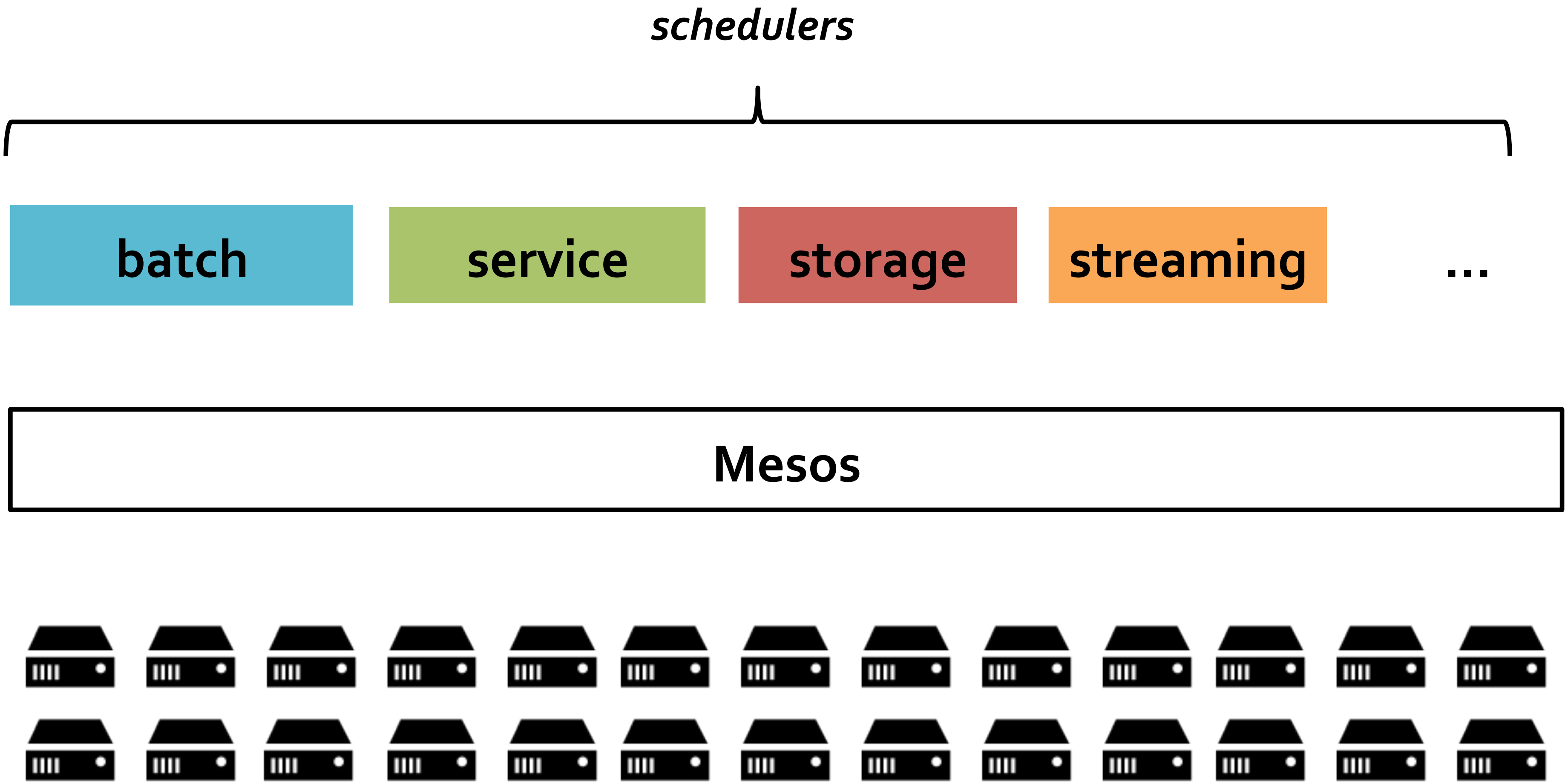
service

storage

streaming

...

Mesos



schedulers

batch

service

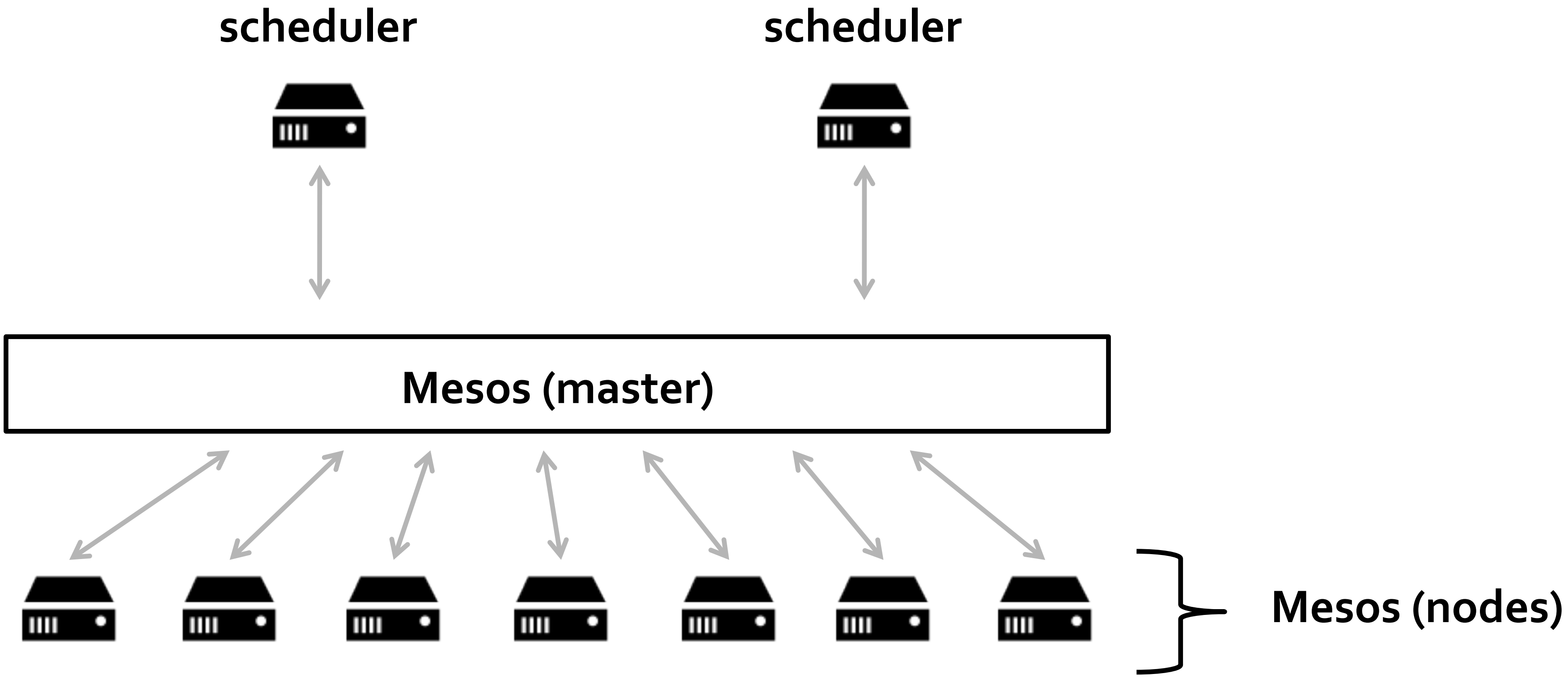
storage

streaming

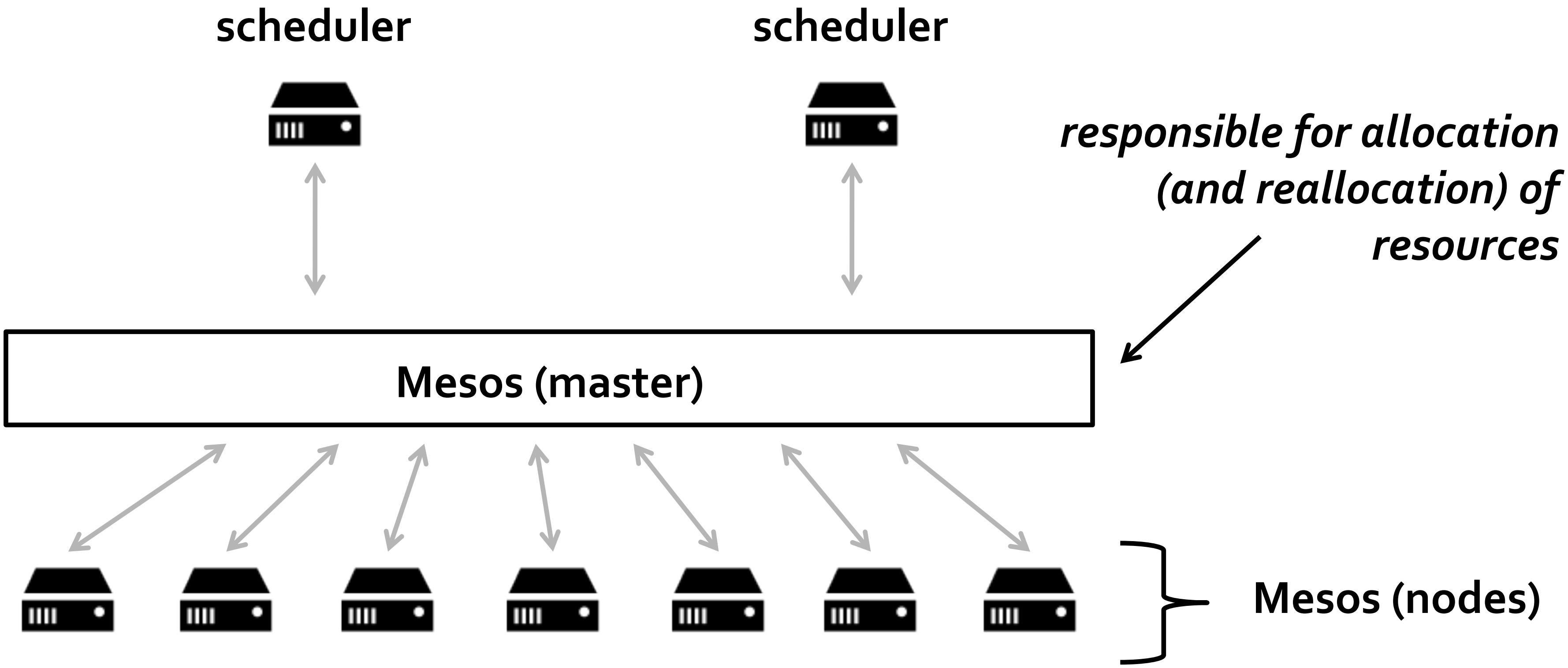
...

Mesos

Mesos: level of indirection



Mesos: level of indirection



scheduler

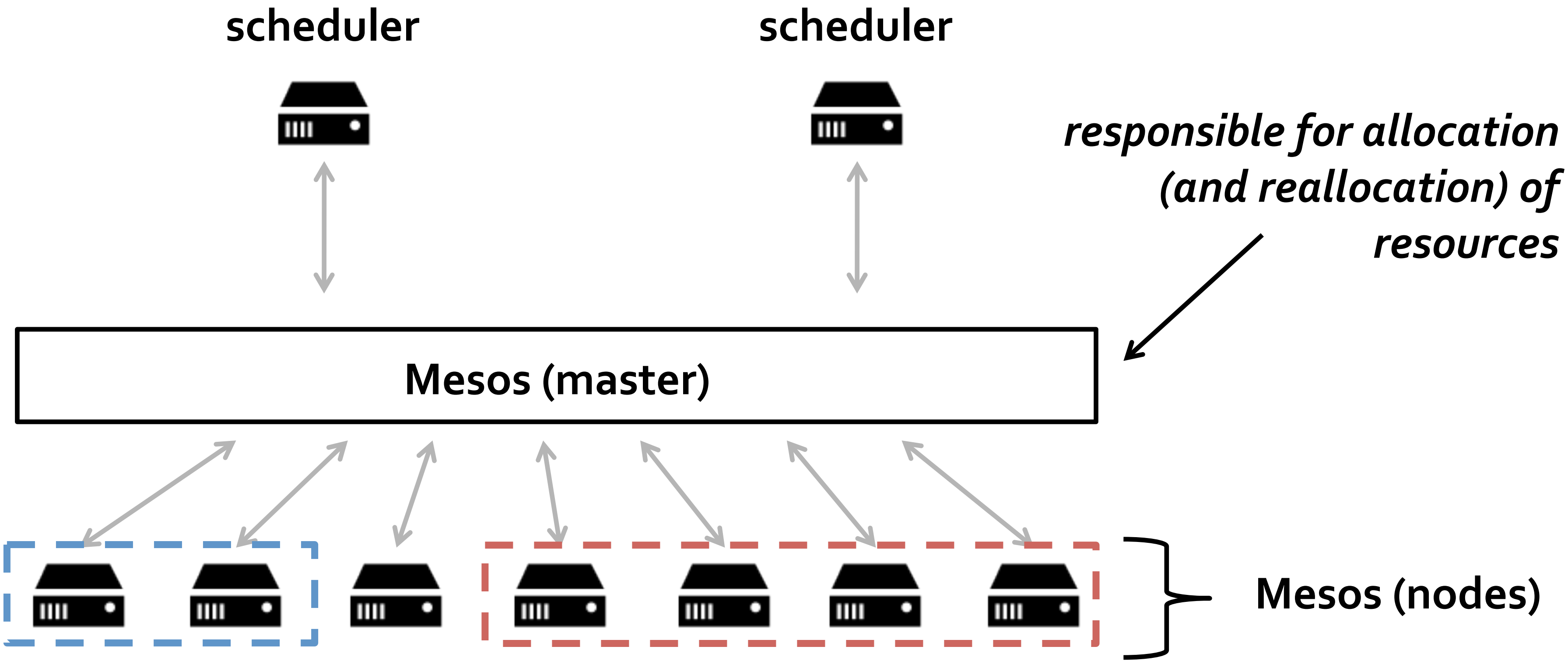
scheduler

responsible for allocation (and reallocation) of resources

Mesos (master)

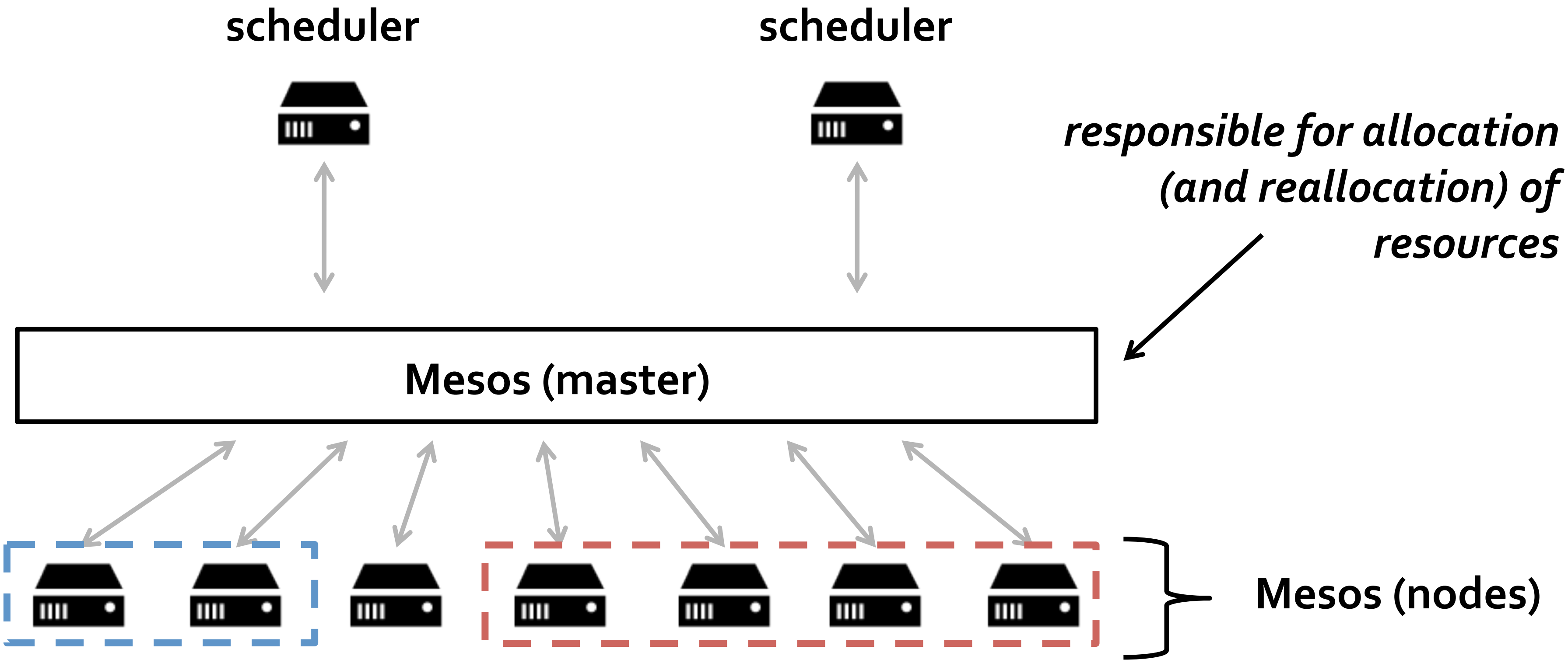
Mesos (nodes)

