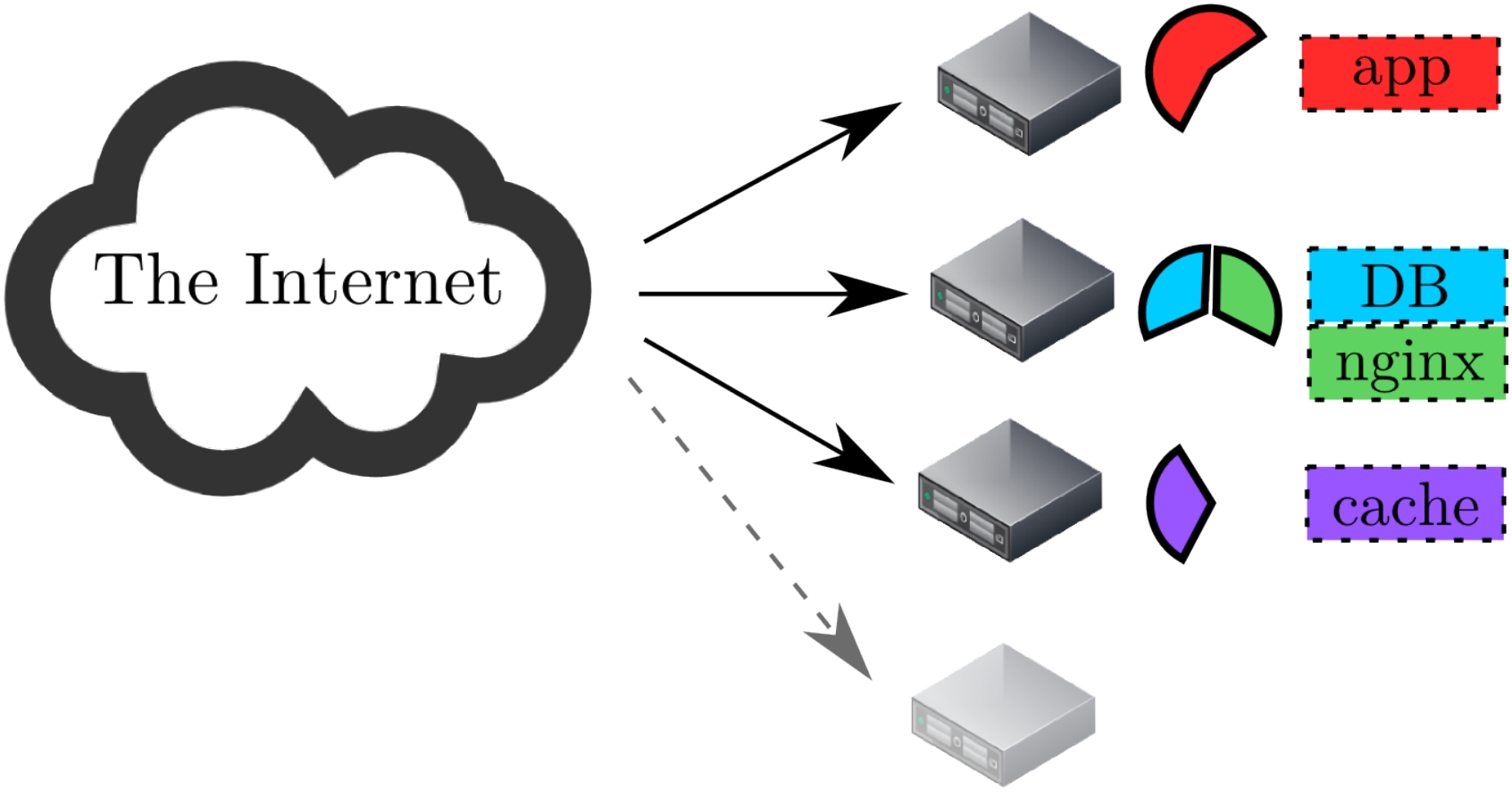
MESOS

Tomas Barton (@barton_tomas)

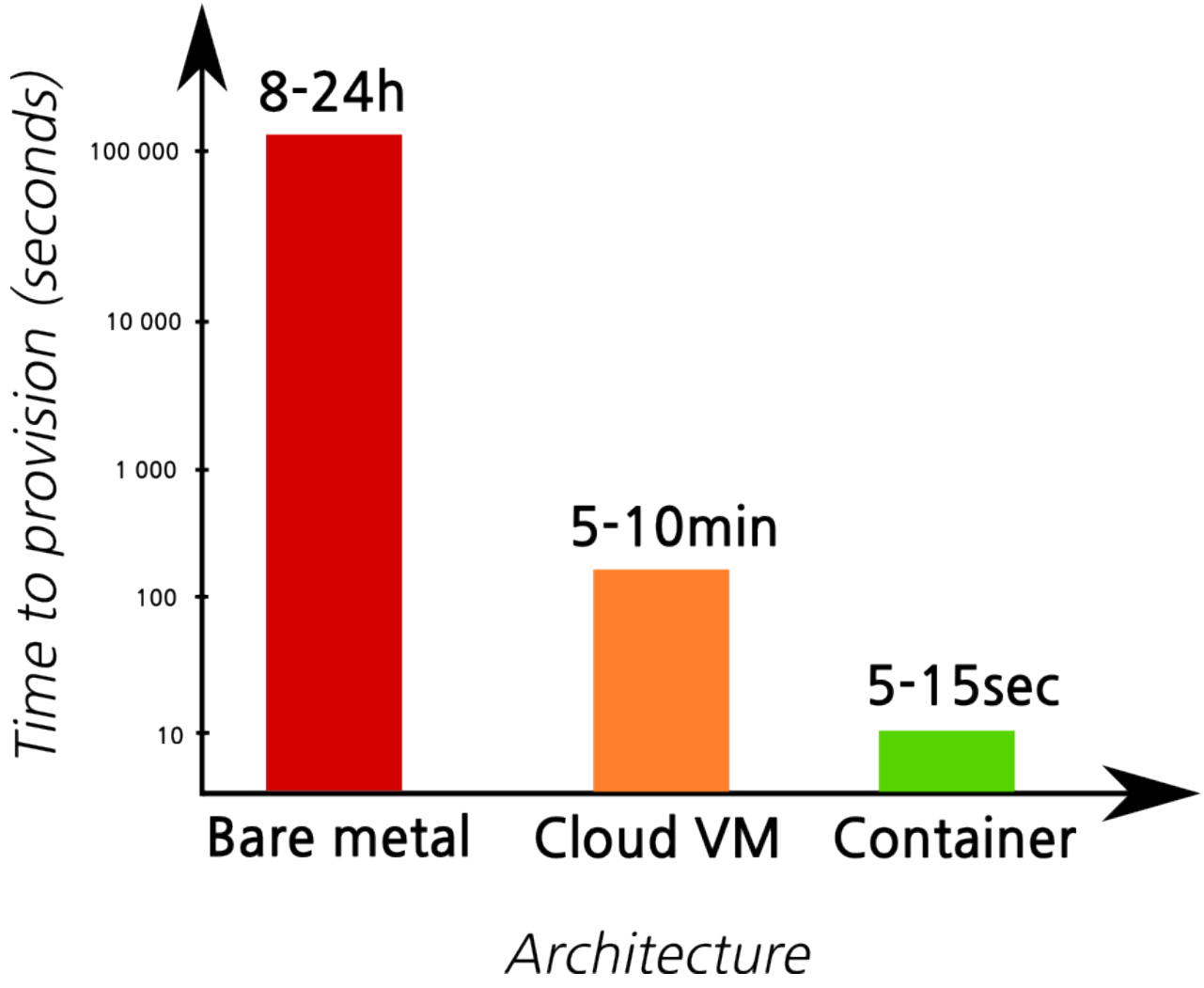
Load balancing

- one server is not enough



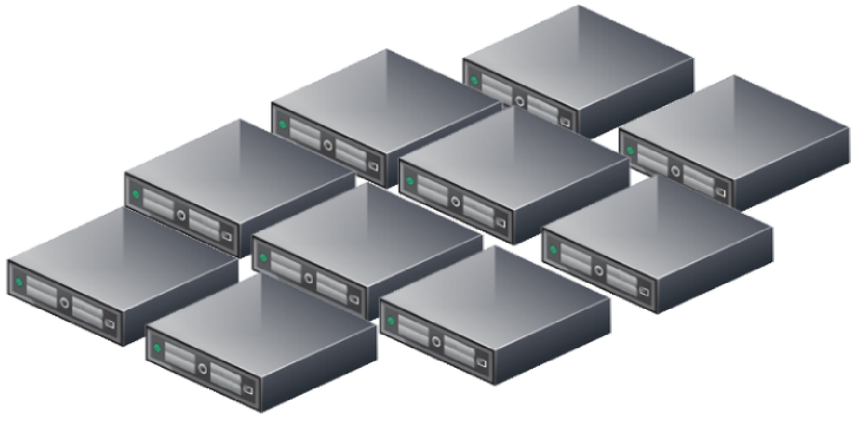
Time to launch new instance

- new server up and runing in a few seconds



Job allocation problem

- run important jobs first



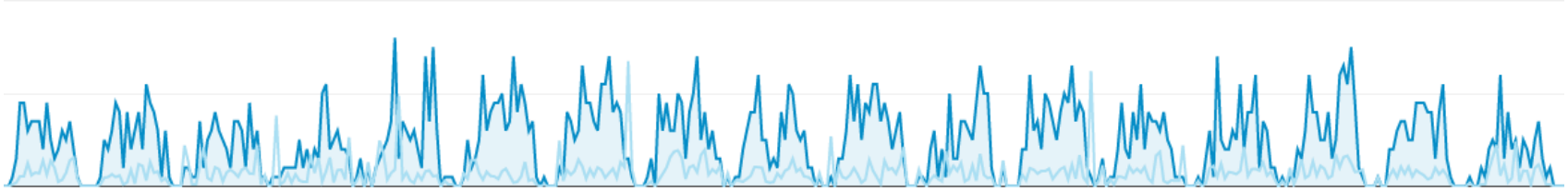
15x service

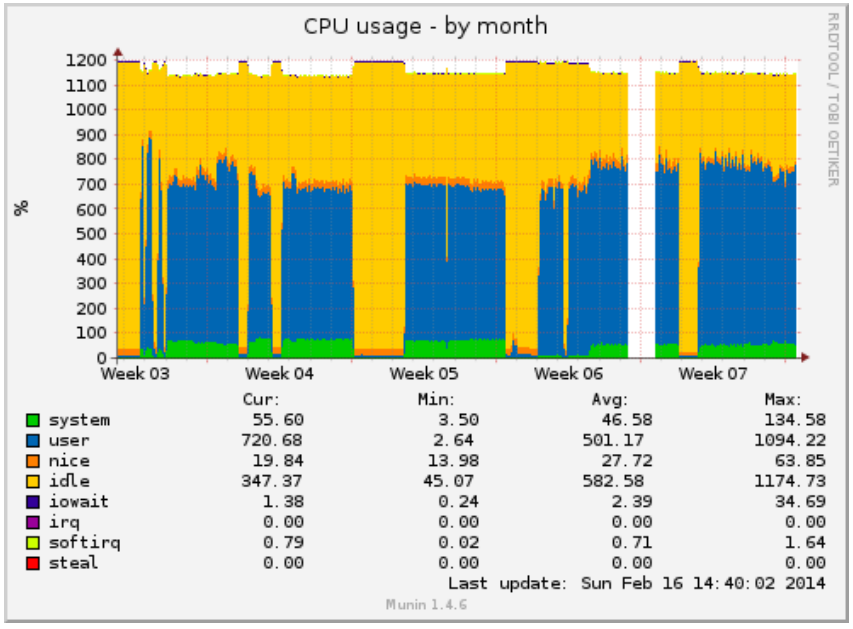
100x batch job

- how many servers do we need?

