Apache Mesos

mesos.apache.org

@ApacheMesos

Benjamin Hindman – @benh

download and install

http://www.apache.org/dyn/closer.cgi/mesos/0.12.1

```
$ tar zxf mesos-0.12.1.tar.gz

$ cd mesos-0.12.1

$ ./configure --prefix=/path/to/install/directory

$ make install
```

releases

maintained	stable	development
0.12.1	0.13.0 (0.13.0-rc7)	0.14.0 (0.14.0-rc1)

development release

```
$ git clone https://git.apache.org/mesos.git

$ cd mesos

$ ./bootstrap

$ ./configure --prefix=/path/to/install/directory

$ make install
```

packages

https://s3.amazonaws.com/mesos-pkg/ubuntu/12.10/mesos_0.14.0_amd64.deb

https://s3.amazonaws.com/mesos-pkg/ubuntu/12.04/mesos_0.14.0_amd64.deb

https://s3.amazonaws.com/mesos-pkg/debian/7.0/mesos_0.14.0_amd64.deb

Contact info@mesosphe.re for more information or other packages

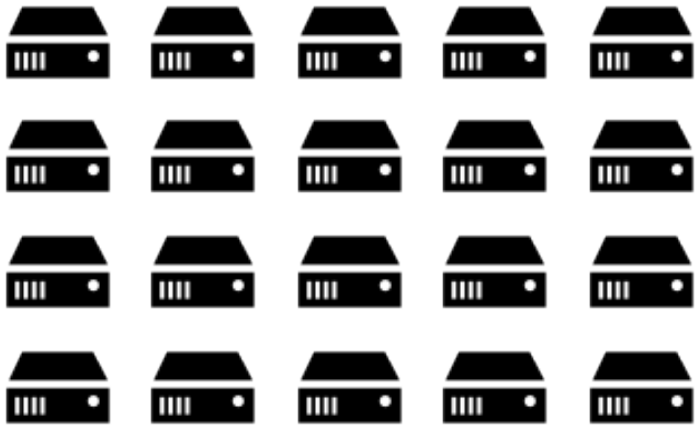
packages

https://github.com/deric/mesos-deb-packaging

packaging support in 0.15.0

nightly/weekly snapshots of development

mesos



starting a master

$ mesos-master --help

$ mesos-master --ip=a.b.c.d

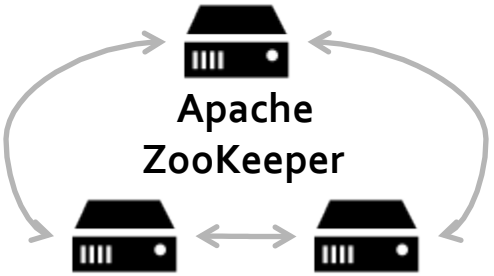
$ MESOS_ip=a.b.c.d mesos-master

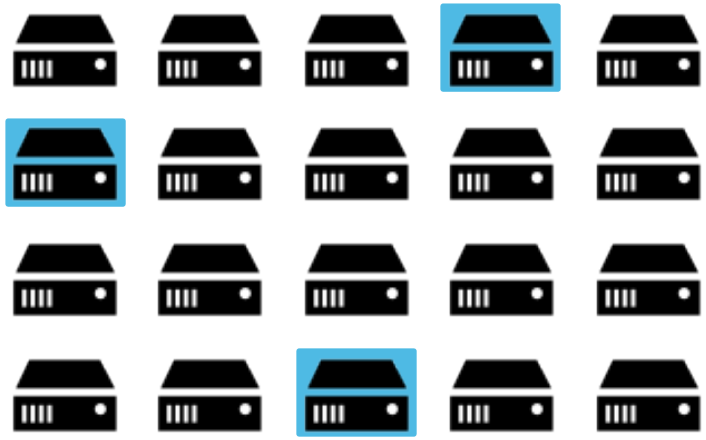
mesos

starting a (fault-tolerant) master

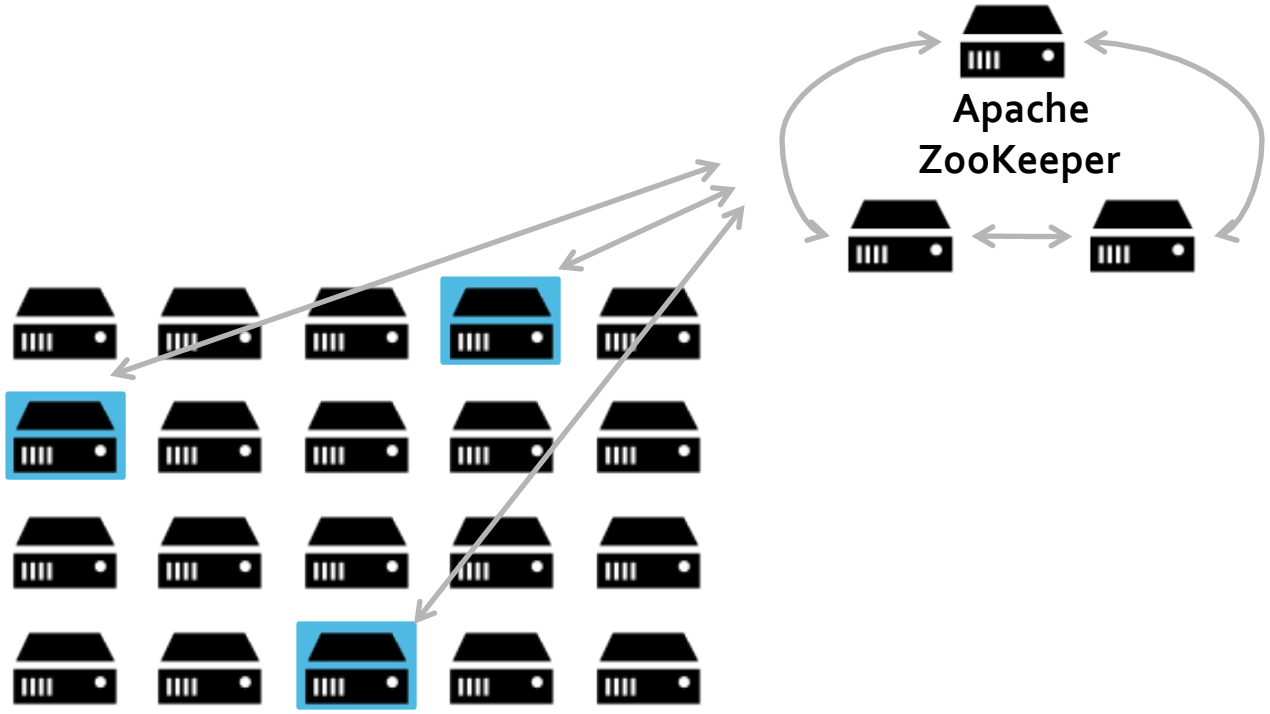
```
$ mesos-master --zk=zk://ip1:port1,ip2:port2,…/mesos
```