Mesos: level of indirection



responsible for allocation (and reallocation) of resources

challenges: failures/maintenance



scheduler

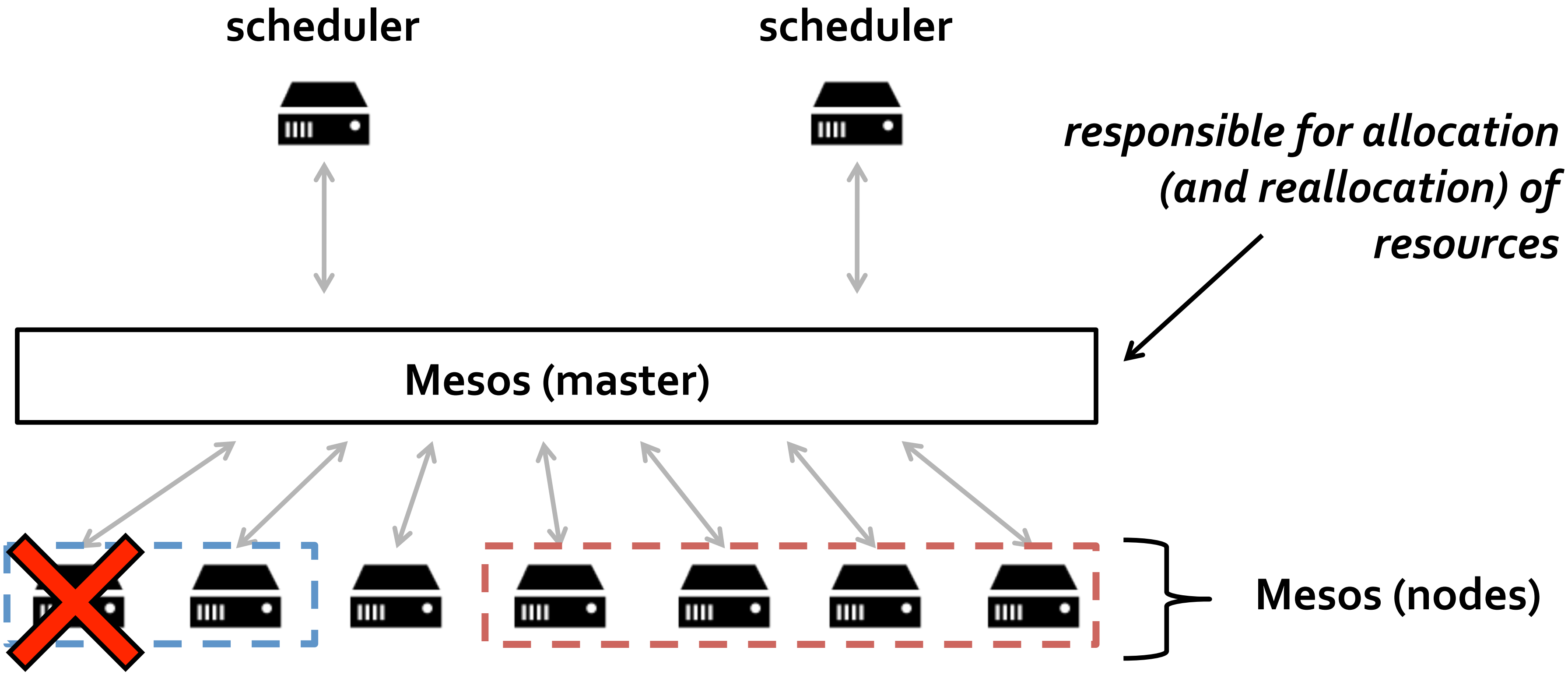
scheduler

responsible for allocation (and reallocation) of resources

Mesos (master)

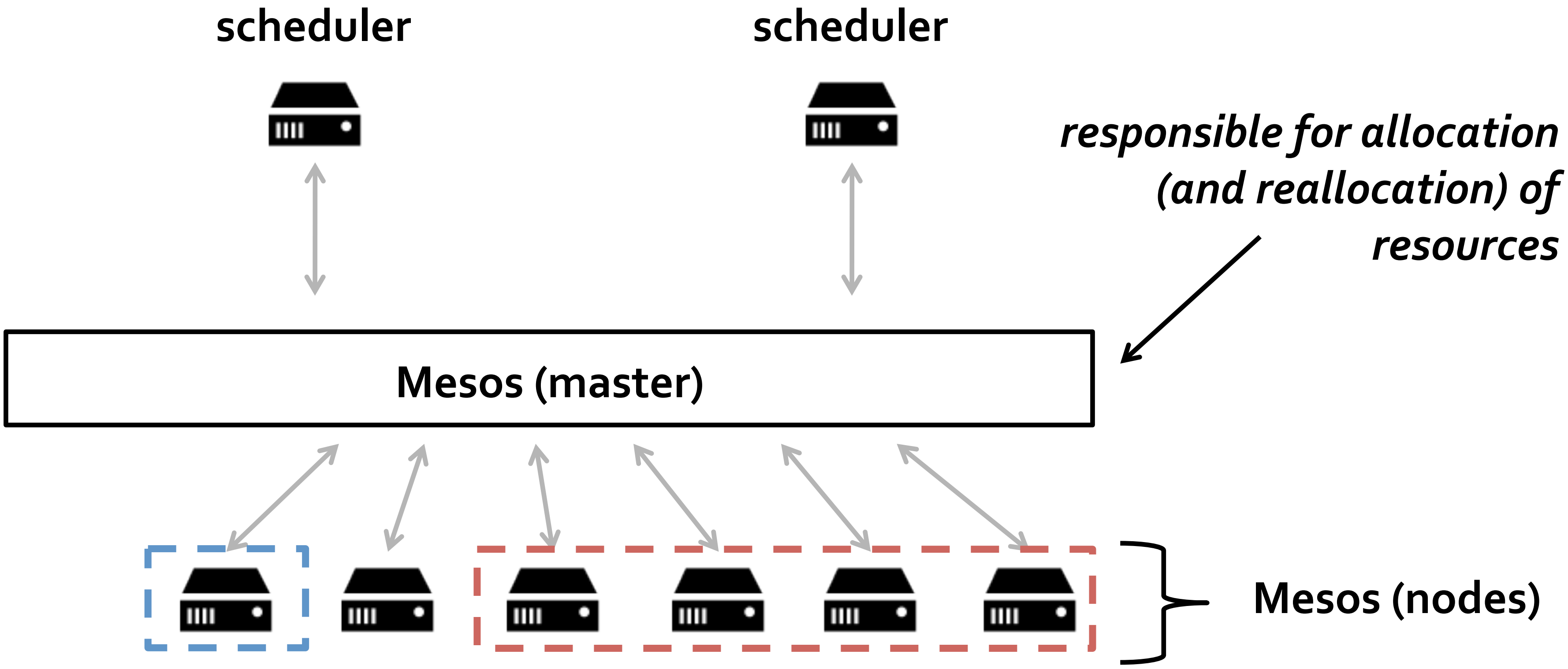
Mesos (nodes)

challenges: failures/maintenance



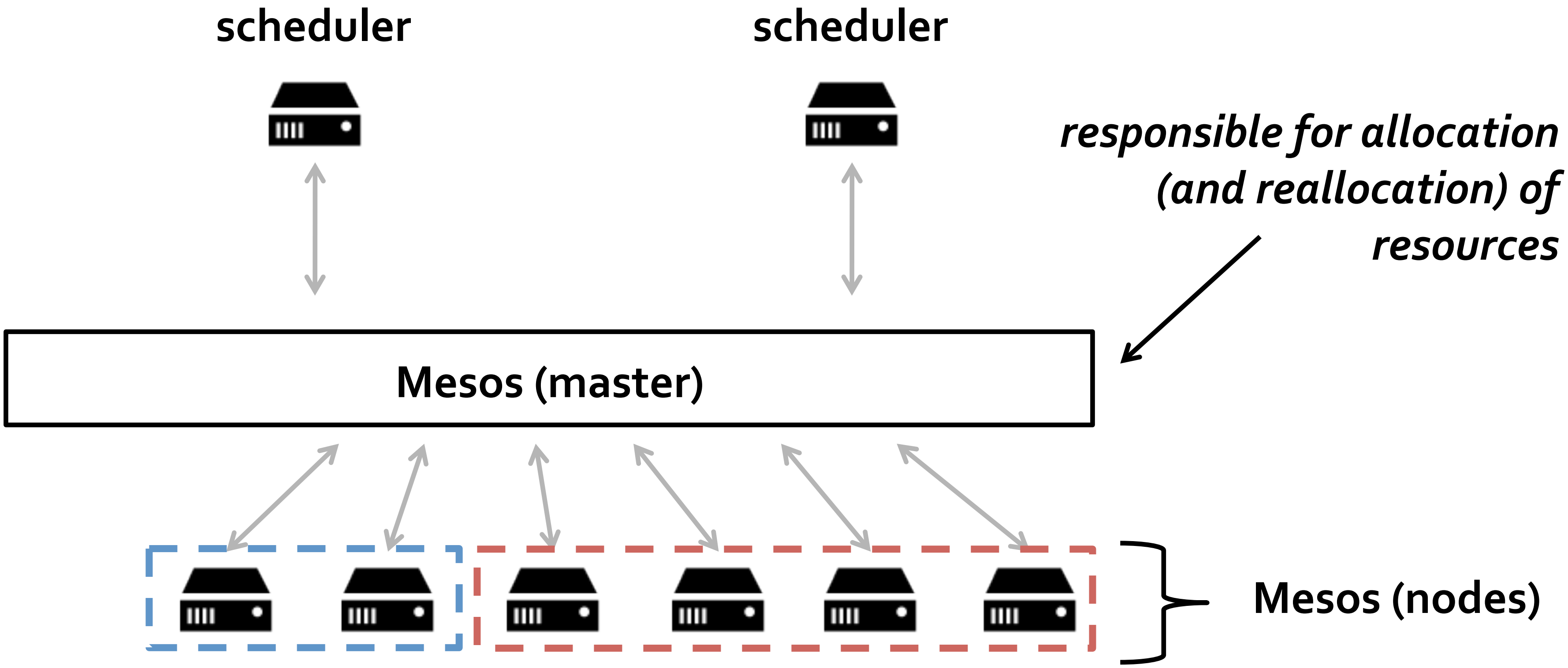
responsible for allocation (and reallocation) of resources

challenges: failures/maintenance

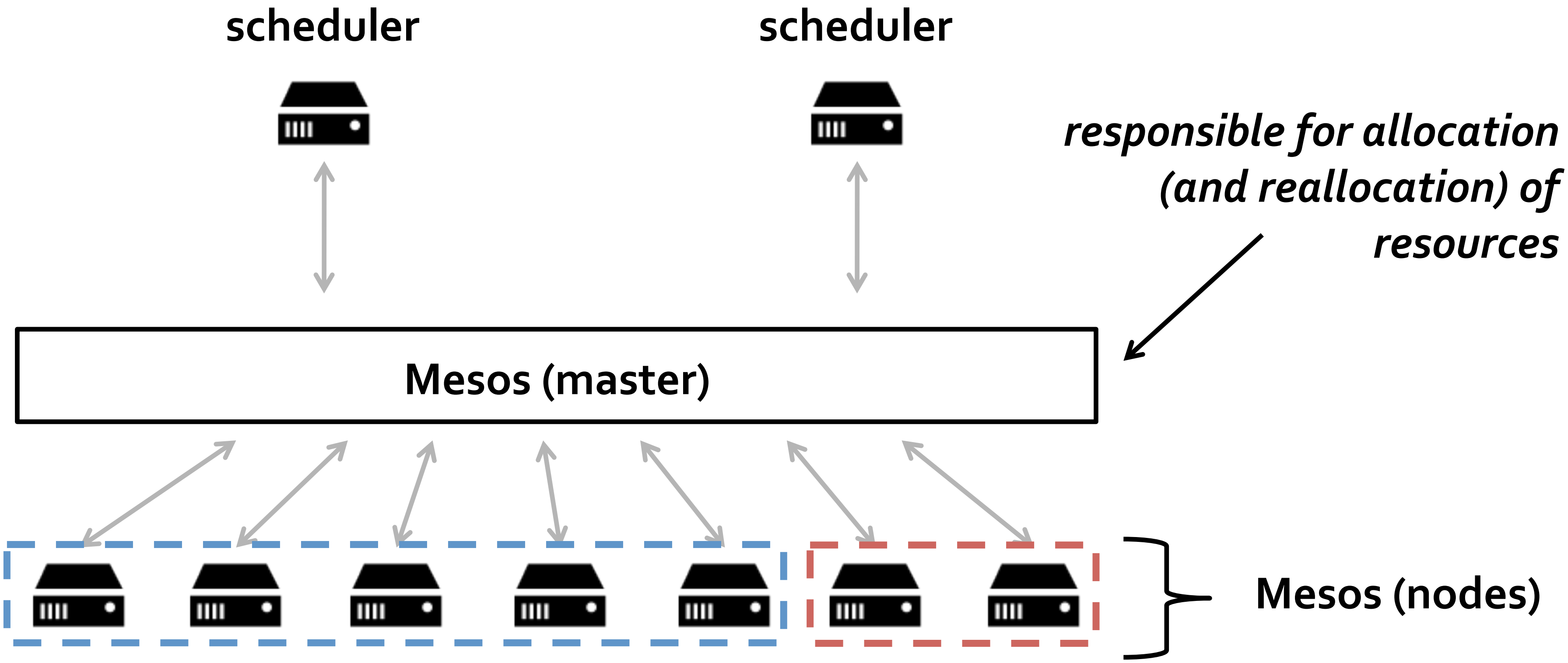


responsible for allocation (and reallocation) of resources

challenges: failures/maintenance

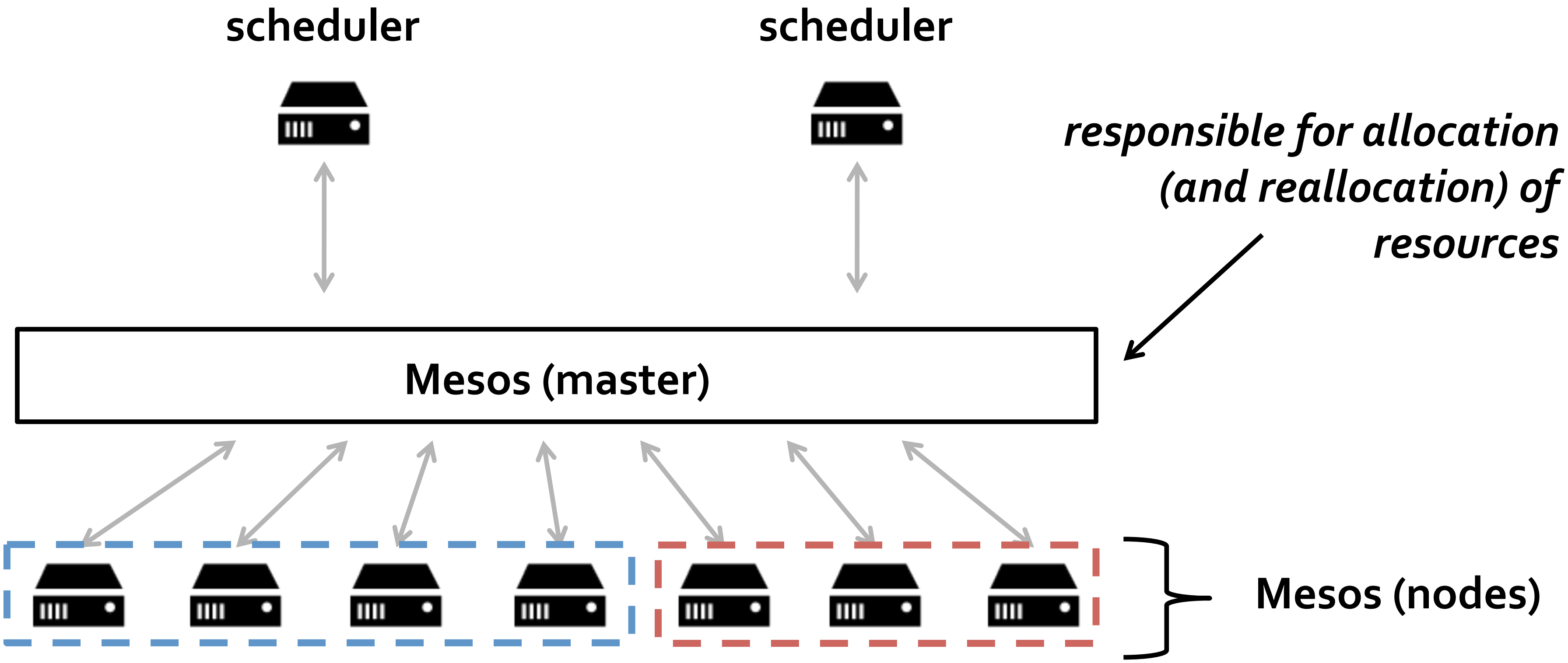

scheduler
scheduler
responsible for allocation (and reallocation) of resources
Mesos (master)
Mesos (nodes)

challenges: utilization

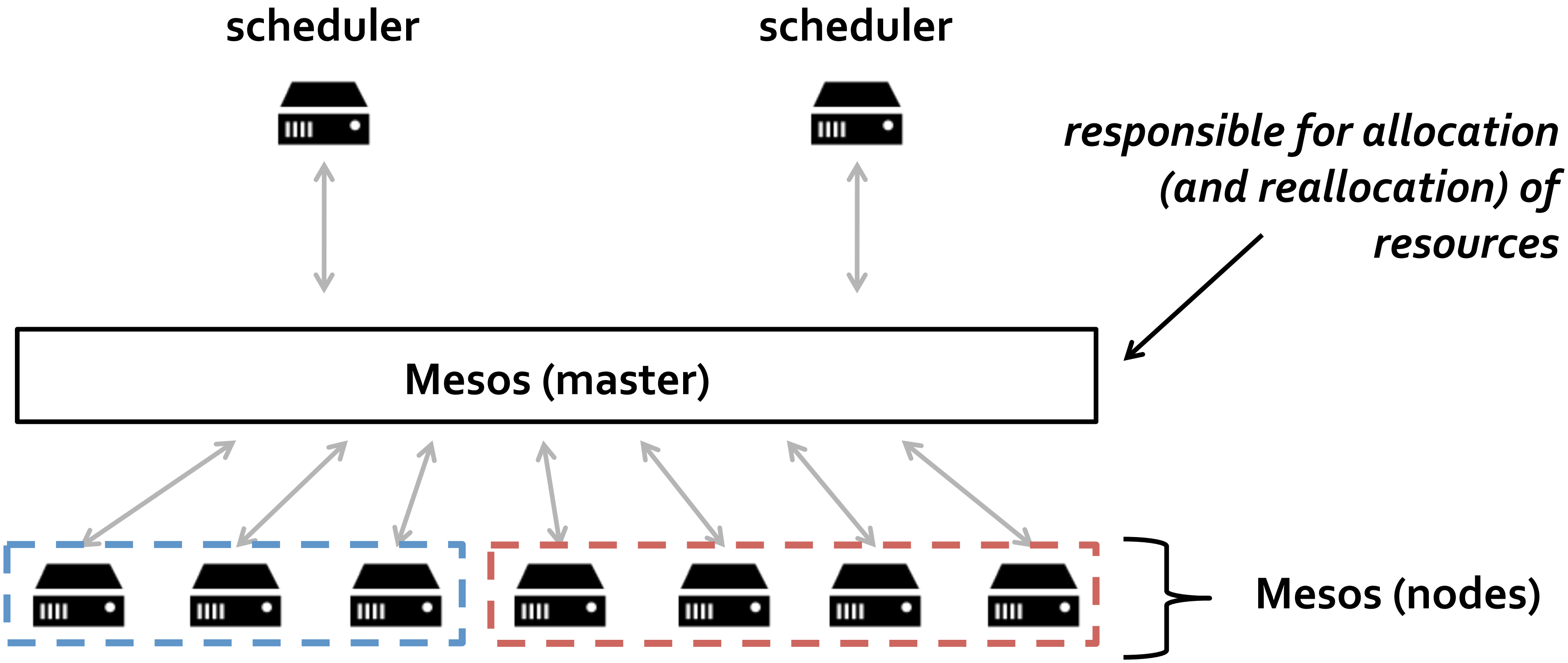


responsible for allocation (and reallocation) of resources

challenges: utilization


scheduler
scheduler
responsible for allocation (and reallocation) of resources
Mesos (master)
Mesos (nodes)

challenges: utilization



scheduler

scheduler

responsible for allocation (and reallocation) of resources

Mesos (master)