Load trends

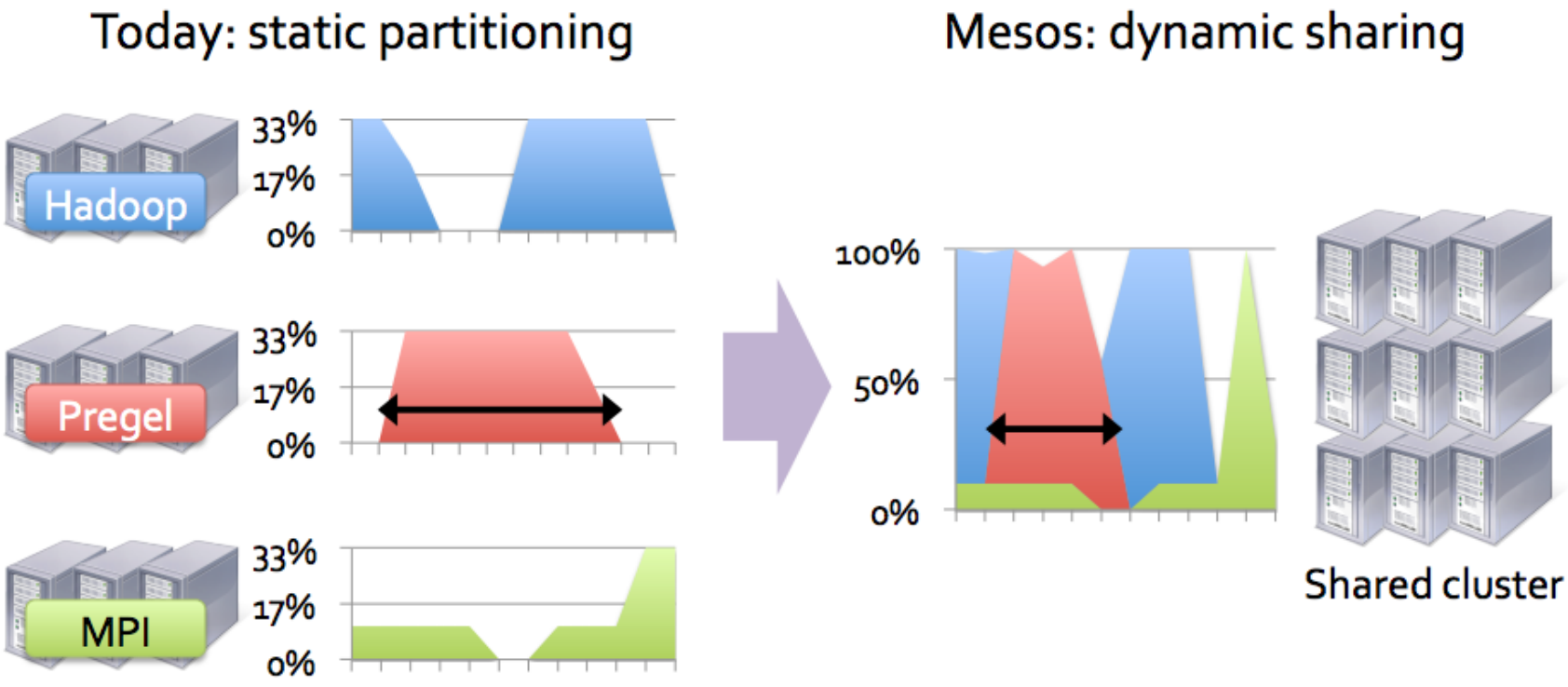
- various load patterns





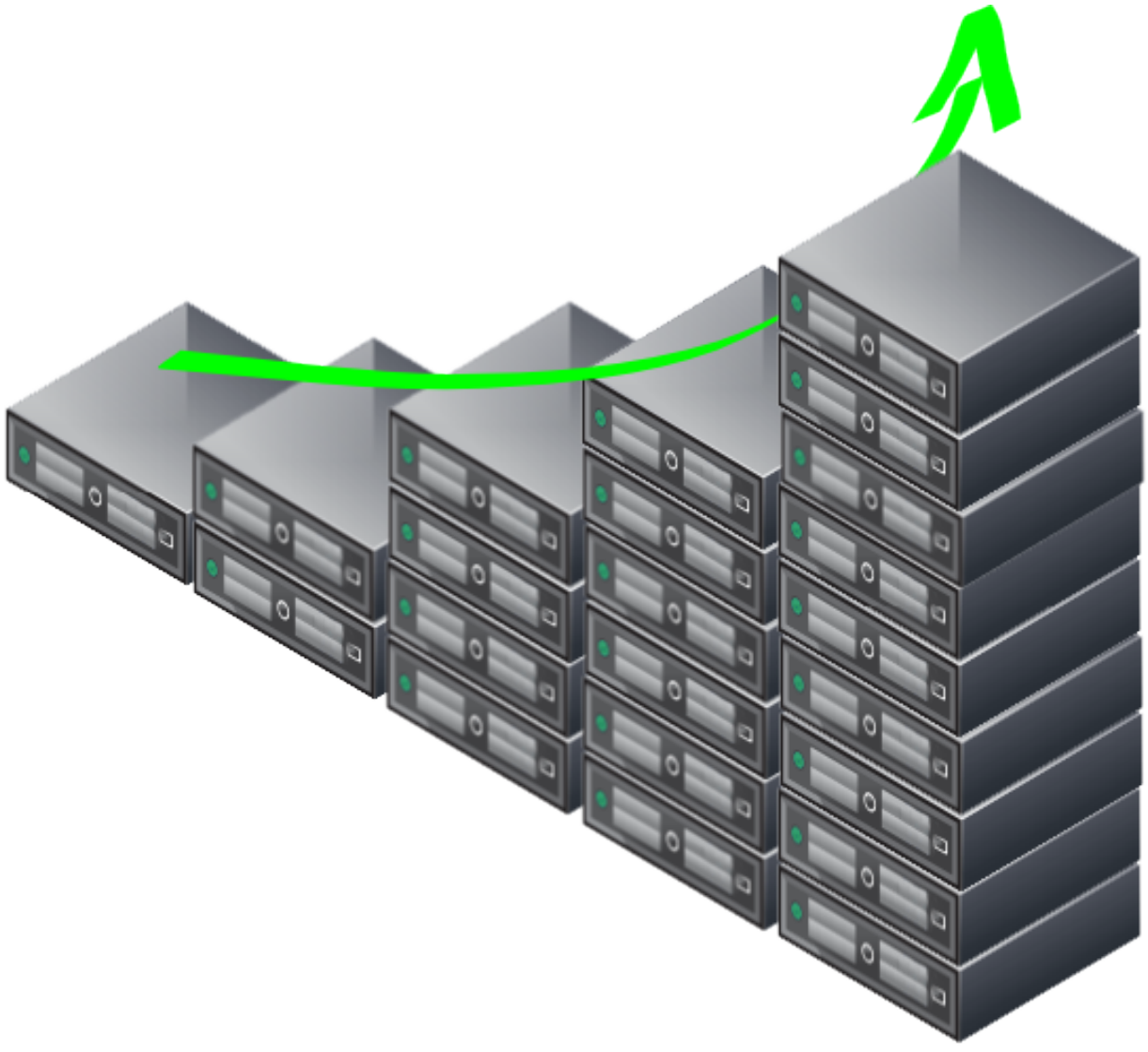
Goals

- effective usage of resources



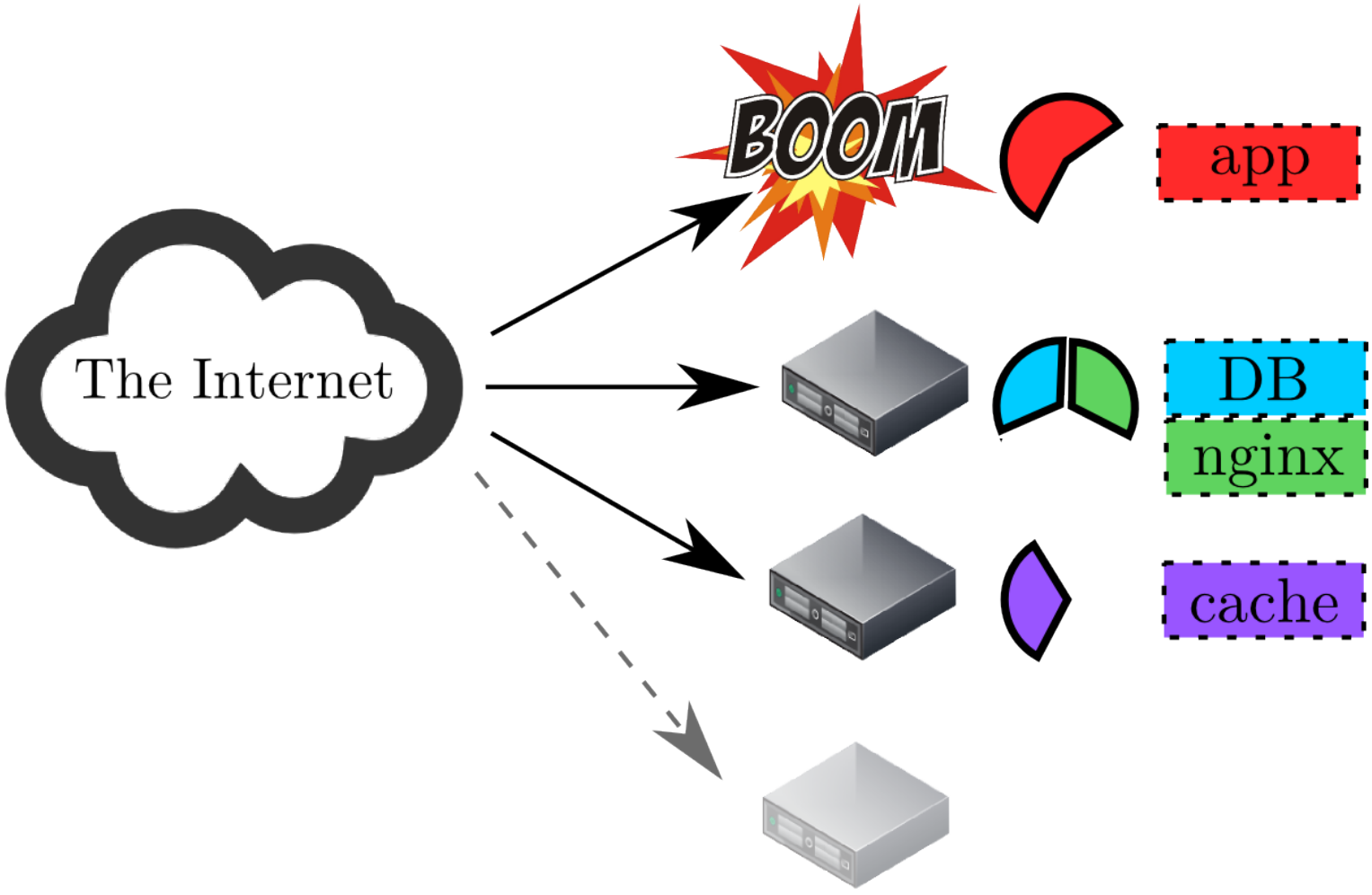
Goals

- scalable



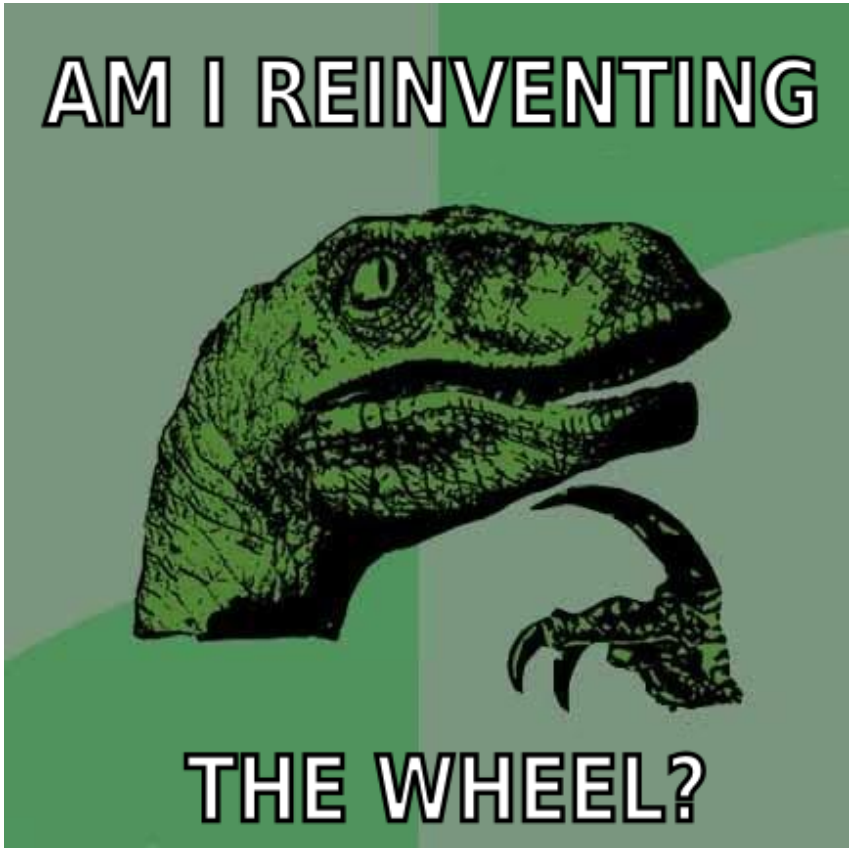
Goals

- fault tolerant



Infrastructure

1. scalable
2. fault-tolerant
3. load balancing
4. high utilization




Someone must
have done it before …


Someone must
have done it before …
Yes, it was Google

Google Research

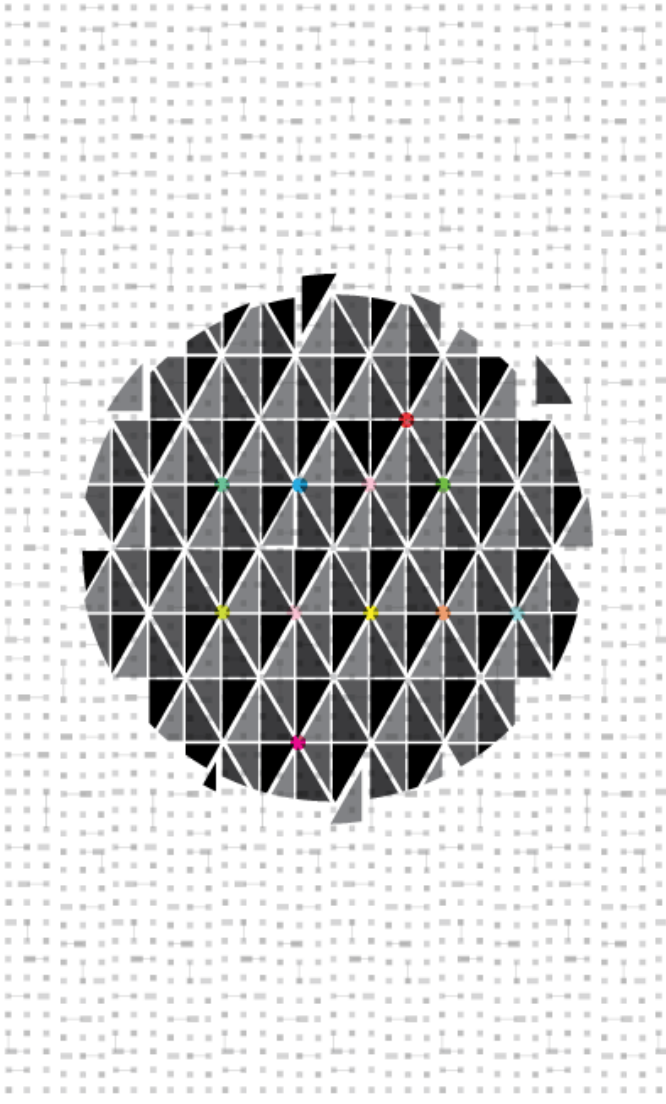
2004 - **MapReduce paper**

- MapReduce: Simplified Data Processing on Large Clusters
  by Jeffrey Dean and Sanjay Ghemawat from Google Lab

$\Rightarrow$ 2005 - **Hadoop**

- by Doug Cutting and Mike Cafarella

Google's secret weapon



"I prefer to call it the system that will not be named."