mesos





mesos



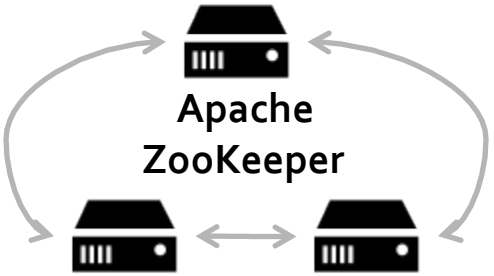
starting a slave

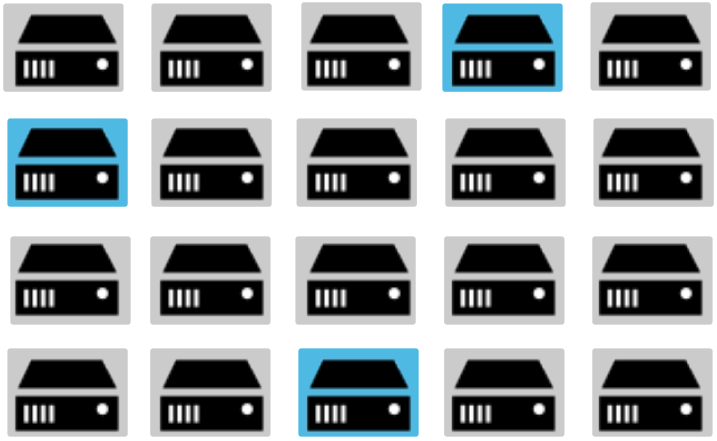
$ mesos-slave –help

$ mesos-slave --master=ip:port

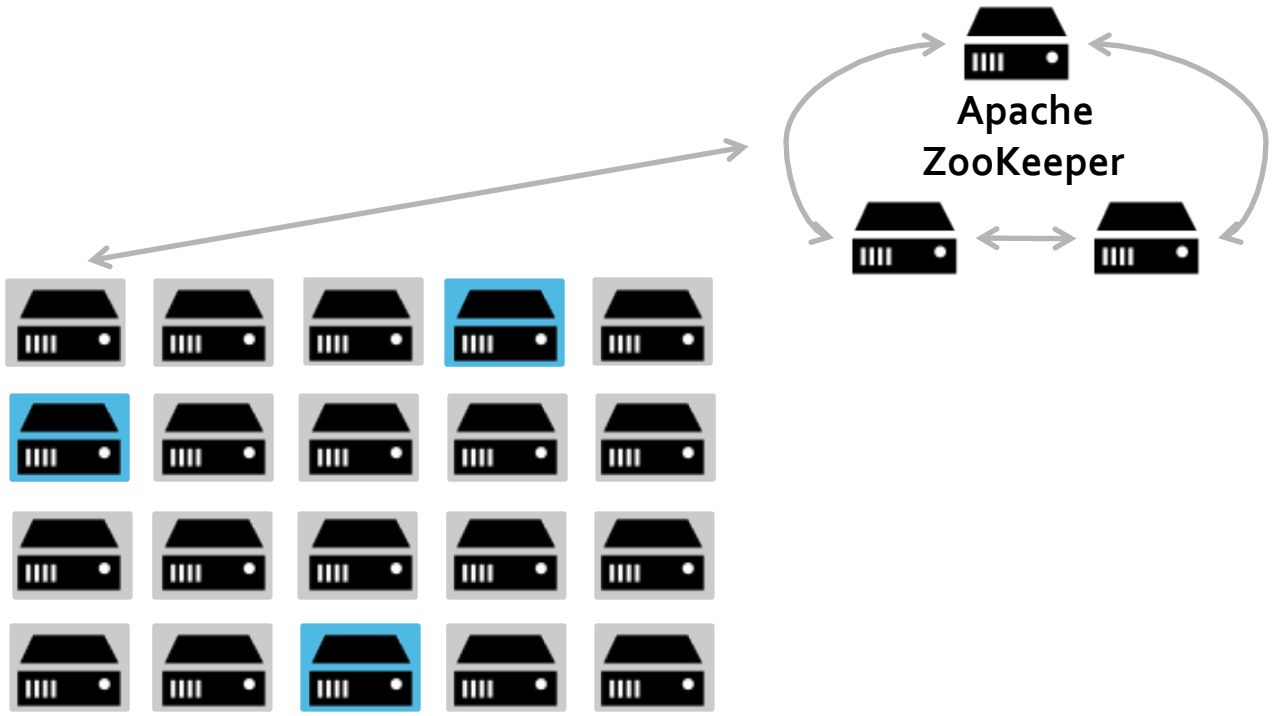
$ mesos-slave --master=zk://ip1:port1,ip2:port2,…/mesos

mesos

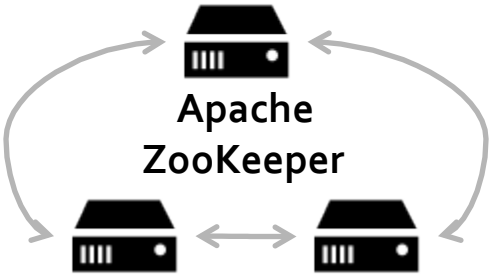


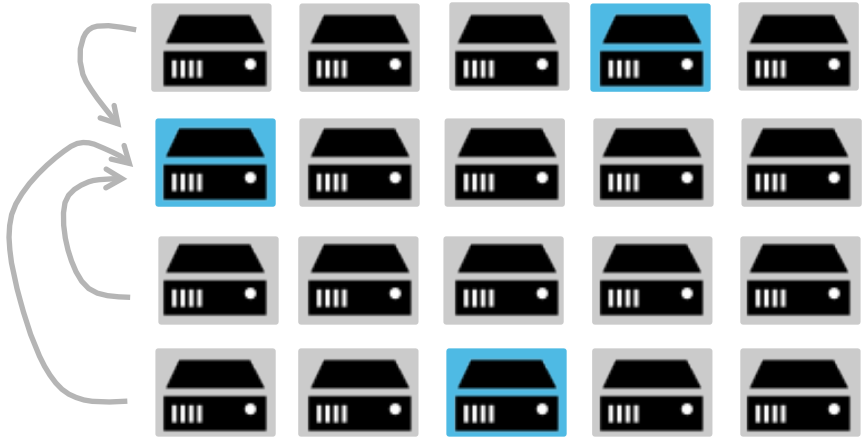


mesos



mesos





now what?

launch frameworks

what's a framework?

framework

≈

distributed system

frameworks

- Hadoop (github.com/mesos/hadoop)
- Spark (github.com/mesos/spark)
- DPark (github.com/douban/dpark)
- Storm (github.com/nathanmarz/storm)
- Chronos (github.com/airbnb/chronos)
- MPICH2 (not well maintained, email mailing list)