Mesos (nodes)

two-level scheduling

Mesos influenced by operating system supported *user-space scheduling* (and scheduler activations)

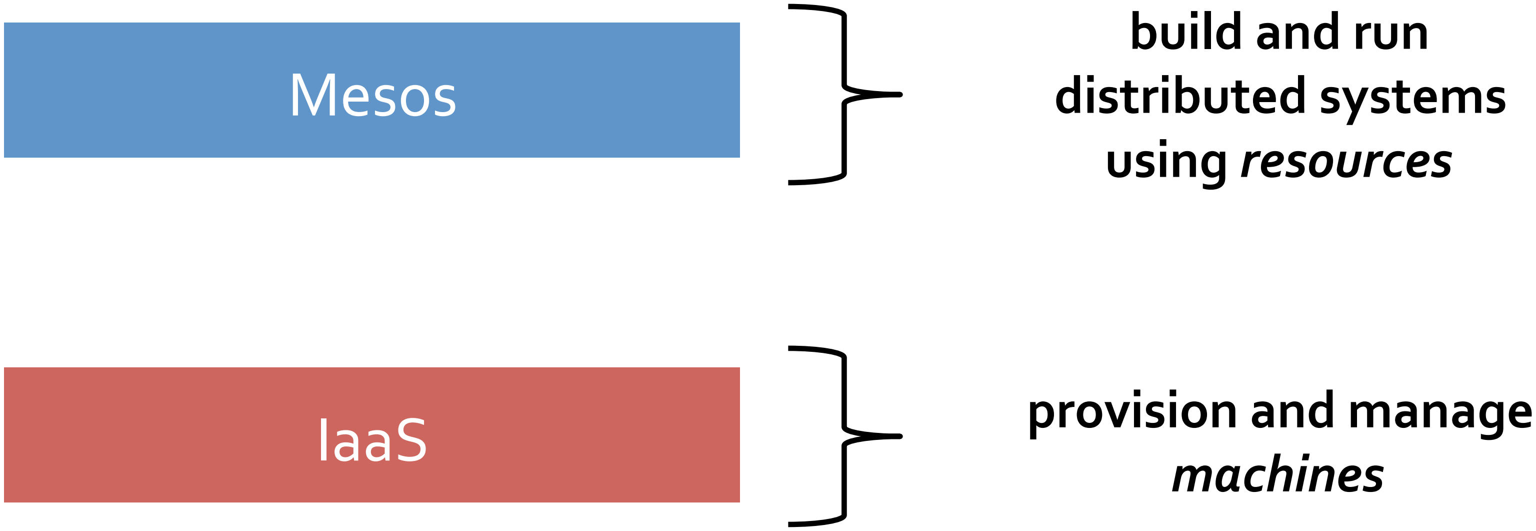
Mesos is designed less like a "cluster manager" and more like an operating system kernel

Mesos: level of abstraction

Mesos

build and run
distributed systems
using *resources*

Mesos: level of abstraction



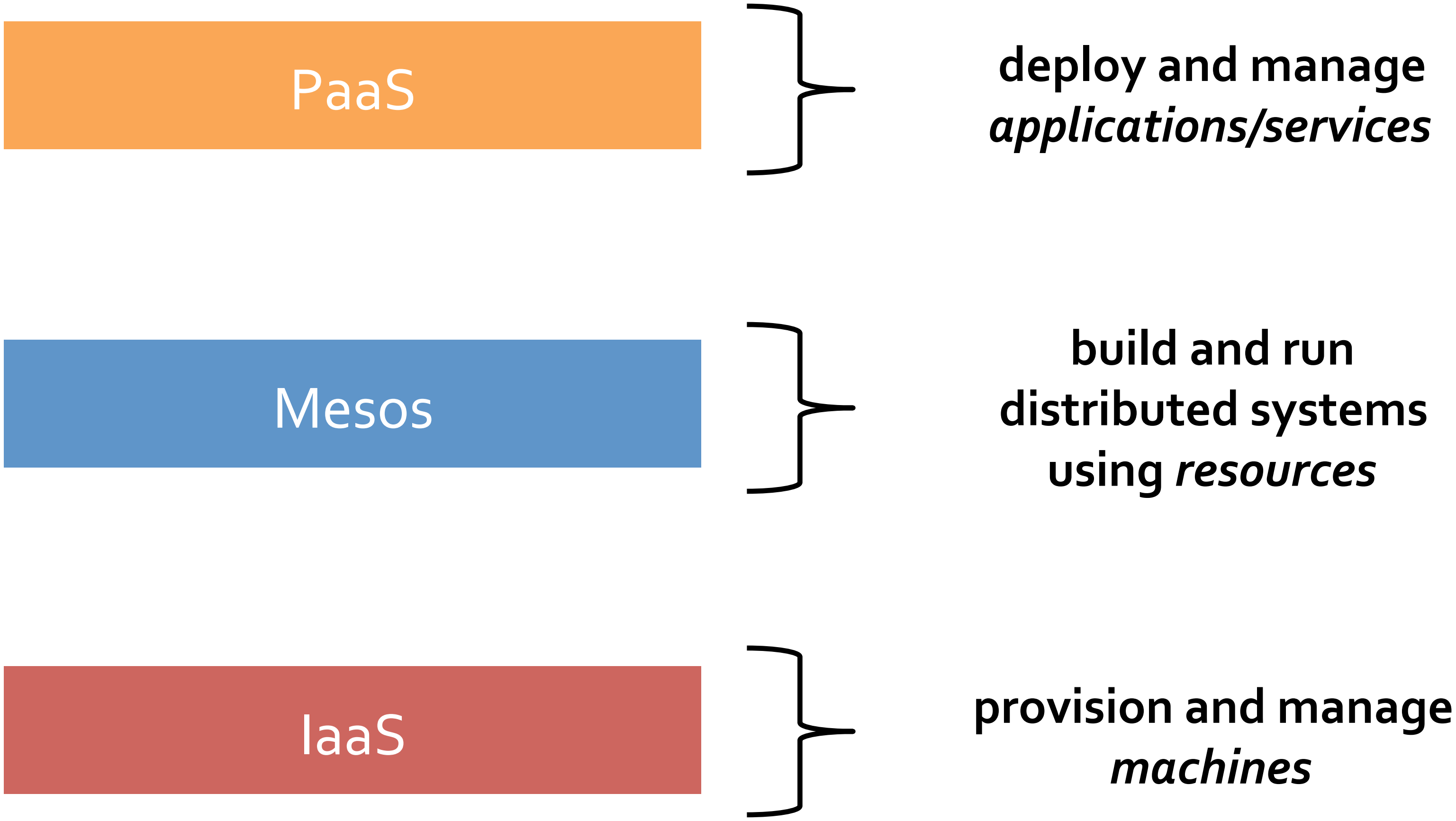
Mesos

build and run distributed systems using *resources*

IaaS

provision and manage *machines*

Mesos: level of abstraction



PaaS

deploy and manage *applications/services*

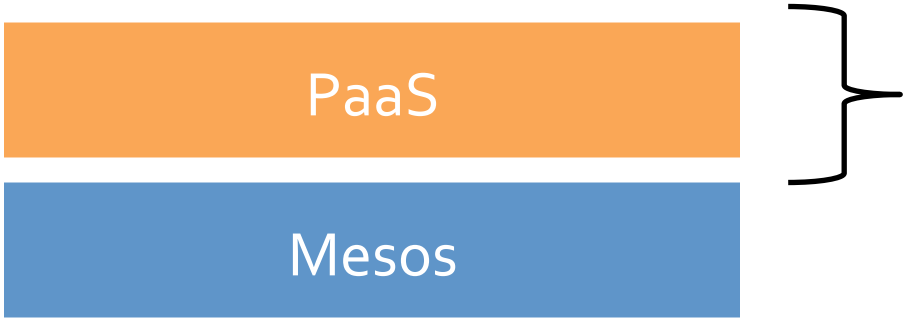
Mesos

build and run distributed systems using *resources*

IaaS

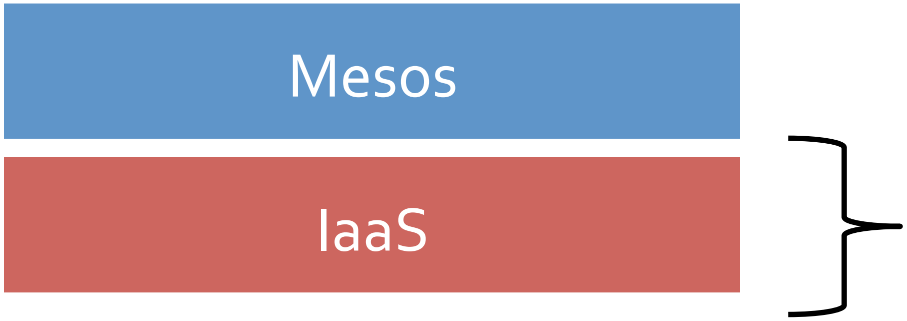
provision and manage *machines*

PaaS on Mesos



build and run a PaaS on top of Mesos: Marathon

Mesos on IaaS

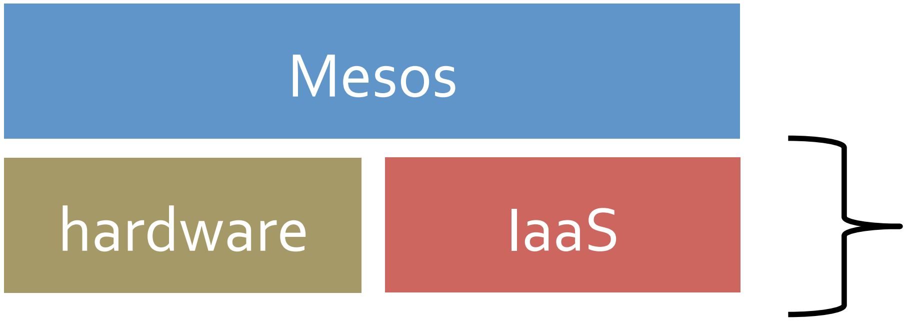


Mesos

IaaS

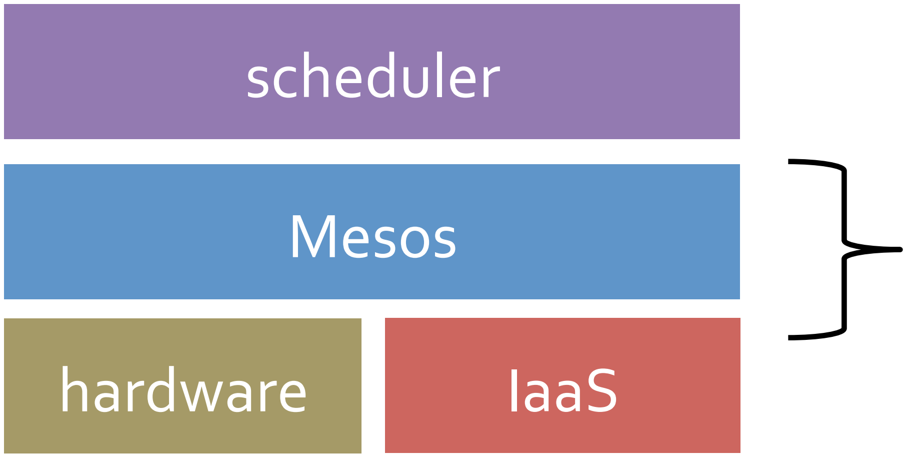
use OpenStack or EC2
to run Mesos

Mesos on IaaS/bare metal



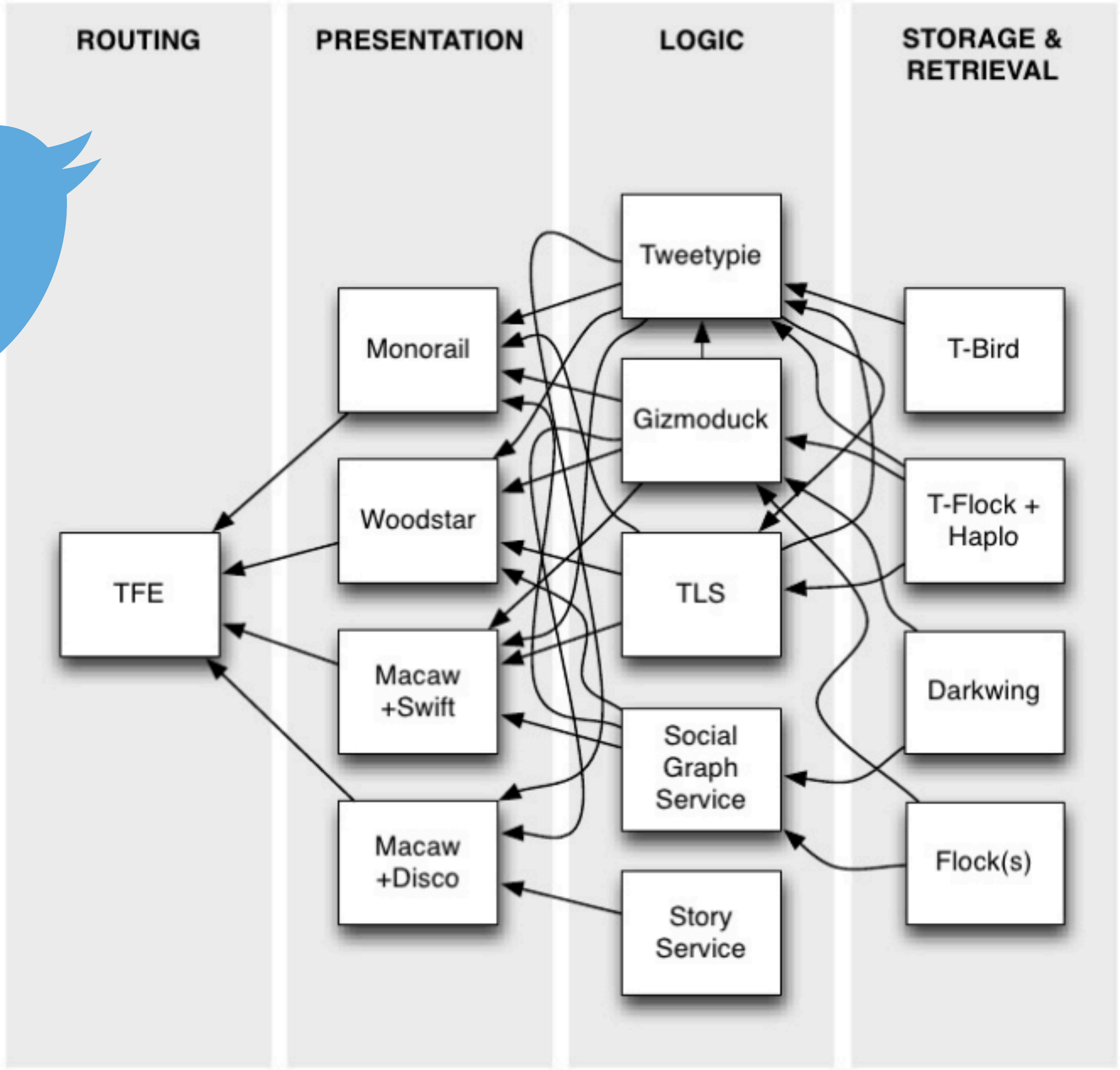
use OpenStack or EC2
or physical machines
to run Mesos