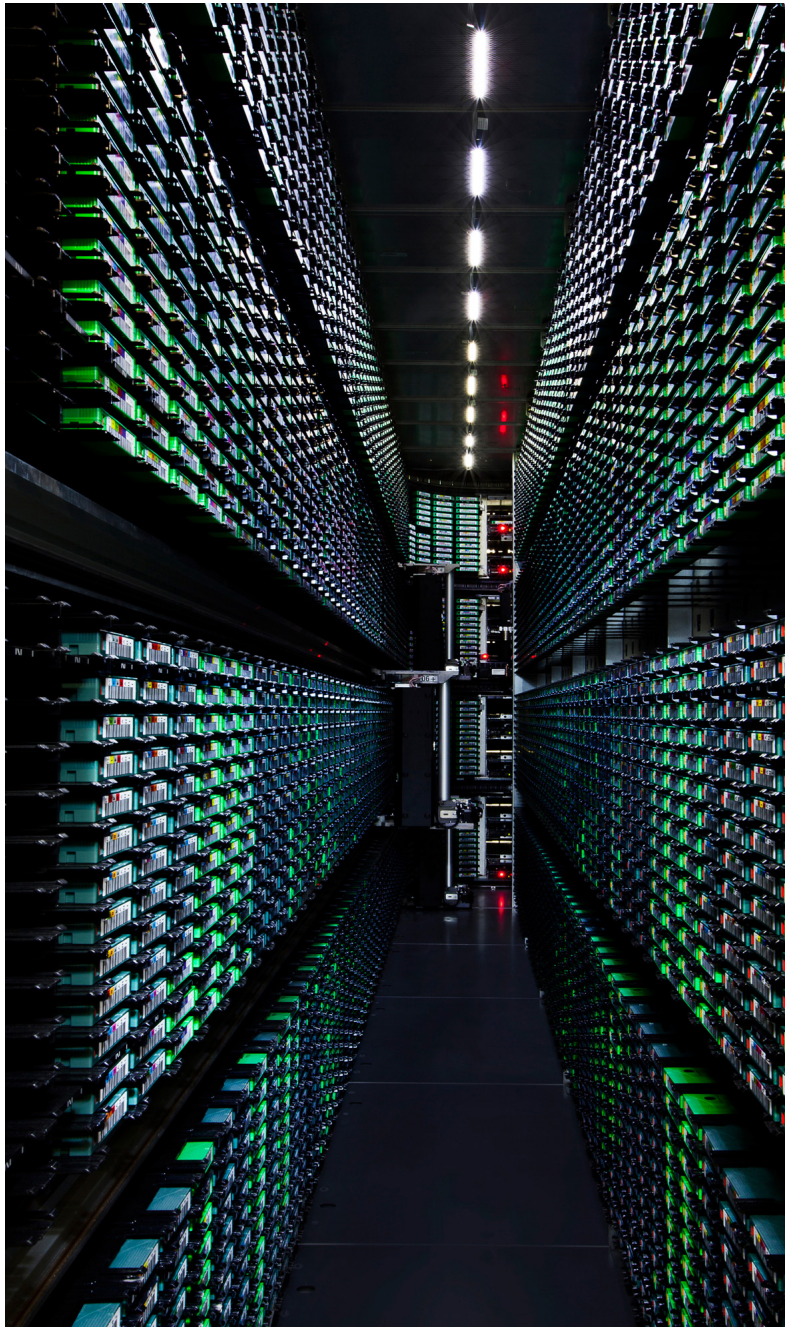
John Wilkes


NOT THIS ONE

“Borg”

unofficial name

- ✓ distributes jobs between computers
- ✓ saved cost of building at least one entire data center
- – centralized, possible bottleneck
- – hard to adjust for different job types



“Borg”

200x – no Borg paper at all

2011 – Mesos: a platform for fine-grained resource sharing in the data center.

- Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Berkeley, CA, USA

amplab UC BERKELEY

The future?

2013 – Omega: flexible, scalable schedulers for large compute clusters

- Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, John Wilkes (SIGOPS European Conference on Computer Systems (EuroSys), ACM, Prague, Czech Republic (2013), pp. 351-364)

- compares different schedulers
  - monolithic (Borg)
  - Mesos (first public release, version 0.9)
  - Omega (next generation of Borg)

Benjamin Hindman

- PhD. at UC Berkley
- research on multi-core processors

"Sixty-four cores or 128 cores on a single chip looks a lot like 64 machines or 128 machines in a data center"

"We wanted people to be able to program for the data center just like they program for their laptop."

Evolution of computing

Datacenter

- low utilization of nodes
- long time to start new node (30 min – 2 h)

Evolution of computing

Datacenter

Virtual Machines

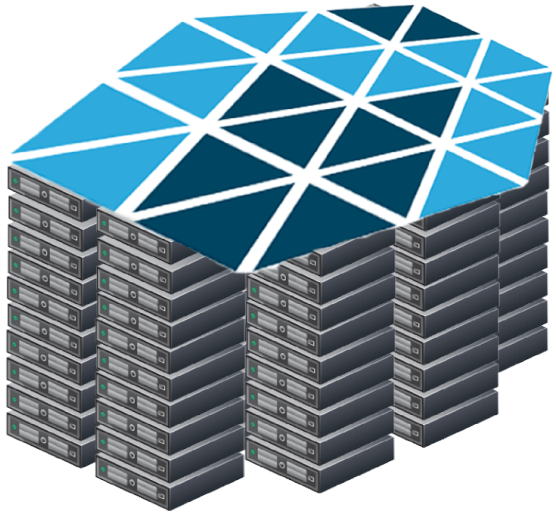
- even more machnines to manage
- high virtualization costs
- VM licensing

Evolution of computing

Datacenter

Virtual Machines

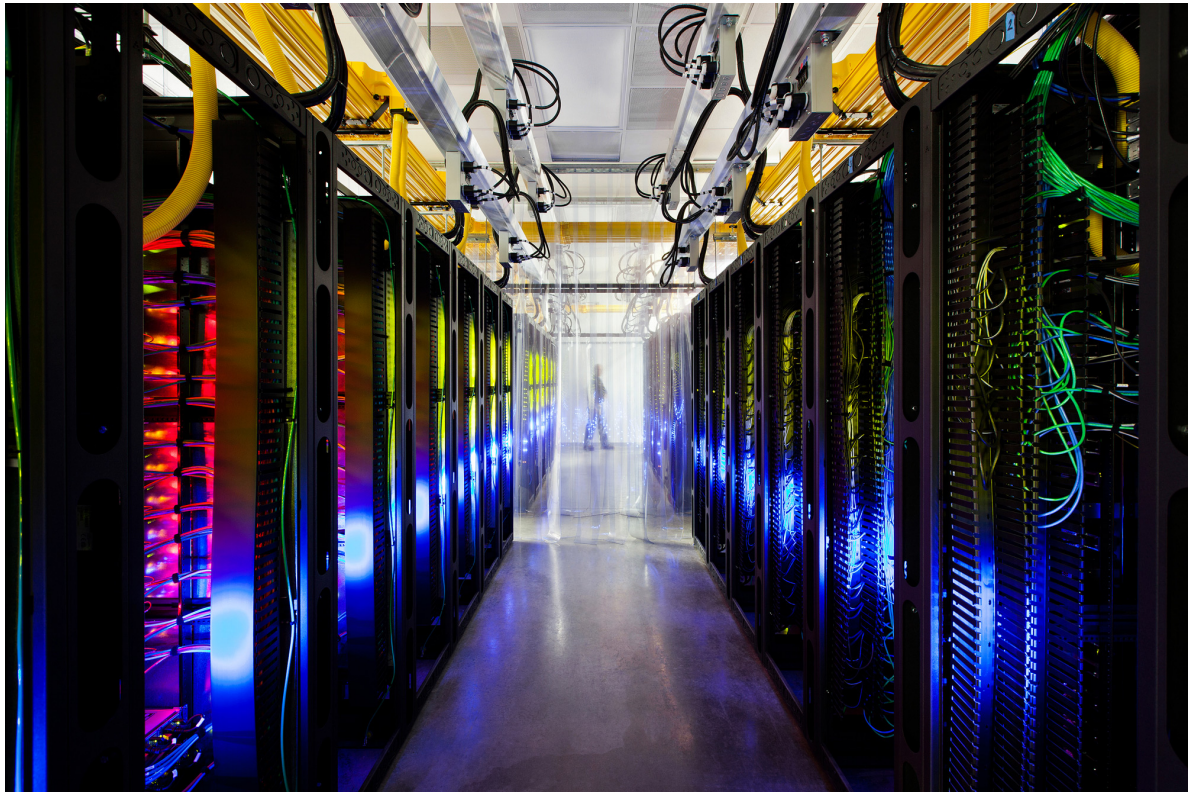
Mesos



- sharing resources
- fault-tolerant

Supercomputers

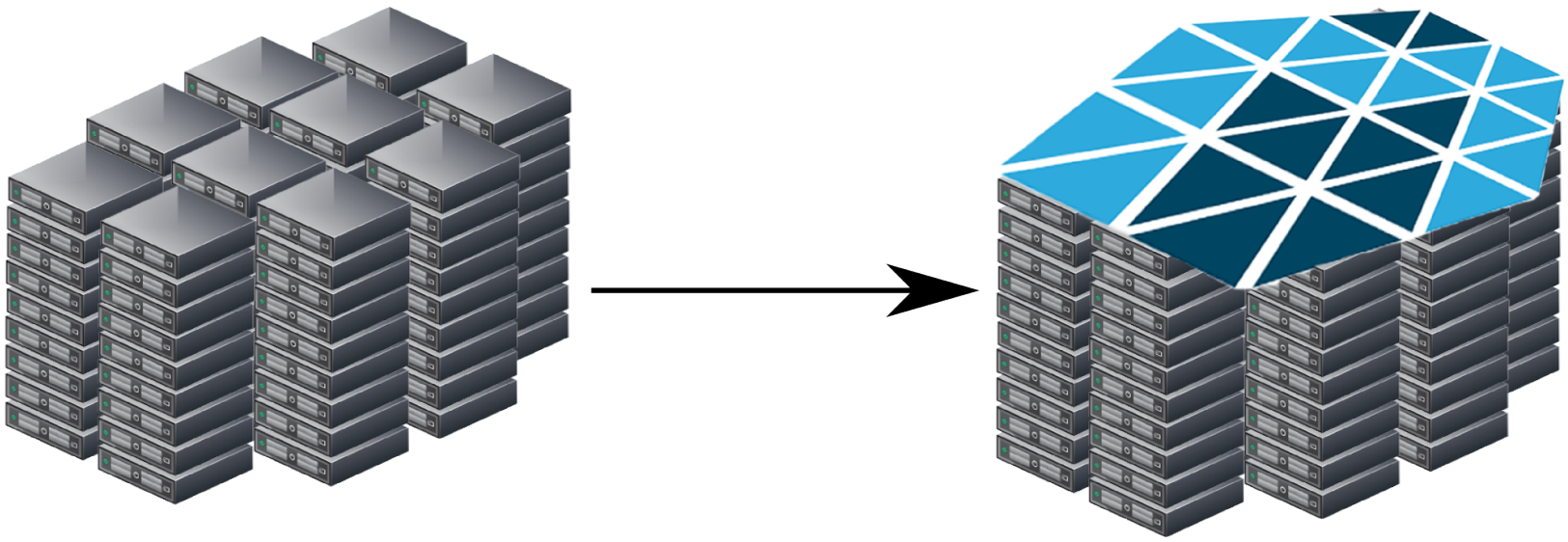
- supercomputers aren't affordable
- single point of failure?