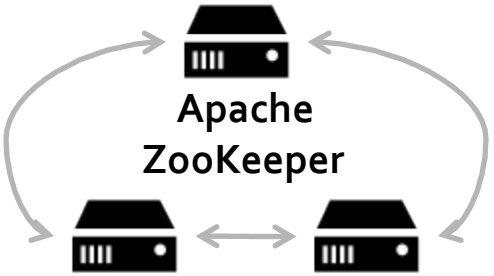
framework commonality

run processes simultaneously (*distributed*)

handle process failures (*fault-tolerance*)

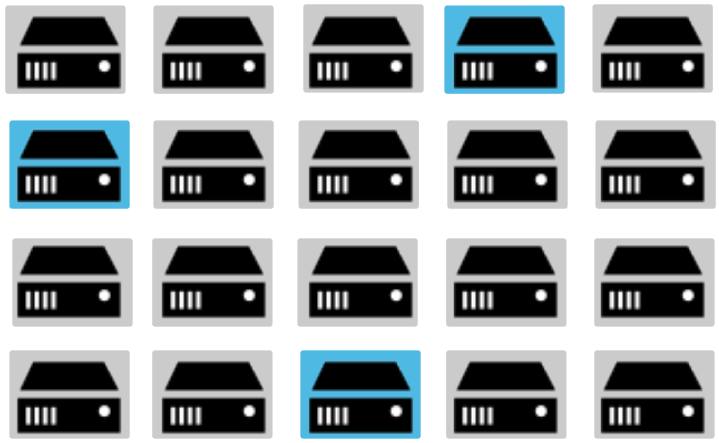
optimize execution (*elasticity, scheduling*)