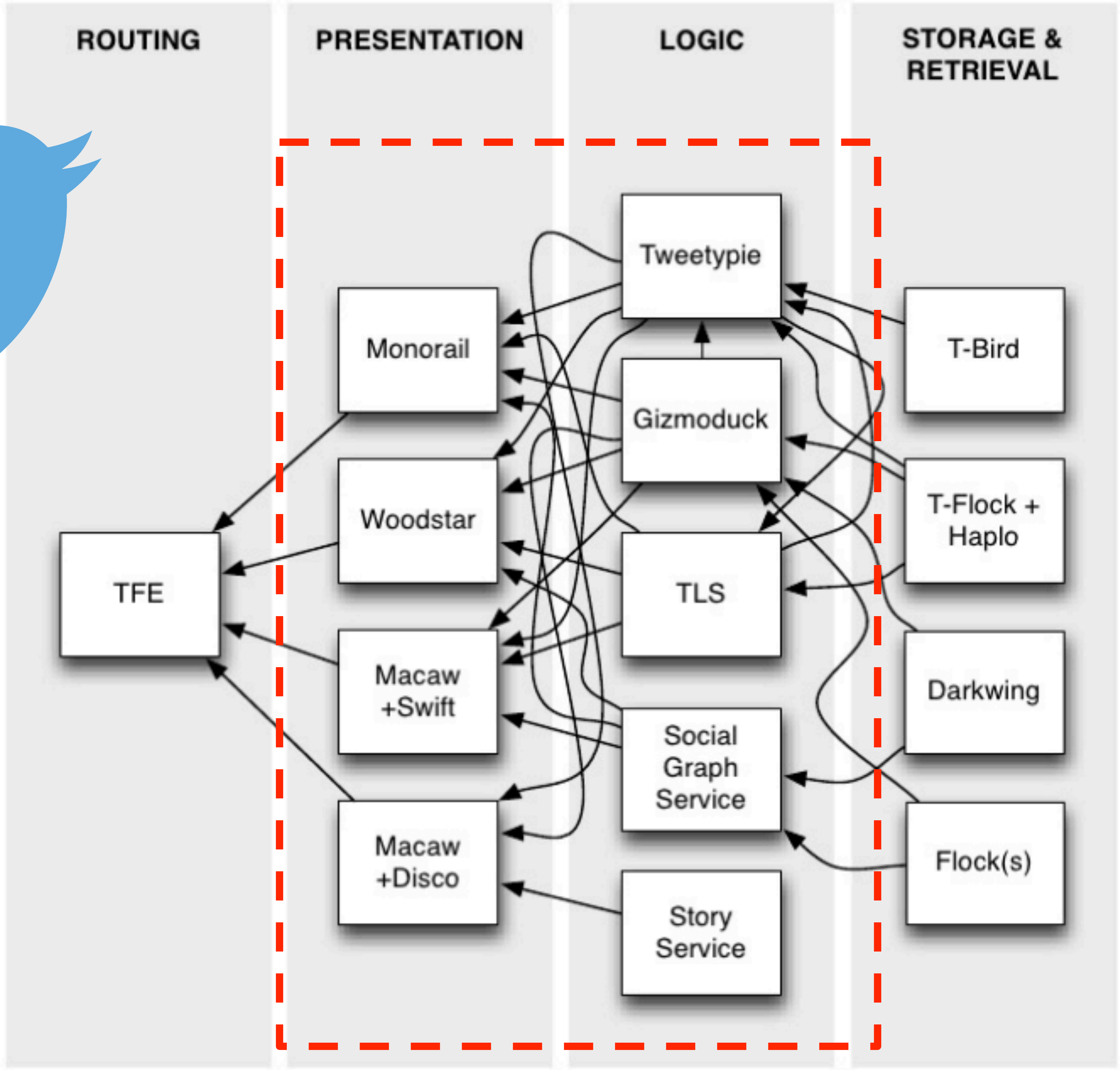
Mesos: datacenter kernel

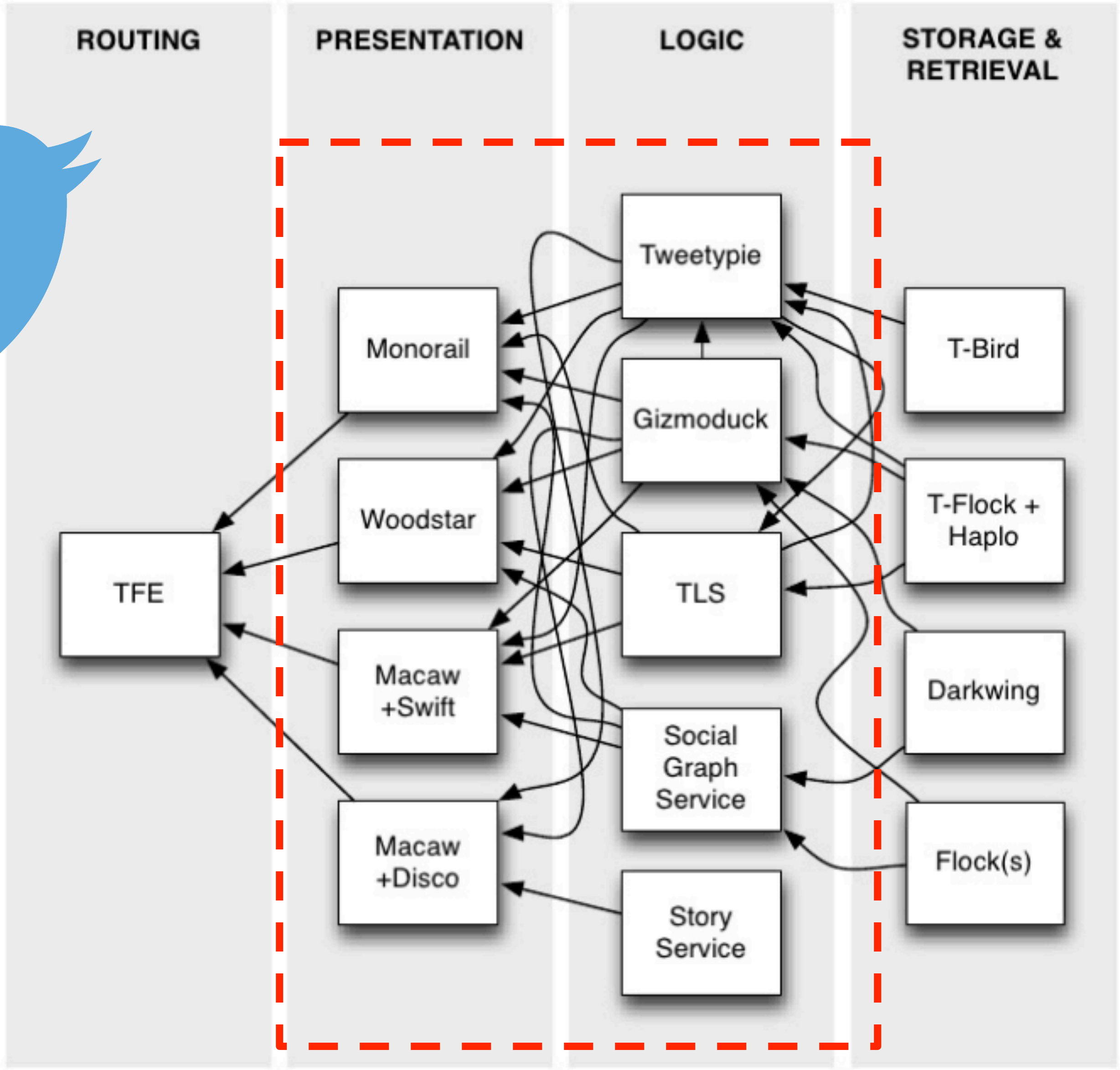


provide common functionality
via an API (kernel)

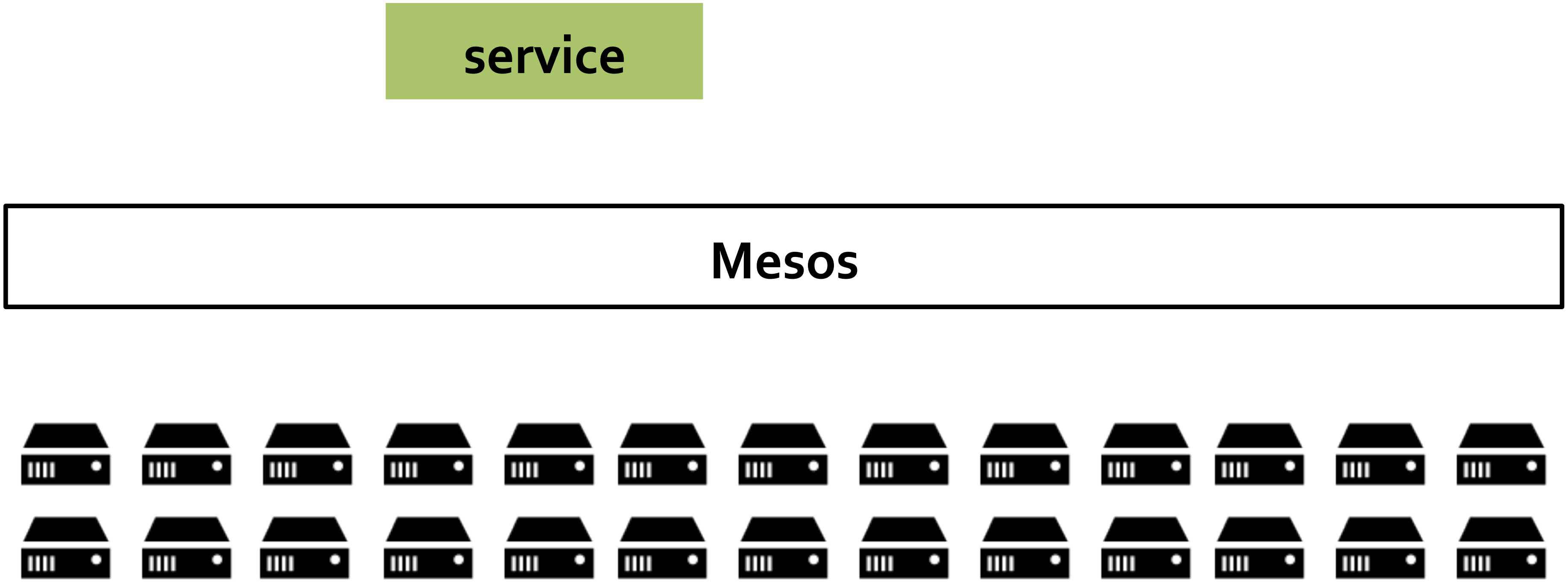
but how should we run datacenter applications?


ROUTING
PRESENTATION
LOGIC
STORAGE & RETRIEVAL
Tweetypie
Monorail
Gizmoduck
T-Bird
Woodstar
T-Flock + Haplo
TFE
TLS
Macaw +Swift
Darkwing
Social Graph Service
Macaw +Disco
Flock(s)
Story Service


ROUTING
PRESENTATION
LOGIC
STORAGE & RETRIEVAL
TFE
Monorail
Woodstar
Macaw +Swift
Macaw +Disco
Tweetypie
Gizmoduck
TLS
Social Graph Service
Story Service
T-Bird
T-Flock + Haplo
Darkwing
Flock(s)



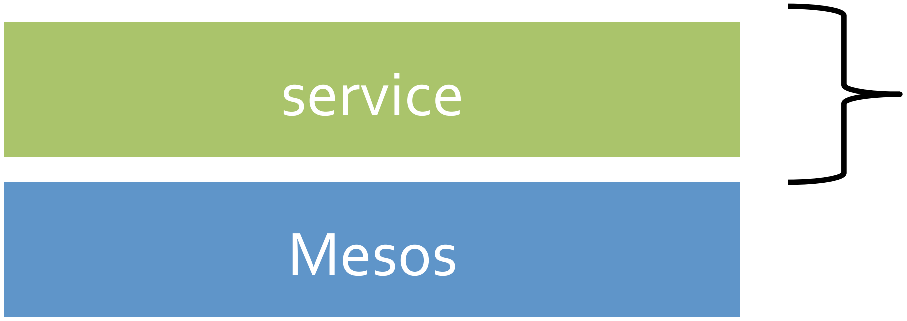
stateless services!



service

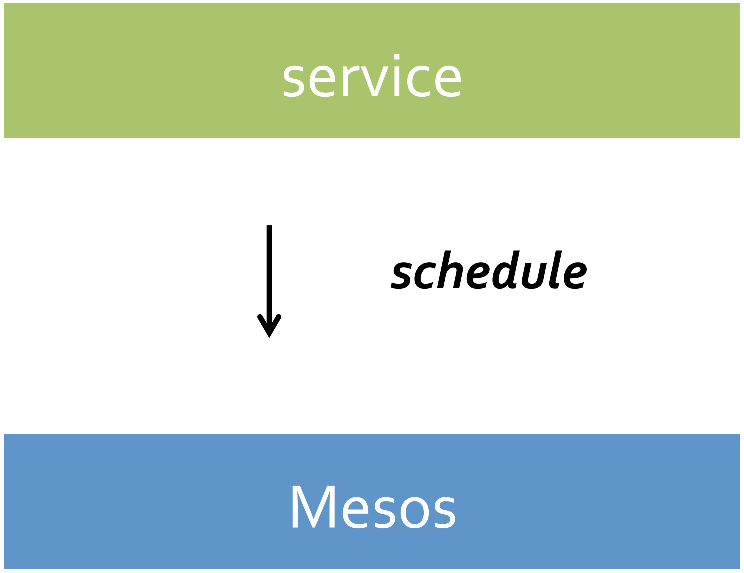
Mesos

service scheduler (PaaS)

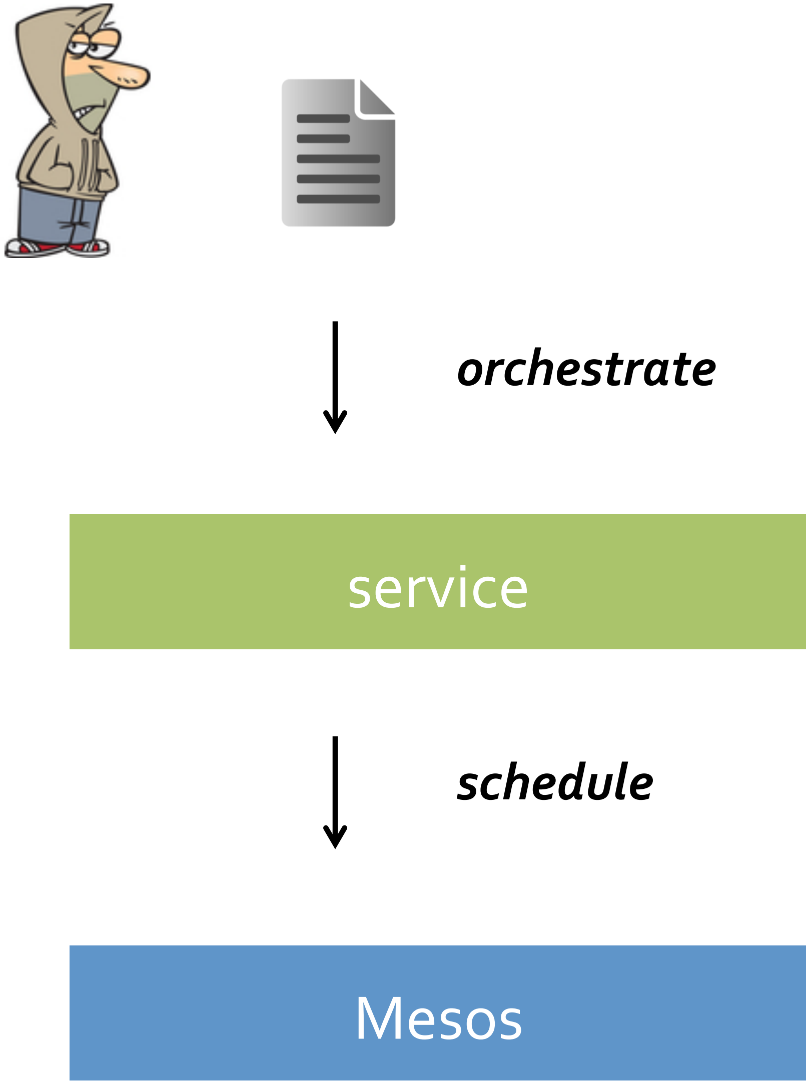


orchestrate services
on top of Mesos

orchestration vs scheduling



orchestration vs scheduling



orchestrate

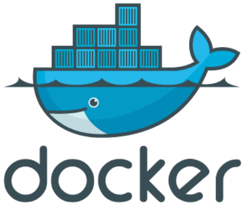
service

schedule

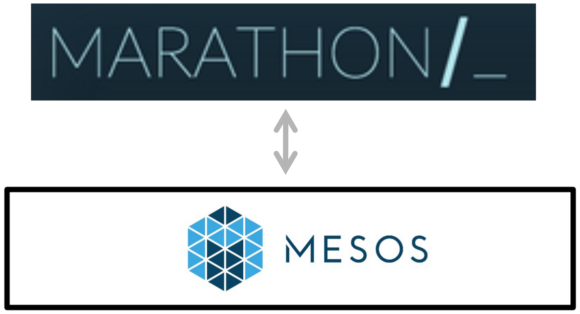
Mesos

orchestration with Marathon

① configuration/package management



② deployment



③ service discovery

configuration/package management

developers

(1) bundle services as jar, tar/gzip, or using Docker

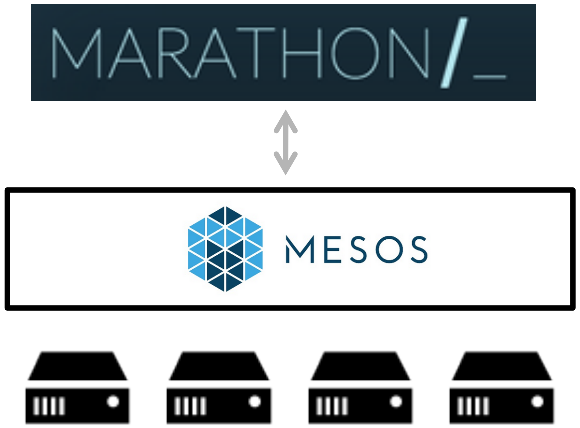
(2) upload to HDFS (or use a Docker registry)