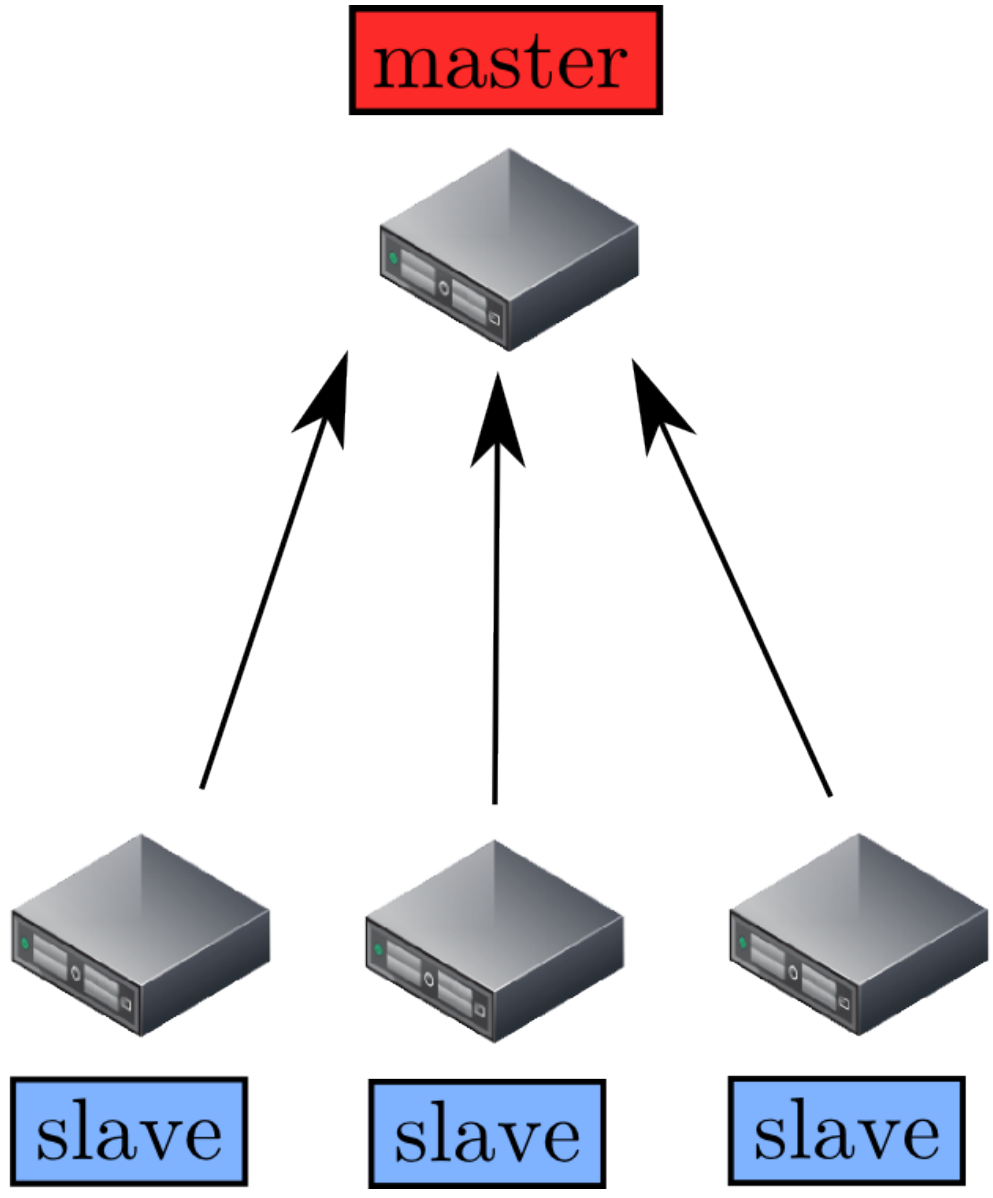
IaaC

Infrastructure as a computer

- build on commodity hardware



Scalablity



First Rule of Distributed Computing

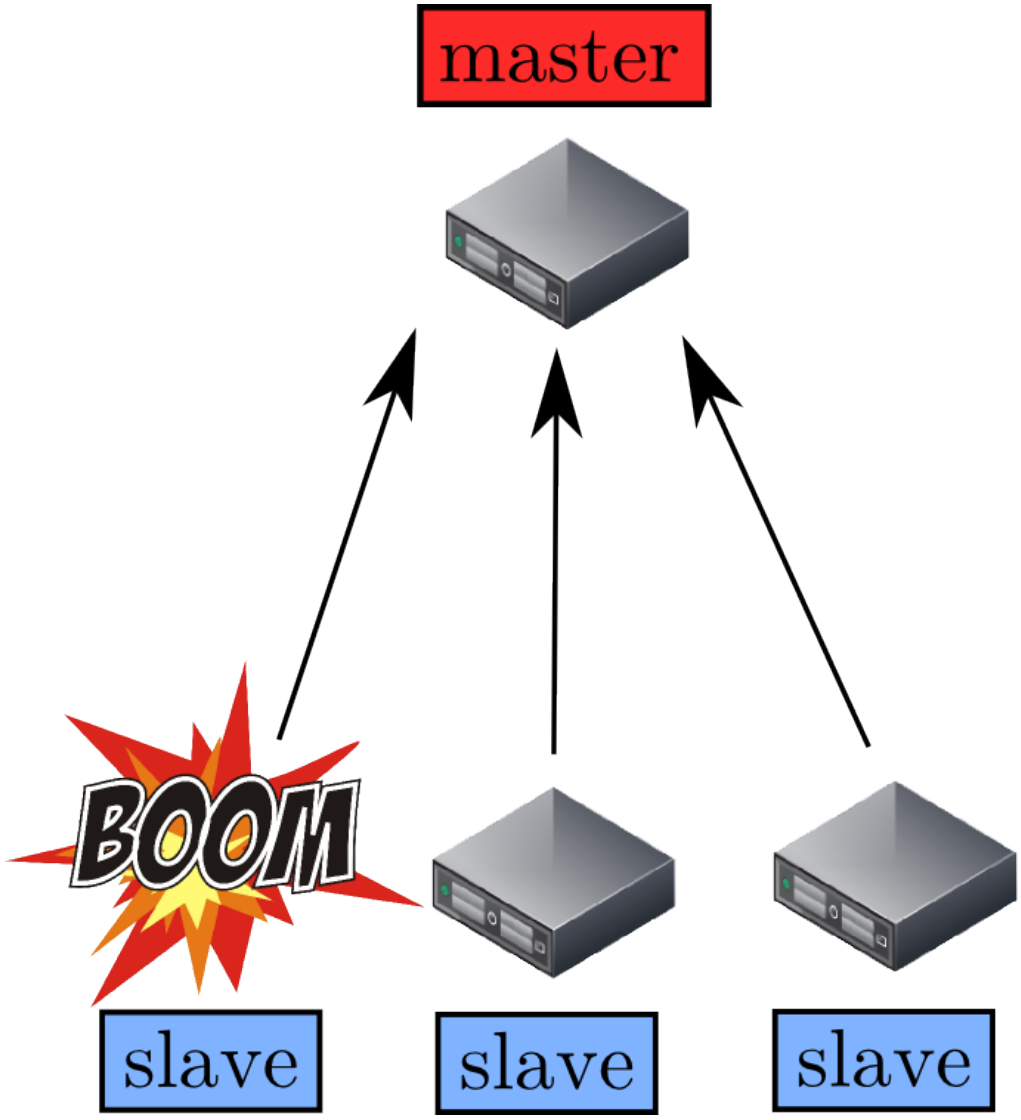
"Everything fails all the time."

— Werner Vogels, CTO of Amazon

```
[1526689.666251] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000
00
[1526689.725426] Process java (pid: 24365, threadinfo ffff8803851e6000, task ff
f8803284d8000)
[1526689.784462] Stack:
[1526689.812815]  ffff88040e233ec0 0000000000000000 ffff88040e223ed0 ffffffff81
03567
[1526689.870570]  ffff88040e22e8b8 ffffffff81c1af00 ffff88040e223f18 ffffffff81
03b45
[1526689.929768]  ffffffff810b00dc ffff88040e223f38 ffff88040e22e800 0000000000
00000
[1526689.989048] Call Trace:
[1526690.018031]  <IRQ>
[1526690.018375]  [<ffffffff810b3567>] get_posix_clock.isra.0+0x27/0x50
[1526690.075394]  [<ffffffff810b3b45>] posix_clock_poll+0x35/0x80
[1526690.103702]  [<ffffffff810b00dc>] ? ktime_get_update_offsets+0x4c/0xd0
[1526690.131720]  [<ffffffff810841ef>] hrtimer_interrupt+0x6f/0x240
[1526690.159145]  [<ffffffff816ffd19>] smp_apic_timer_interrupt+0x69/0x99
[1526690.186251]  [<ffffffff816fec5d>] apic_timer_interrupt+0x6d/0x80
[1526690.212862]  <EOI>
[1526690.213208] Code: 6e dd c6 ff 48 83 c4 08 5b 5d c3 0f 1f 80 00 00 00 00 66
66 66 66 90 55 48 89 e5 53 48 89 fb 48 83 ec 08 e8 6a 0a 00 00 48 89 d8 <f0> 48
ff 00 79 05 e8 0c dd c6 ff 48 83 c4 08 5b 5d c3 55 48 8d
[1526690.317237] RIP  [<ffffffff816f3439>] down_read+0x19/0x2b
[1526690.342569]  RSP <ffff88040e223ea0>
[1526690.367420] CR2: 00000000000000f0
[1526690.425275] ---[ end trace fec7704b6488d8b5 ]---
[1526691.430539] Kernel panic - not syncing: Fatal exception in interrupt
```