mesos

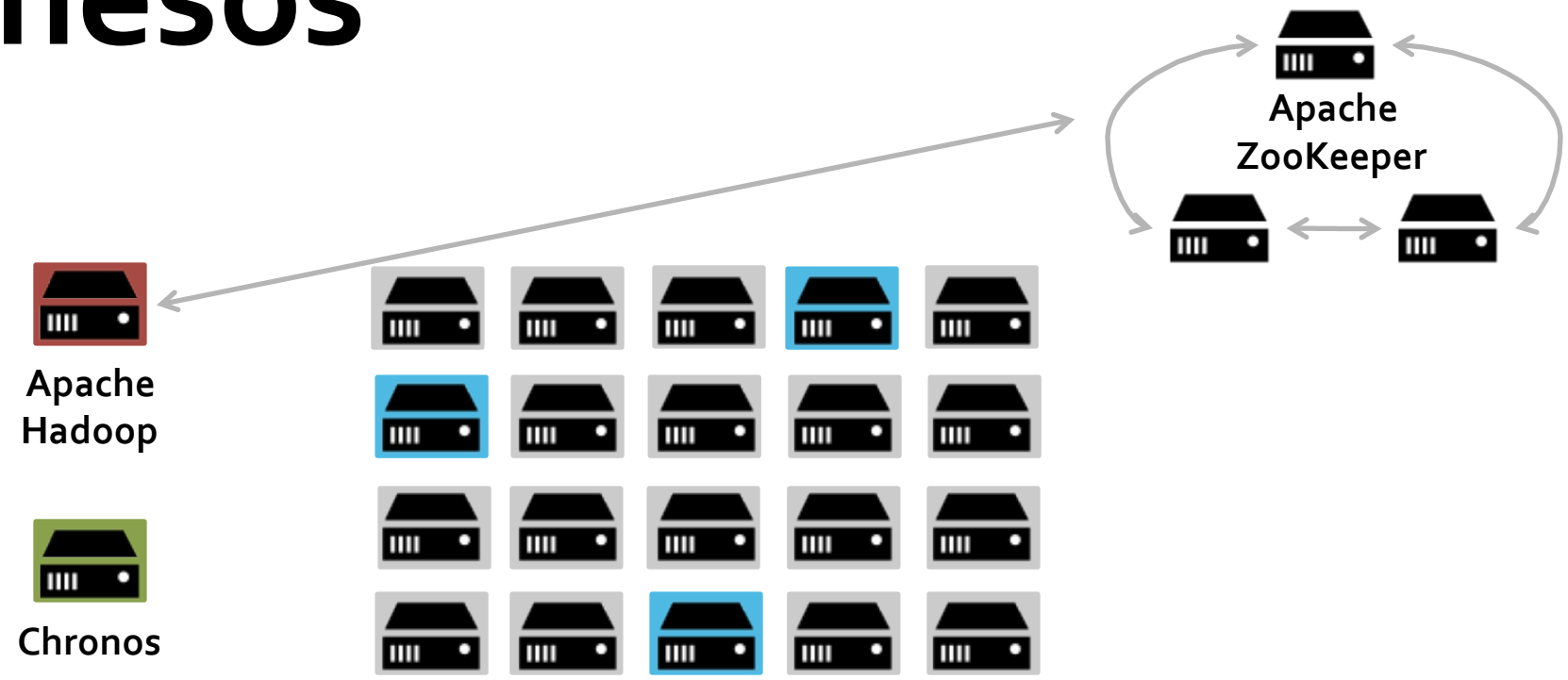


Apache
Hadoop

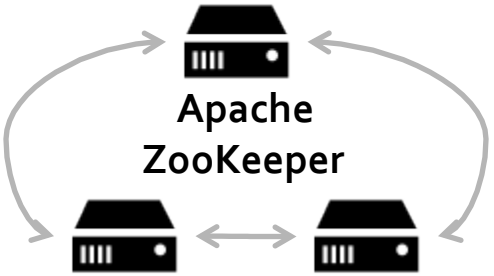
Chronos

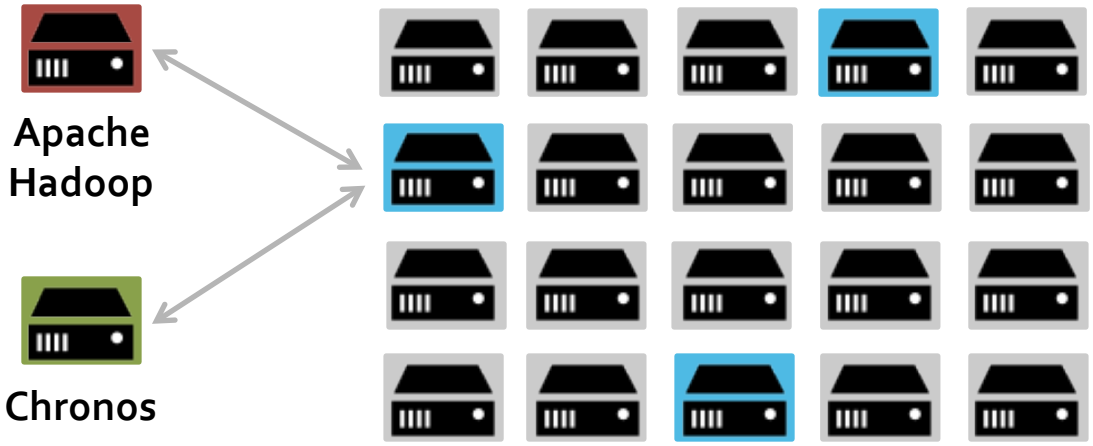


mesos

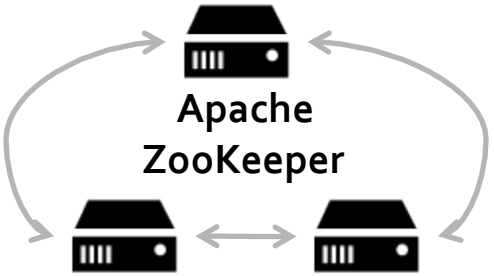


mesos





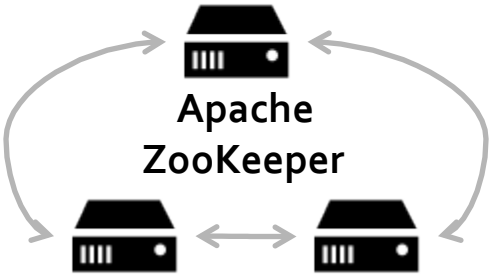
mesos



Apache
Hadoop

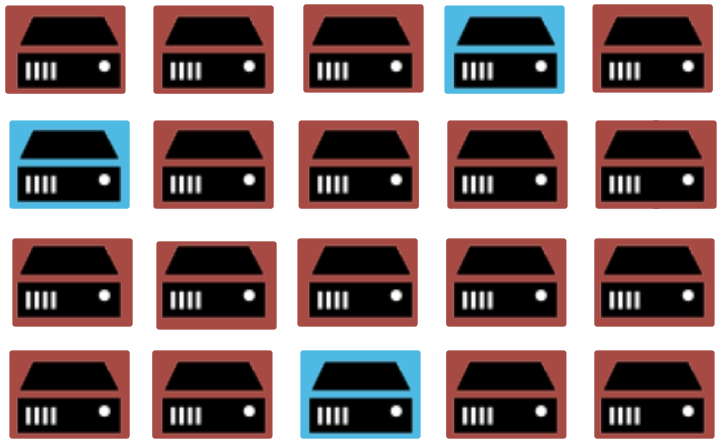
Chronos

mesos



Apache
Hadoop

Chronos



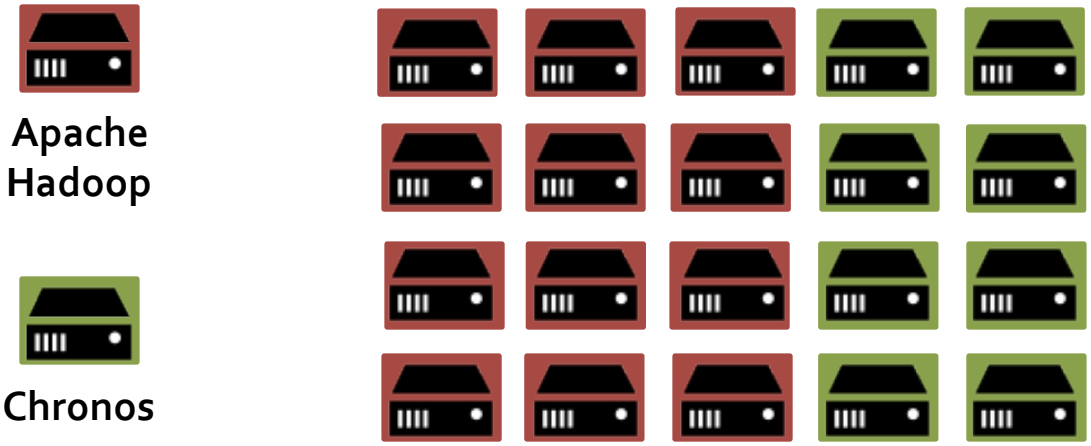
but why?

origins

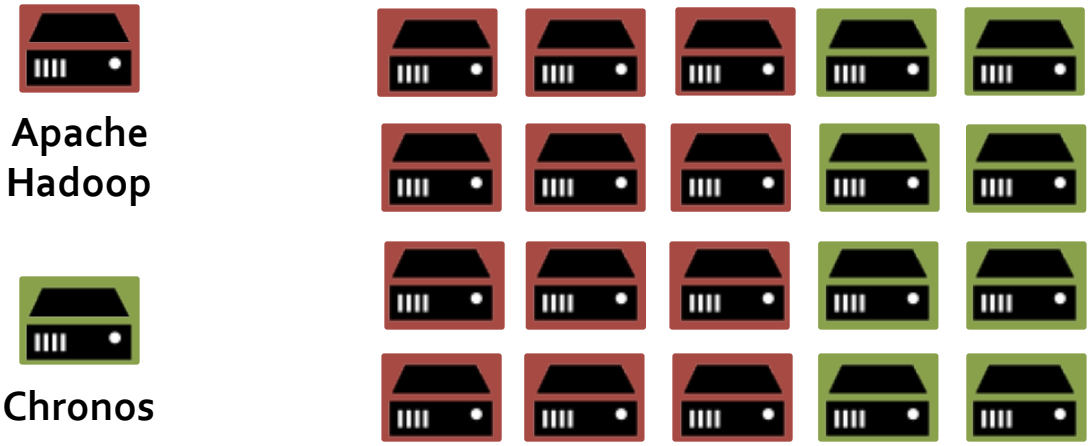
Berkeley research project including Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica

mesos.apache.org/documentation

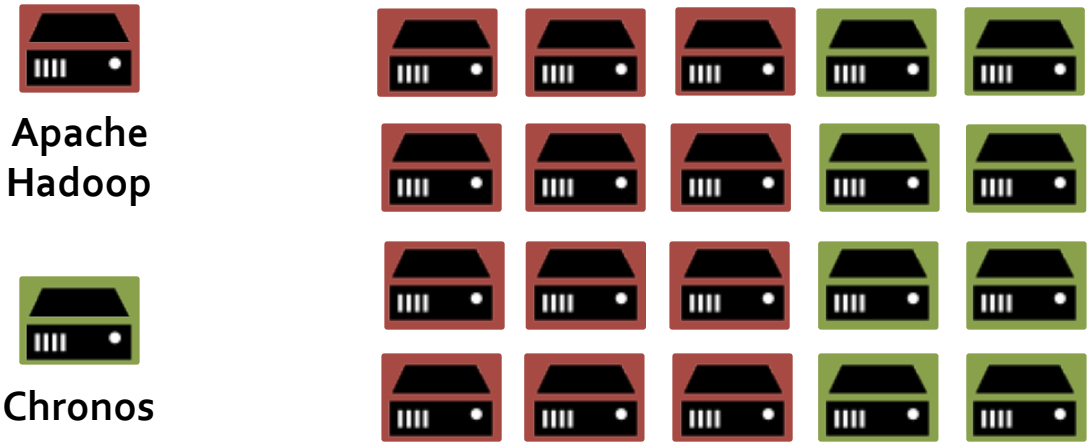
static partitioning



static partitioning considered harmful

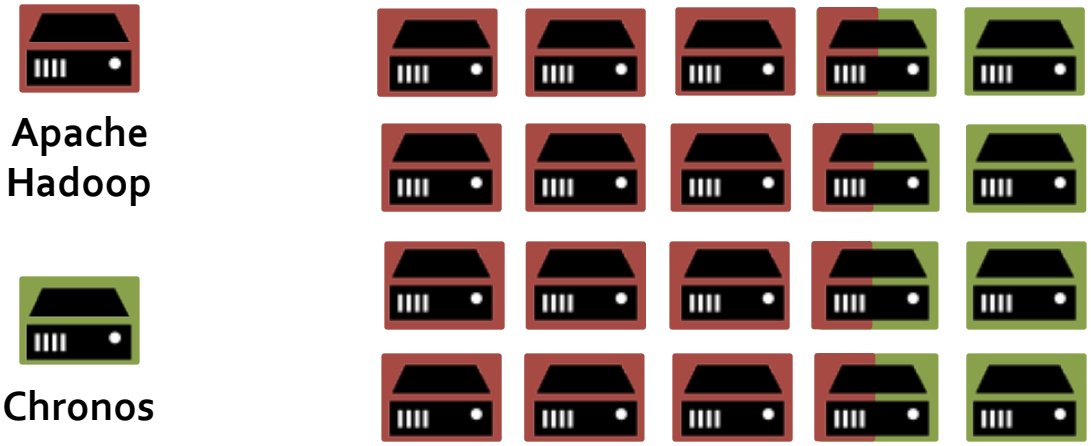


static partitioning considered harmful



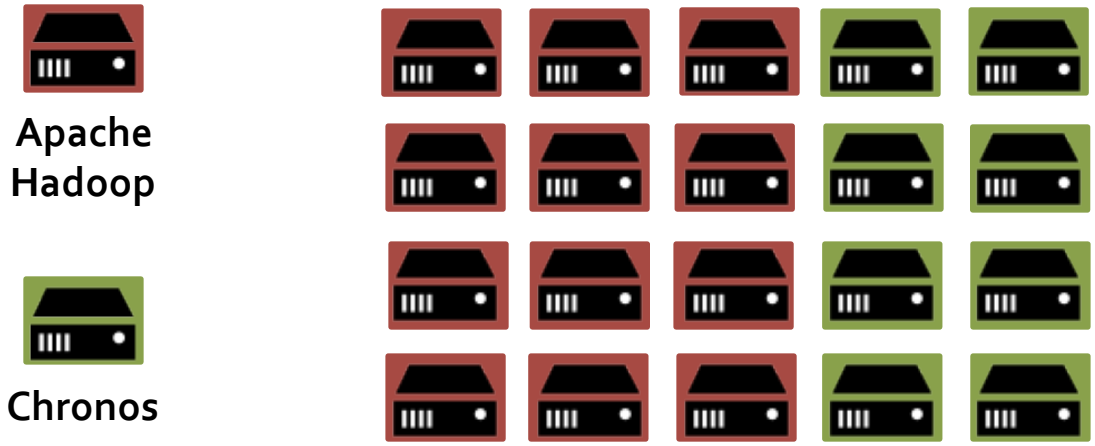
(1) hard to ***utilize*** machines
(e.g., 72 GB RAM and 24 CPUs)

static partitioning considered harmful



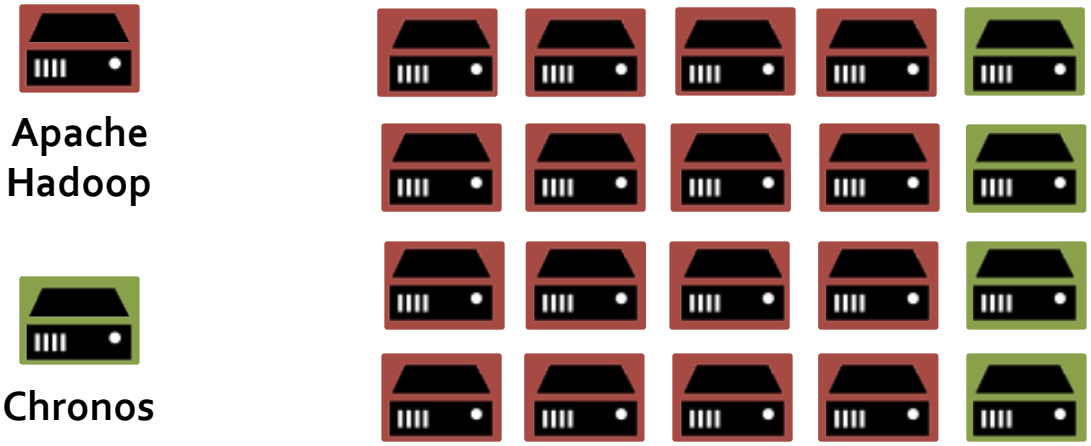
(1) hard to ***utilize*** machines
(e.g., 72 GB RAM and 24 CPUs)

static partitioning considered harmful



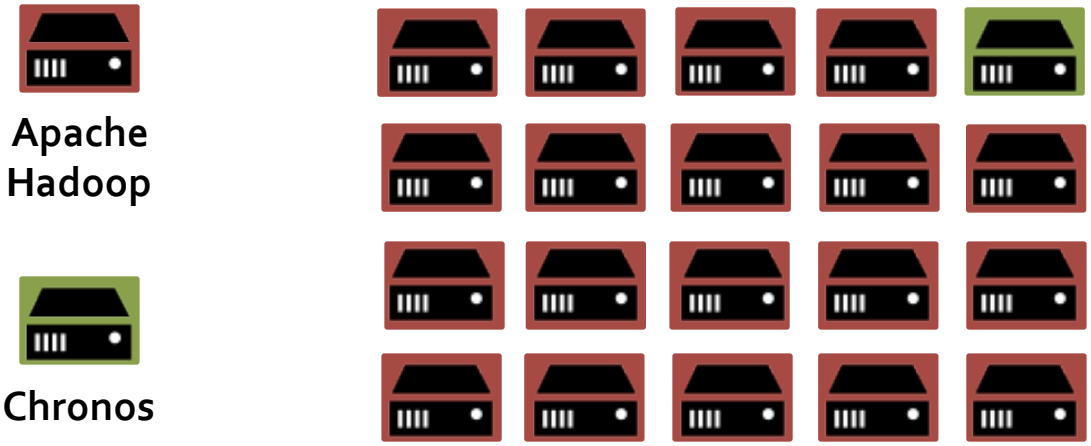
(2) hard to scale ***elastically***
(to take advantage of statistical multiplexing)