deployment

developers

(1) describe services using JSON

(2) submit services to Marathon via REST

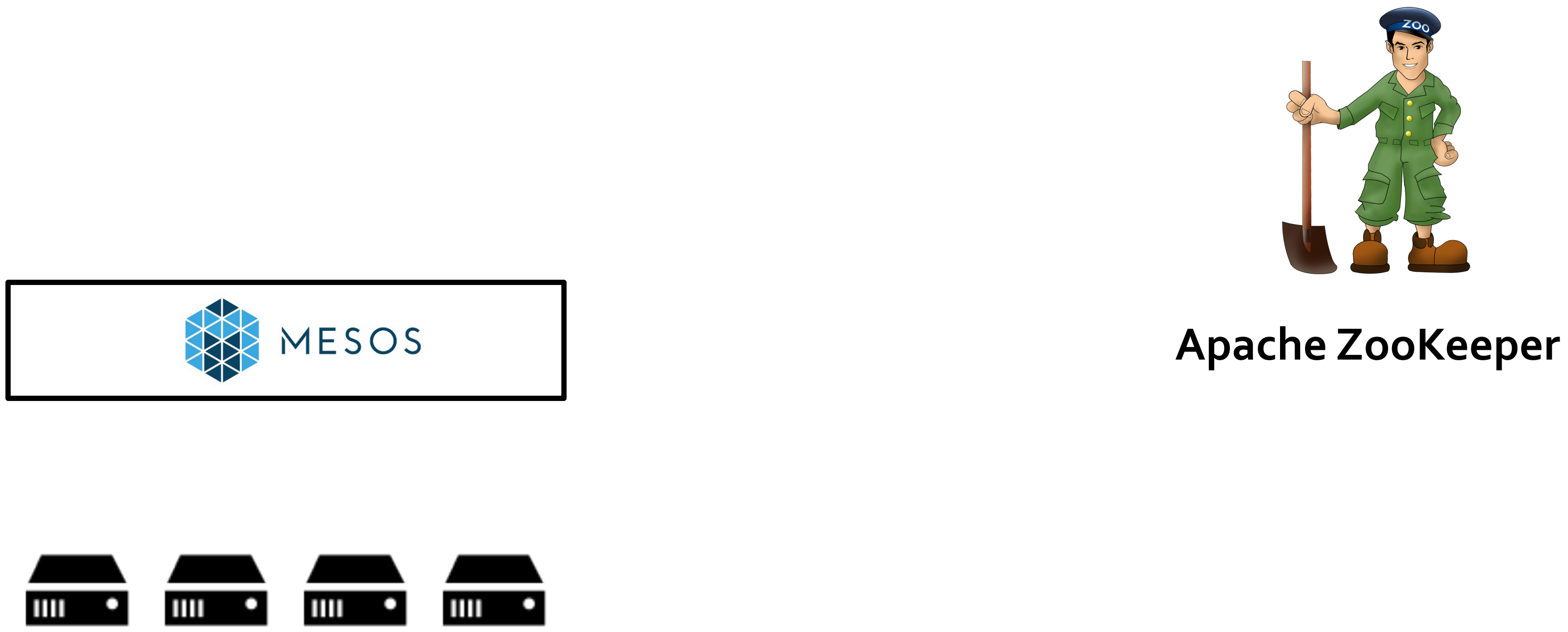


example-docker.json

```
{
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "libmesos/ubuntu"
    },
    "volumes" : [
      {
        "containerPath": "/etc/a",
        "hostPath": "/var/data/a",
        "mode": "RO"
      },
      {
        "containerPath": "/etc/b",
        "hostPath": "/var/data/b",
        "mode": "RW"
      }
    ]
  },
  "id": "ubuntu",
  "instances": 1,
  "cpus": 0.5,
  "mem": 512,
  "cmd": "while sleep 10; do date -u +%T; done"
}
```

service discovery

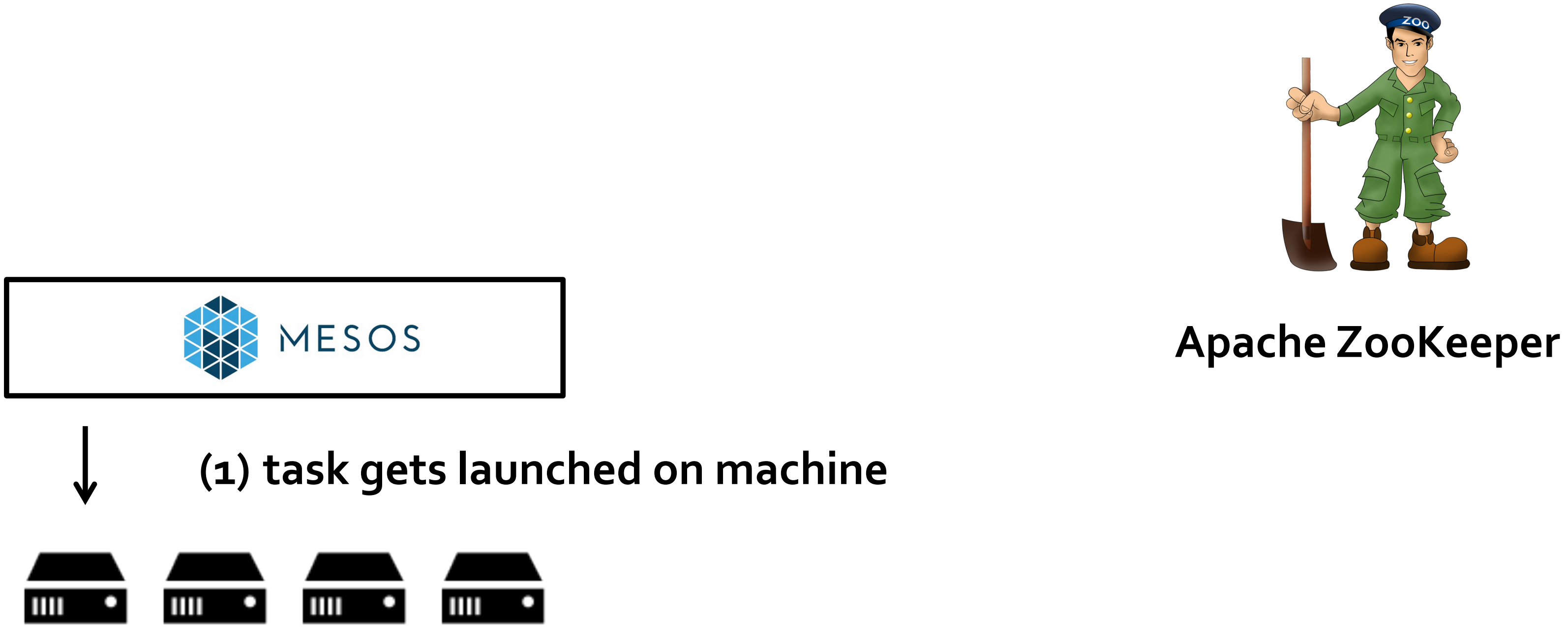
using Apache ZooKeeper and *server sets* (github.com/twitter/commons)



Apache ZooKeeper

service discovery

using Apache ZooKeeper and *server sets* (github.com/twitter/commons)

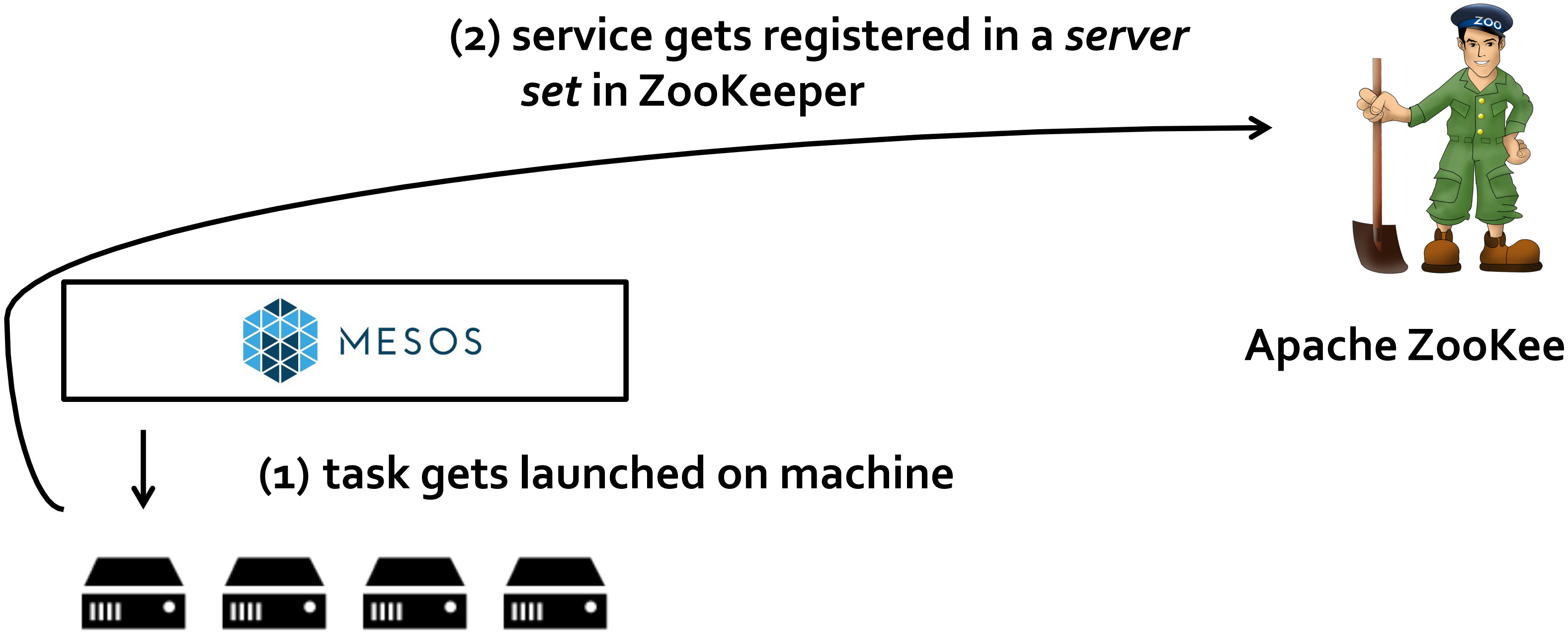


(1) task gets launched on machine

Apache ZooKeeper

service discovery

using Apache ZooKeeper and *server sets* (github.com/twitter/commons)



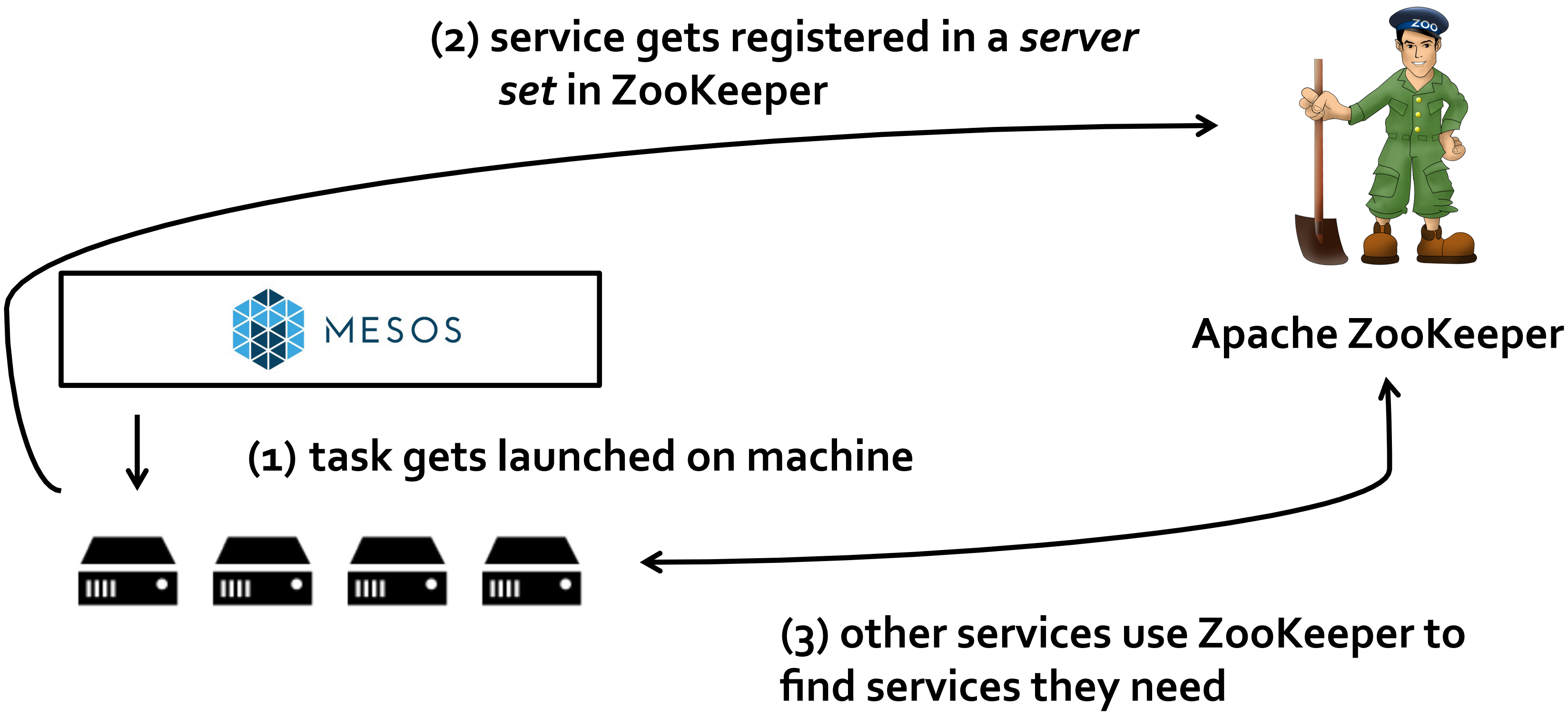
(2) service gets registered in a *server set* in ZooKeeper

(1) task gets launched on machine

Apache ZooKeeper

service discovery

using Apache ZooKeeper and *server sets* (github.com/twitter/commons)



(2) service gets registered in a *server set* in ZooKeeper

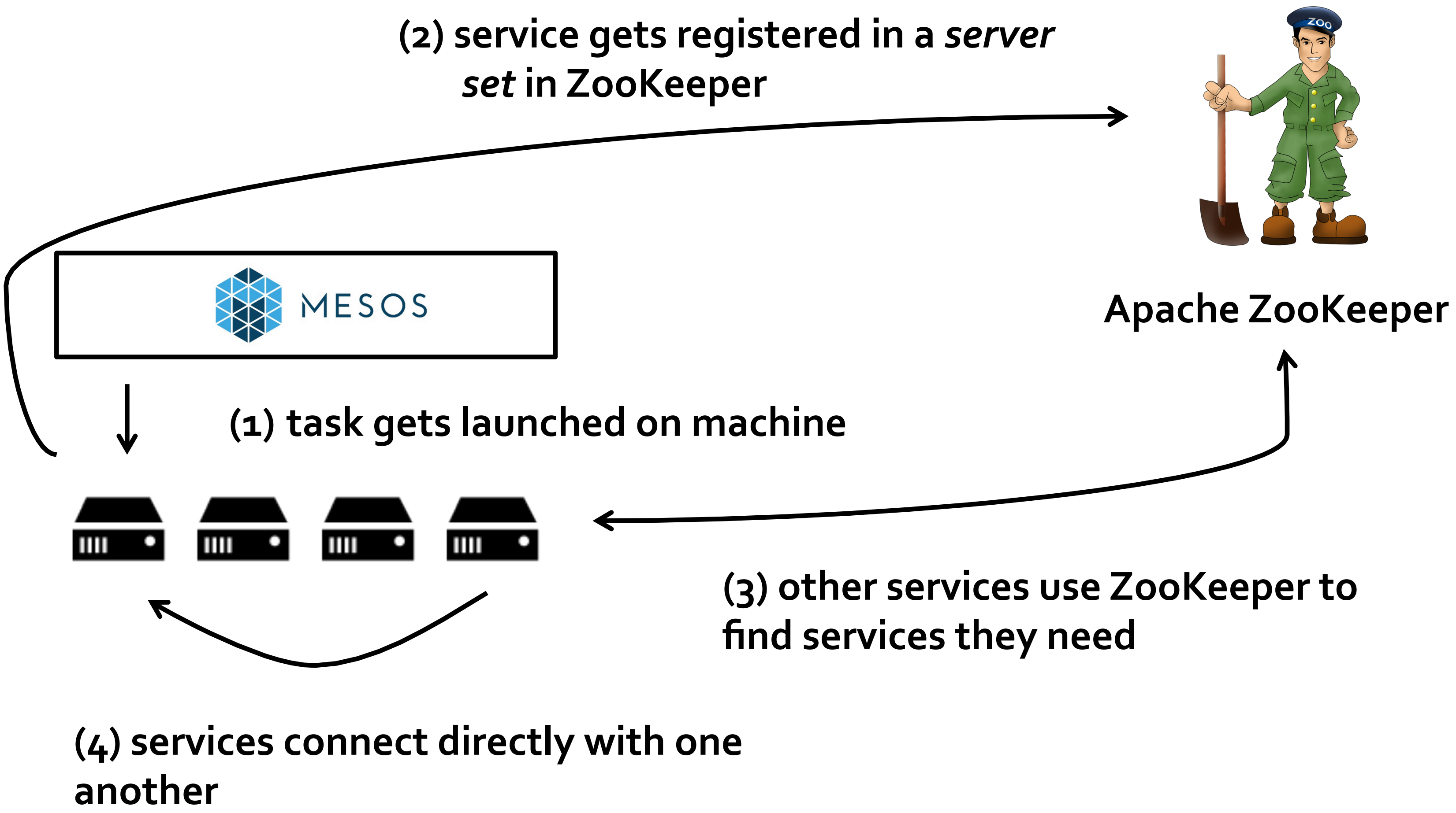
Apache ZooKeeper

(1) task gets launched on machine

(3) other services use ZooKeeper to find services they need

service discovery

using Apache ZooKeeper and *server sets* (github.com/twitter/commons)



