Mesos
for Spark Users

mesos.apache.org

@ApacheMesos

Benjamin Hindman – @benh

agenda

① Mesos ←

② Spark on Mesos

③ why Mesos?

  ① multi-tenancy
  ② fine-grained sharing
  ③ why not?

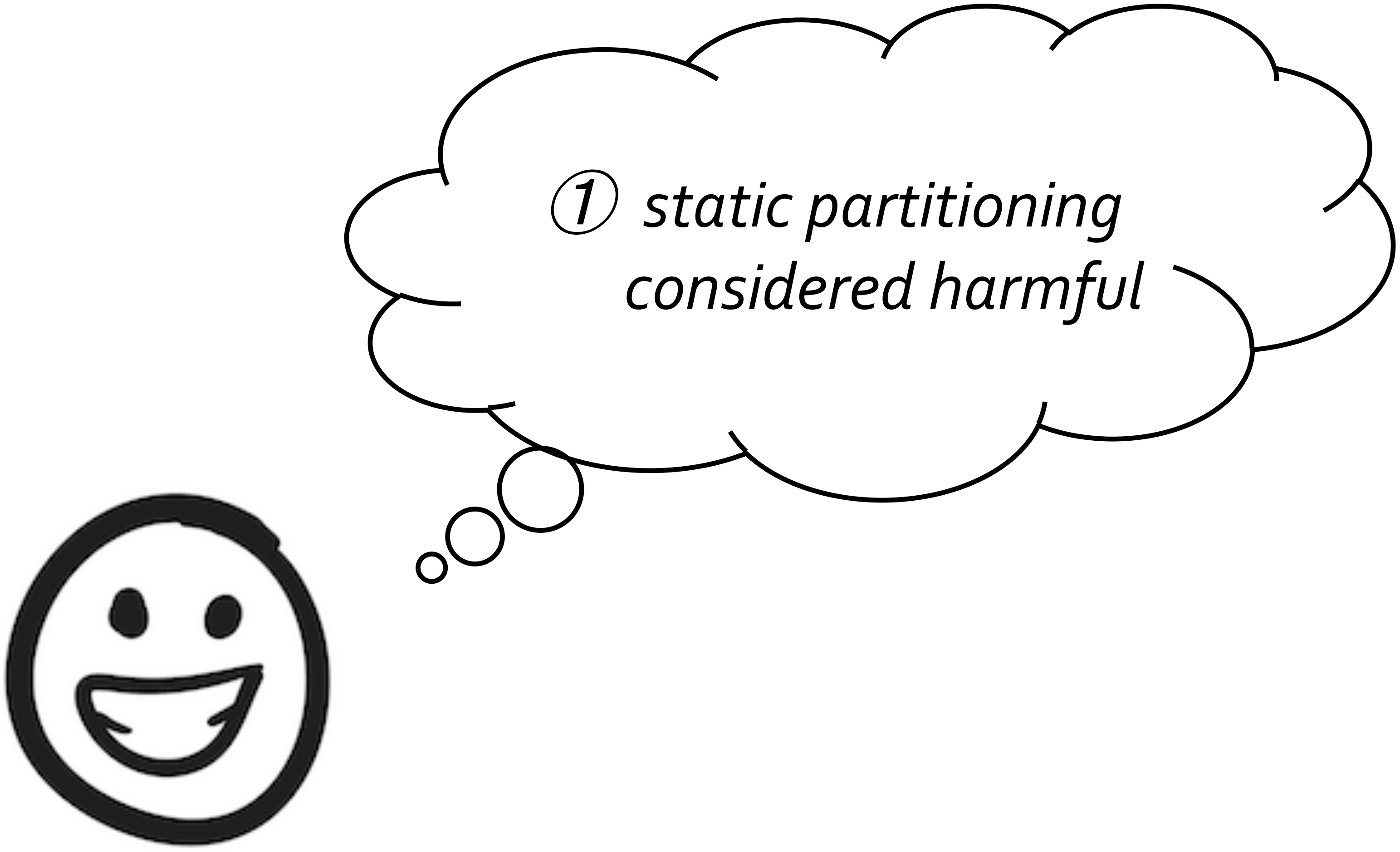
④ long-lived services and other frameworks

a little history

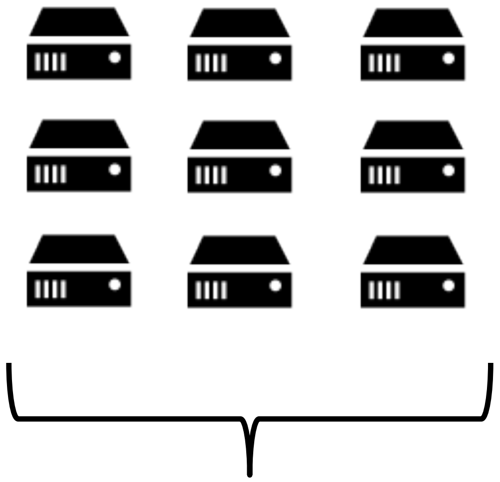
Mesos started as a research project at Berkeley in early 2009 by Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica

our motivation

our intuition

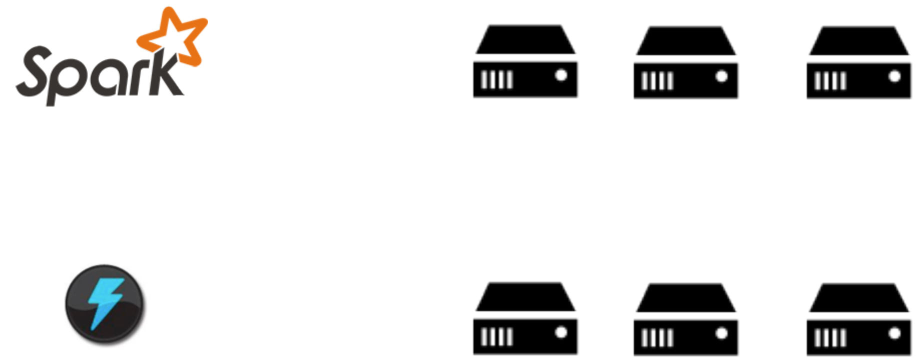


static partitioning considered harmful

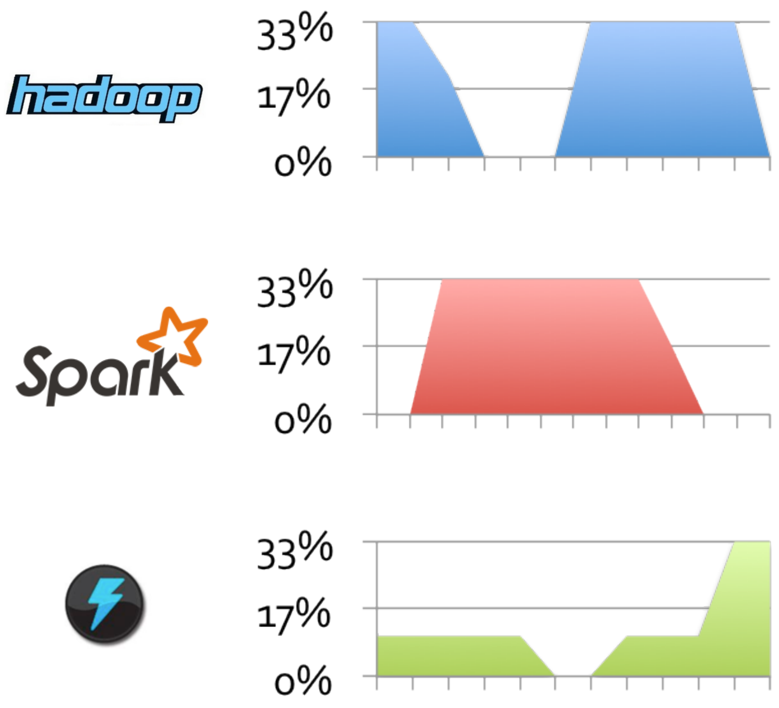


datacenter

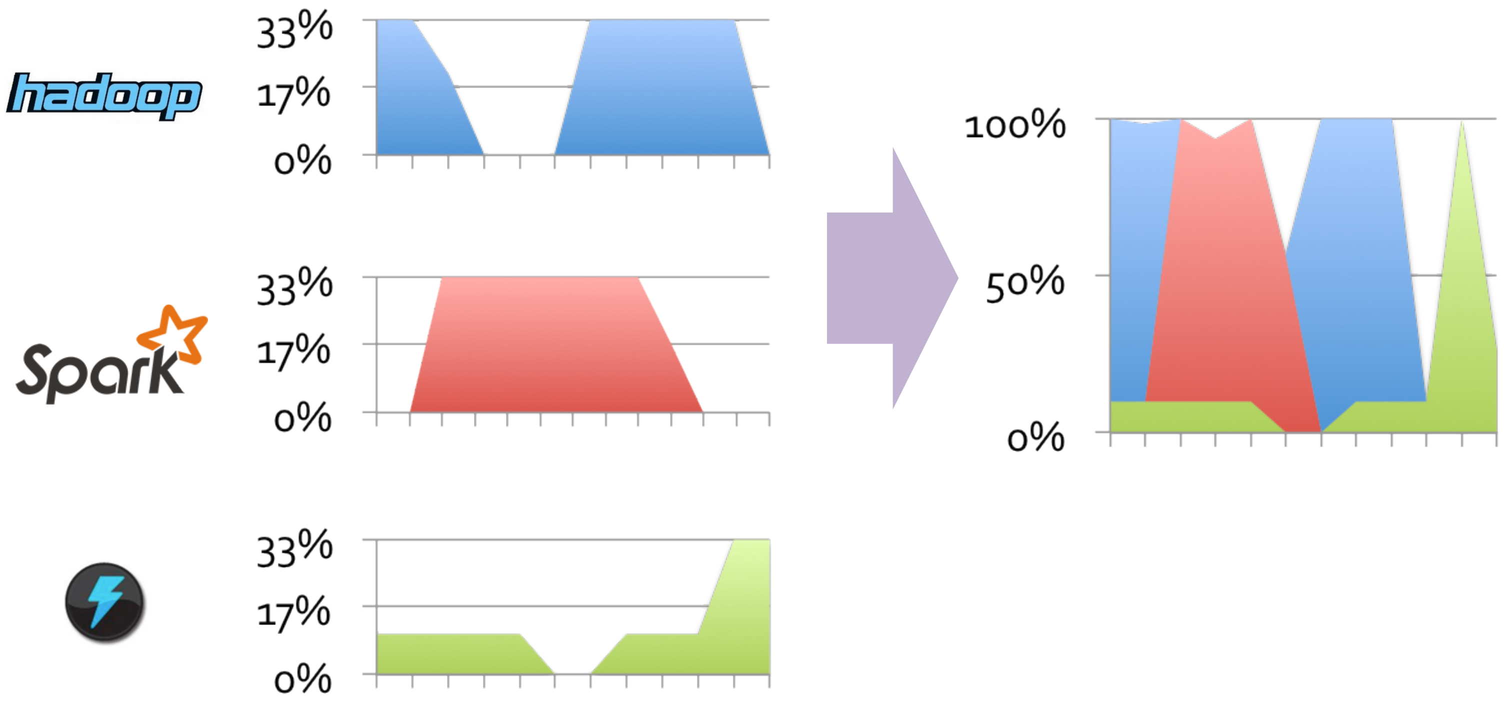
static partitioning considered harmful



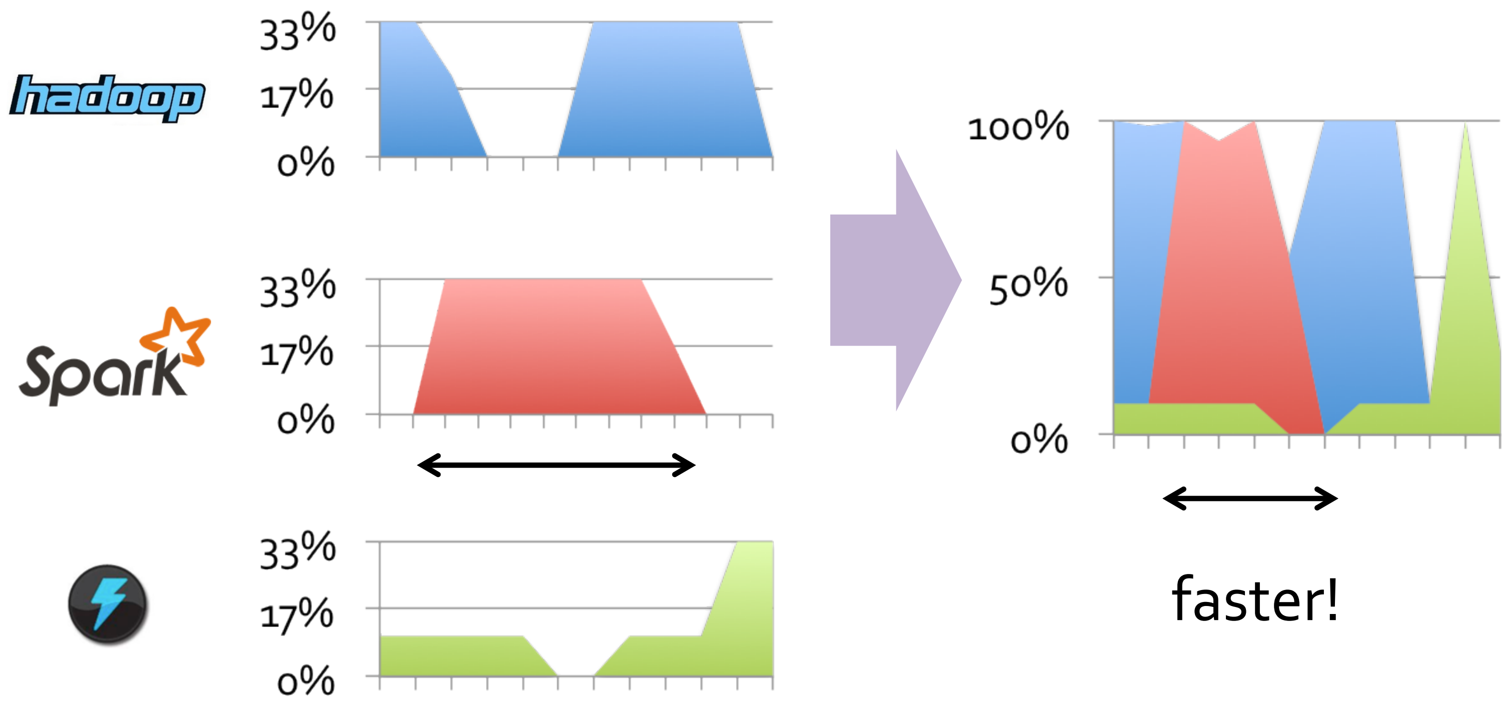
static partitioning considered harmful



static partitioning considered harmful

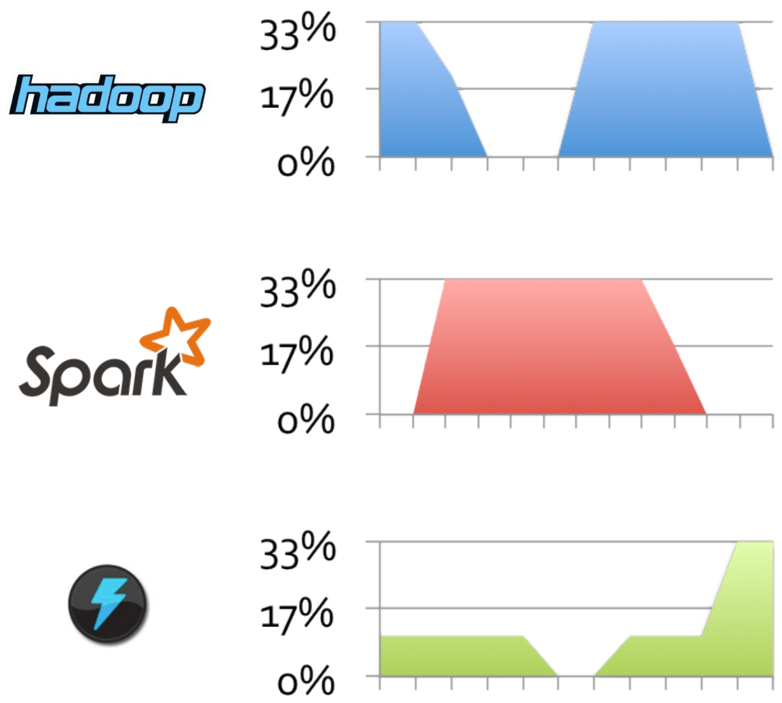


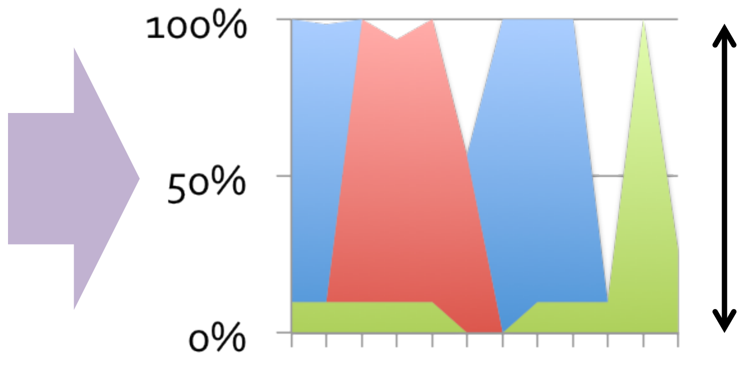
static partitioning considered harmful



faster!

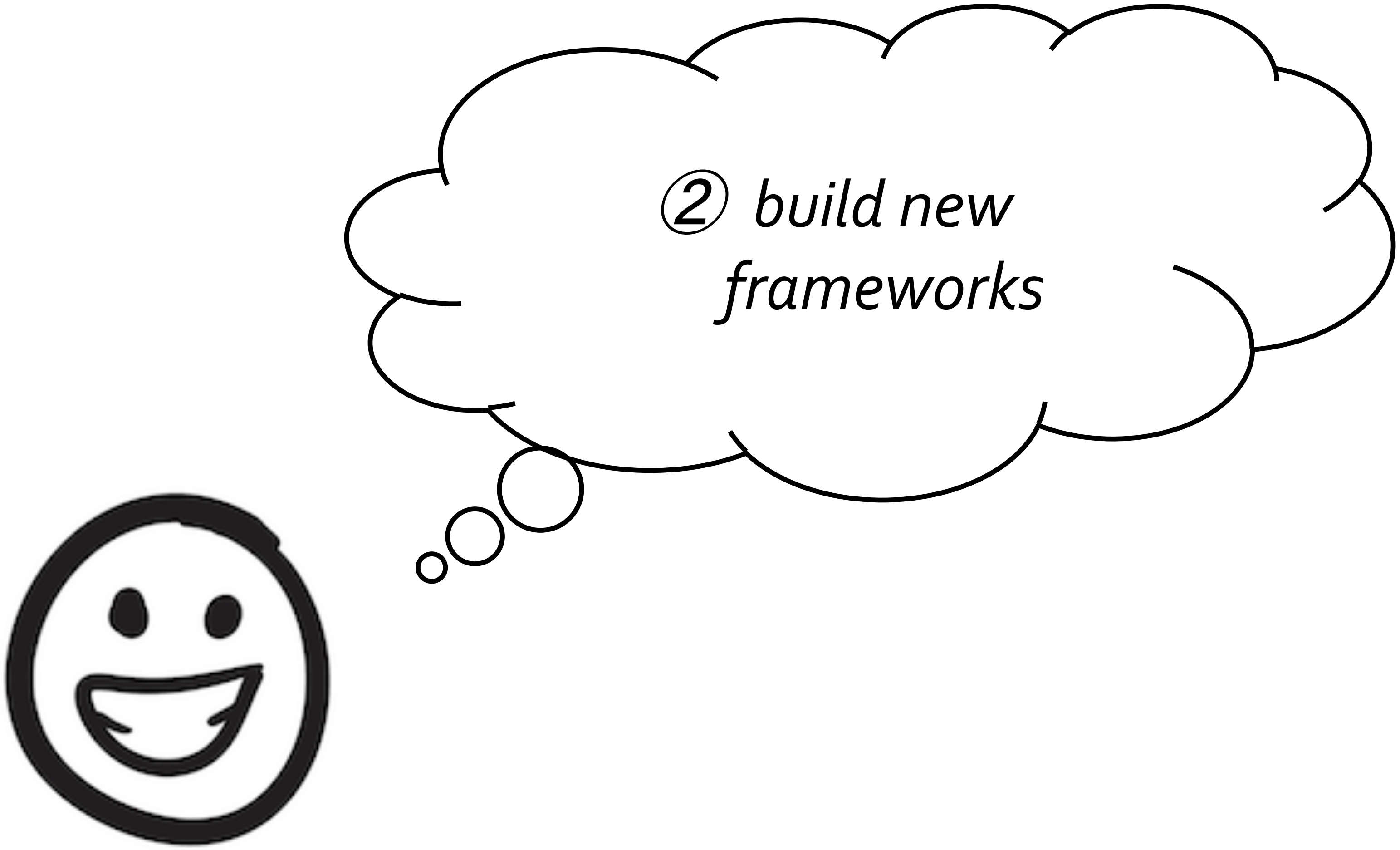
static partitioning considered harmful





higher utilization!

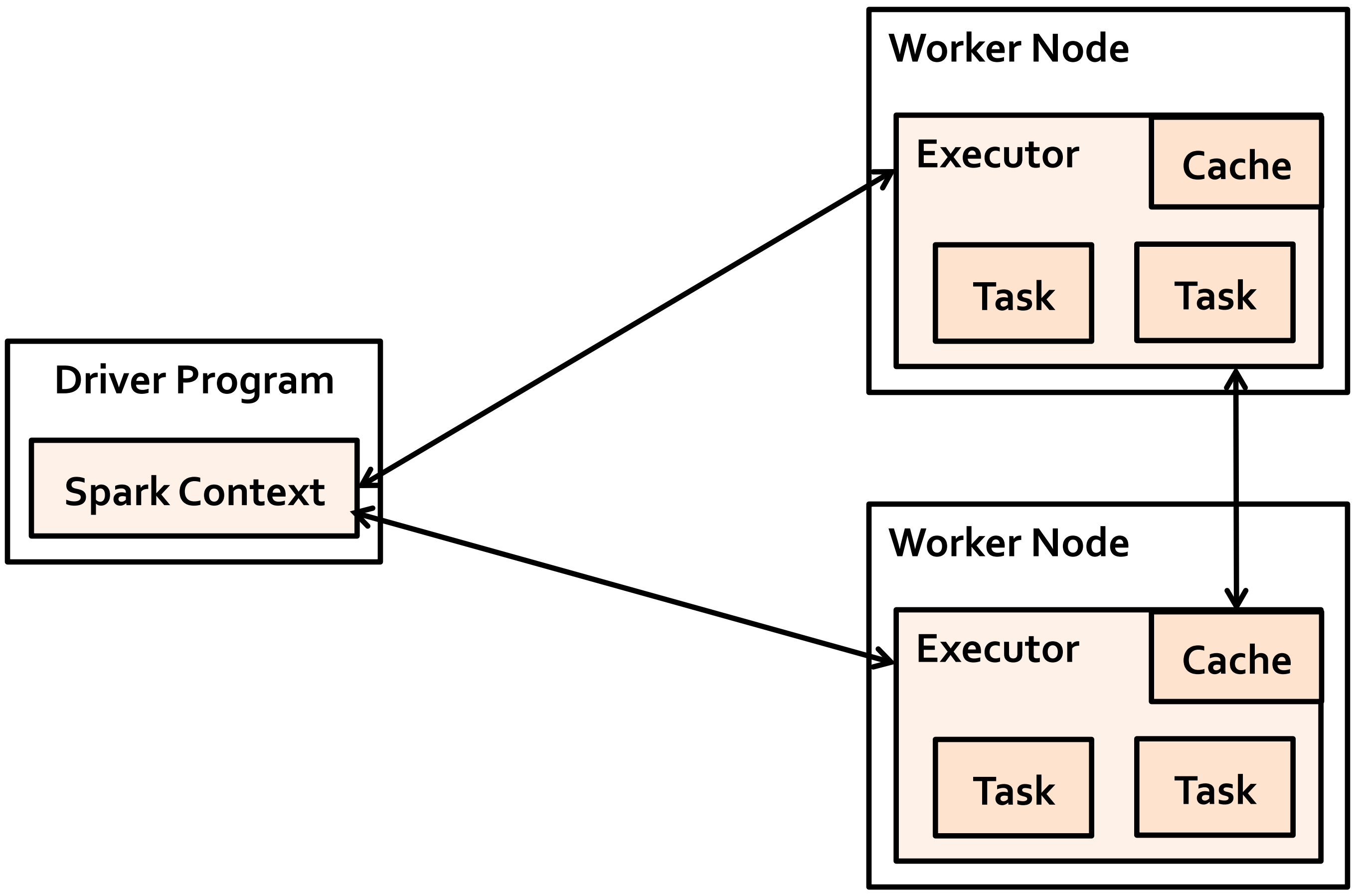
our intuition



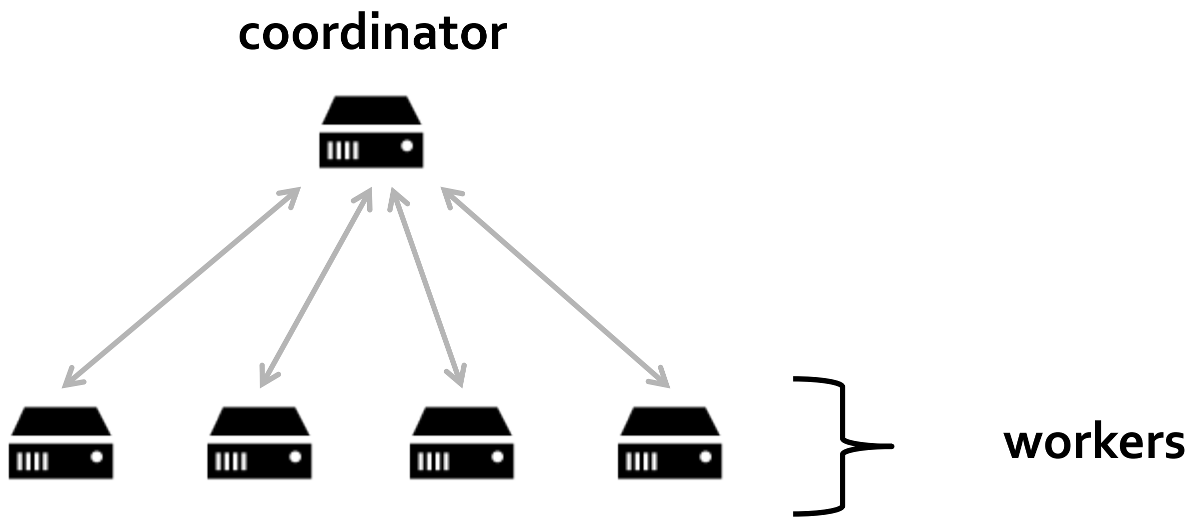
"Map/Reduce is a big hammer, but not everything is a nail!"