static partitioning considered harmful



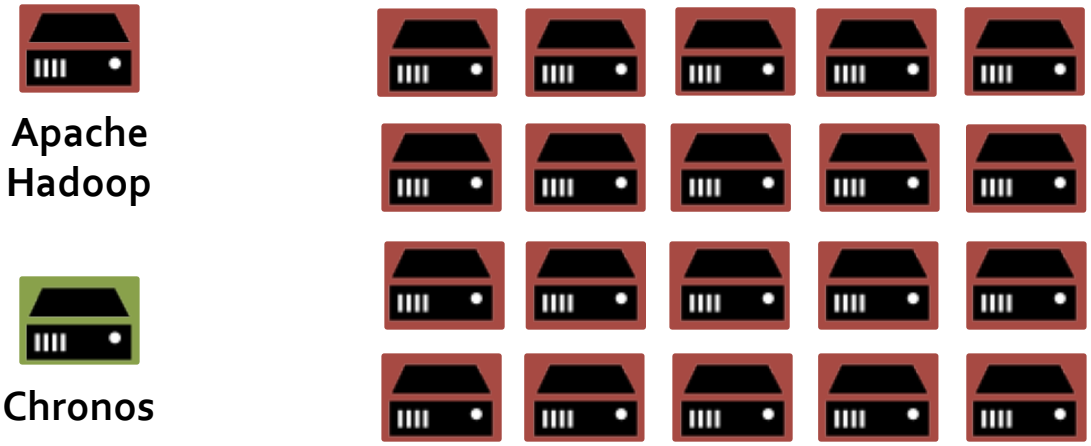
(2) hard to scale ***elastically***
(to take advantage of statistical multiplexing)

static partitioning considered harmful



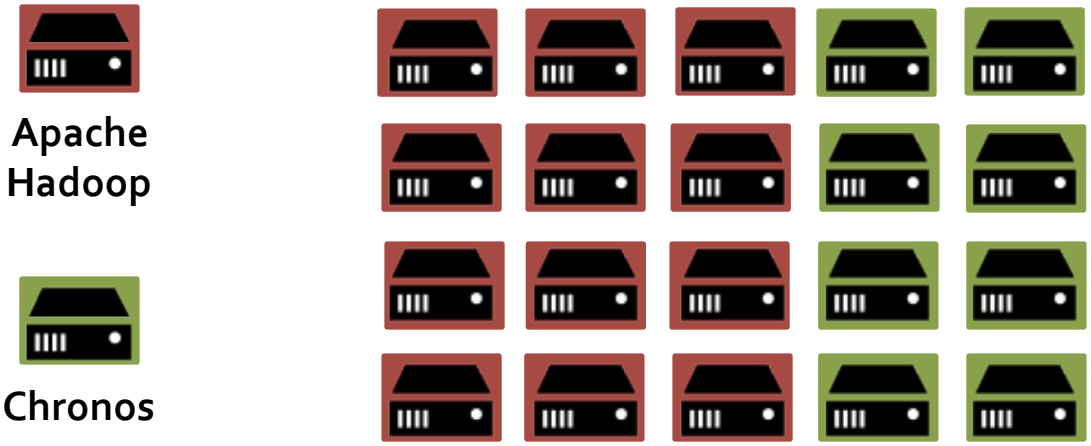
(2) hard to scale ***elastically***
(to take advantage of statistical multiplexing)

static partitioning considered harmful



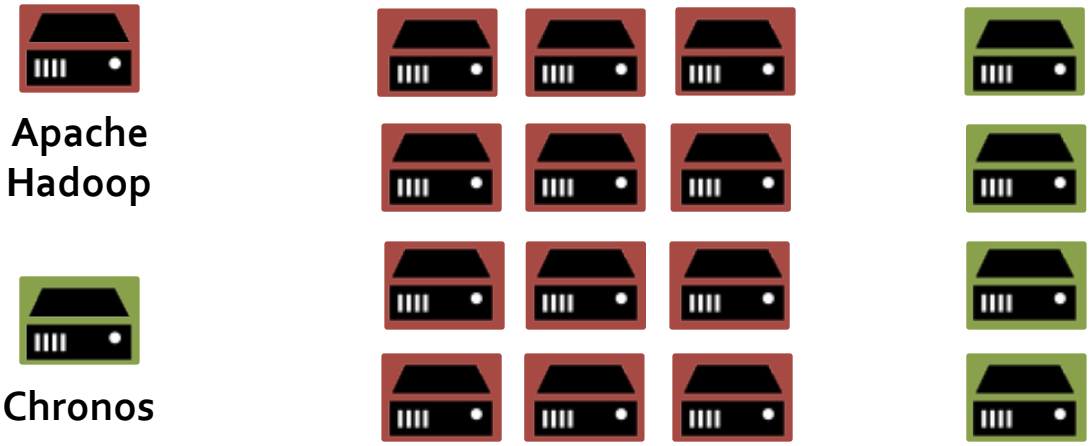
(2) hard to scale ***elastically***
(to take advantage of statistical multiplexing)