(2) service gets registered in a *server set* in ZooKeeper

Apache ZooKeeper

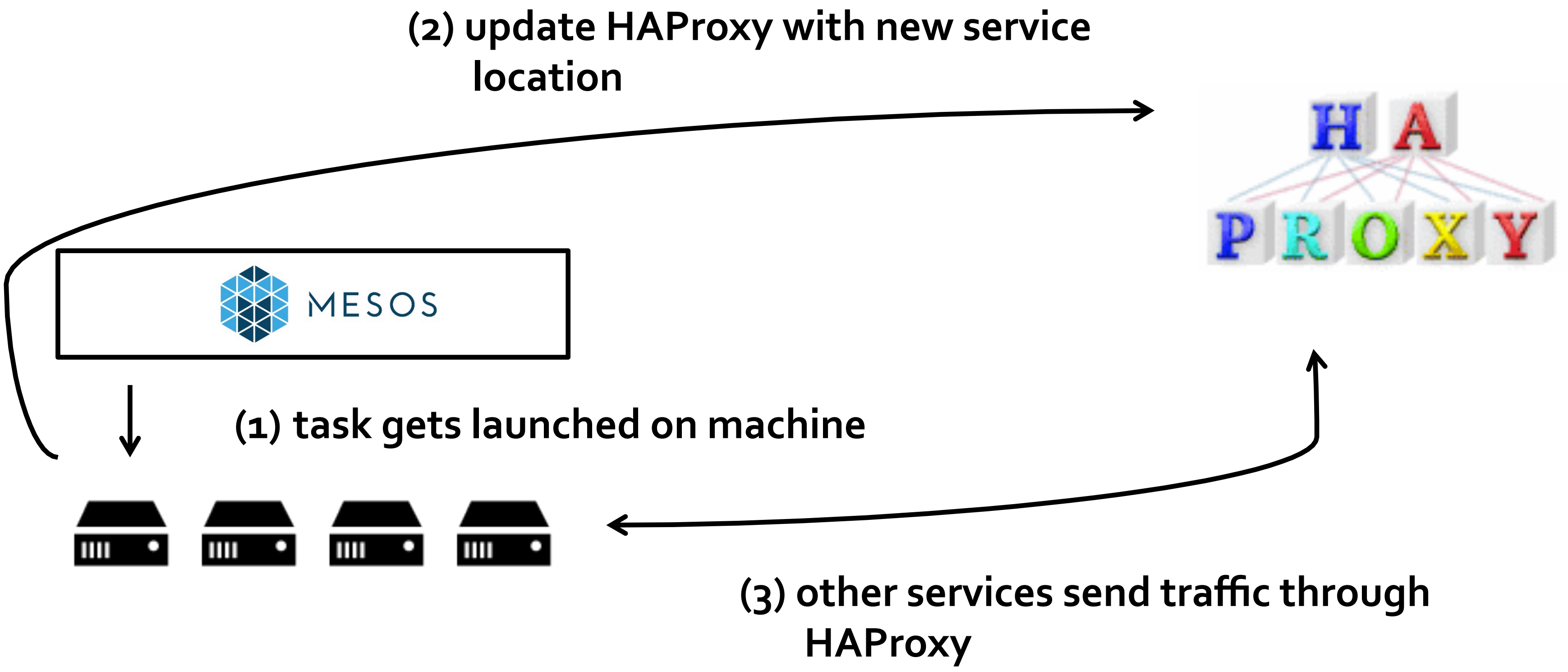
(1) task gets launched on machine

(3) other services use ZooKeeper to find services they need

(4) services connect directly with one another

service discovery alternative

ZooKeeper/server sets requires injecting code into your clients!



(2) update HAProxy with new service location

(1) task gets launched on machine

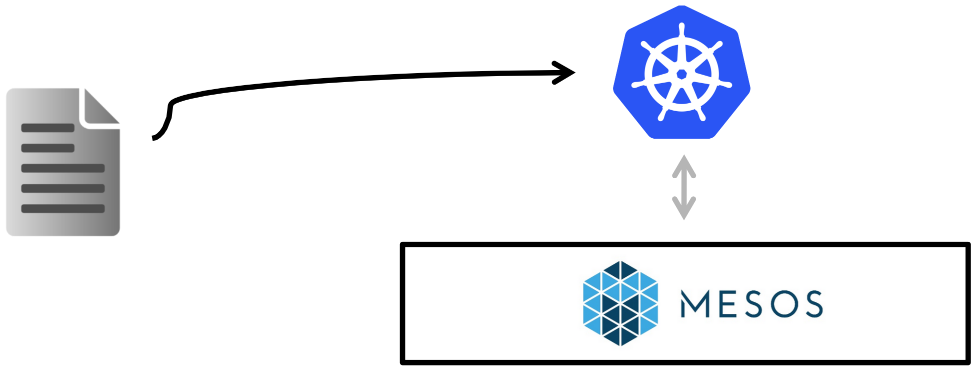
(3) other services send traffic through HAProxy

orchestration w/ Kubernetes (on Mesos)

① configuration/package management

docker

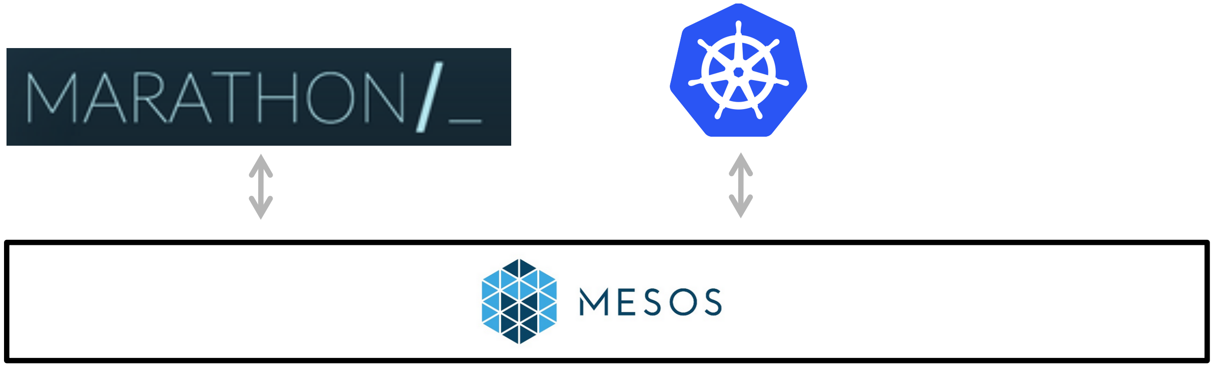
② deployment



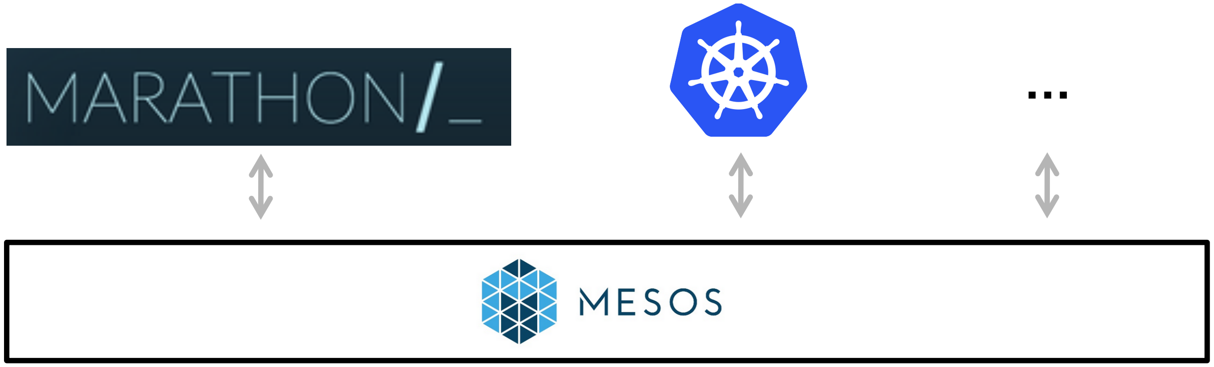
③ service discovery

✓

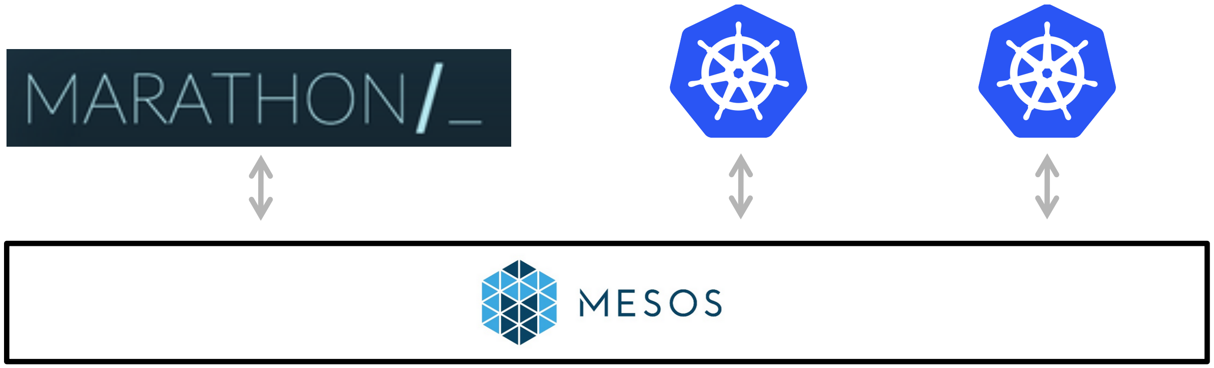
multiple schedulers!



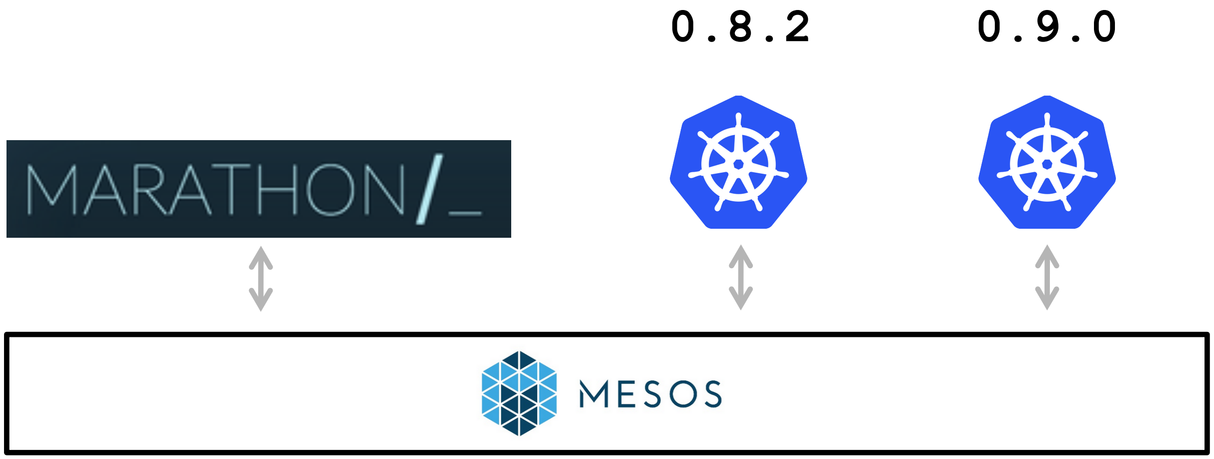
multiple schedulers!



multiple schedulers!



multiple schedulers!



agenda

① what are datacenter applications?

② how should we run datacenter applications?

③ how should we program datacenter applications?

distributed systems: dev

- leader election
- state management (working set)
- task management (launch, isolate, kill, etc)
- machine management (monitoring)
- …

distributed systems: ops

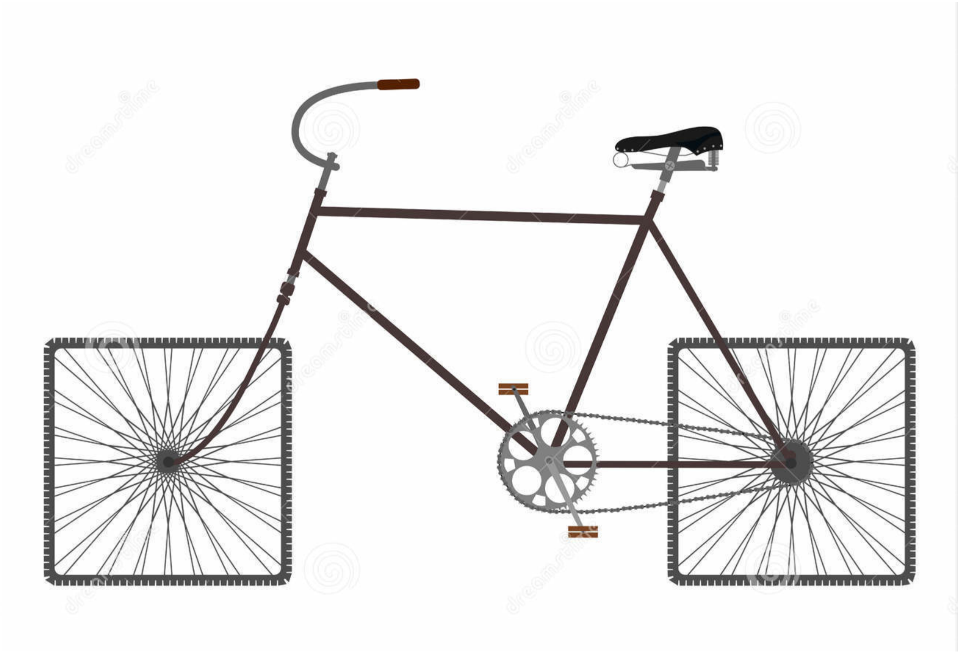
- what to do if task/machine fails?
- what happens when more resources/tasks are needed? does everything scale proportionally? should tasks take on different or more specialized functions/roles?

distributed systems: ops

- *what to do if task/machine fails?*
- what happens when more resources/tasks are needed? does everything scale proportionally? should tasks take on different or more specialized functions/roles?

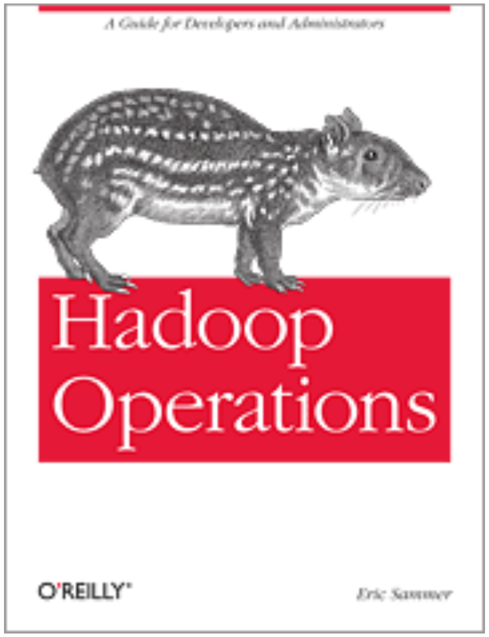
distributed systems: dev

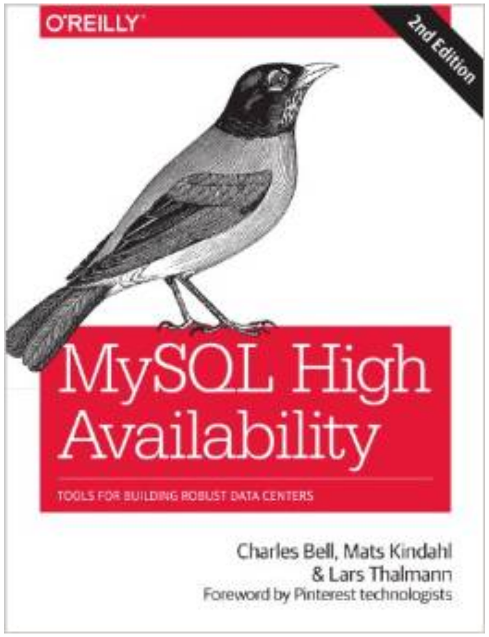
everybody keeps reinventing the wheel

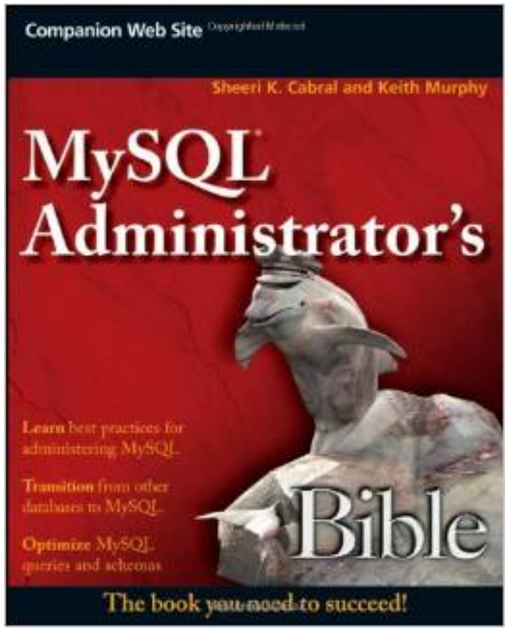


distributed systems: ops

UX of operating distributed systems is horrible




O'REILLY
2nd Edition
MySQL High Availability
TOOLS FOR BUILDING ROBUST DATA CENTERS
Charles Bell, Mats Kindahl & Lars Thalmann
Foreword by Pinterest technologists


Companion Web Site
Sheeri K. Cabral and Keith Murphy
MySQL Administrator's
Bible
Learn best practices for administering MySQL
Transition from other databases to MySQL
Optimize MySQL queries and schemas
The book you need to succeed!

① why does each distributed system have to implement the same things?