Spark

anatomy of Spark



anatomy of a framework

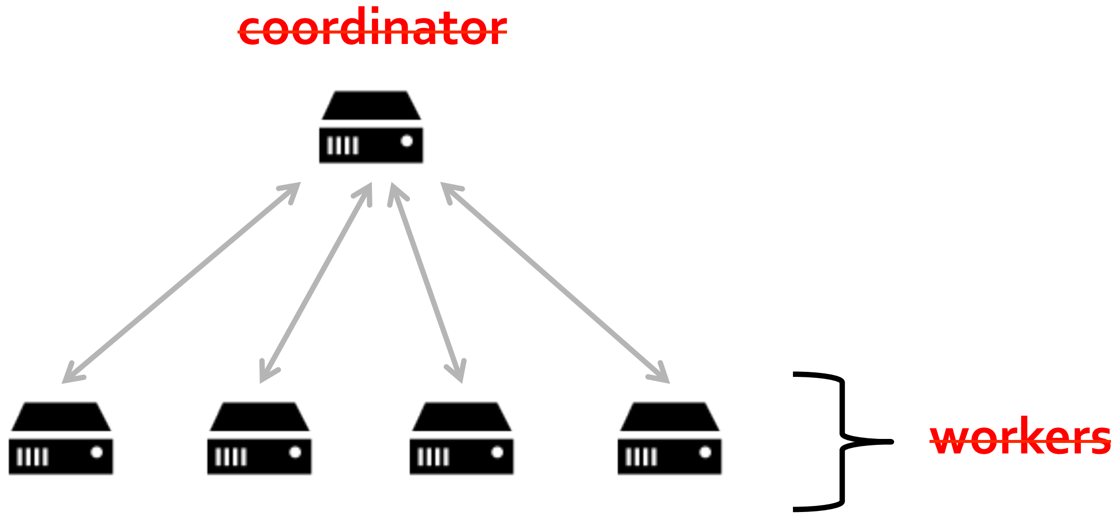


framework

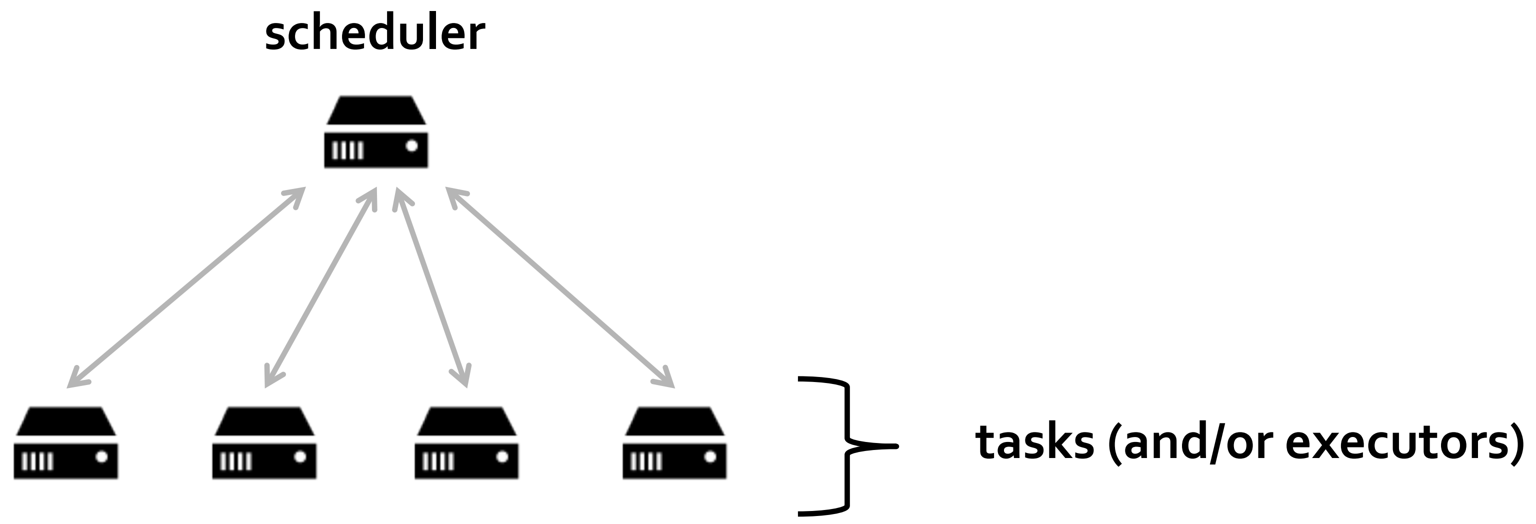
≈

distributed system

anatomy of a framework



anatomy of a framework

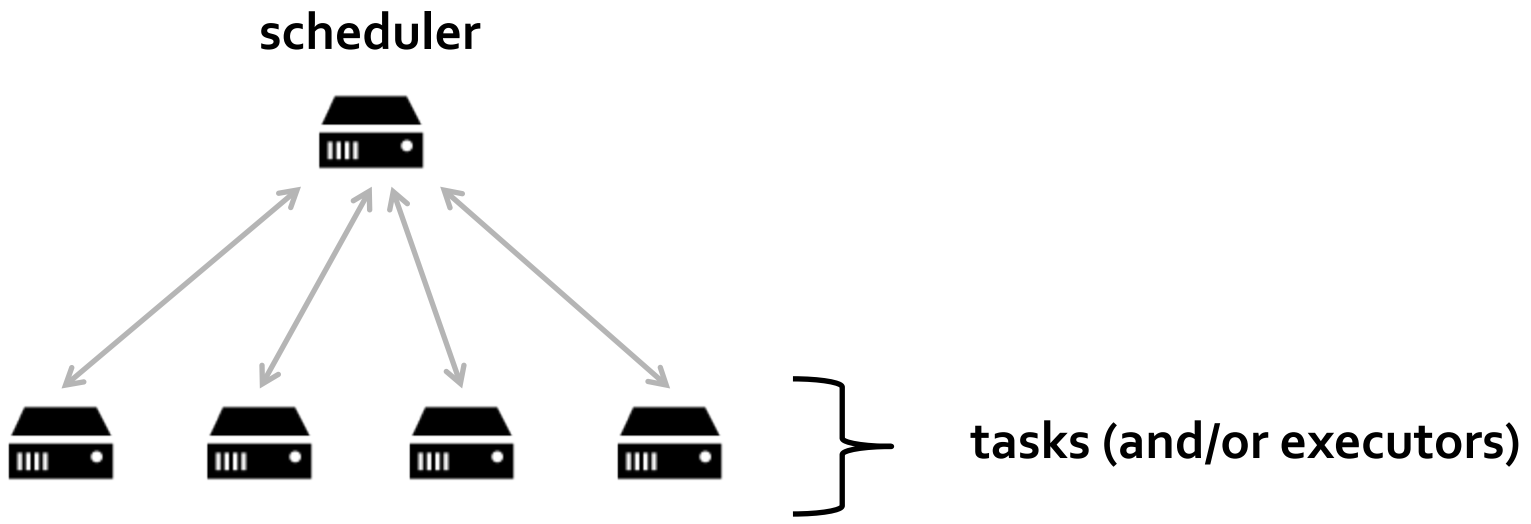


execution coordination

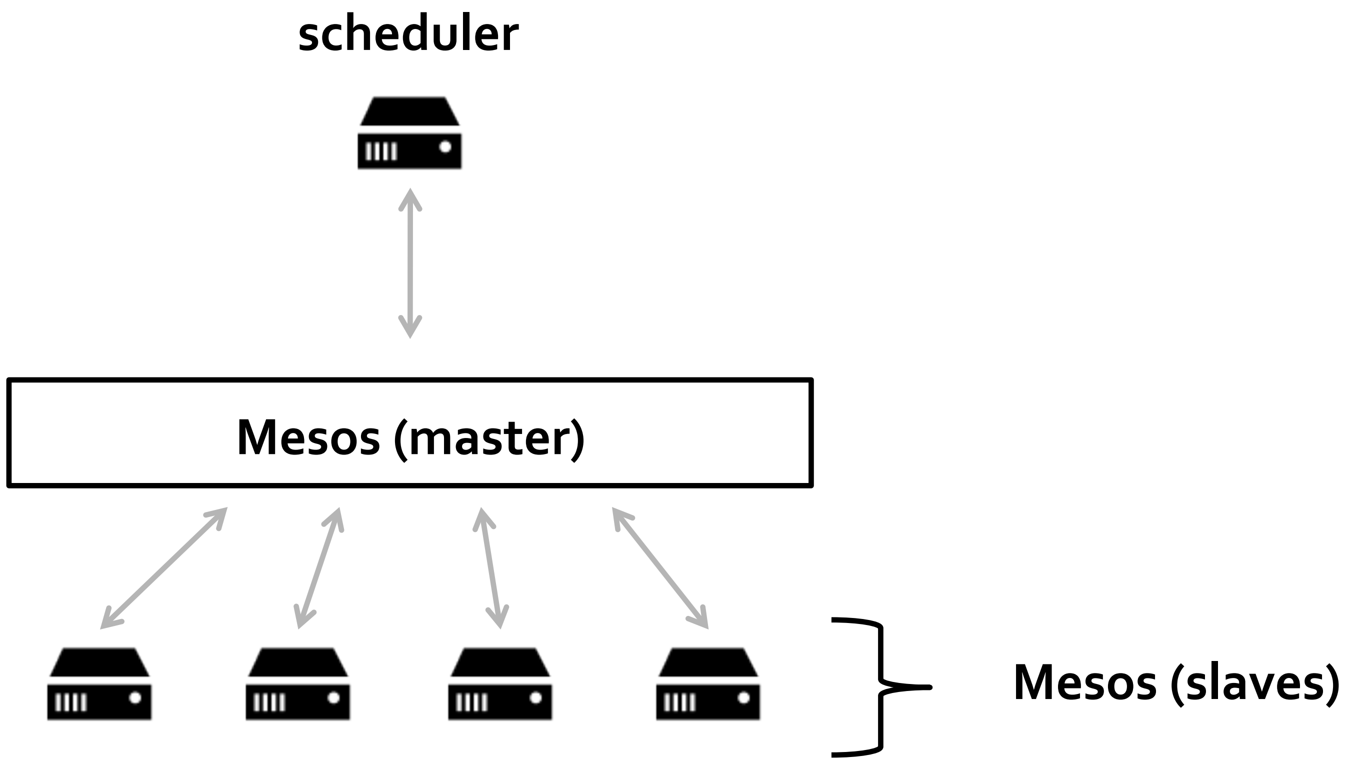
==

scheduling

Mesos: level of indirection



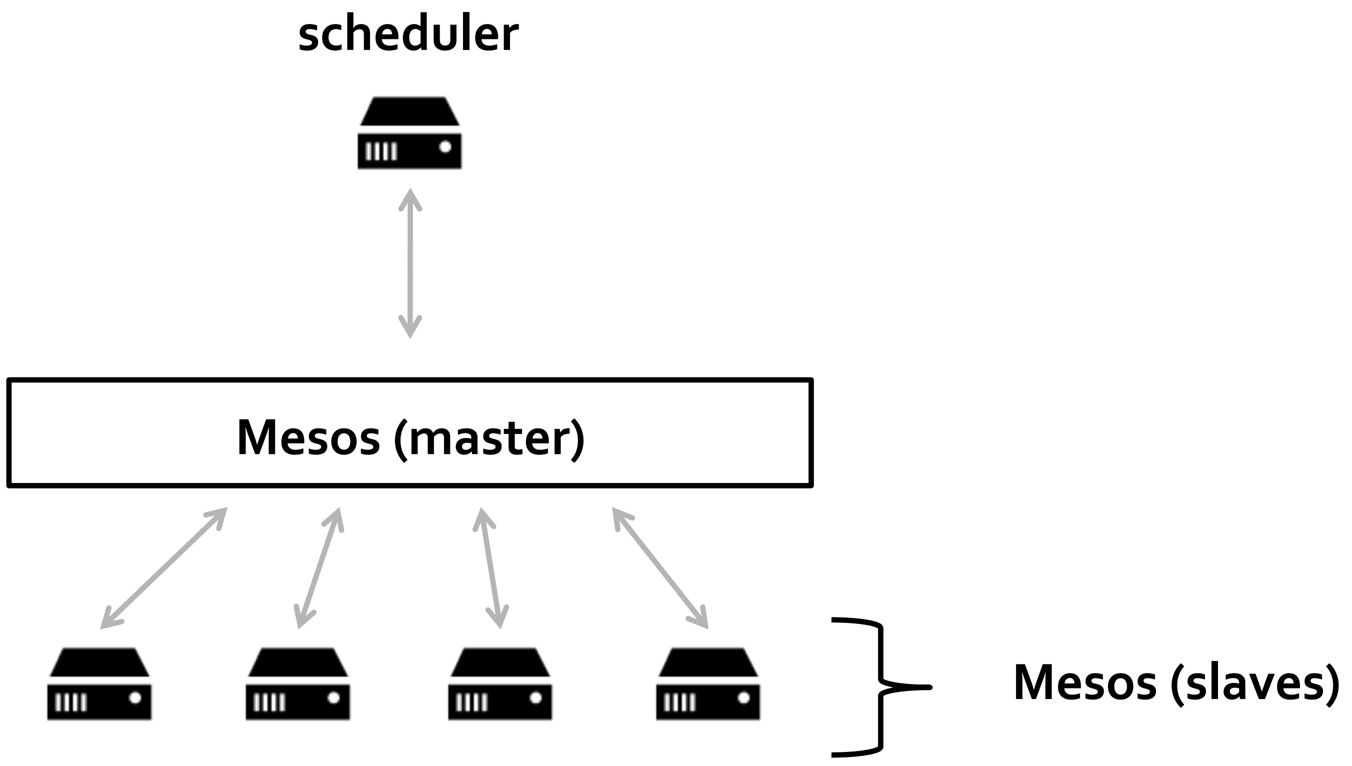
Mesos: level of indirection



Mesos: a level of indirection

+ provide common functionality every new distributed system *re-implements* like failure detection, task distribution, task starting, task monitoring, task killing, task cleanup!