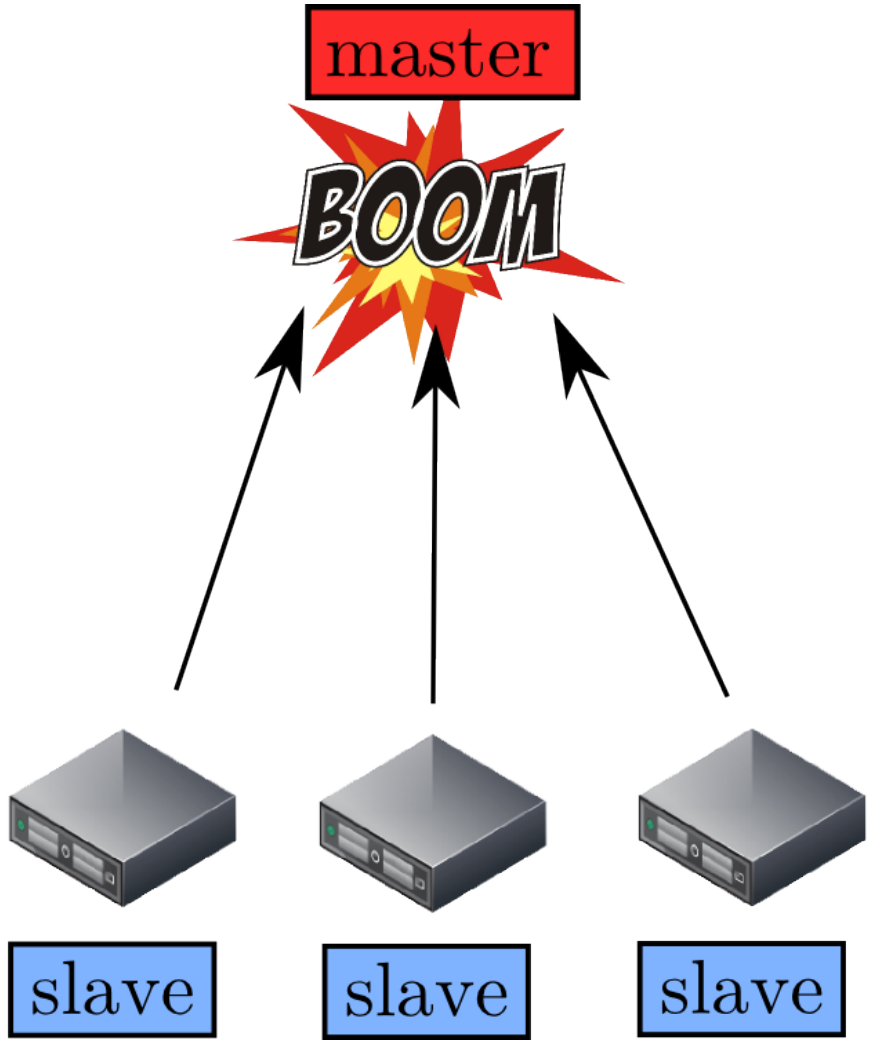
Slave failure

- no problem



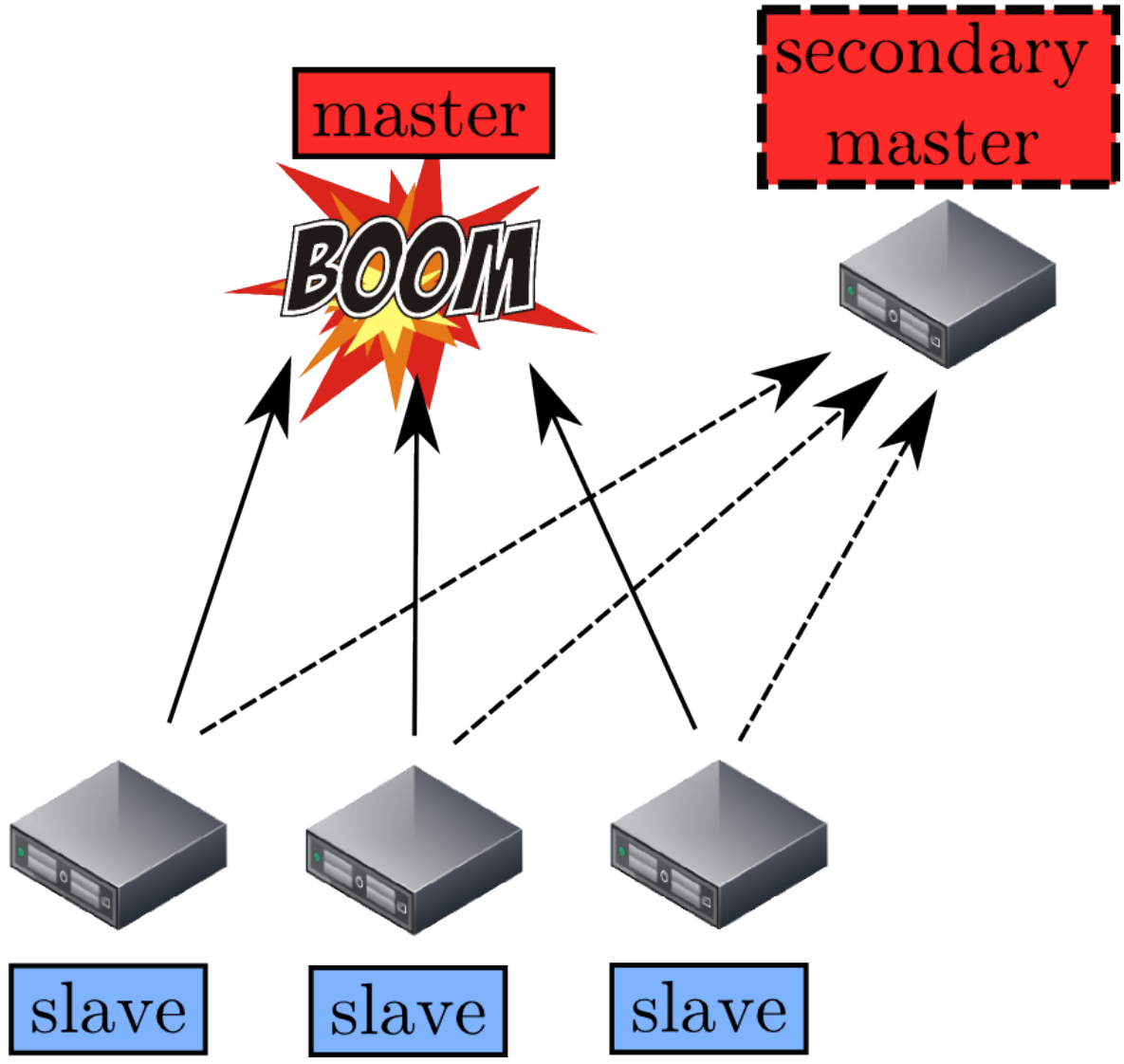
Master failure

- big problem
- single point of failure

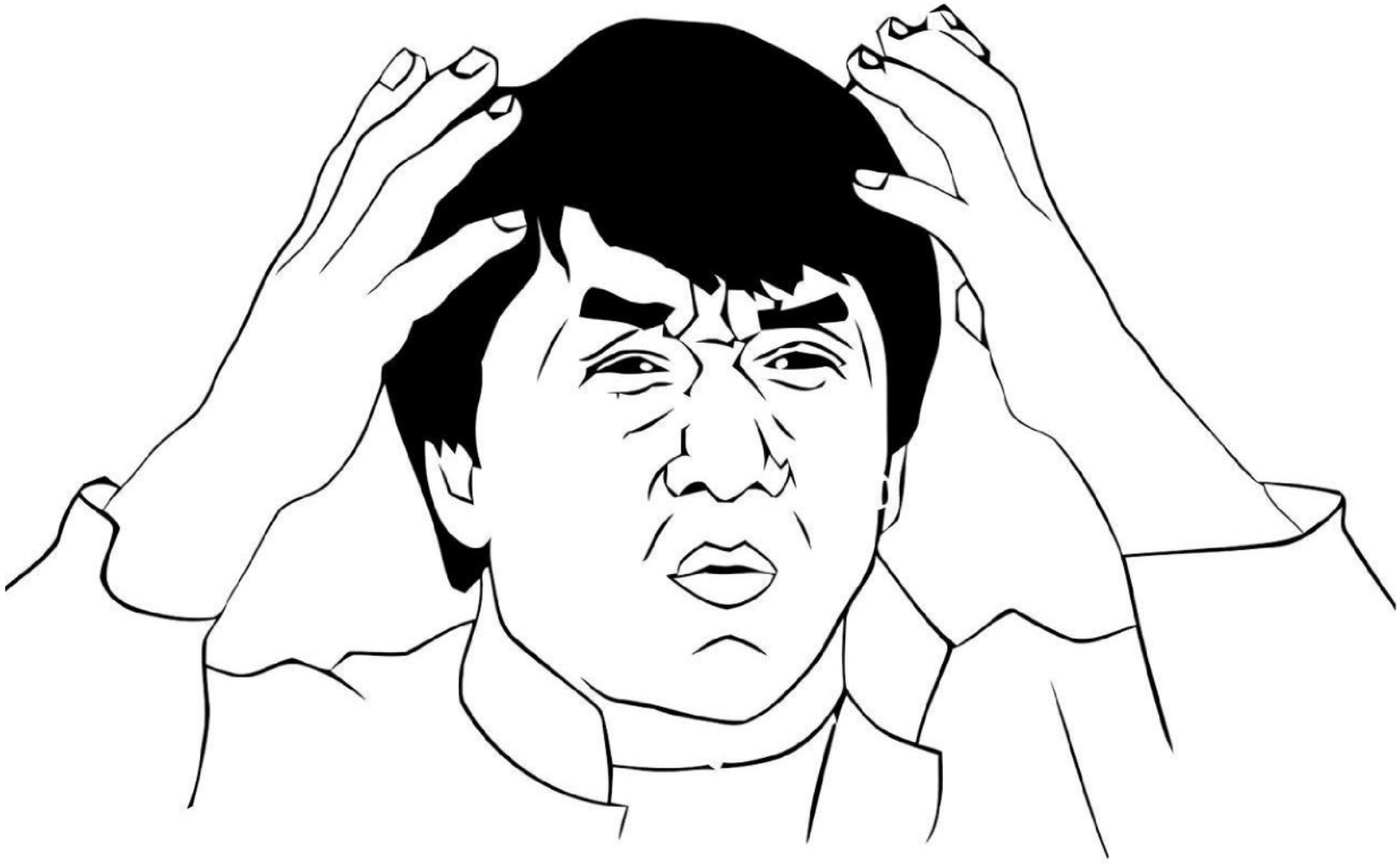


Master failure – solution

- ok, let's add secondary master

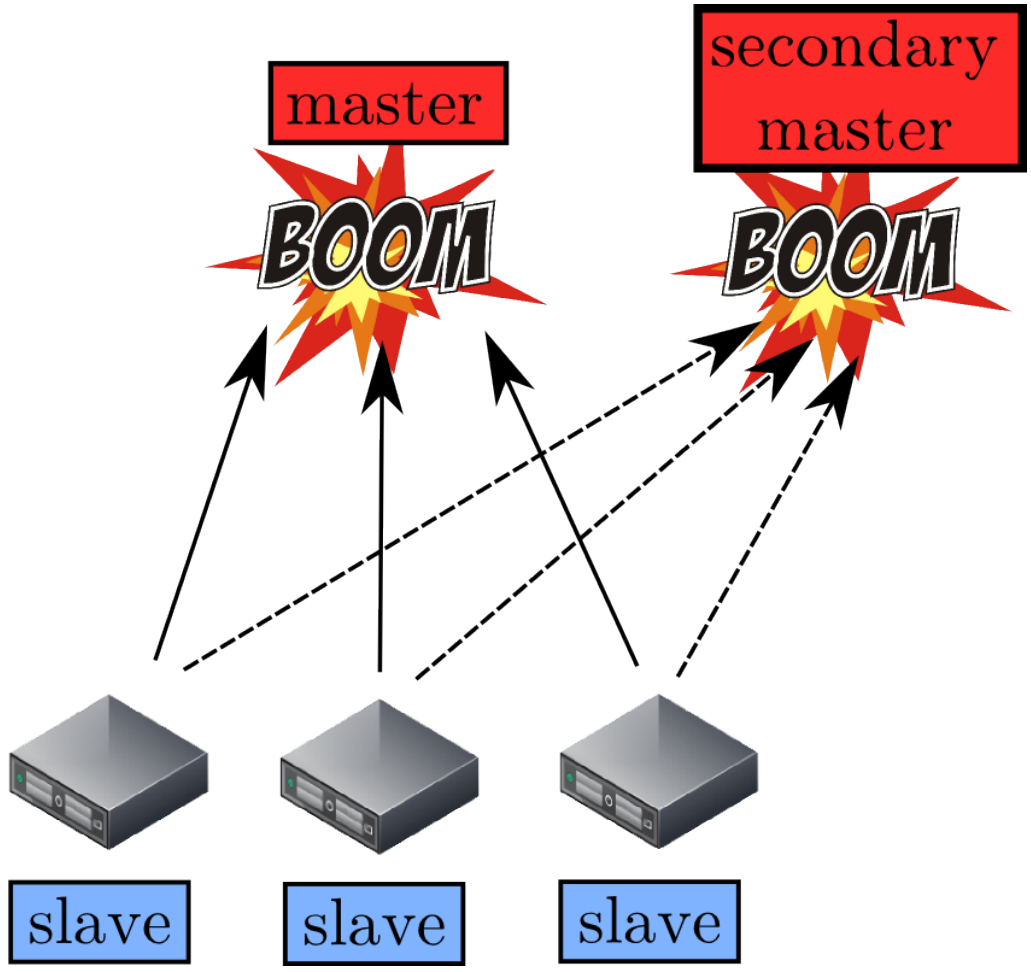


OMG!



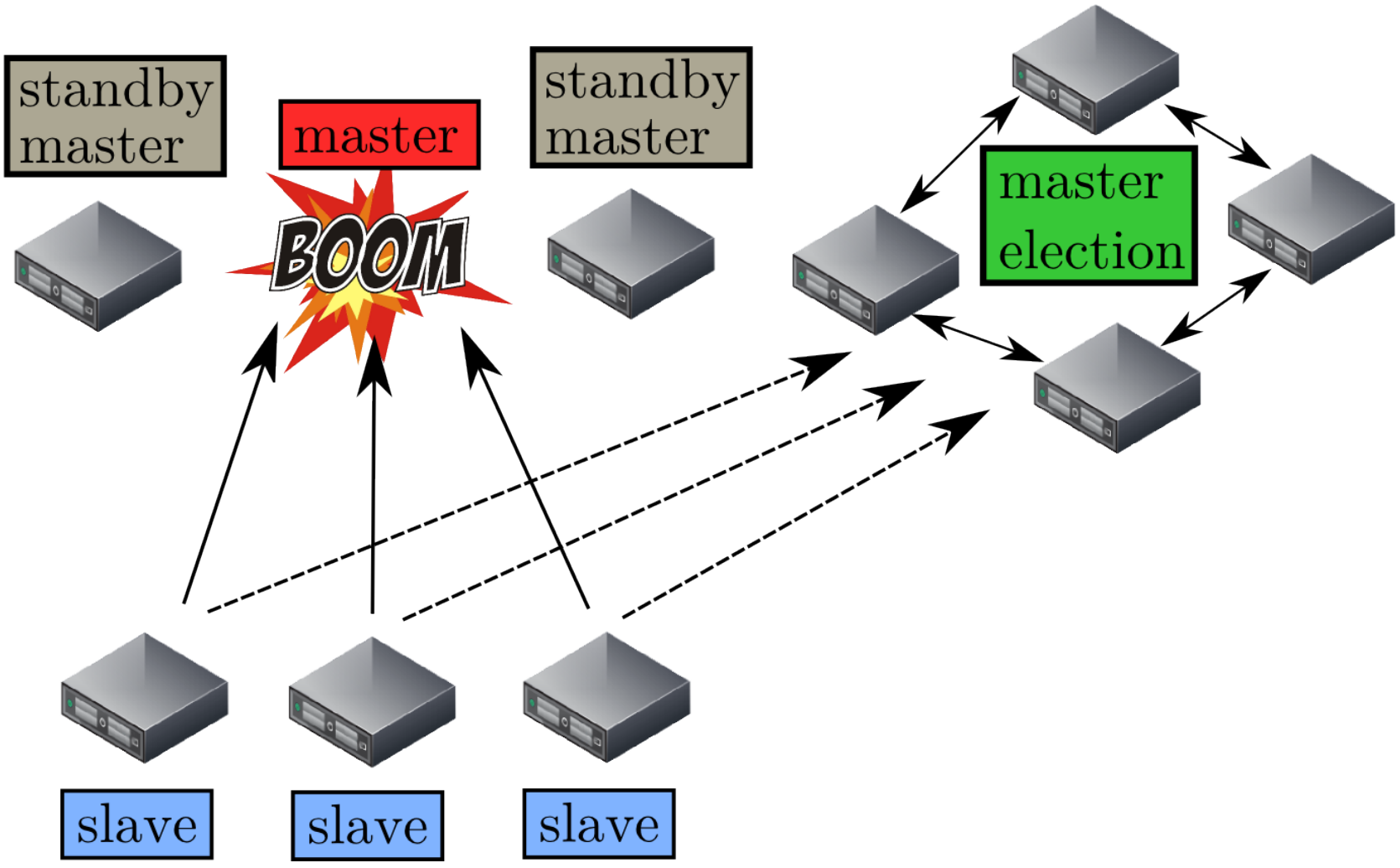
That's like mid 2000's!

Secondary master failure



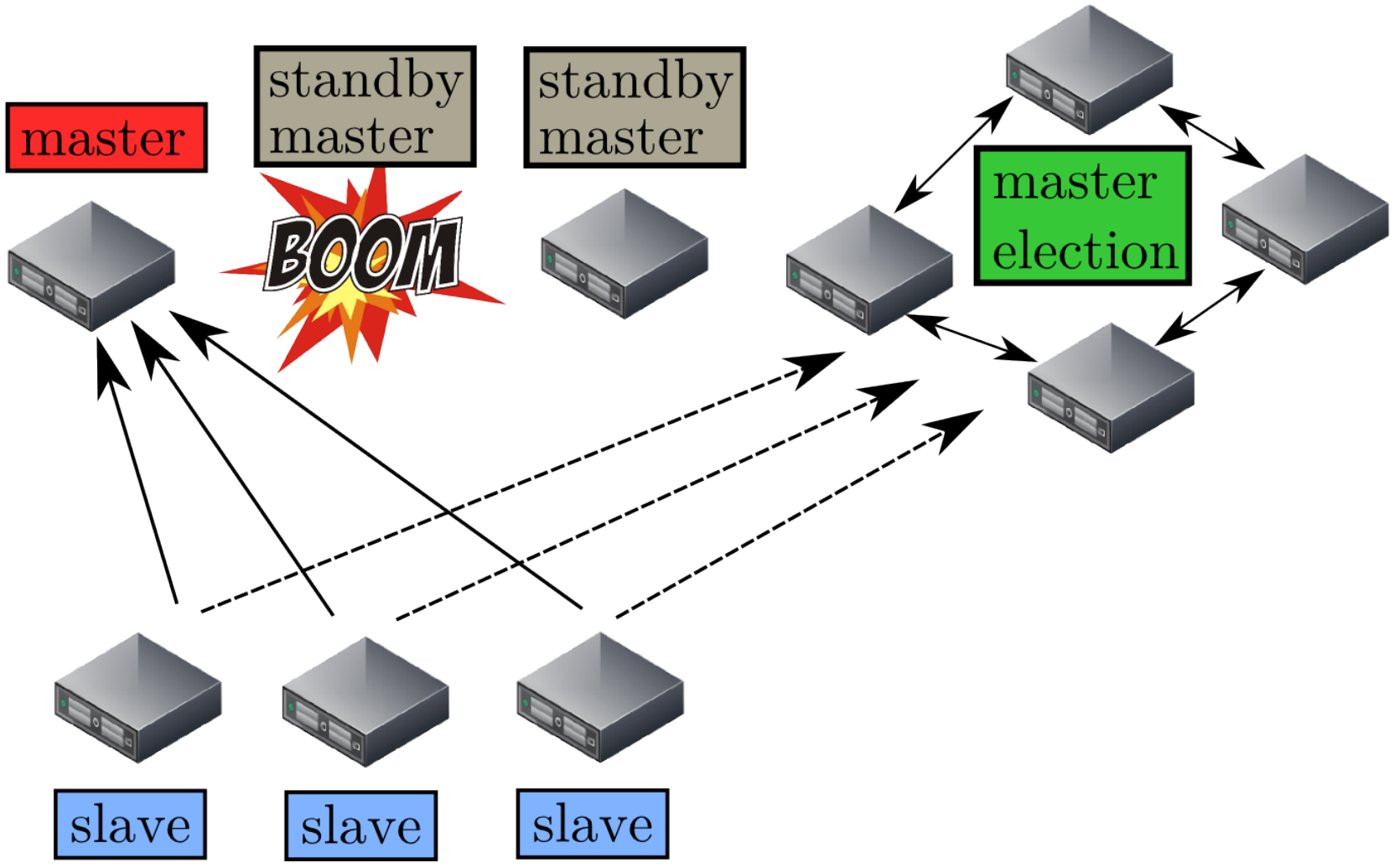
- × not resistant to secondary master failure
- × masters IPs are stored in slave's config
- • will survive just 1 server failure