② why can‘t we build distributed systems that incorporate and automate operations?

thesis: we need a common abstraction layer upon which all distributed systems can be built and …

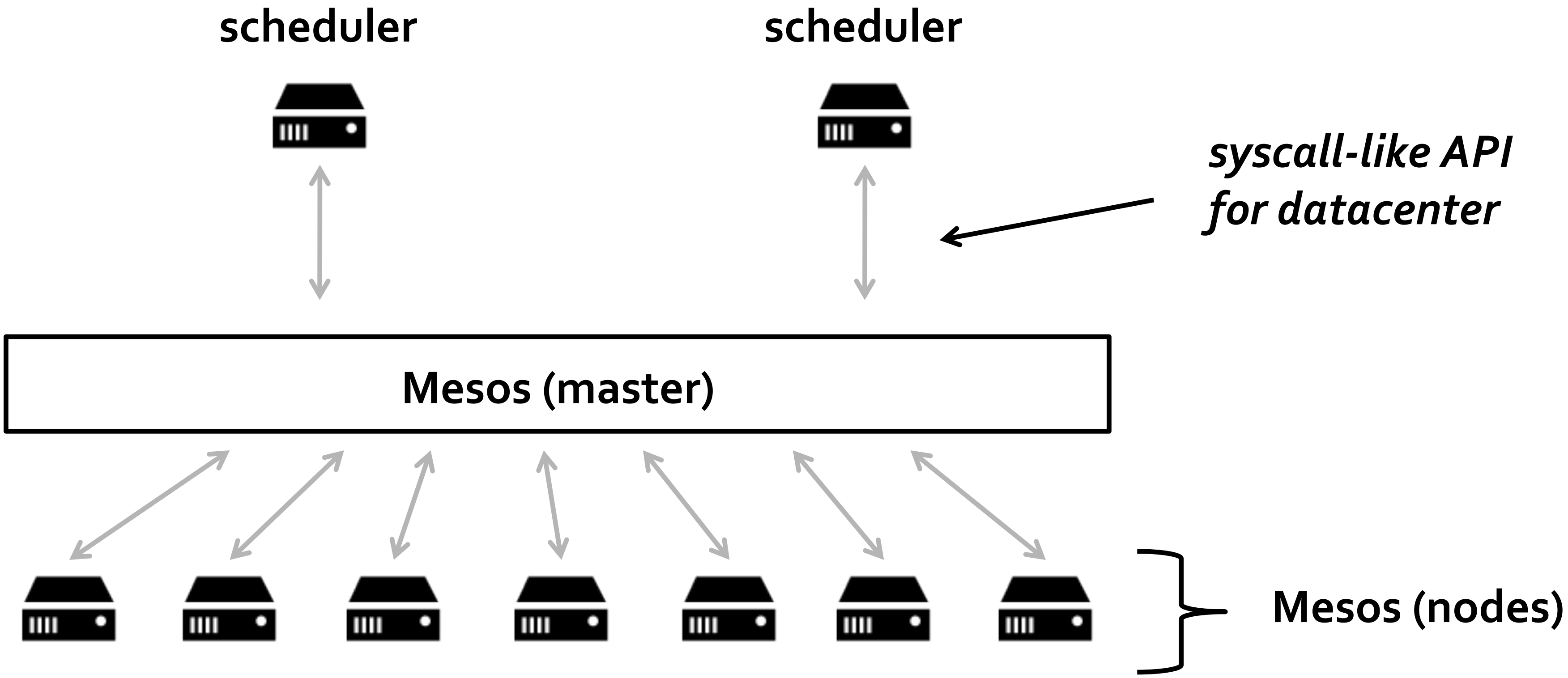
we need to build distributed systems with *schedulers* (on top of said common abstraction layer)

Apache Mesos is a
distributed system
for running and building
other distributed systems

Apache Mesos: distributed systems kernel

Apache Mesos: datacenter kernel

Mesos: datacenter kernel



Mesos: datacenter kernel

+ provide common functionality every new distributed system *re-implements*

Mesos: datacenter kernel

+ enable running multiple distributed systems on the same cluster of machines and dynamically share the resources more efficiently!

build against Mesos:

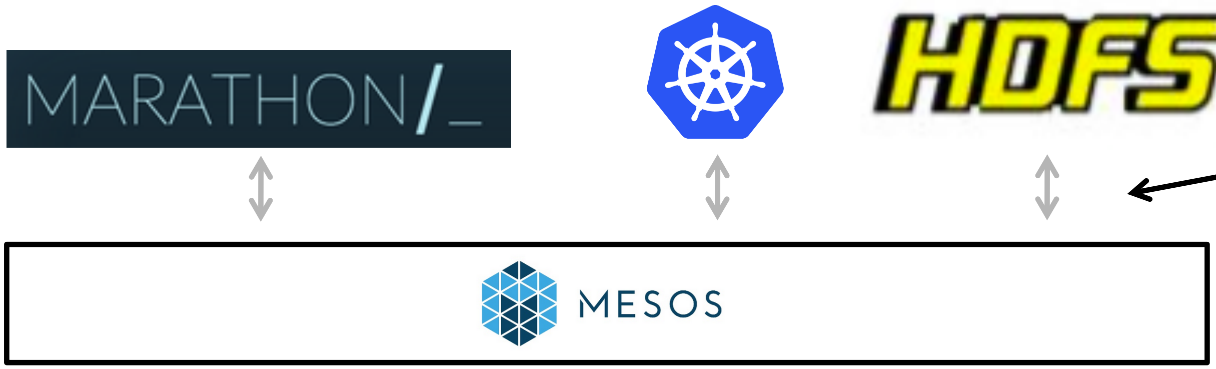
① abstract cloud, i.e., “hardware”

② leverage primitives to implement/automate failures, maintenance, etc.

Mesos primitives

- principals, users, roles
- advanced fair-sharing allocation algorithms
- high-availability (even during upgrades)
- resource monitoring
- preemption/revocation
- volume management
- reservations (dynamic/ static)
- …

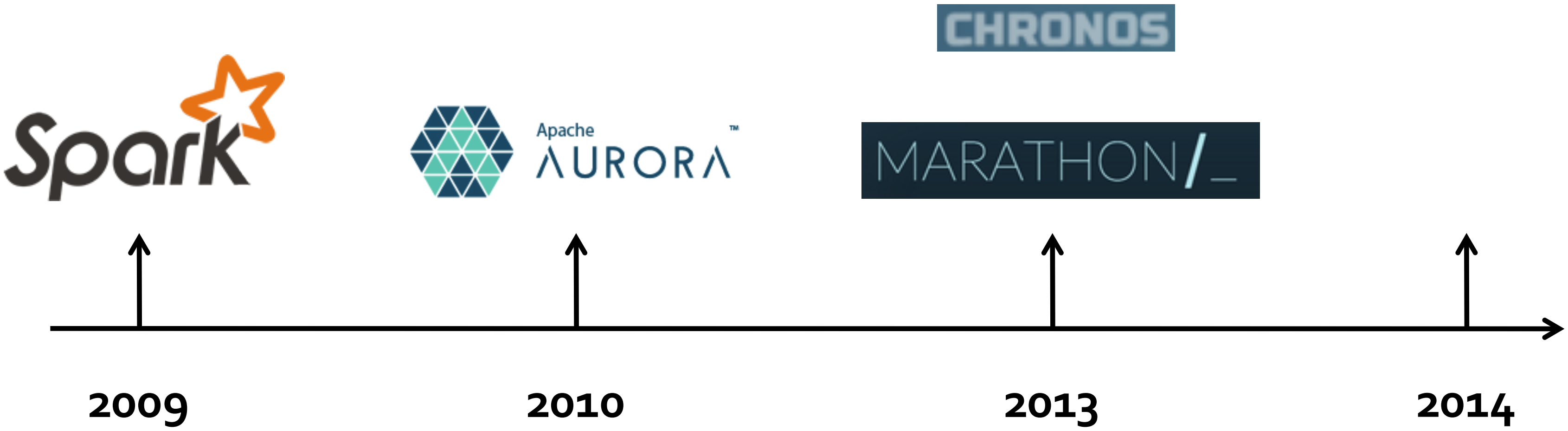
maintenance via primitives



propagate machine restart, shutdown, downtime, etc there is a lot of stuff in a kernel

there is a lot of stuff in a datacenter kernel

built on Mesos:



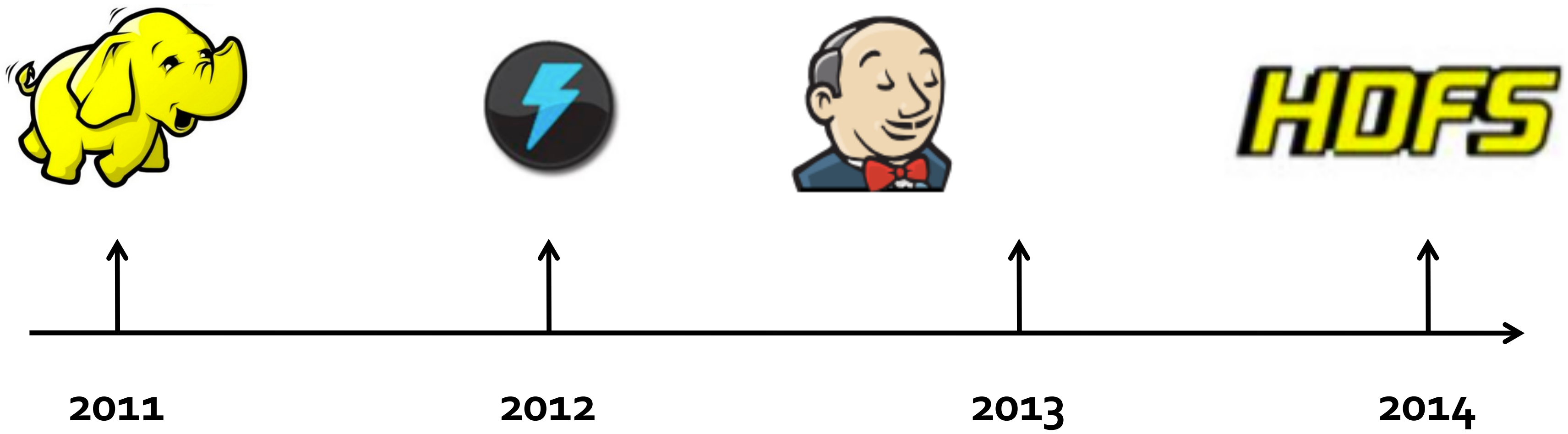
2009

2010

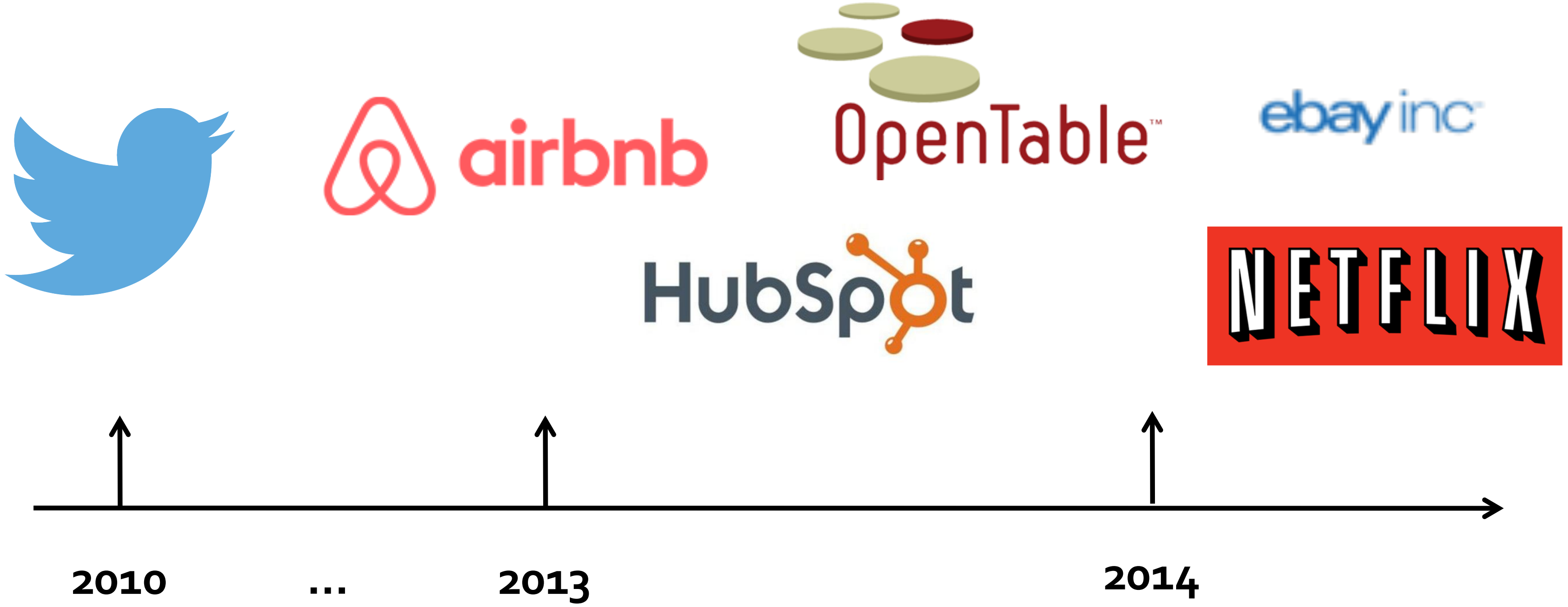
2013

2014

ported to Mesos:



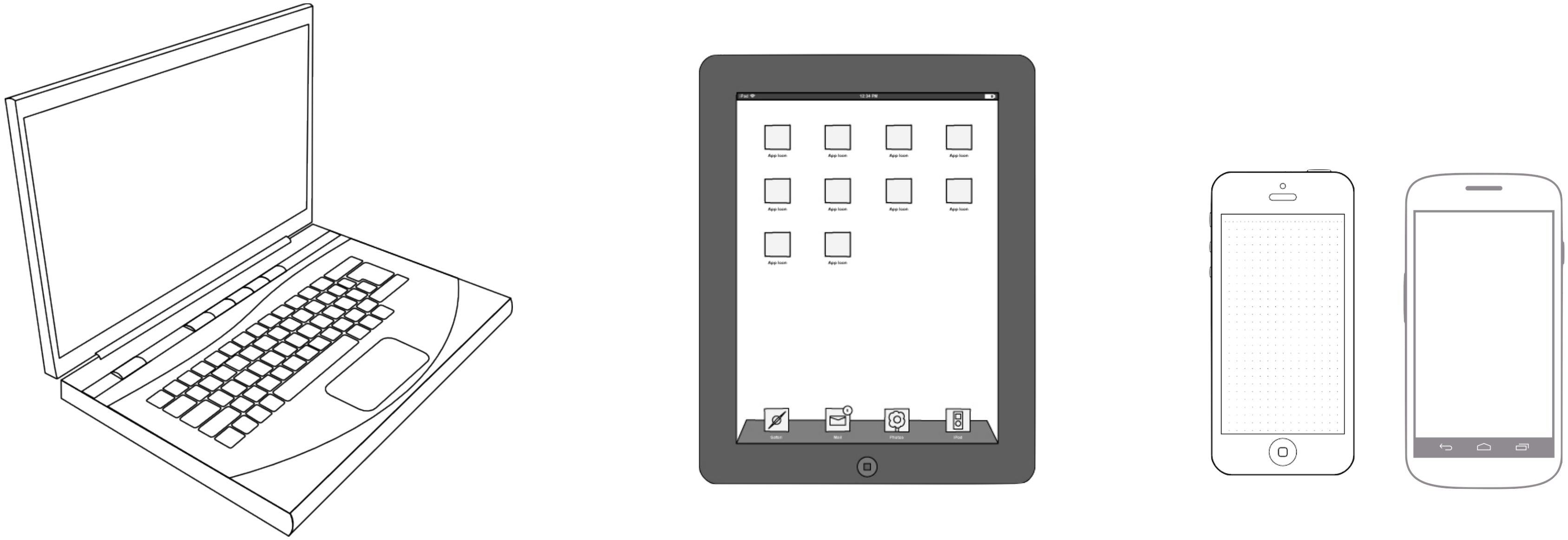
some of our adopters …



2010 … 2013 2014

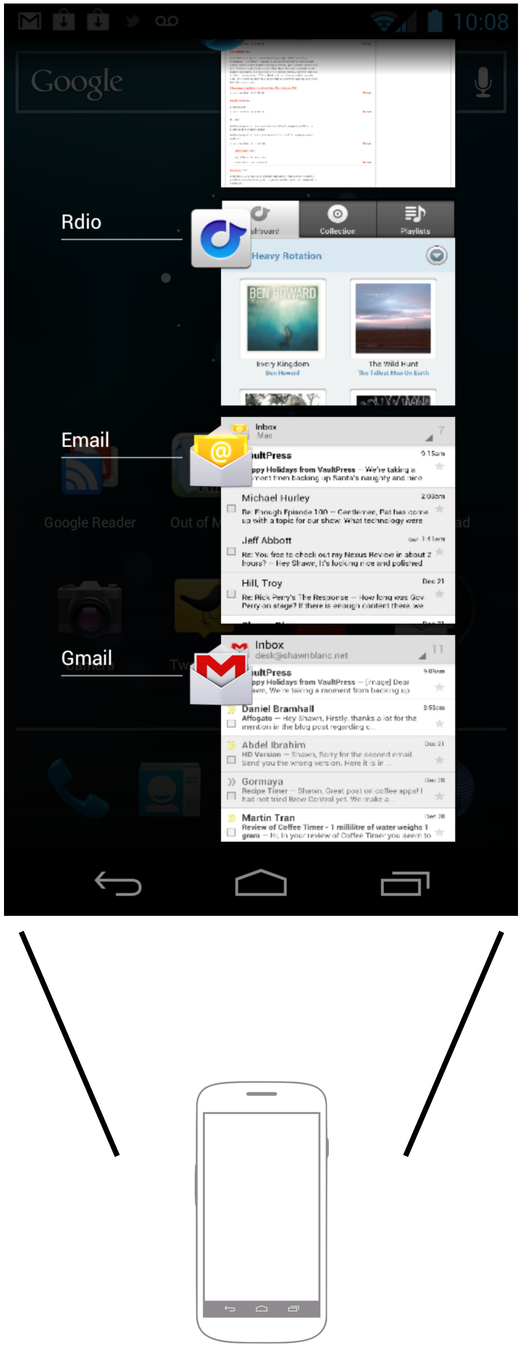
conclusion

the datacenter
is just another form factor



the datacenter is just another form factor

why can‘t we run applications on our datacenters just like we run applications on our mobile phones?