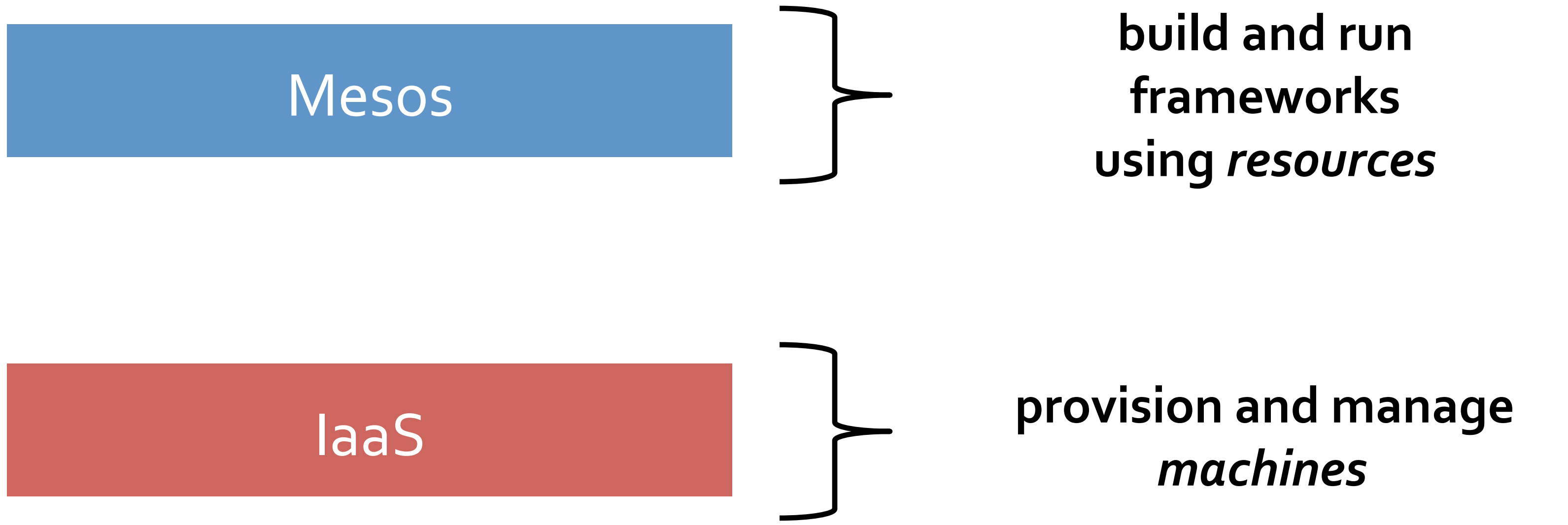
Mesos: level of abstraction



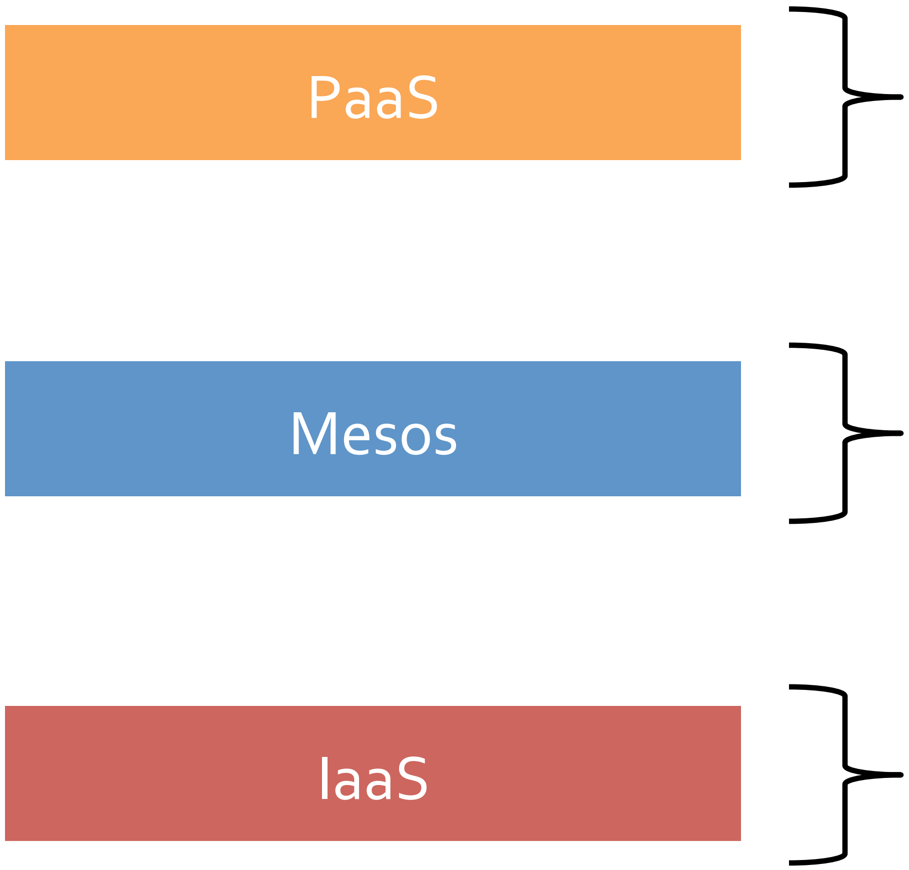
Mesos: level of abstraction

build and run
frameworks
using *resources*

Mesos: level of abstraction


Mesos
build and run frameworks using *resources*
IaaS
provision and manage *machines*

Mesos: level of abstraction

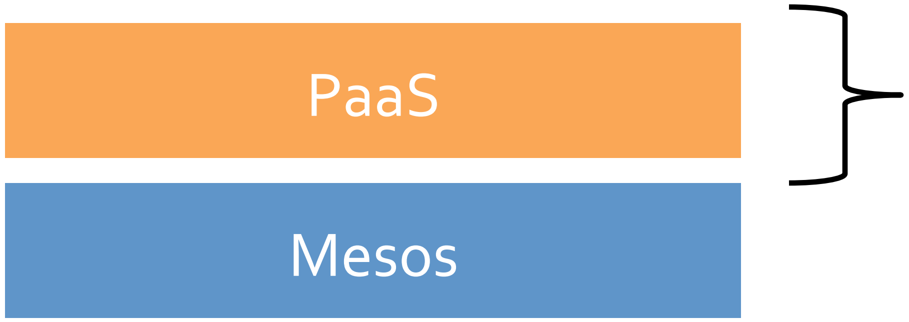


deploy and manage
applications/services

build and run
frameworks
using *resources*

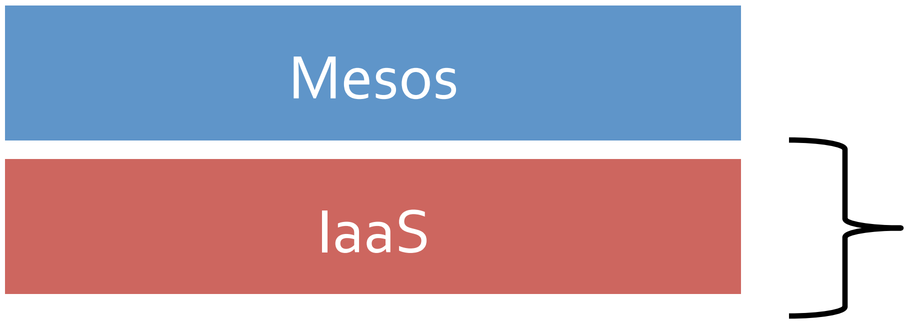
provision and manage
machines

PaaS on Mesos



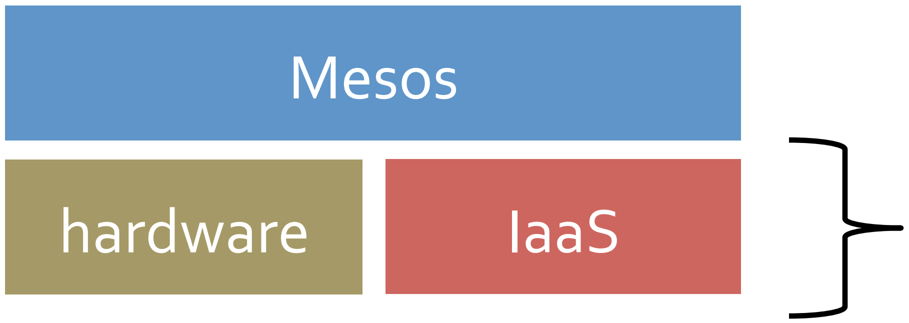
build and run a PaaS
on top of Mesos:
Apache Aurora and Marathon

Mesos on IaaS



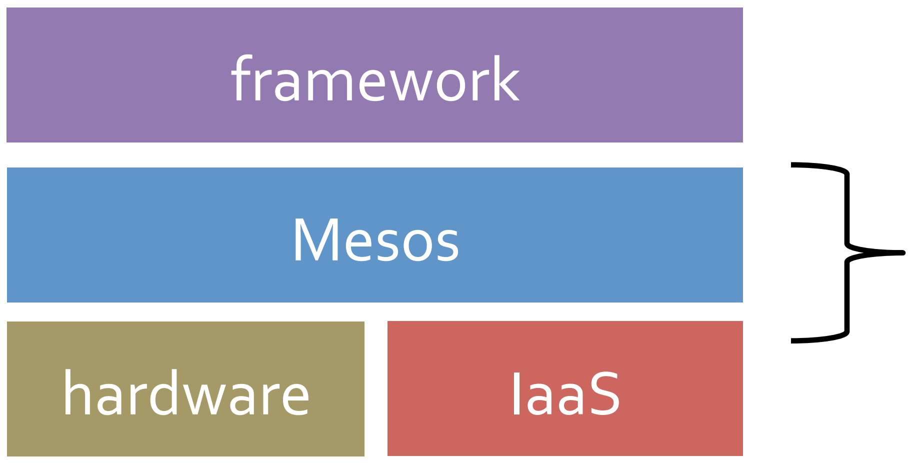
use OpenStack or EC2
to run Mesos

Mesos on IaaS++



use OpenStack or EC2
or physical machines
to run Mesos

Mesos: datacenter kernel



provide common functionality
via an API (kernel)