static partitioning considered harmful



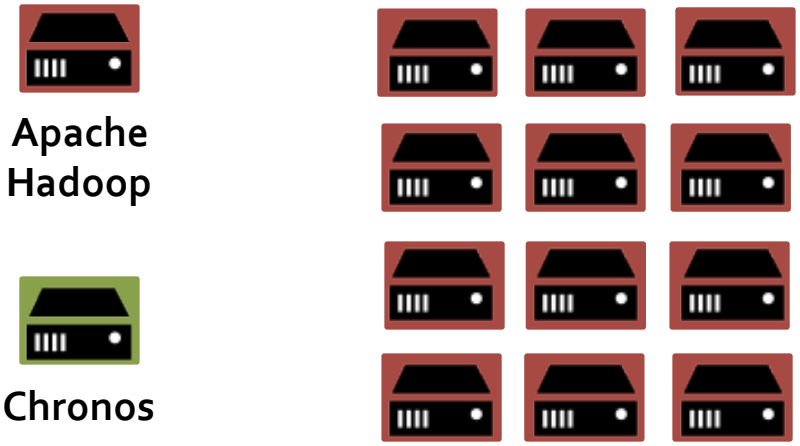
hard to deal with ***failures***

static partitioning considered harmful



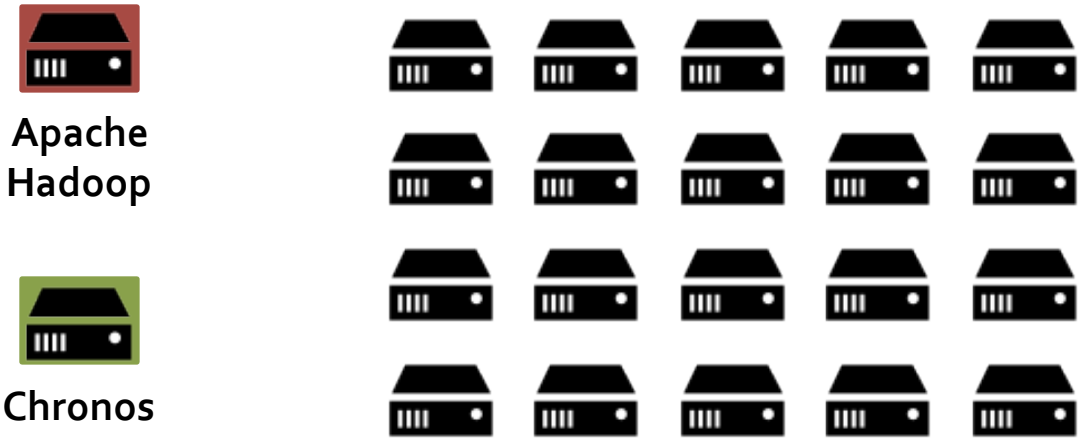
hard to deal with ***failures***

static partitioning considered harmful



hard to deal with ***failures***

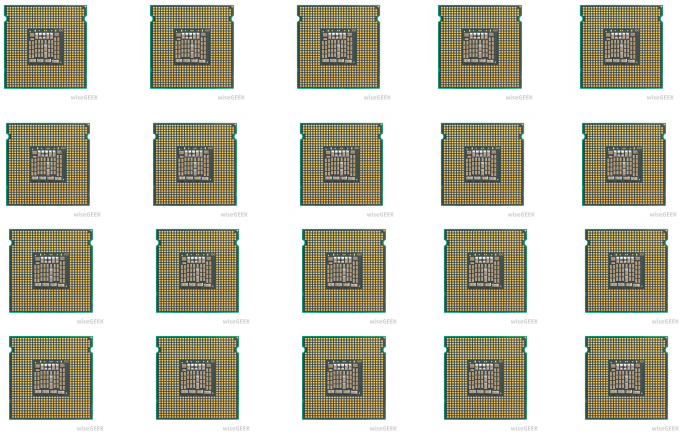
mesos – level of indirection



mesos – level of indirection

Apache Hadoop

Chronos



mesos – level of indirection

Apache
Hadoop

Chronos

a “kernel” for the datacenter

Apache
Hadoop

Chronos

primitives

scheduler – distributed system "master"

(***executor*** – lower-level control of task execution, optional)

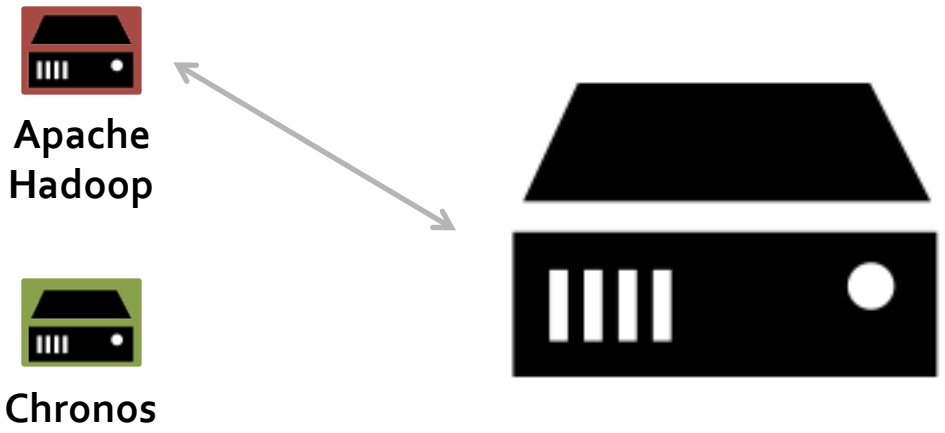
requests/offers – resource allocations

tasks – "threads" of the distributed system

state – working set of the distributed system

…

scheduler



scheduler

(1) brokers for resources (with master)

(2) launches tasks

(3) handles task termination

brokering for resources

(1) make resource ***requests***

2 CPUs

1 GB RAM

slave *

(2) respond to resource ***offers***

4 CPUs

4 GB RAM

slave foo.bar.com

offers: non-blocking resource allocation

exist to answer the question:

"*what should mesos do if it can't satisfy a request?"*

(1) wait until it can

(2) ***offer*** the best allocation it can immediately

offers: non-blocking resource allocation

exist to answer the question:

"*what should mesos do if it can't satisfy a request?*"

(1) wait until it can

(2) ***offer*** the best allocation it can immediately

“two-level scheduling”

mesos: controls resource allocations to schedulers

schedulers: make decisions about what to run given allocated resources

end-to-end principle

"application-specific functions ought to reside in the end hosts of a network rather than intermediary nodes"

tasks

either a concrete command line or an opaque description (which requires a framework executor to execute)

a *consumer* of resources

task operations

launching/killing

health monitoring/reporting (failure detection)

resource usage monitoring (statistics)

state (and replicated log)

… when your distributed system needs *state (*the "working set", often 10's to 100's of MB), what do you do?

- » a database is overkill (yet another system to manage)
- » ZooKeeper can work (but you probably want to use a higher level abstraction, and if you have more than 1MB you could be out of luck, and …)
- » can build your own distributed state machine …

state (and replicated log)

you probably don't want Paxos , you want Multi-Paxos

and Multi-Paxos is just a replicated log (i.e., a replicated log is an implementation of Multi-Paxos but with a nicer interface)

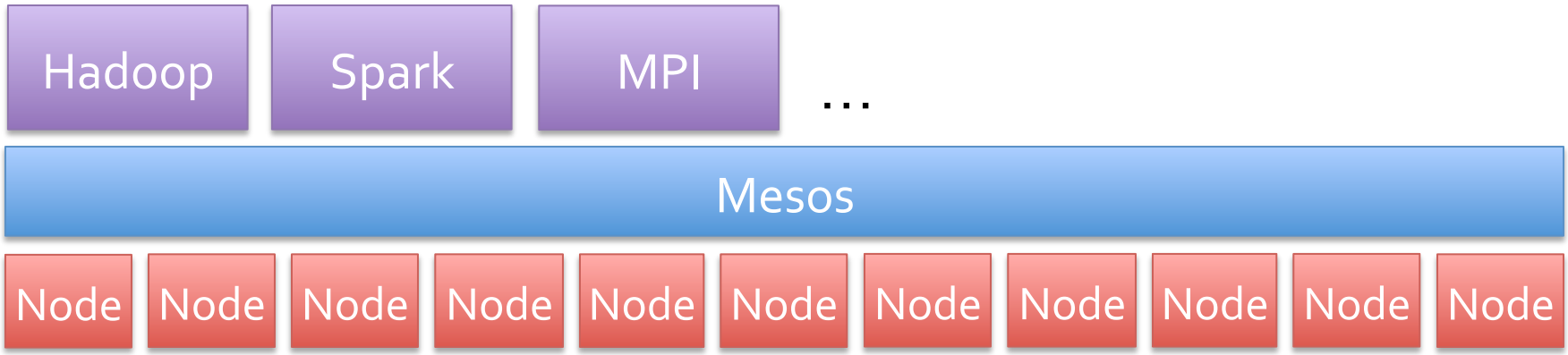
in Mesos since 0.9.0 (including Java/JNI bindings), used in production for ~2 years

state (and replicated log)