Inbox
Mac
7

ultPress 9:15am
ppy Holidays from VaultPress — We're taking a
ment from backing up Santa's naughty and nice

Michael Hurley 2:03am
Re: Enough Episode 100 — Gentlemen, Pat has come up with a topic for our show: What technology were

Jeff Abbott 1:41am
Re: You free to check out my Nexus Review in about 2 hours? — Hey Shawn, It's looking nice and polished

Hill, Troy Dec 21
Re: Rick Perry's The Response — How long was Gov. Perry on stage? If there is enough content there, we

Inbox
desk@shawnblanc.net
11

ultPress 9:03am
ppy Holidays from VaultPress — [image] Dear
awn, We're taking a moment from backing up

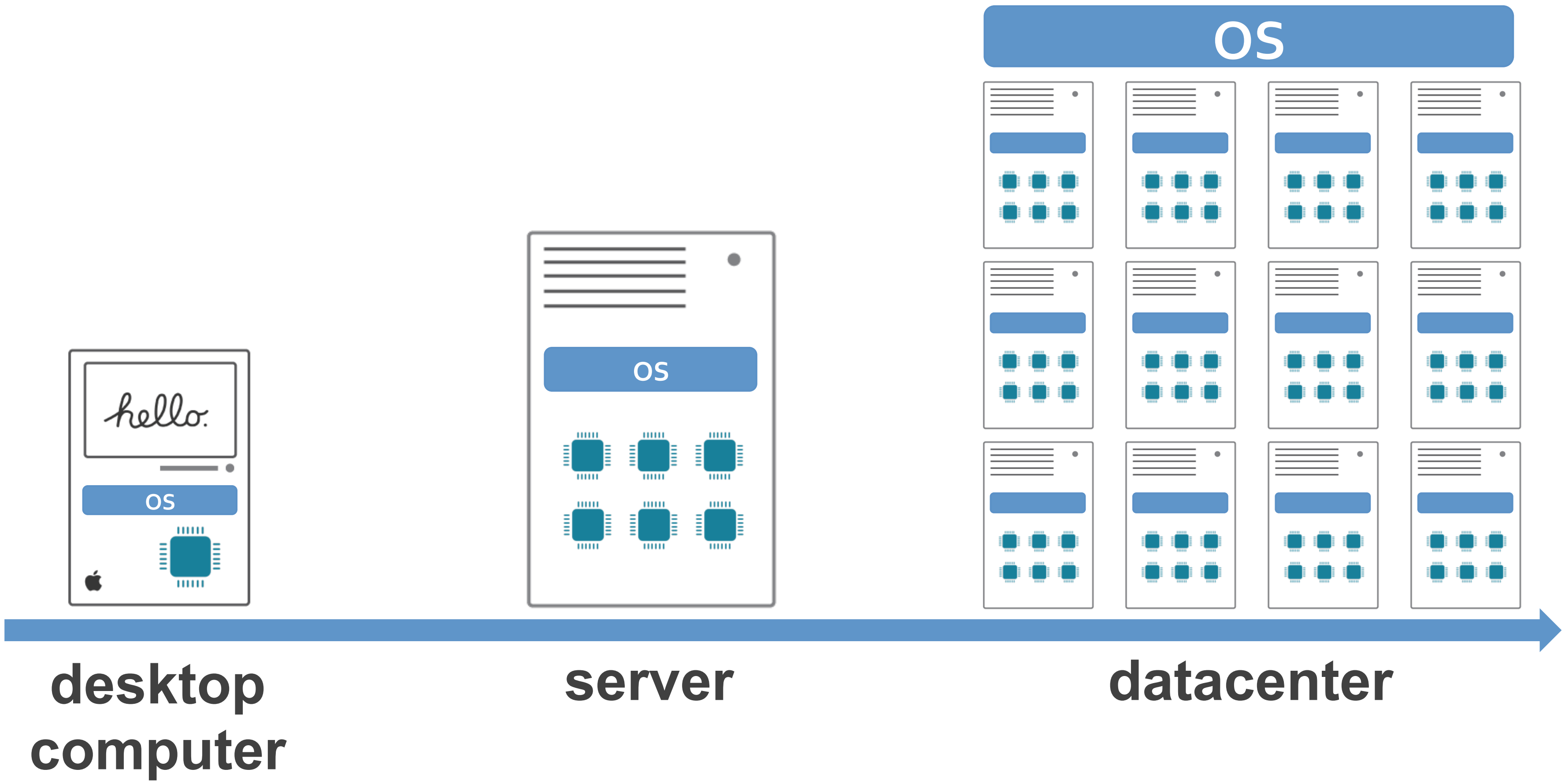
Daniel Bramhall 5:53am
Affogato — Hey Shawn, Firstly, thanks a lot for the mention in the blog post regarding c...

Abdel Ibrahim Dec 21
HD Version — Shawn, Sorry for the second email. Send you the wrong version. Here it is in ...

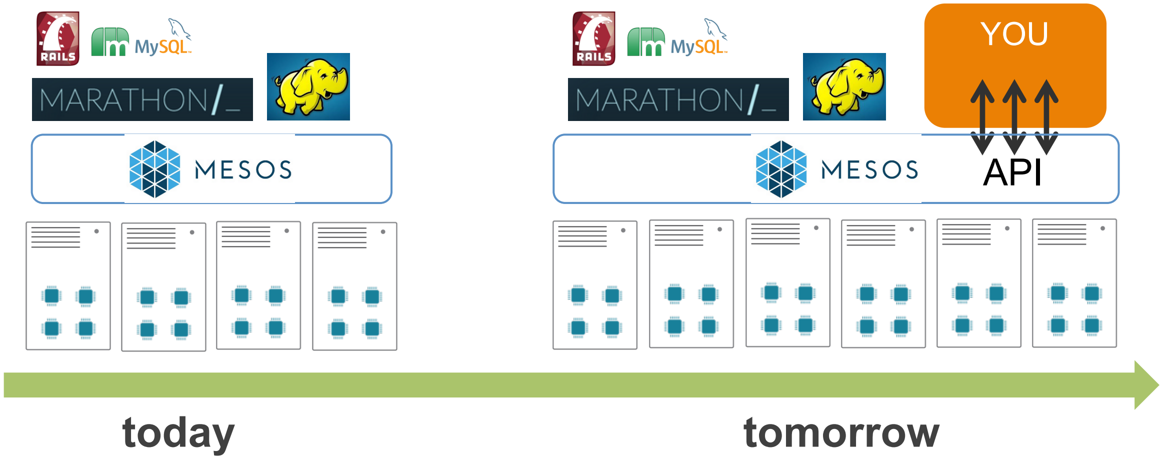
Gormaya Dec 20
Recipe Timer — Shawn, Great post on coffee apps! I had not tried Brew Control yet. We make a ...

Martin Tran Dec 20
Review of Coffee Timer - 1 millilitre of water weighs 1 gram — Hi, In your review of Coffee Timer you seem to

the datacenter computer needs an operating system



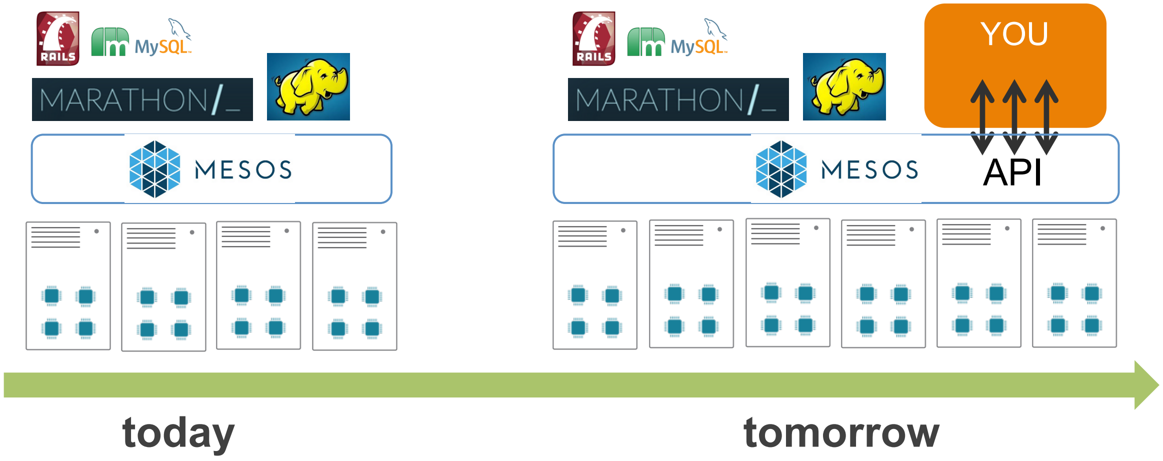
Mesos: datacenter kernel



provides common functionality every new distributed system *re-implements*:

- failure detection
- package distribution
- task starting
- resource isolation
- resource monitoring
- task killing, cleanup
- ...

Mesos: datacenter kernel

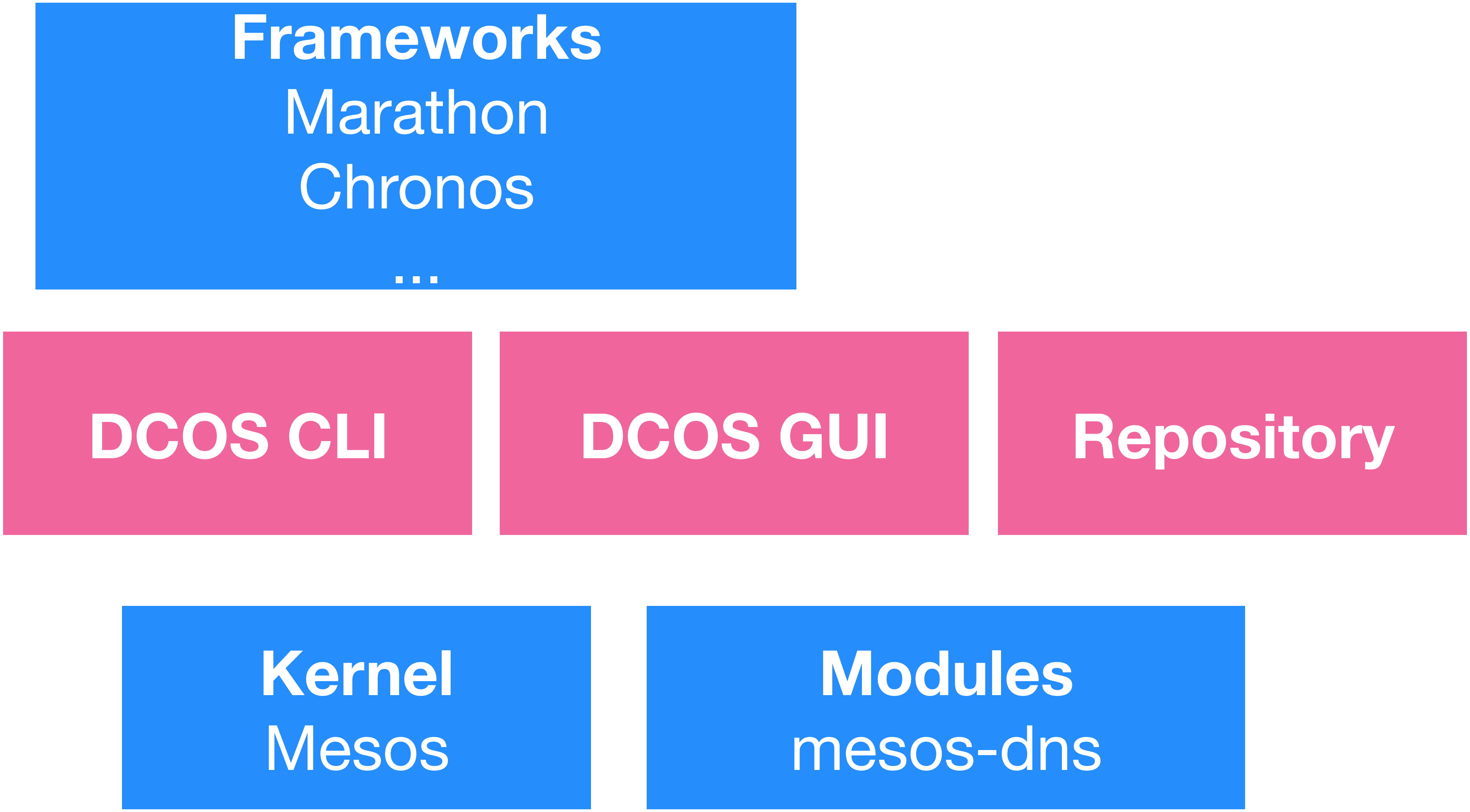


provides common functionality every new distributed system *re-implements*:

- failure detection
- package distribution
- task starting
- resource isolation
- resource monitoring
- task killing, cleanup
- …

don’t reinvent the wheel!

Mesosphere DCOS



Frameworks
Marathon
Chronos
...

DCOS CLI

DCOS GUI

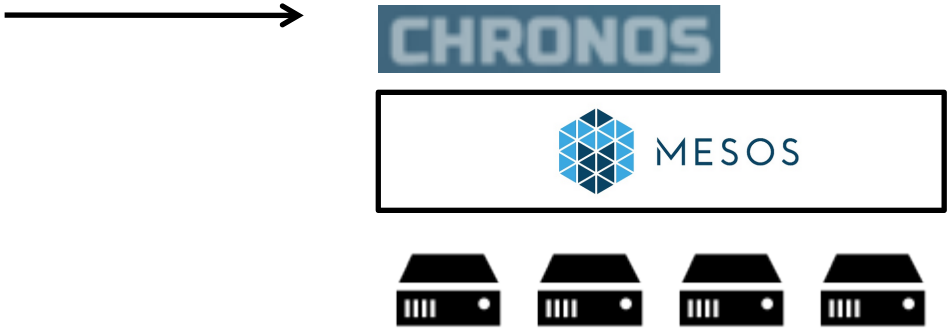
Repository

Kernel
Mesos

Modules
mesos-dns

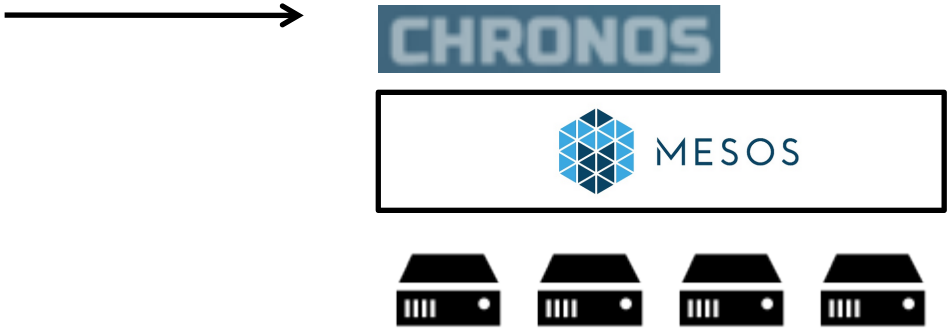
Airbnb's Chronos

Chronos, a scheduler for running *cron jobs with dependencies*



Airbnb's Chronos

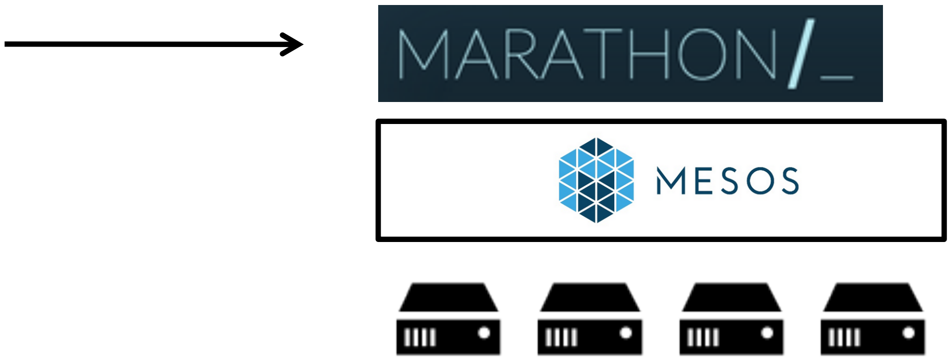
Chronos, a scheduler for running *cron jobs with dependencies*



cron *for the datacenter operating system*

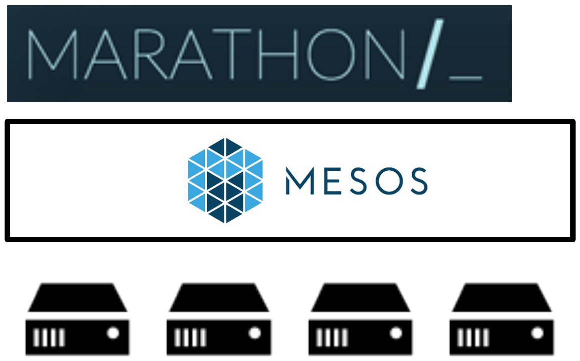
Mesosphere’s Marathon

Marathon, a scheduler for running *stateless services* written in any language



Mesosphere's Marathon

Marathon, a scheduler for running *stateless services* written in any language



__init__ for the datacenter operating system

Q&A

mesos.apache.org

@ApacheMesos

mesosphere.com

@mesosphere