Leader election



✓ in case of master failure new one is elected

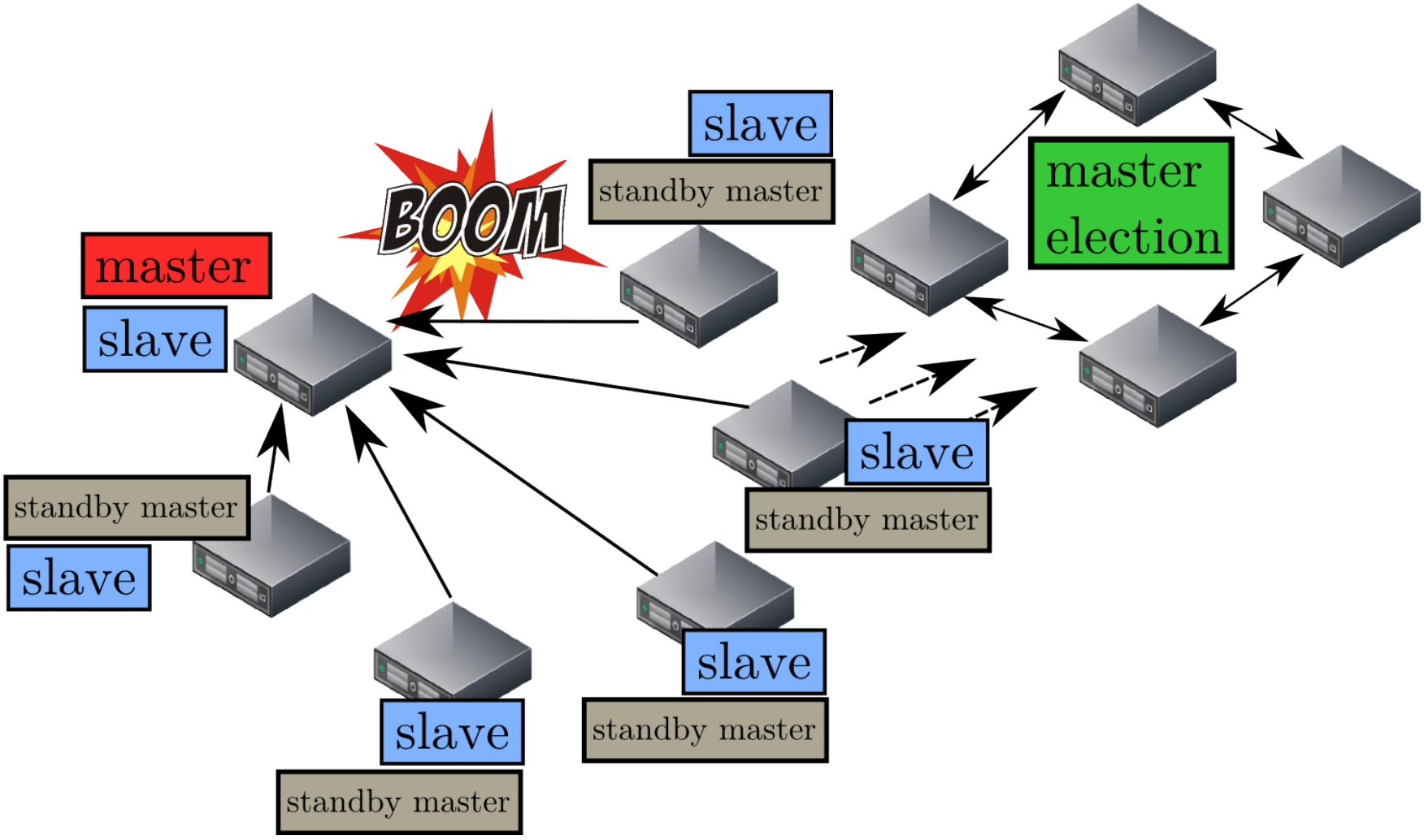
Leader election



✓ in case of master failure new one is elected

Leader election

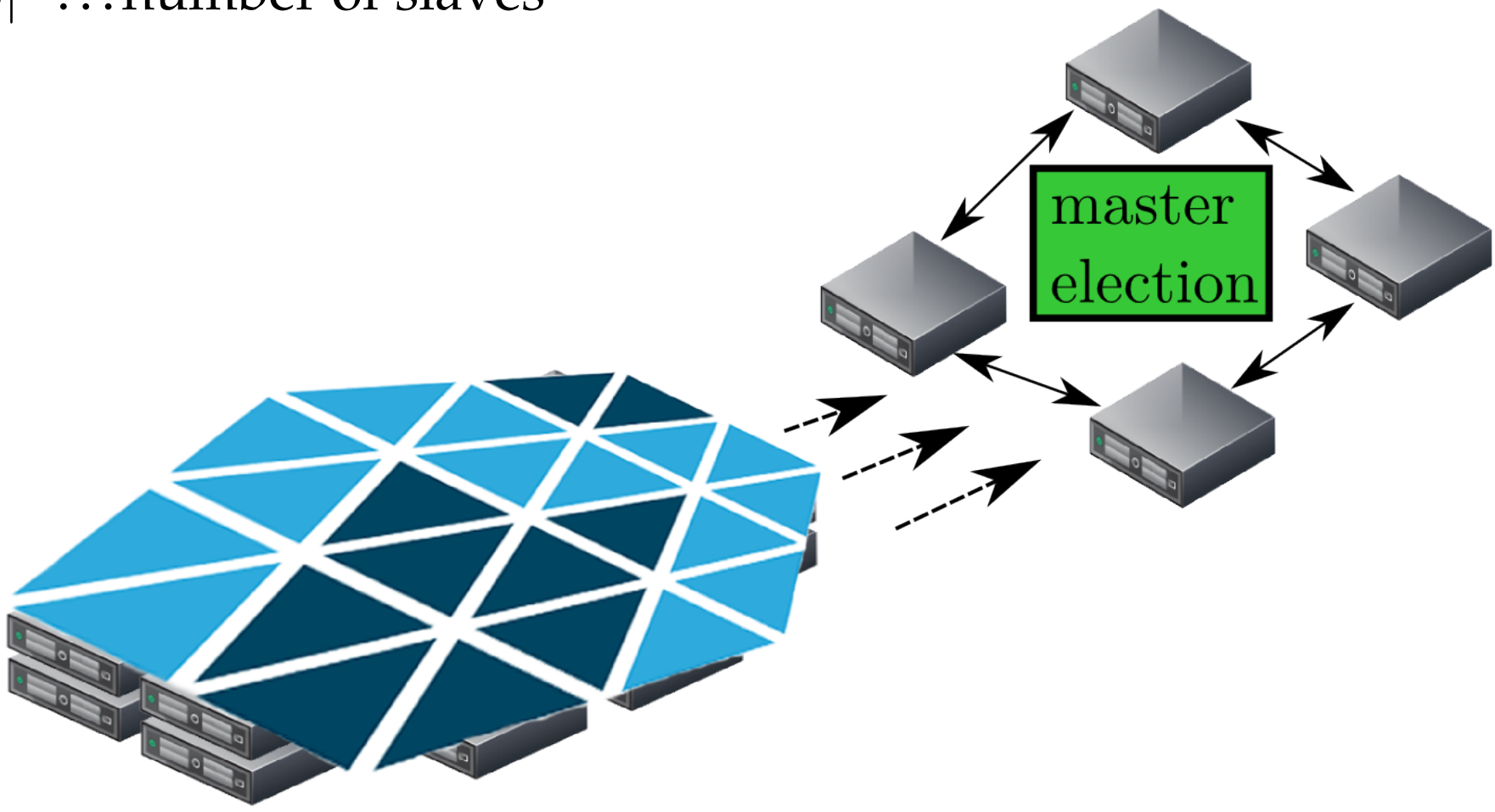
- you might think of the system like this:



Mesos scheme

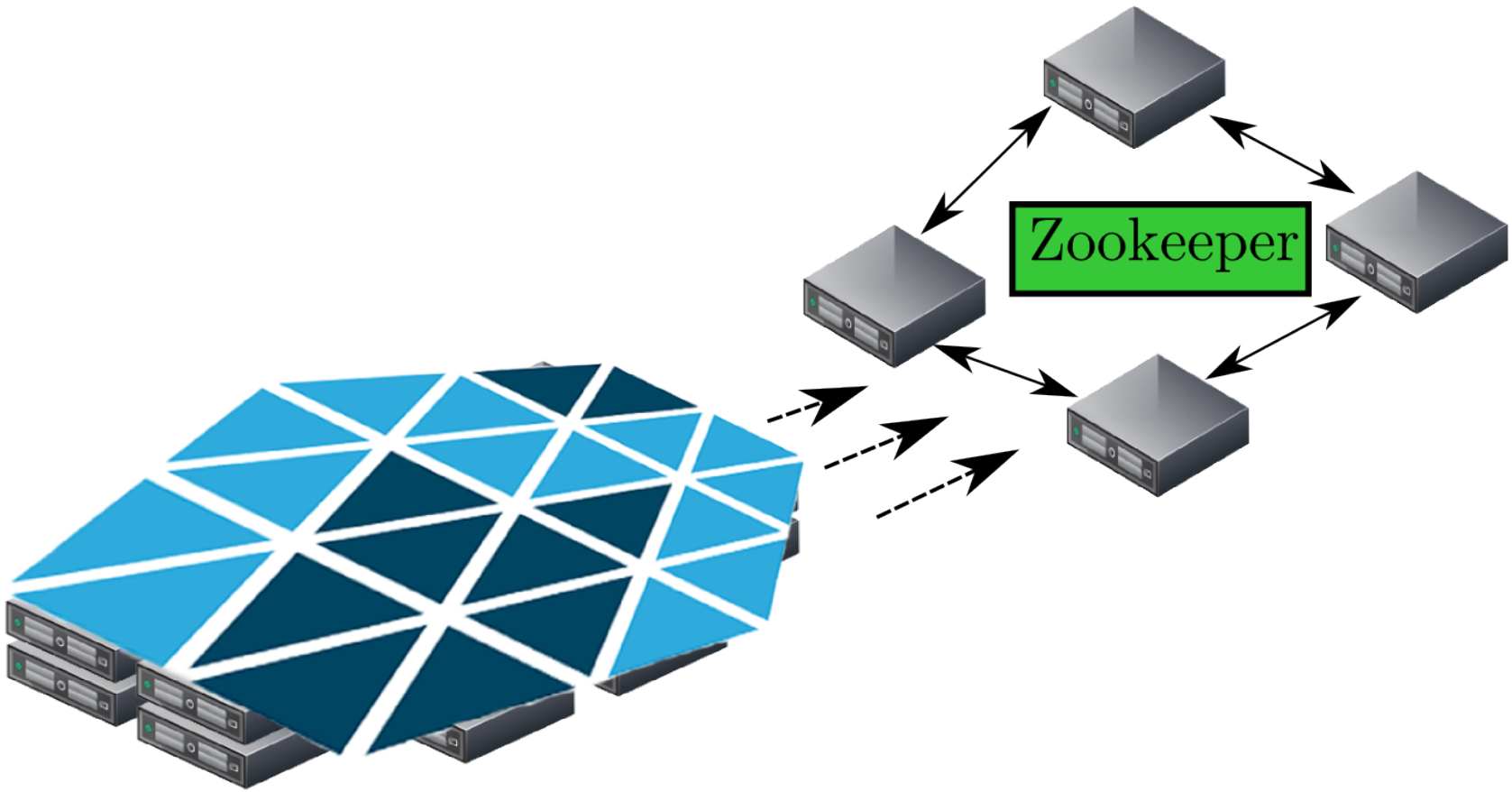
- usually 4-6 standby masters are enough
- tolerant to $|m|$ failures, $|s| \gg |m|$

$|m|$ . . . number of standby masters
$|s|$ . . . number of slaves



Mesos scheme

- each slave obtain master's IP from Zookeeper
- e.g. `zk://192.168.1.1:2181/mesos`



Common delusion

1. Network is reliable
2. Latency is zero
3. Transport cost is zero
4. The Network is homogeneous

Cloud Computing

design you application for failure

- split application into multiple components
- every component must have redundancy
- no common points of failure

"If I seen further than others it is by standing upon the shoulders of giants."

Sir Isaac Newton

ZooKeeper

ZooKeeper


hadoop

mahout


Apache
Solr


HIVE


MESOS


Spark
Lightning-Fast Cluster Computing


ZOO

ZooKeeper


hadoop
mahout
airbnb
YAHOO!
Apache
Solr
HIVE
facebook
NETFLIX
MESOS
Spark
Lightning-Fast Cluster Computing
LinkedIn
YouTube
twitter


ZOO

ZooKeeper

- not a key-value storage
- not a filesystem
- 1 MB item limit

Frameworks

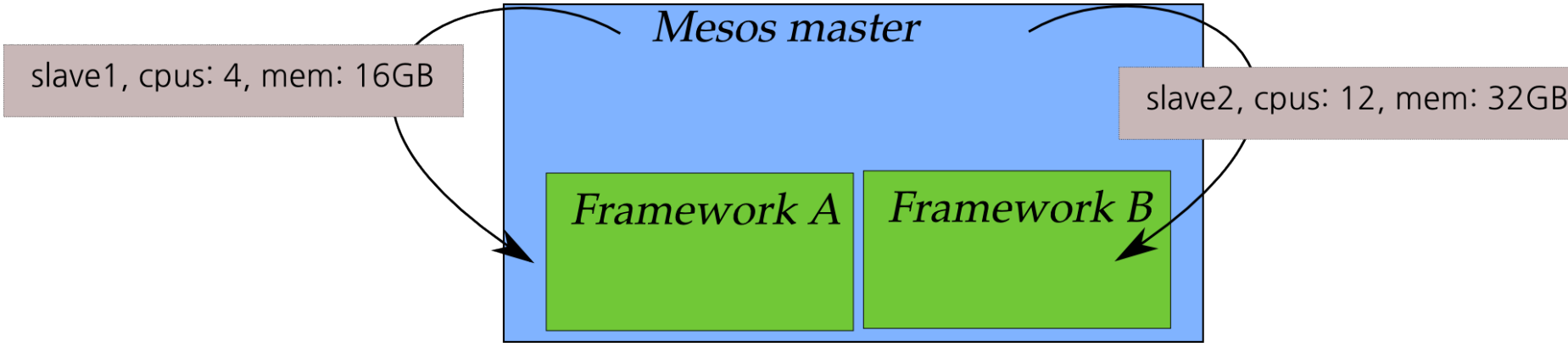
- **framework** – application running on Mesos

two components:

1. scheduler
2. executor

Scheduling

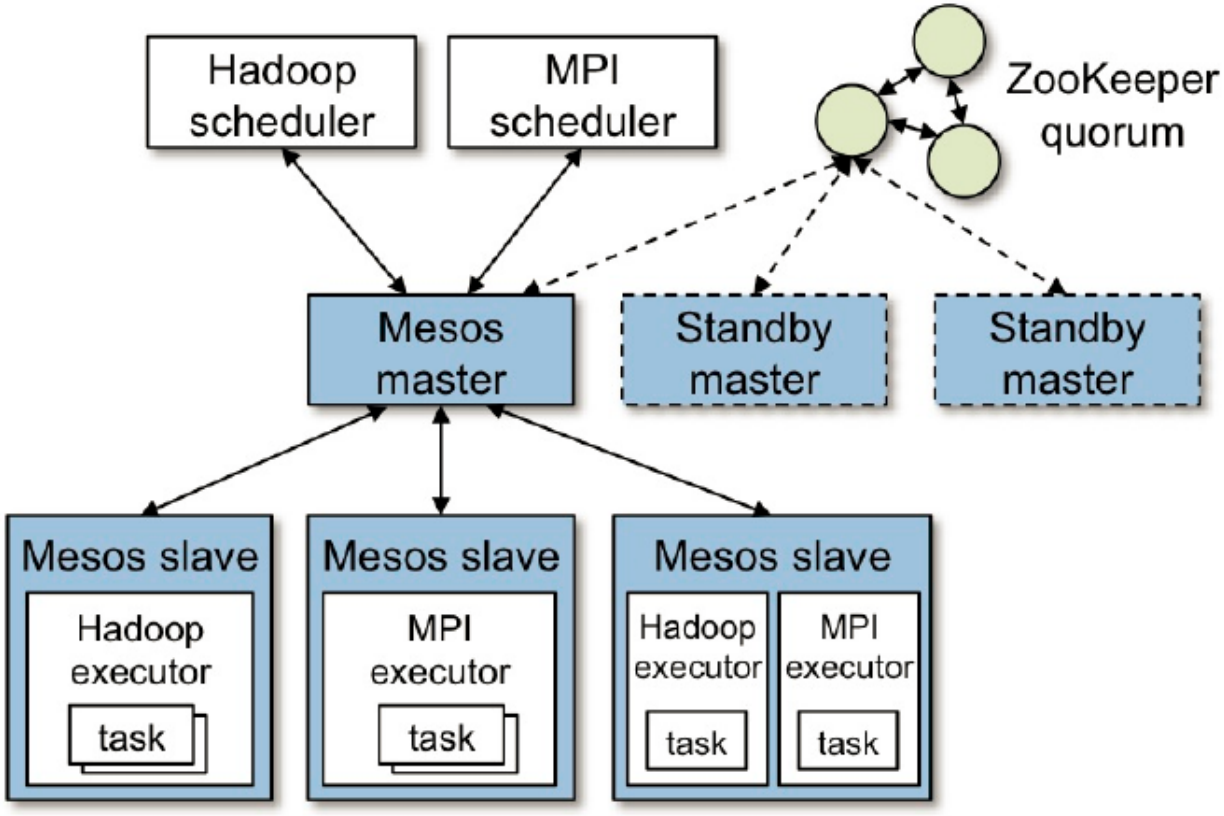
- two levels scheduler



- Dominant Resource Fairness

Mesos architecture

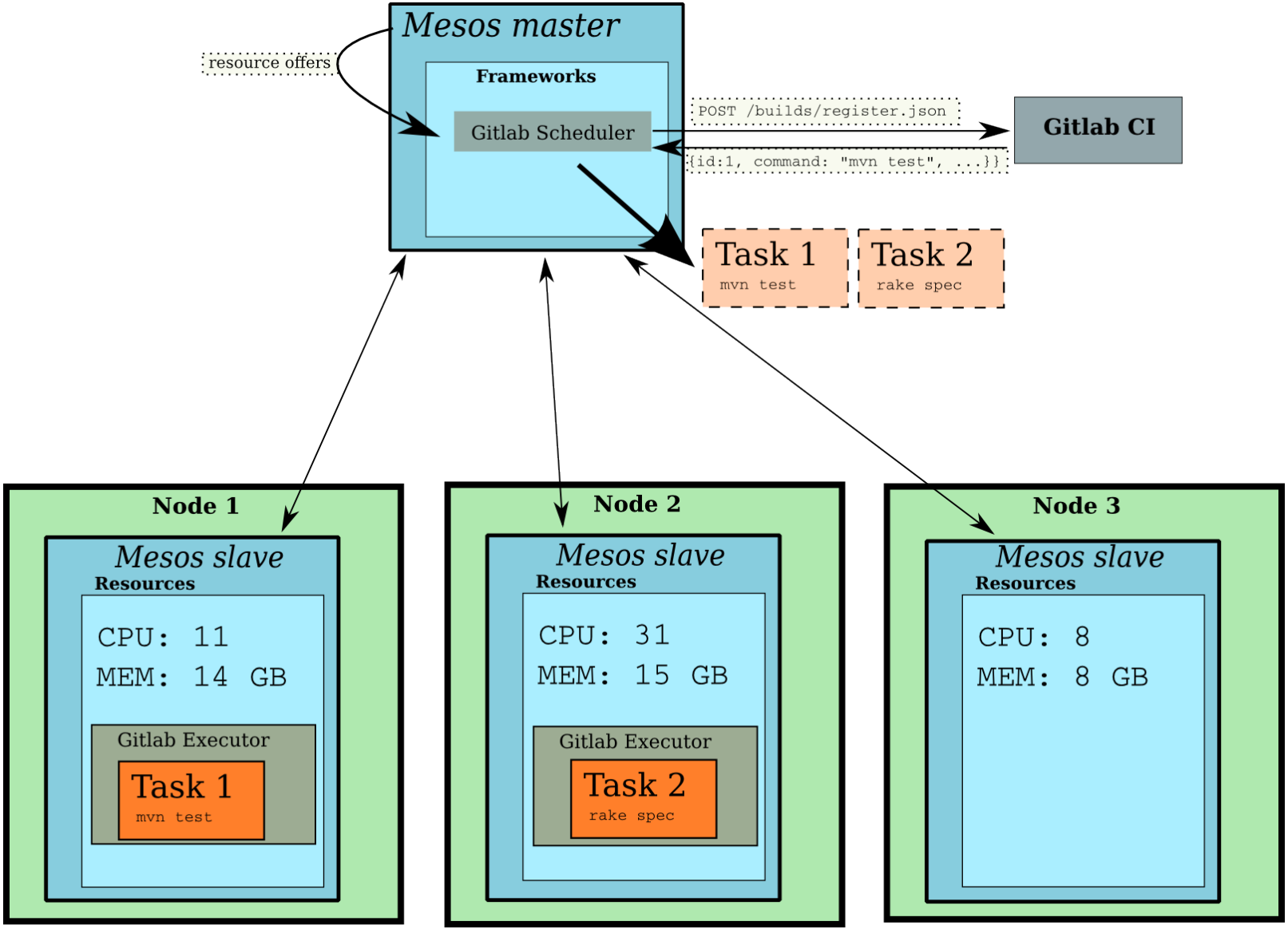
- fault tolerant
- scalable

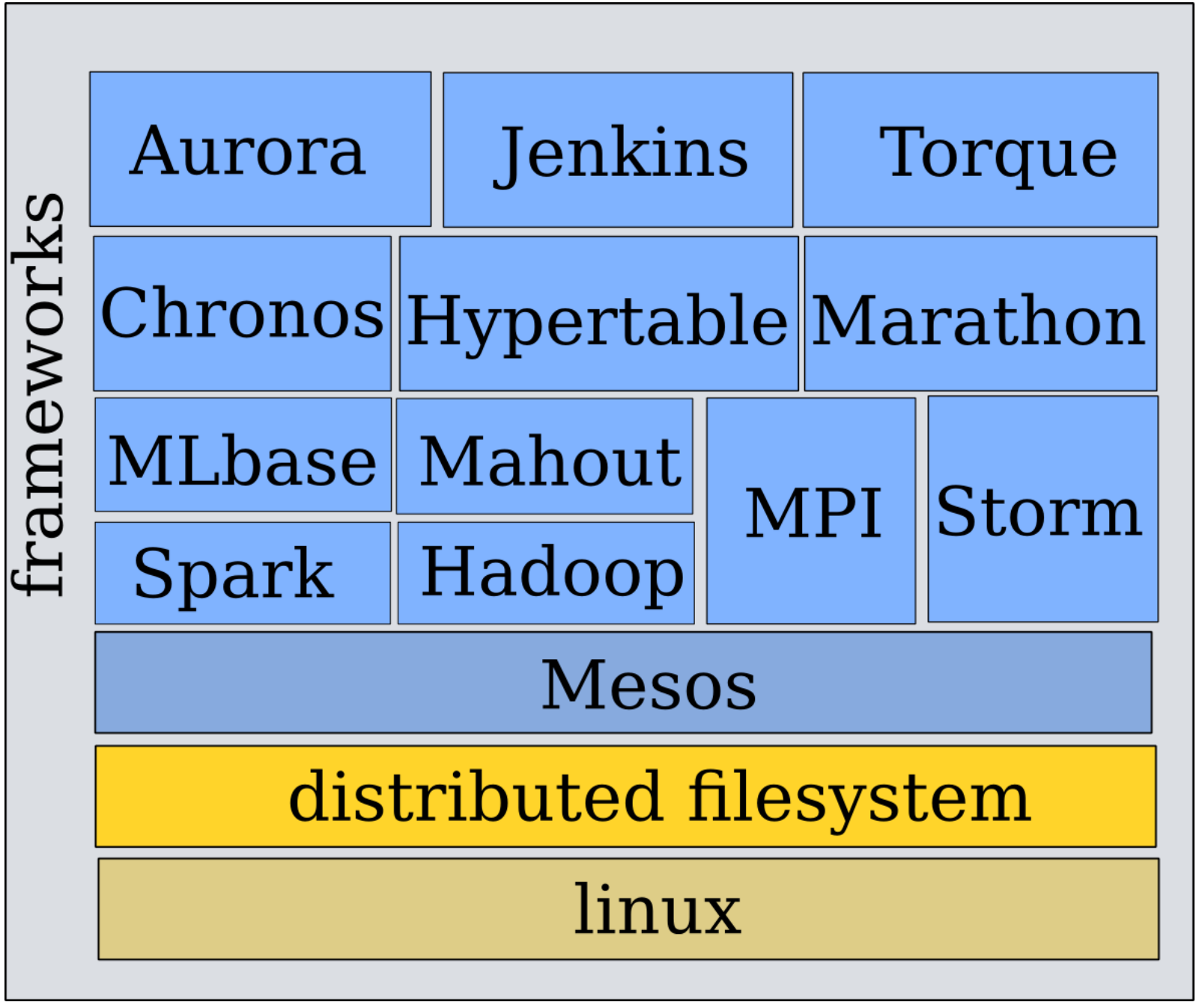


Isolation

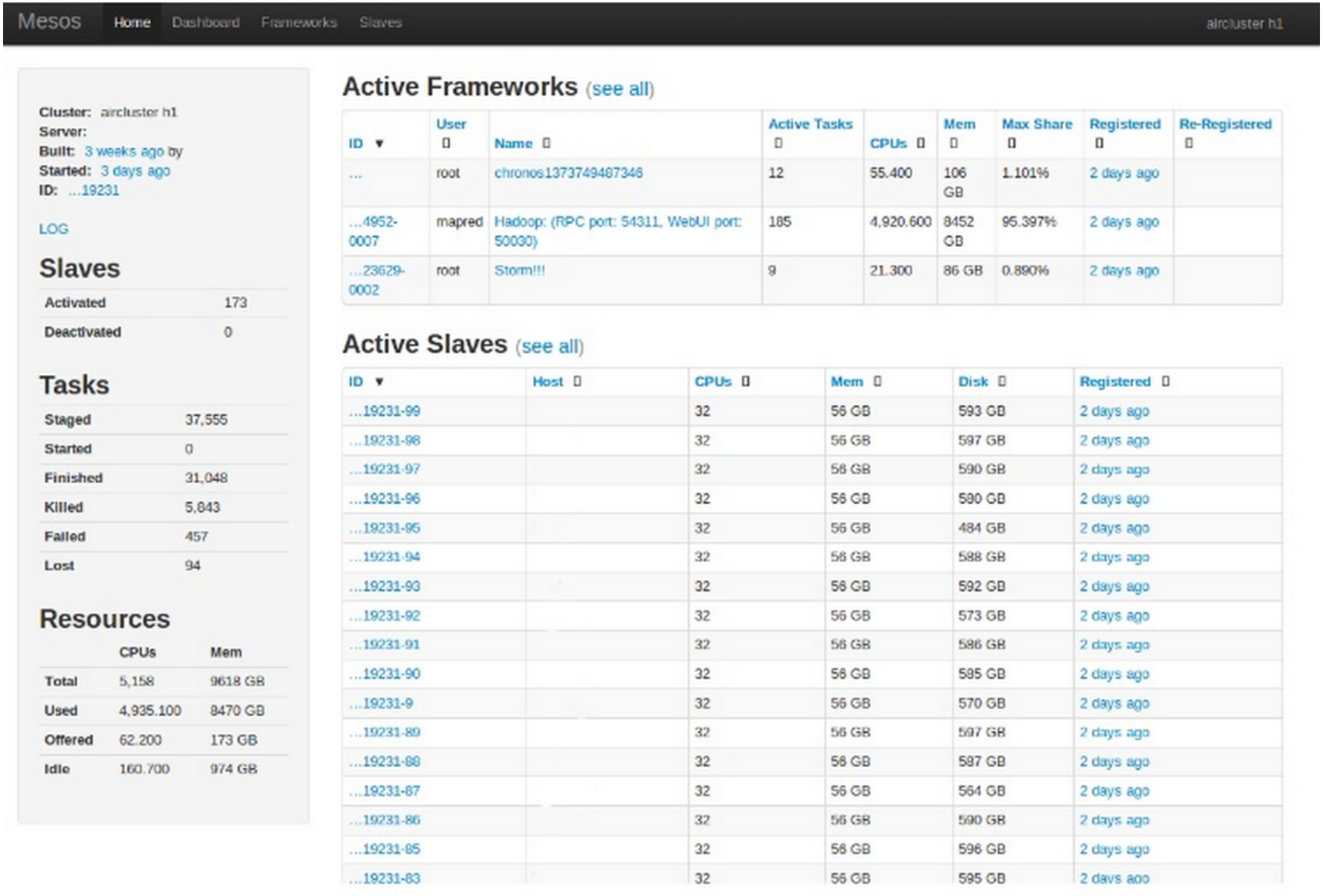
- using cgroups
- isolates:
  - CPU
  - memory
  - I/O
  - network