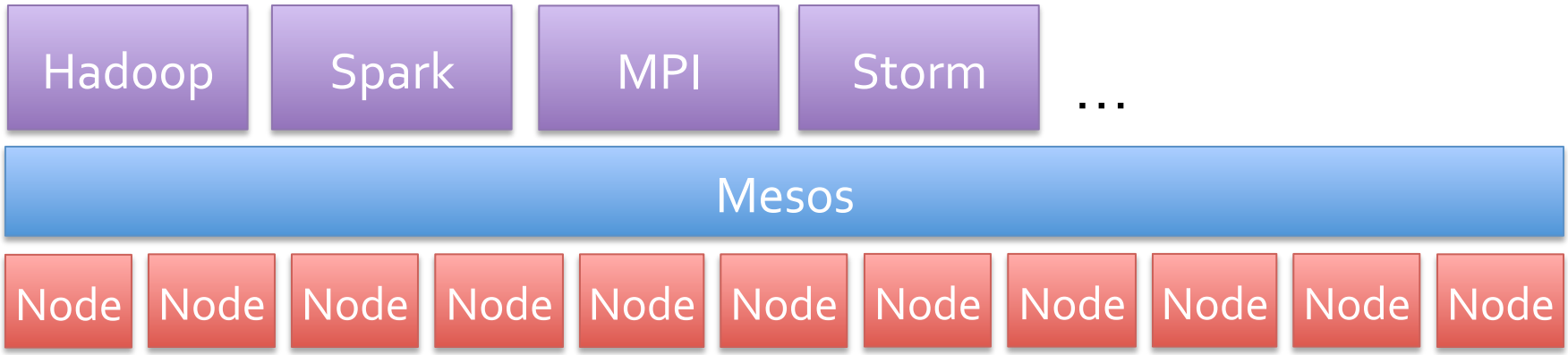
even a replicated log is fairly low-level (one of the reasons ZooKeeper is so popular) … enter "state"

```
State* state = new State(new ReplicatedLogStorage());
Future<Variable<Registry>> fetch = state->fetch("registry");
Variable<Registry> variable = fetch.get();
Registry registry = variable.get();

registry.makeSomeUpdates();

Variable<Registry> variable_ = variable.mutate(registry);
Future<Option<Variable<Registry>>> store = state->store(variable_);
```

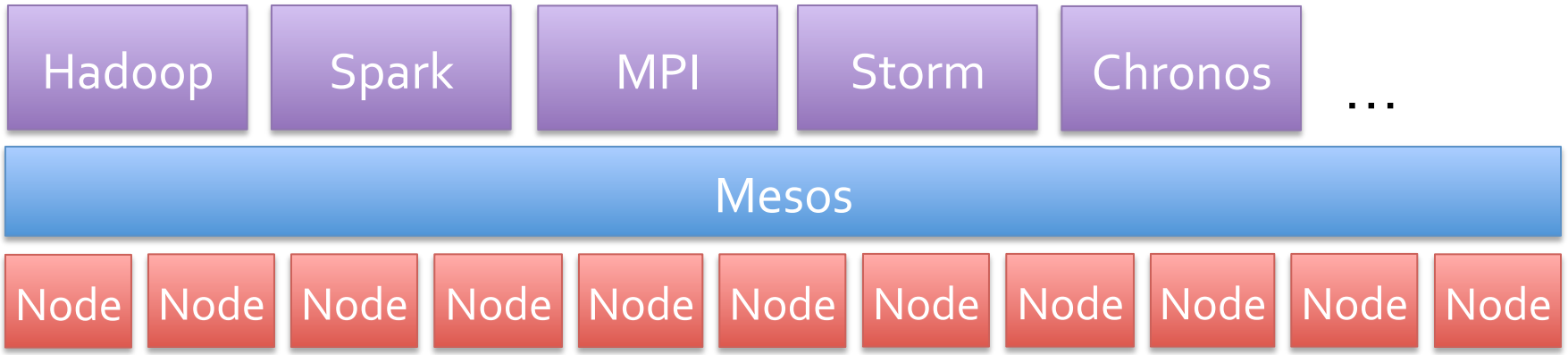
Mesos


Hadoop
Spark
MPI
…
Mesos
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node

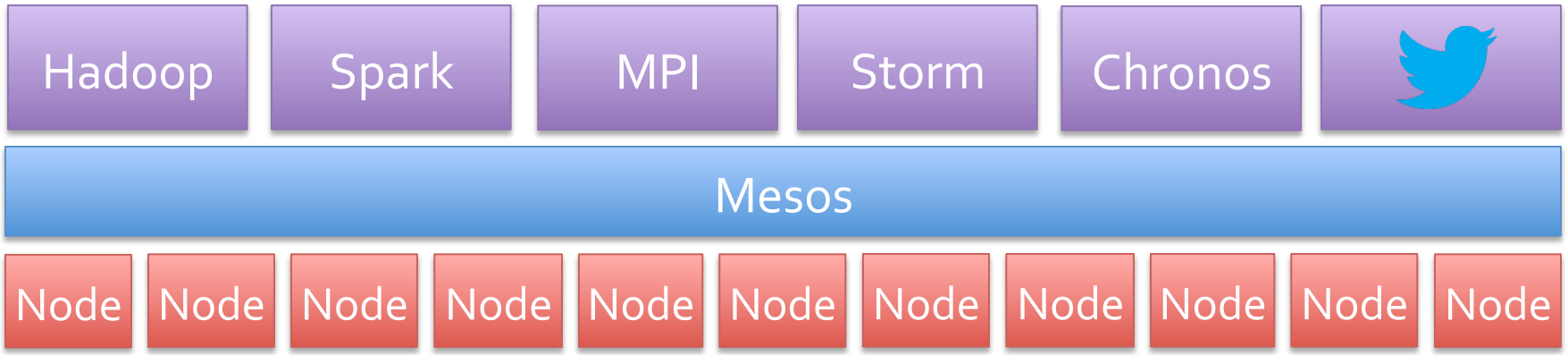
Mesos


Hadoop
Spark
MPI
Storm
...
Mesos
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node

Mesos


Hadoop
Spark
MPI
Storm
Chronos
…
Mesos
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node

Mesos


Hadoop
Spark
MPI
Storm
Chronos
Mesos
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node
Node

Questions ...