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

Mesos is a cluster manager

agenda

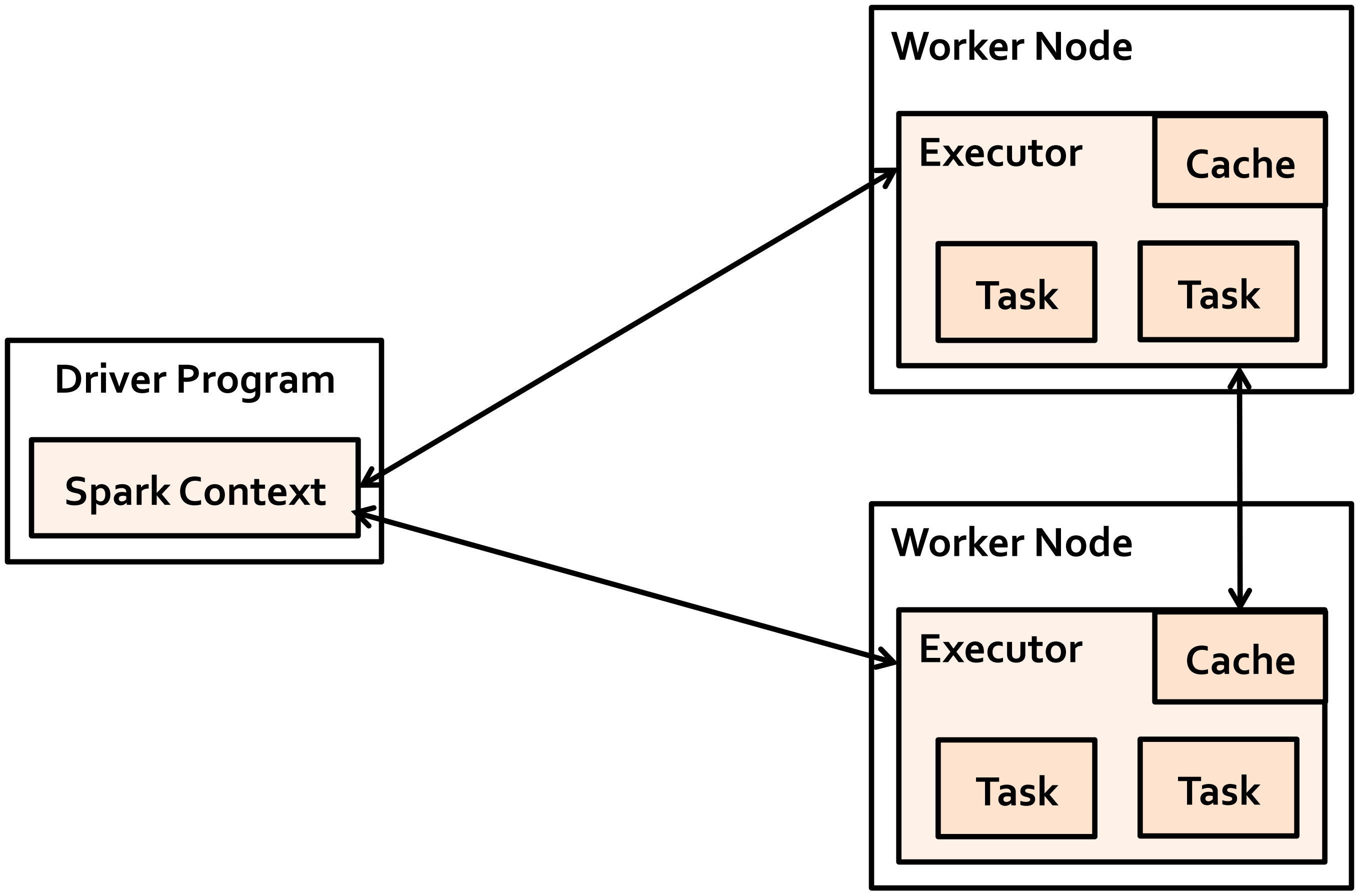
① Mesos

② Spark on Mesos ←

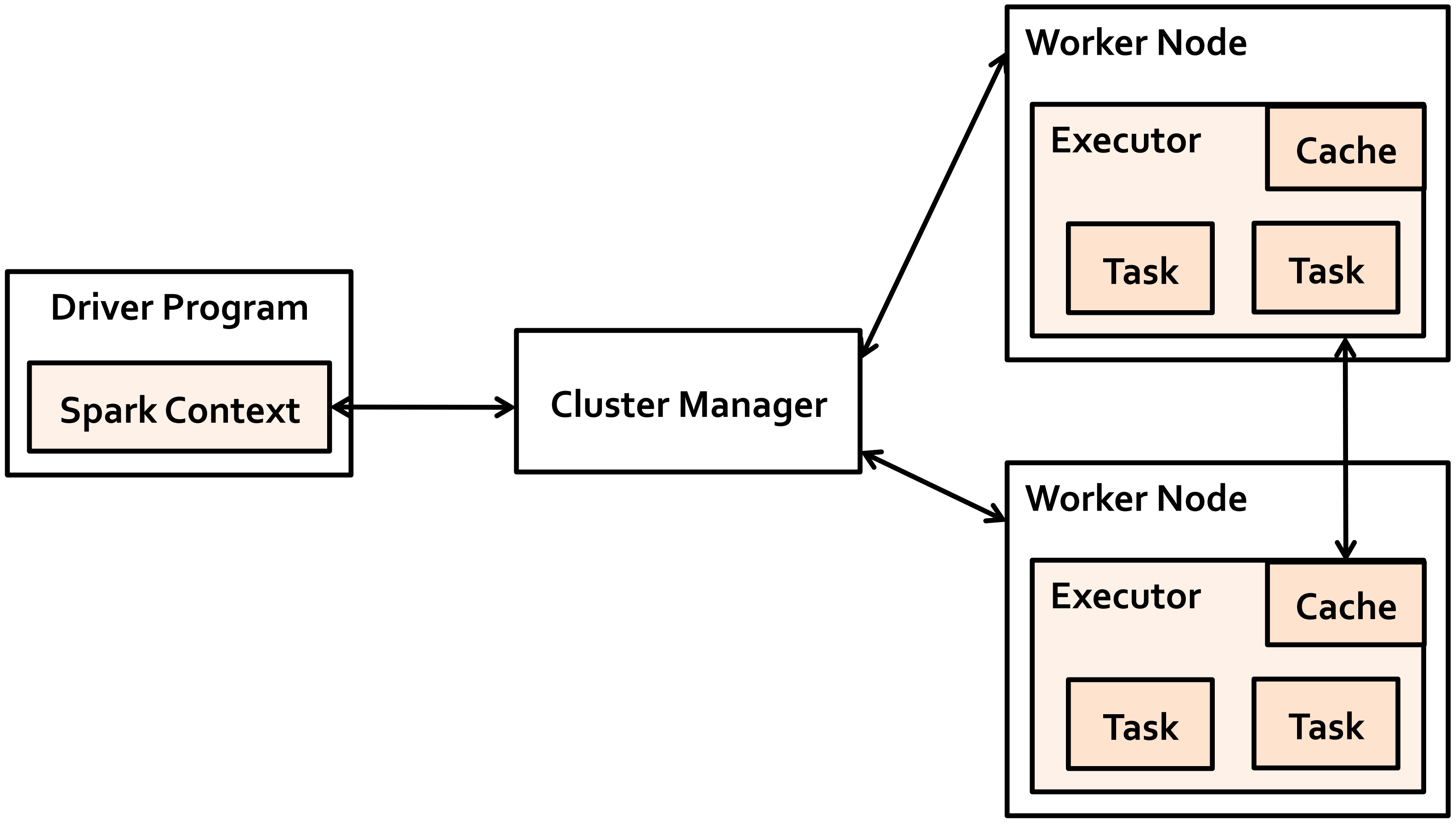
③ why Mesos?
- ① multi-tenancy
- ② fine-grained sharing
- ③ why not?