Example usage – GitLab CI

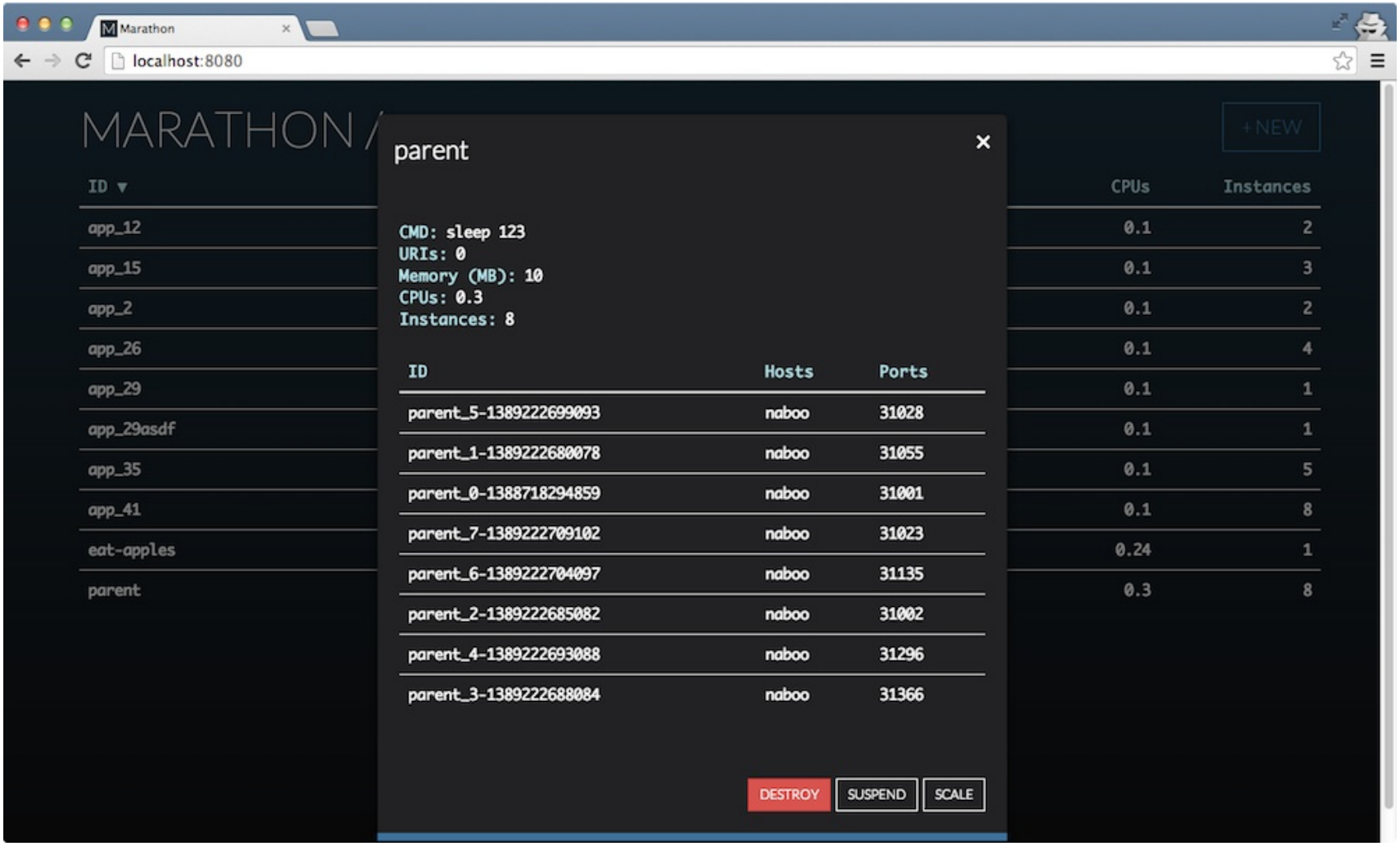



frameworks
Aurora
Jenkins
Torque
Chronos
Hypertable
Marathon
MLbase
Mahout
MPI
Storm
Spark
Hadoop
Mesos
distributed filesystem
linux

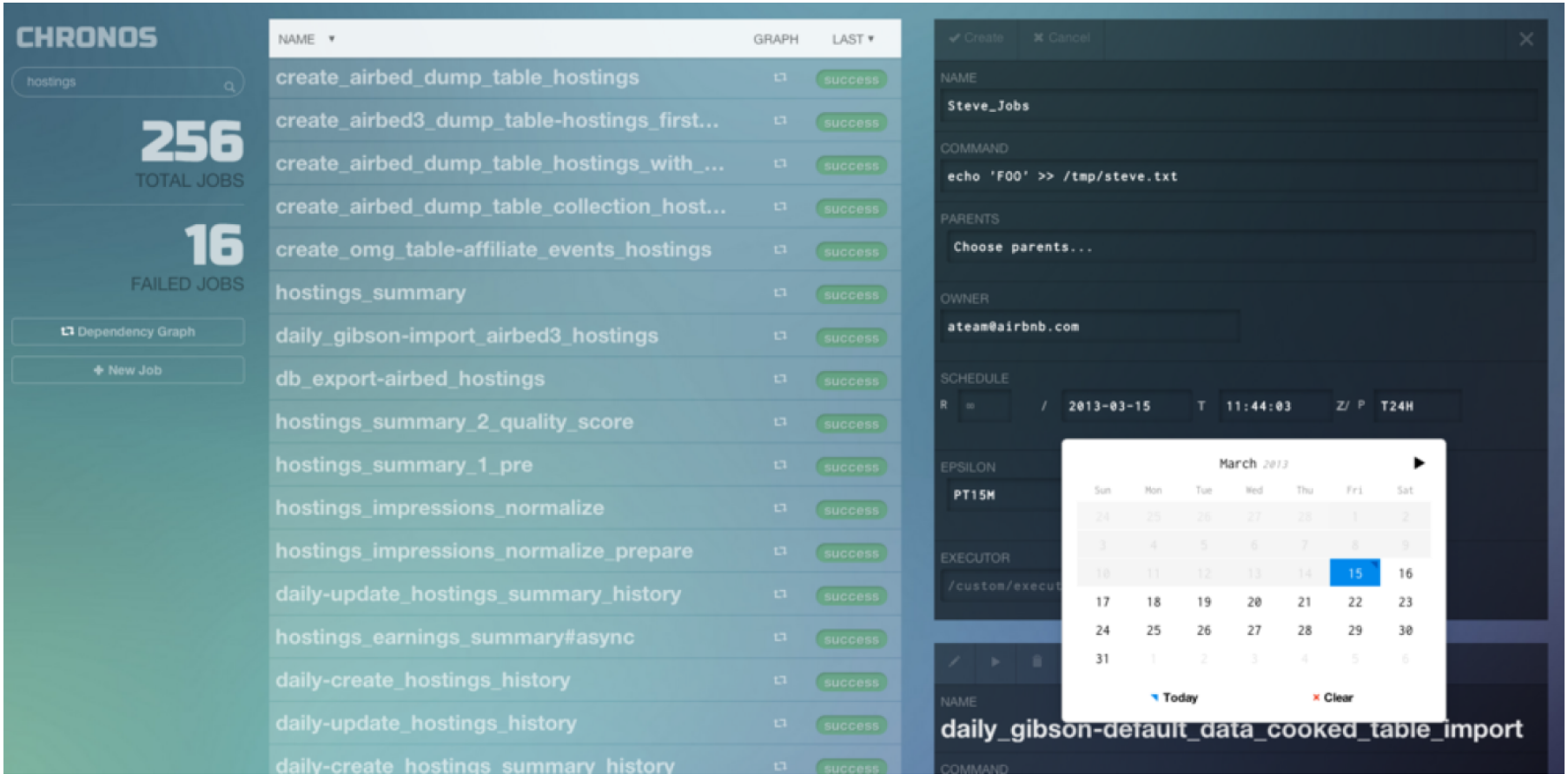
AirBnB


Mesos
Home
Dashboard
Frameworks
Slaves
aircluster h1
Cluster: aircluster h1
Server:
Built: 3 weeks ago by
Started: 3 days ago
ID: ...19231
LOG
Slaves
Activated 173
Deactivated 0
Tasks
Staged 37,555
Started 0
Finished 31,048
Killed 5,843
Failed 457
Lost 94
Resources
CPUs Mem
Total 5,158 9618 GB
Used 4,935.100 8470 GB
Offered 62.200 173 GB
Idle 160.700 974 GB
Active Frameworks (see all)
ID User Name Active Tasks CPUs Mem Max Share Registered Re-Registered
... root chronos1373749487346 12 55.400 106 GB 1.101% 2 days ago
...4952-0007 mapred Hadoop: (RPC port: 54311, WebUI port: 50030) 185 4,920.600 8452 GB 95.397% 2 days ago
...23629-0002 root Storm!!! 9 21.300 86 GB 0.890% 2 days ago
Active Slaves (see all)
ID Host CPUs Mem Disk Registered
...19231-99 32 56 GB 593 GB 2 days ago
...19231-98 32 56 GB 597 GB 2 days ago
...19231-97 32 56 GB 590 GB 2 days ago
...19231-96 32 56 GB 590 GB 2 days ago
...19231-95 32 56 GB 484 GB 2 days ago
...19231-94 32 56 GB 588 GB 2 days ago
...19231-93 32 56 GB 592 GB 2 days ago
...19231-92 32 56 GB 573 GB 2 days ago
...19231-91 32 56 GB 586 GB 2 days ago
...19231-90 32 56 GB 595 GB 2 days ago
...19231-9 32 56 GB 570 GB 2 days ago
...19231-89 32 56 GB 597 GB 2 days ago
...19231-88 32 56 GB 587 GB 2 days ago
...19231-87 32 56 GB 564 GB 2 days ago
...19231-86 32 56 GB 590 GB 2 days ago
...19231-85 32 56 GB 596 GB 2 days ago
...19231-83 32 56 GB 595 GB 2 days ago

Any web application can run on Mesos



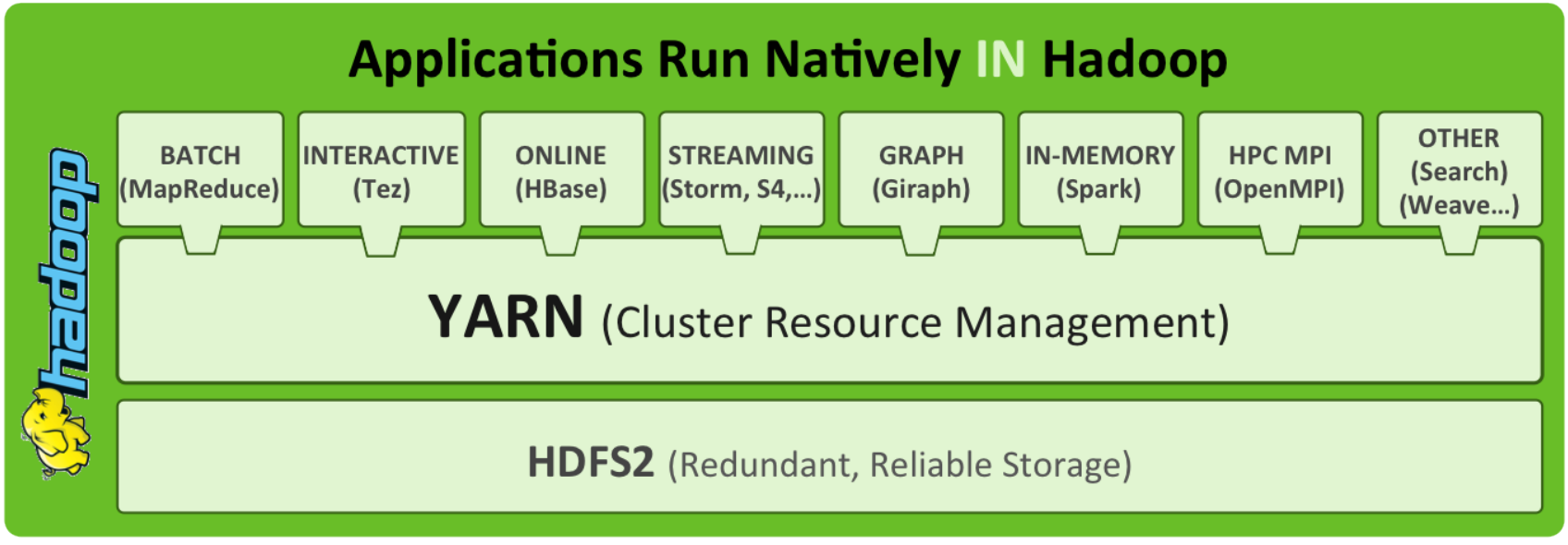
Distributed cron



YARN

Alternative to Mesos

- **Y**et **A**nother **R**esource **N**egotiator



Where to get Mesos?

- tarball – http://mesos.apache.org
- deb, rpm – http://mesosphere.io/downloads/
- custom package – https://github.com/deric/mesos-deb-packaging

- AWS instance in few seconds – https://elastic.mesosphere.io/

Configuration management

- automatic server configuration
- portable
- should work on most Linux distributions (currently Debian, Ubuntu)
- `https://github.com/deric/puppet-mesos`

01000110 Fakulta
01001001 Informačních
01010100 Technologií

DATA SCIENCE LABORATORY

Thank you for attention!

tomas.barton@fit.cvut.cz

Resources

- Containers – Not Virtual Machines – Are the Future Cloud
- Managing Twitter clusters with Mesos
- Return of the Borg: How Twitter Rebuilt Google's Secret Weapon
- Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center, Hindman, Benjamin and Konwinski, Andrew and Zaharia, Matei and Ghodsi, Ali and Joseph, Anthony D. and Katz, Randy H. and Shenker, Scott and Stoica, Ion; EECS Department, University of California, Berkeley 2010