④ long-lived services and other frameworks

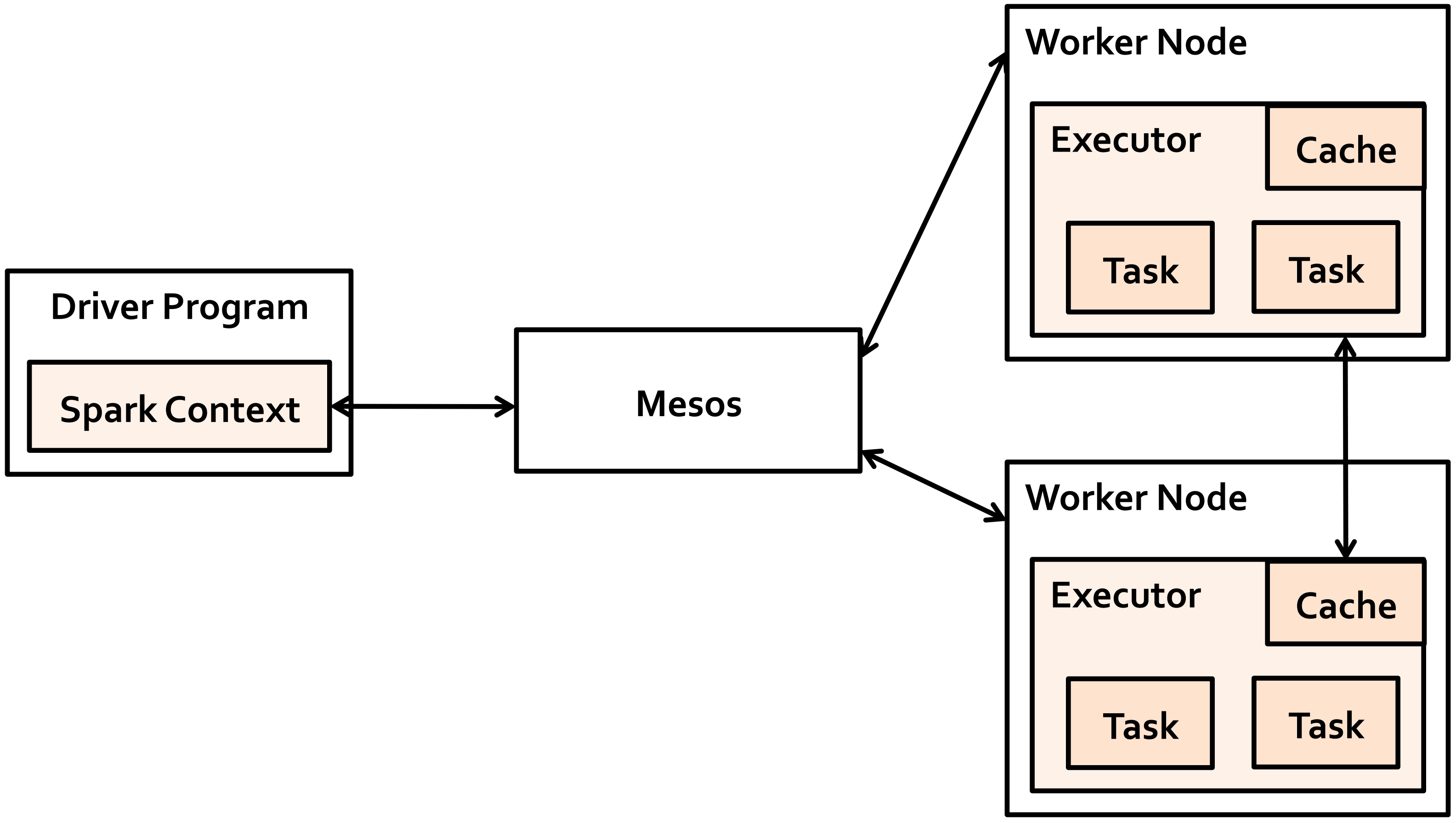
anatomy of Spark



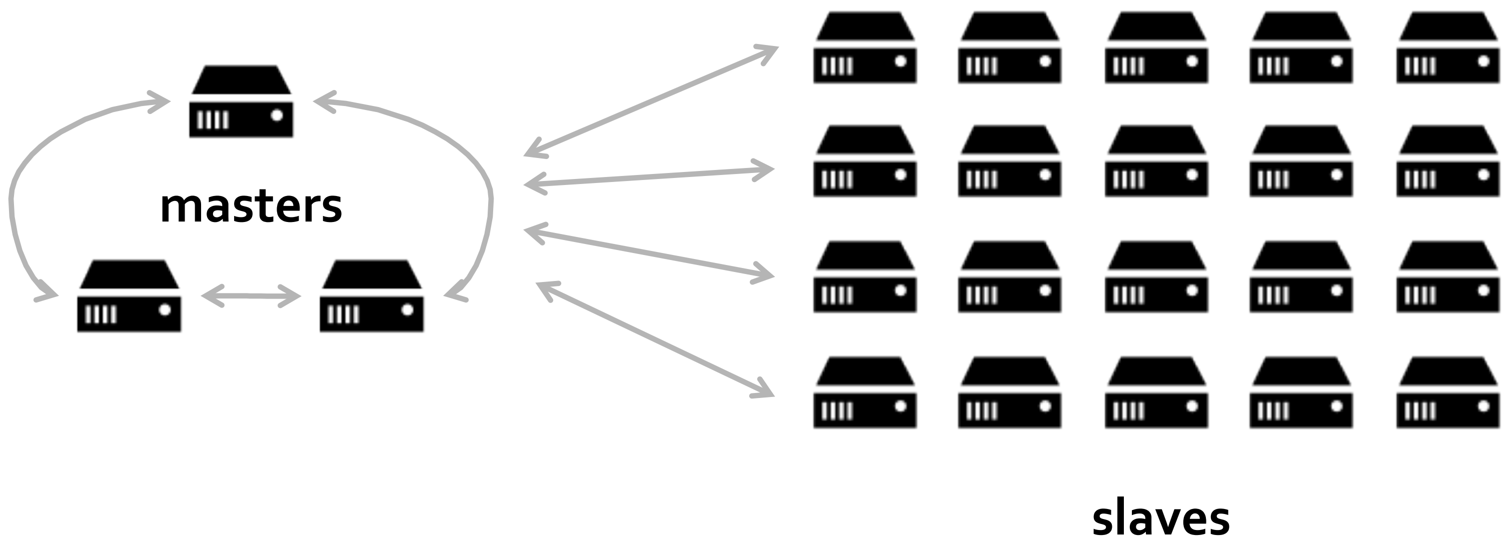
anatomy of Spark



anatomy of Spark

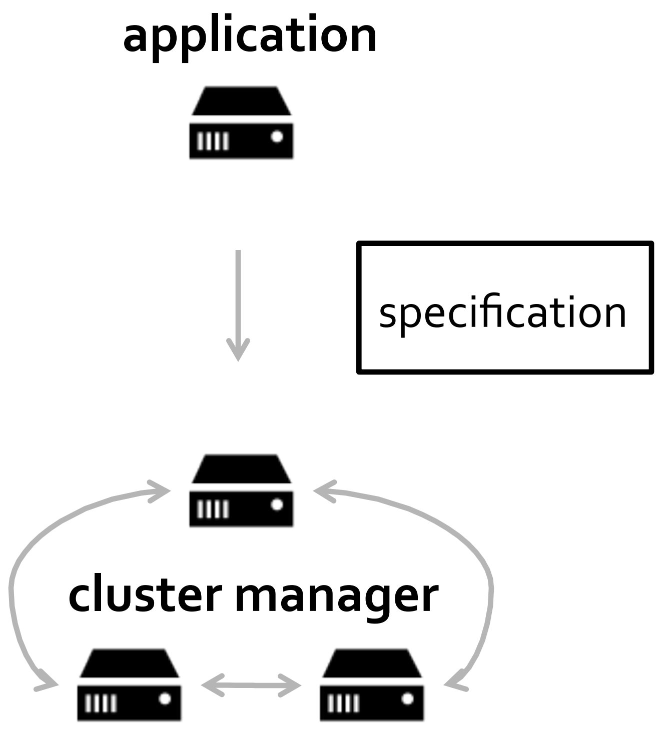


Mesos is a distributed system with a master/slave architecture



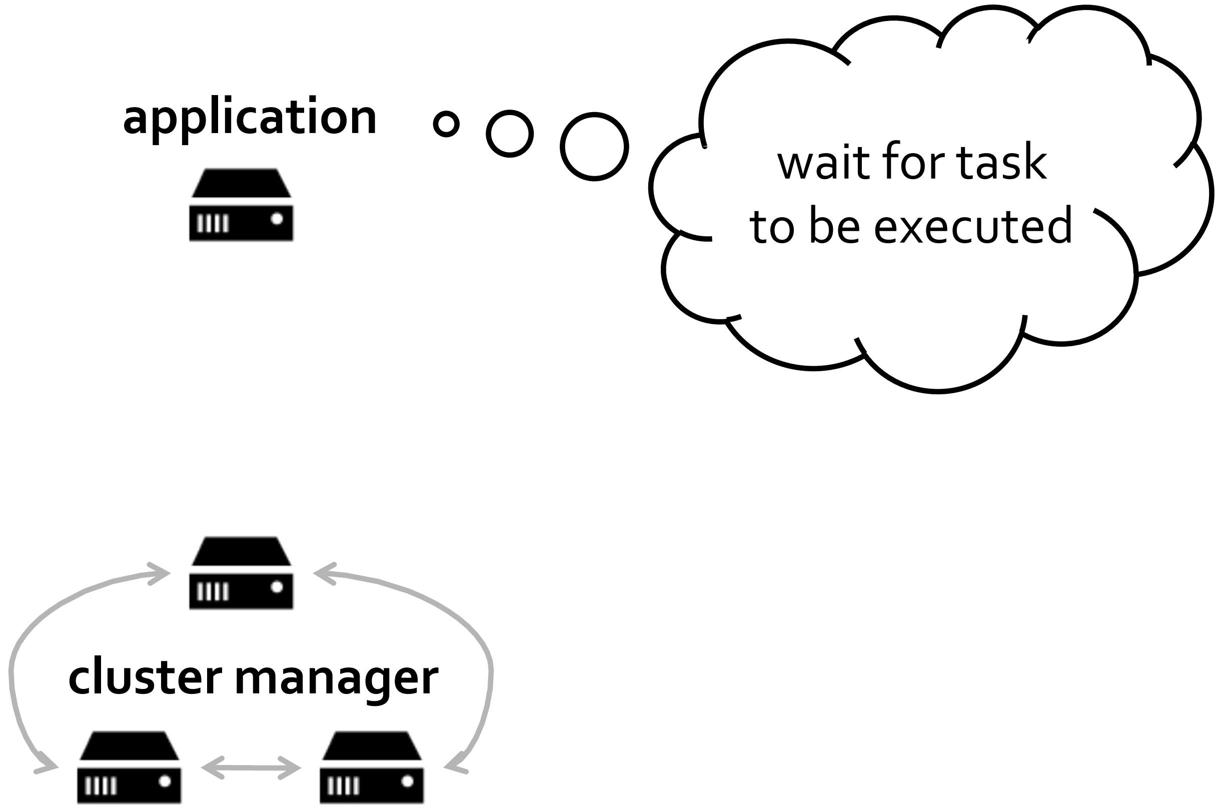
Mesos challenged the status quo of cluster managers

cluster manager status quo

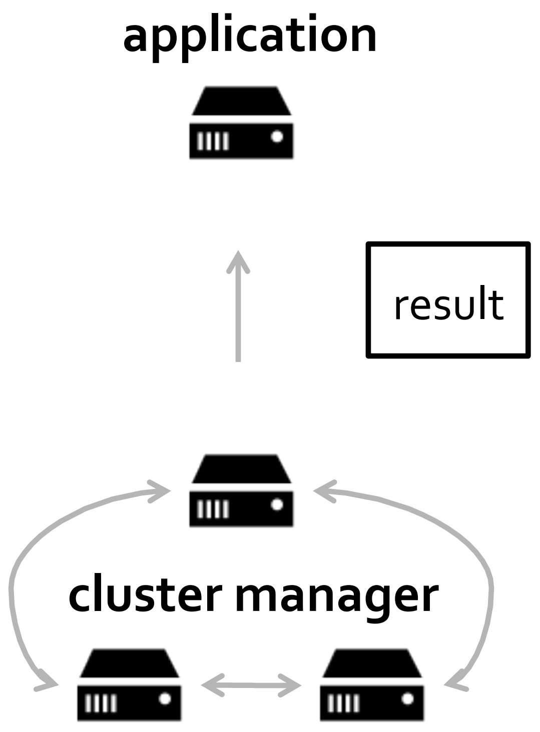


the specification includes as much information as possible to assist the cluster manager in scheduling and execution

cluster manager status quo



cluster manager status quo

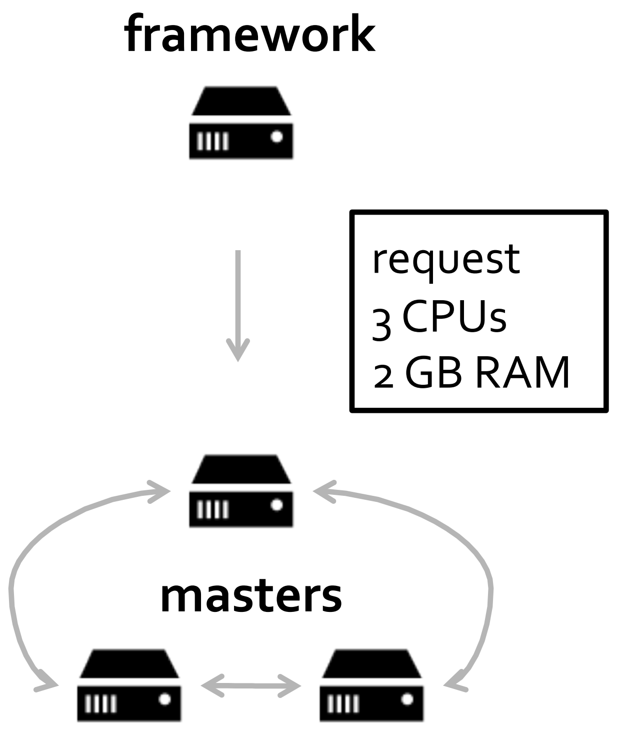


problems with specifications

① hard to specify certain desires or constraints

② hard to update specifications dynamically as tasks executed and finished/failed

an alternative model



a request is purposely simplified subset of a specification, mainly including the required resources

question: what should you do if you can‘t satisfy a request?

question: what should you do if you can't satisfy a request?

① wait until you can …

question: what should you do if you can‘t satisfy a request?

① wait until you can …

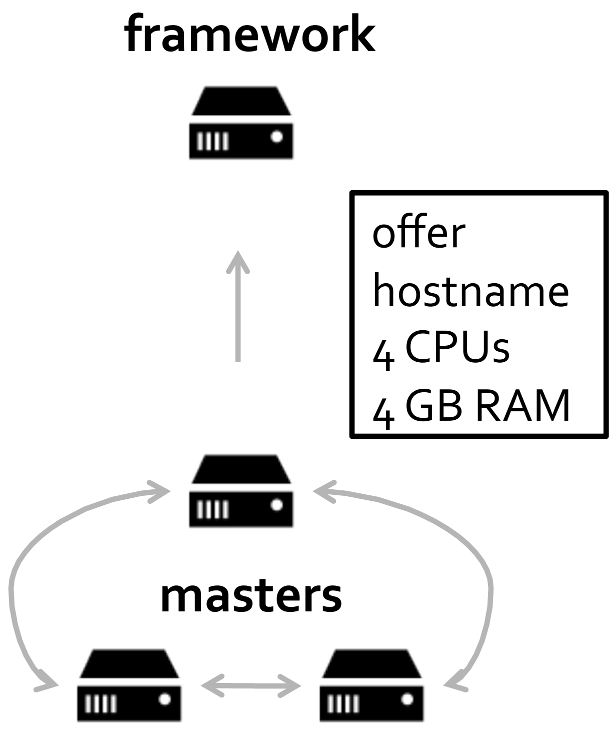
② *offer* best you can <u>immediately</u>

question: what should you do if you can't satisfy a request?

① wait until you can …

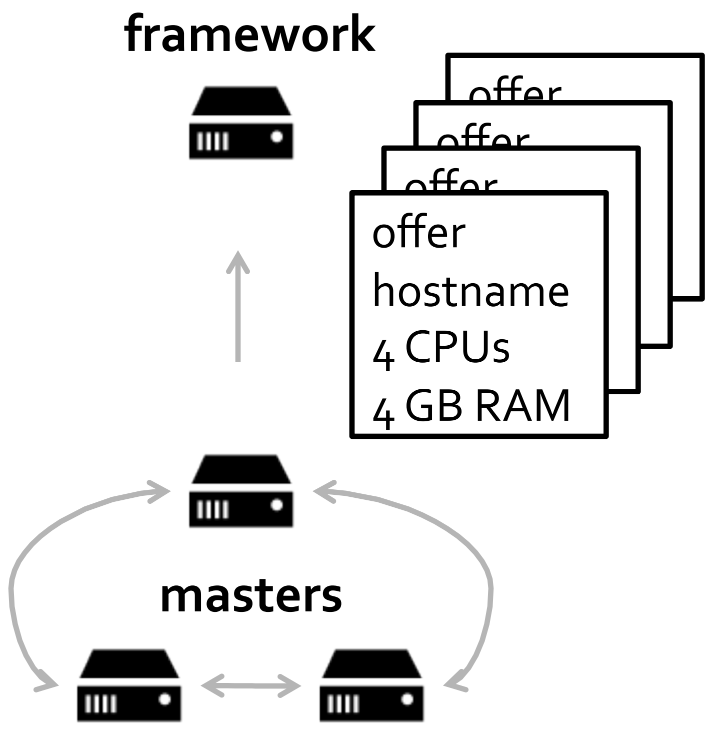
② *offer* best you can <u>immediately</u>

Mesos model



offer
hostname
4 CPUs
4 GB RAM

Mesos model

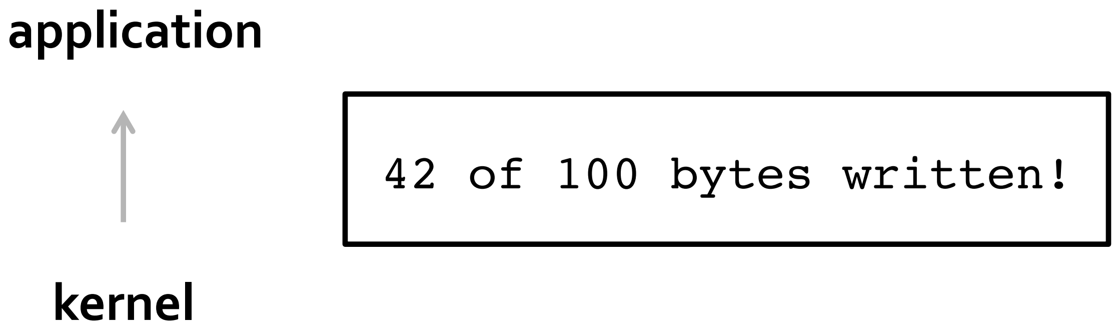

framework
offer
offer
offer
offer
hostname
4 CPUs
4 GB RAM
masters

an analogue: non-blocking sockets

$$\begin{tikzcd}\text{application} \arrow[d] \\ \text{kernel}\end{tikzcd}$$

```
write(s, buffer, size);
```

an analogue: non-blocking sockets

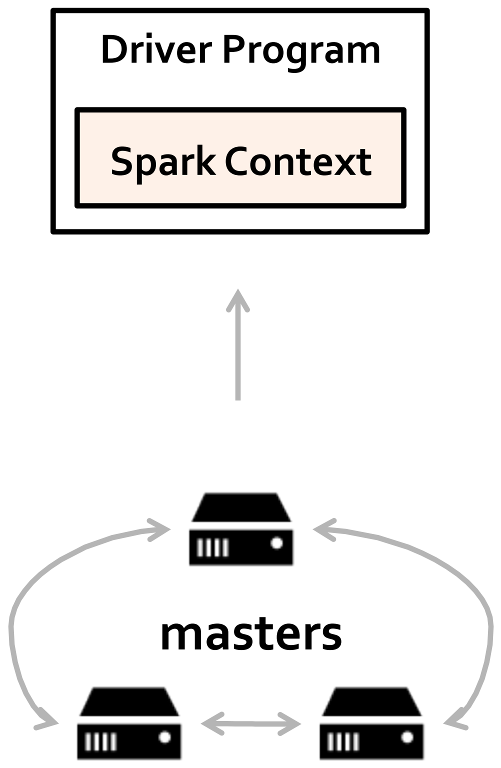


```
42 of 100 bytes written!
```

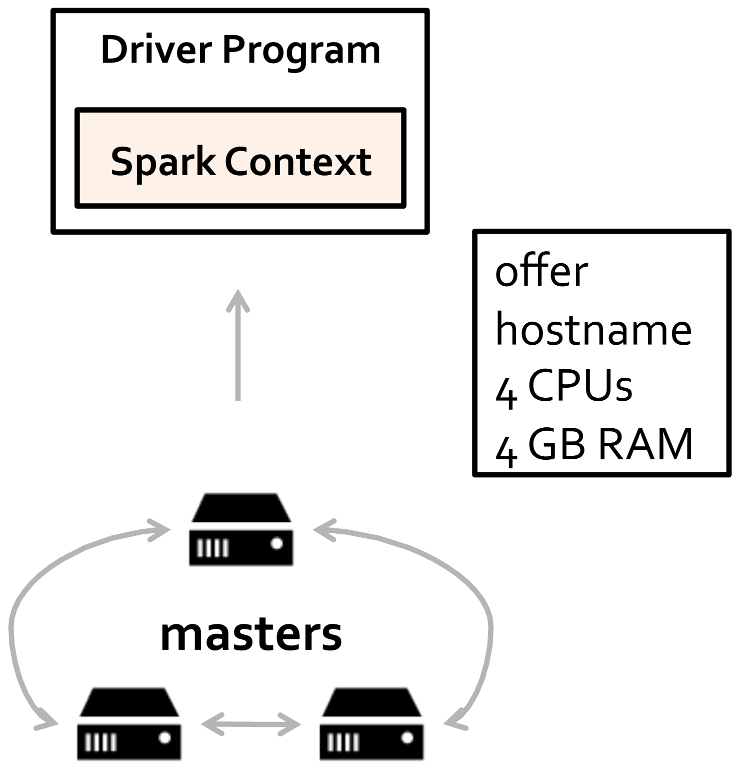
offers represent the current *snapshot* of available resources a framework can use

(requests are complimentary, but not necessary; see Google's Omega)

Spark on Mesos

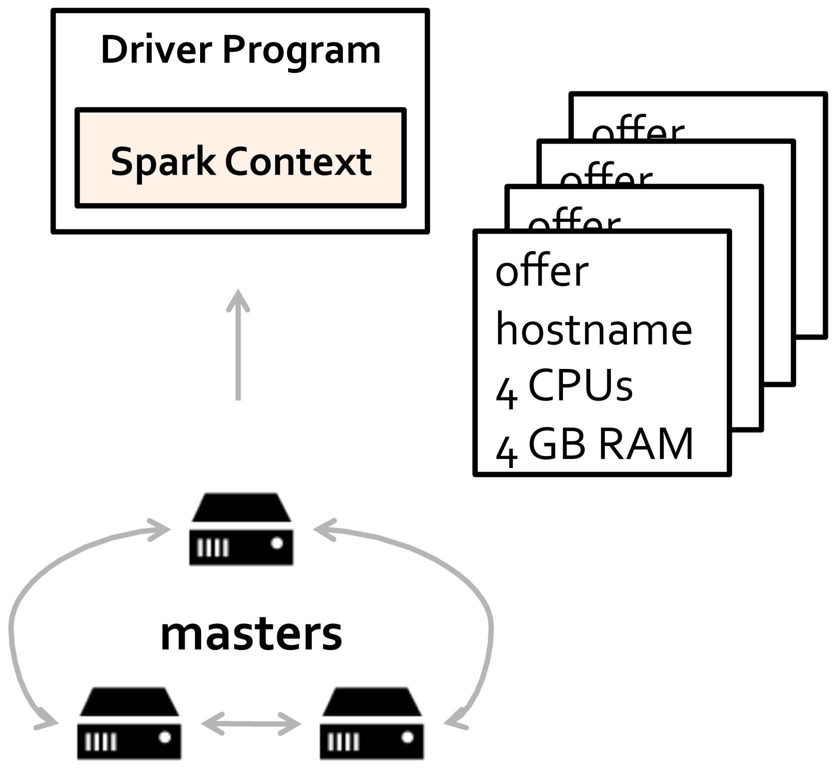


Spark on Mesos

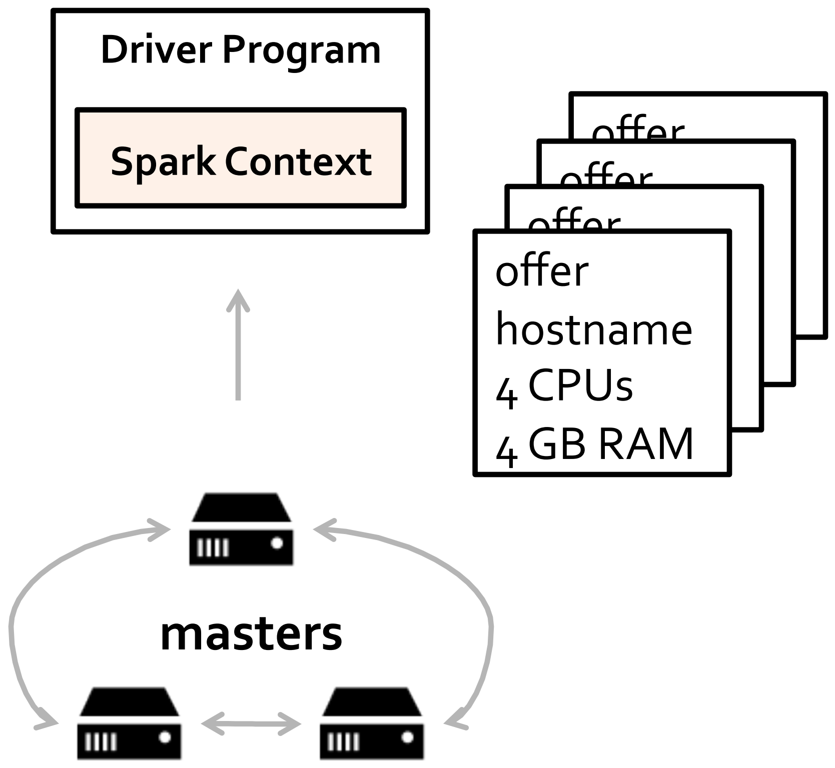


offer
hostname
4 CPUs
4 GB RAM

Spark on Mesos

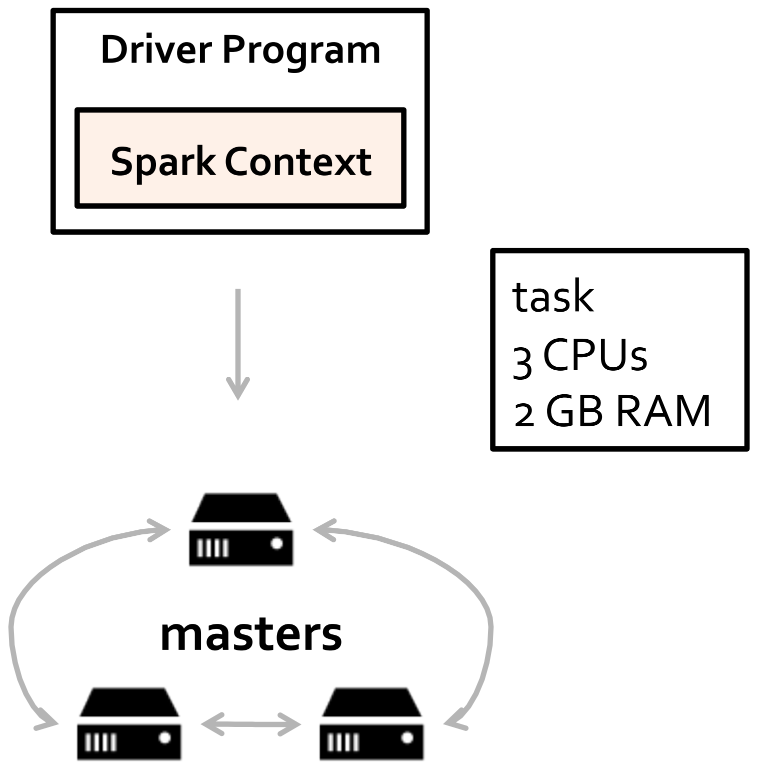


Spark on Mesos



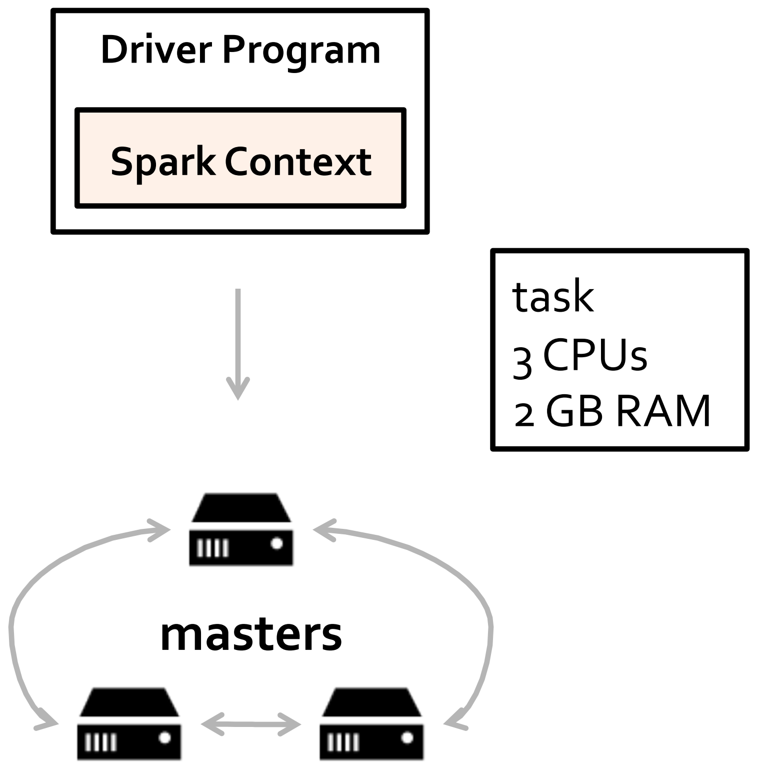
Spark uses the offers to perform it's own scheduling

Spark on Mesos



Spark uses the offers to perform it's own scheduling

Spark on Mesos



Spark uses the offers to perform it's own scheduling

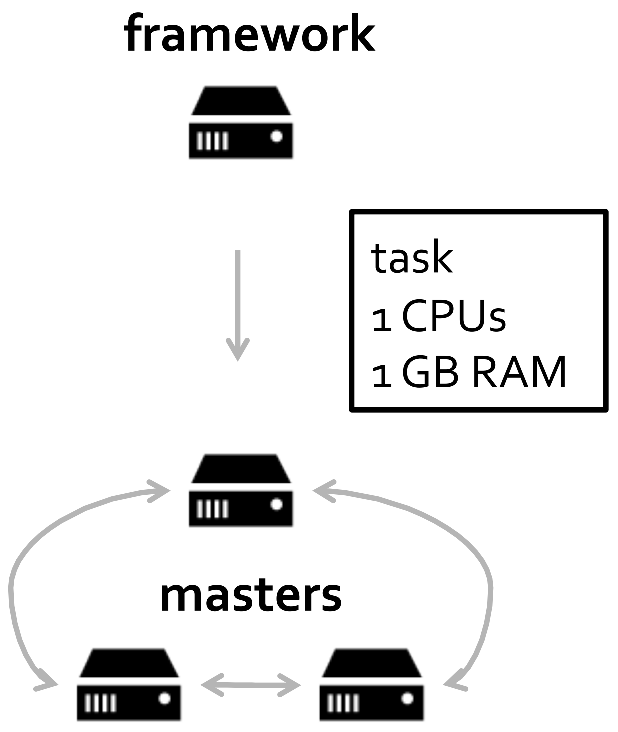
"two-level scheduling"

“two-level scheduling”

Mesos: controls resource *allocations* to Spark

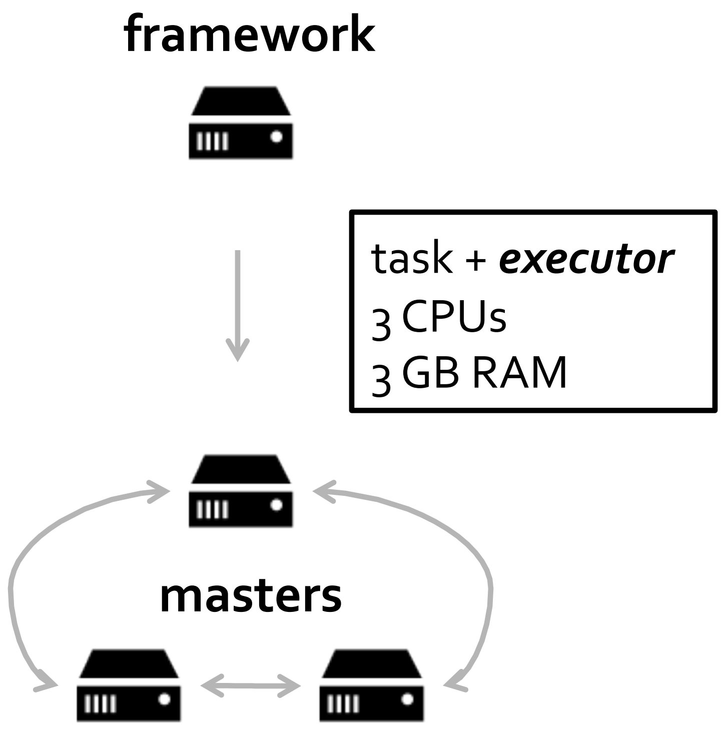
Spark: makes decisions about what tasks to run given available resources

execution



frameworks launch fine-grained tasks for execution

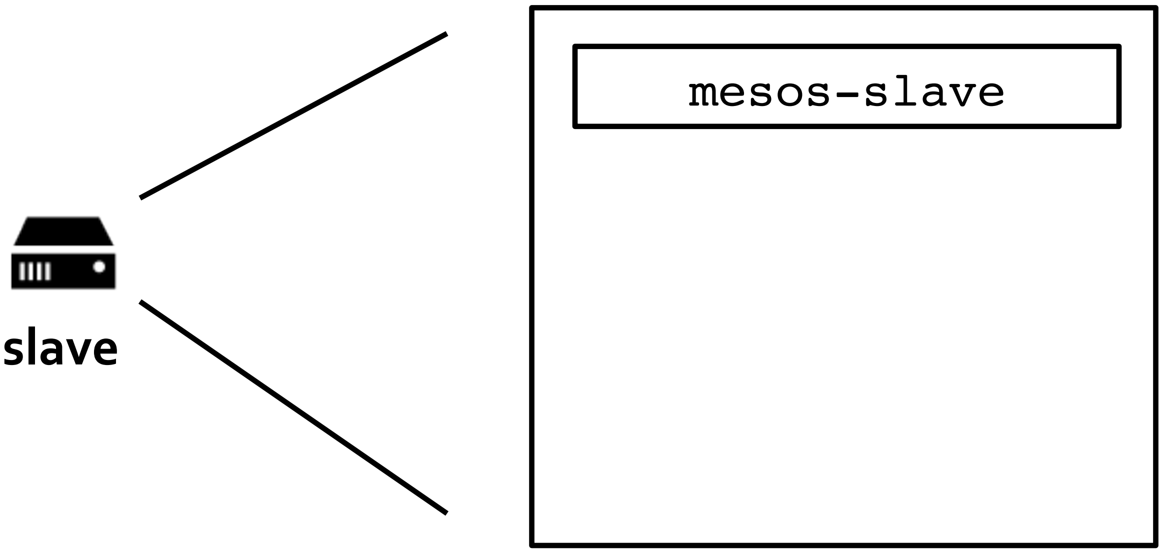
execution



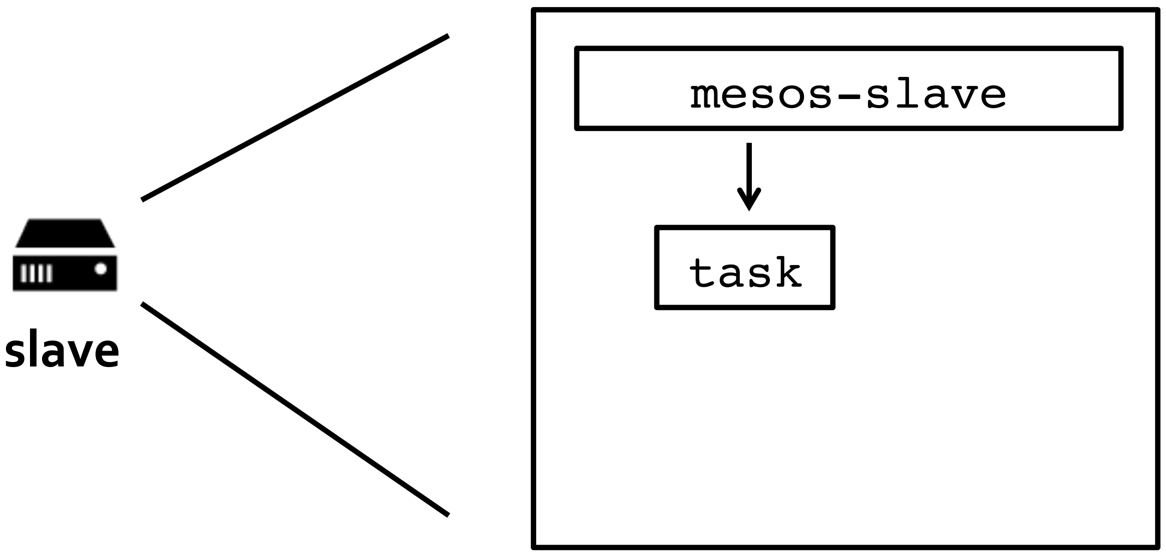
frameworks launch fine-grained tasks for execution

if necessary, a framework can provide an *executor* to handle the execution of a task

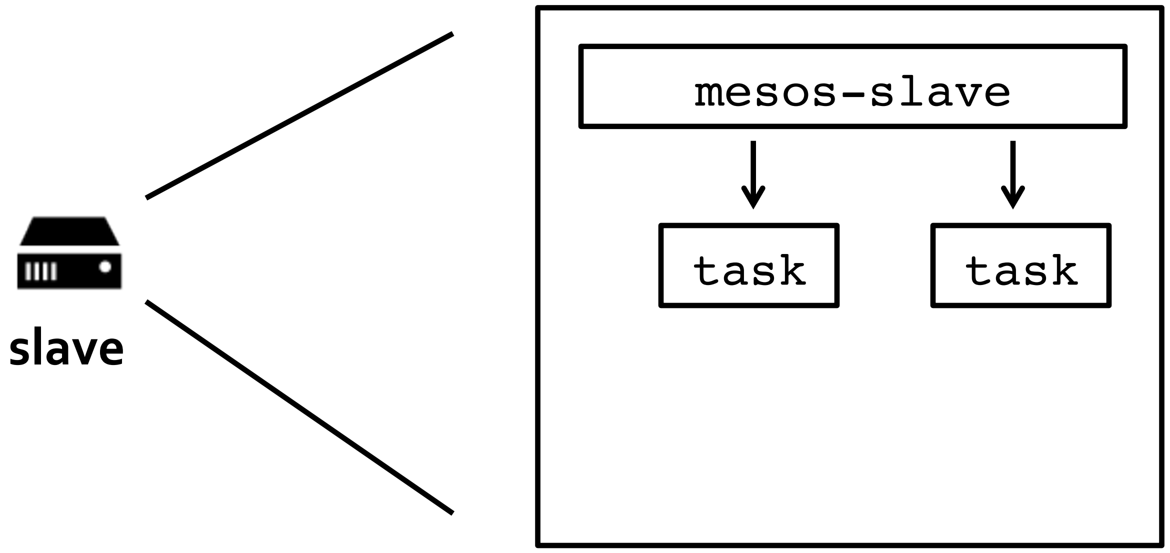
a task with a *command*



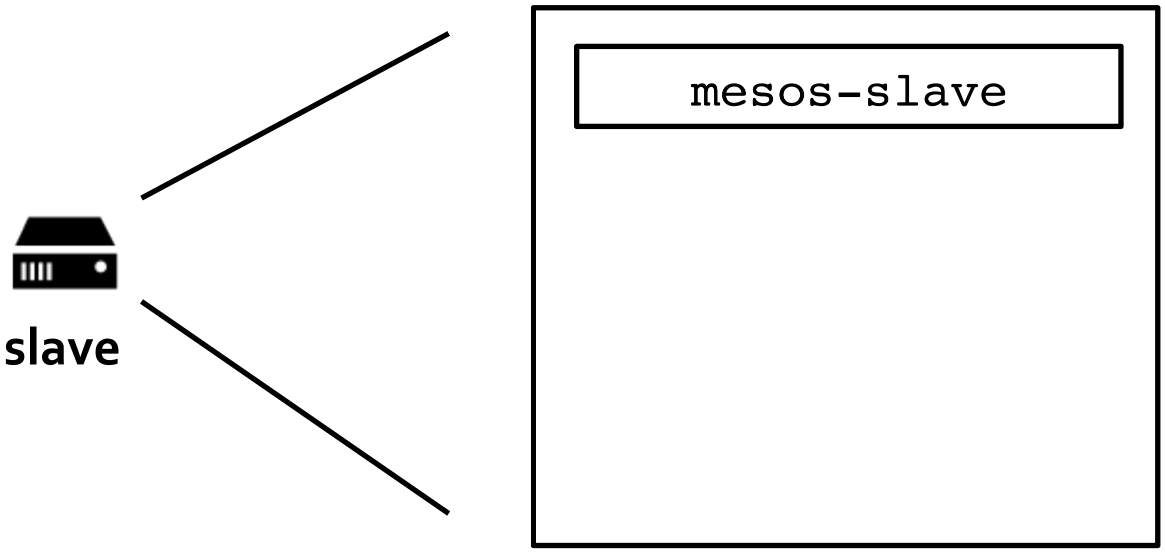
a task with a *command*



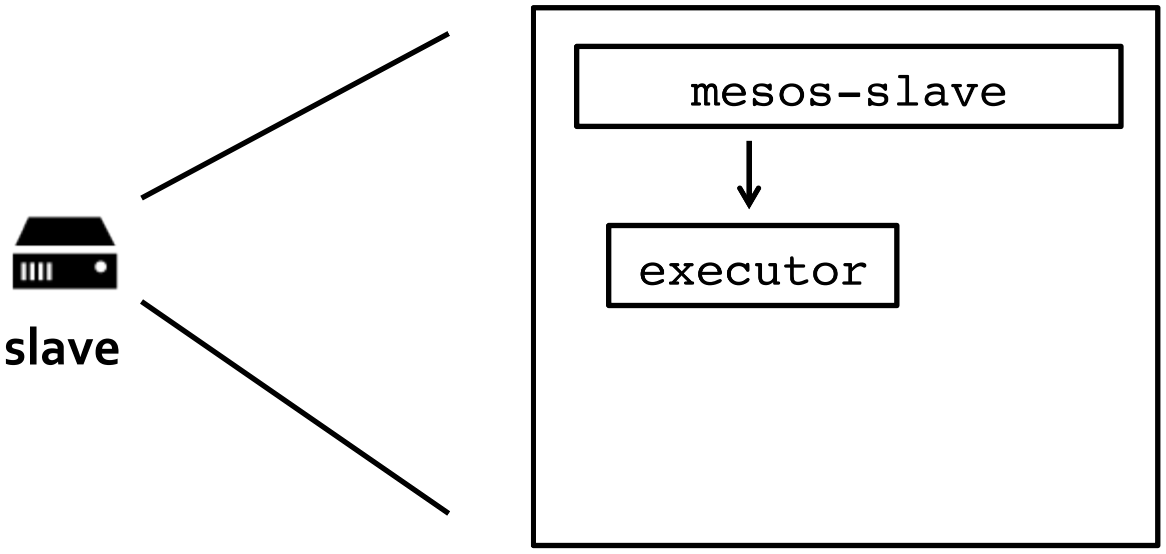
a task with a *command*



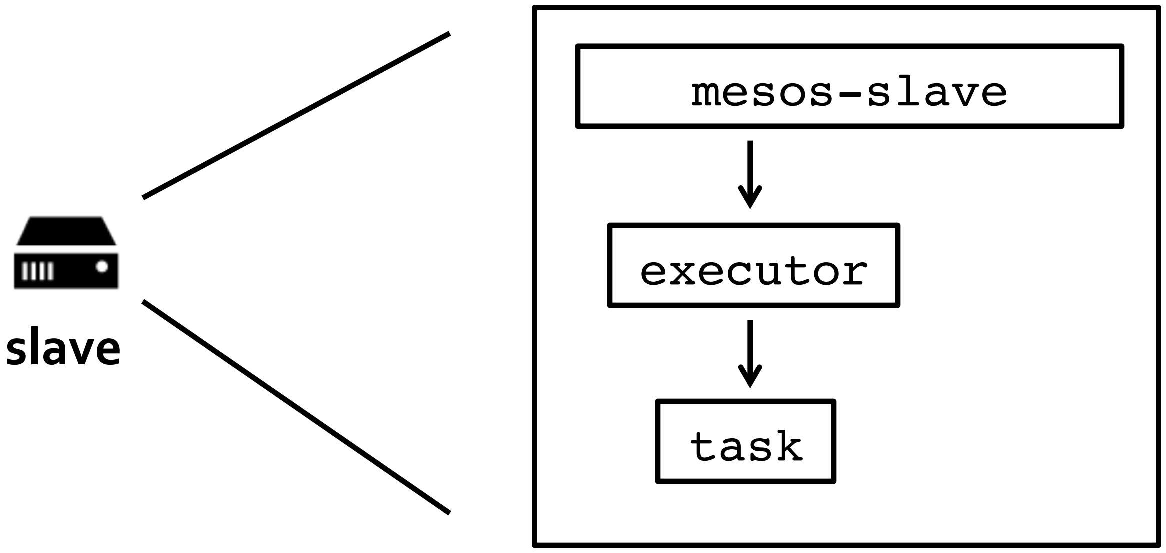
a task with an *executor*



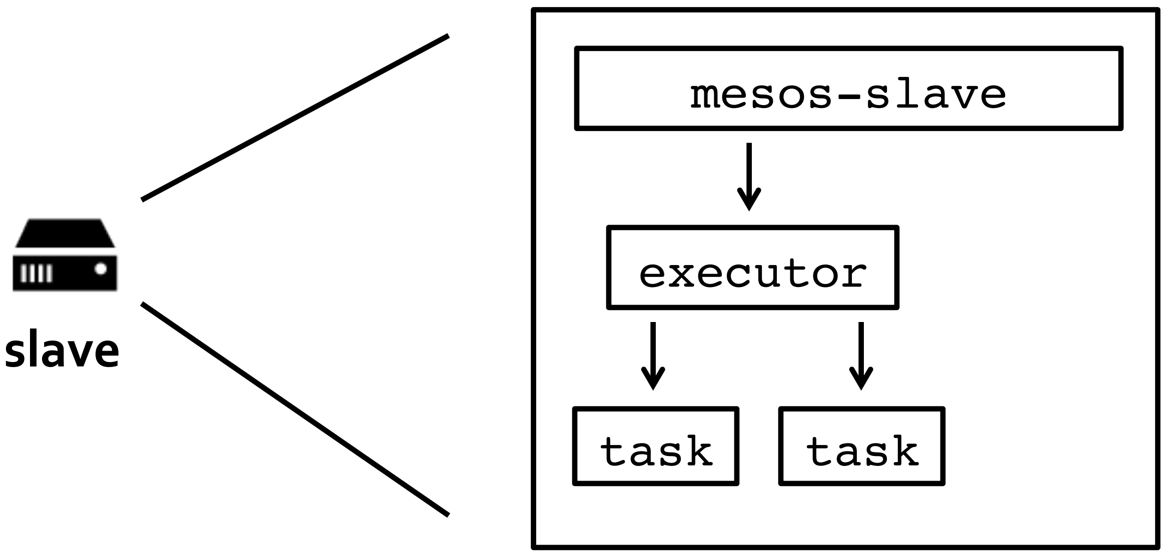
a task with an *executor*



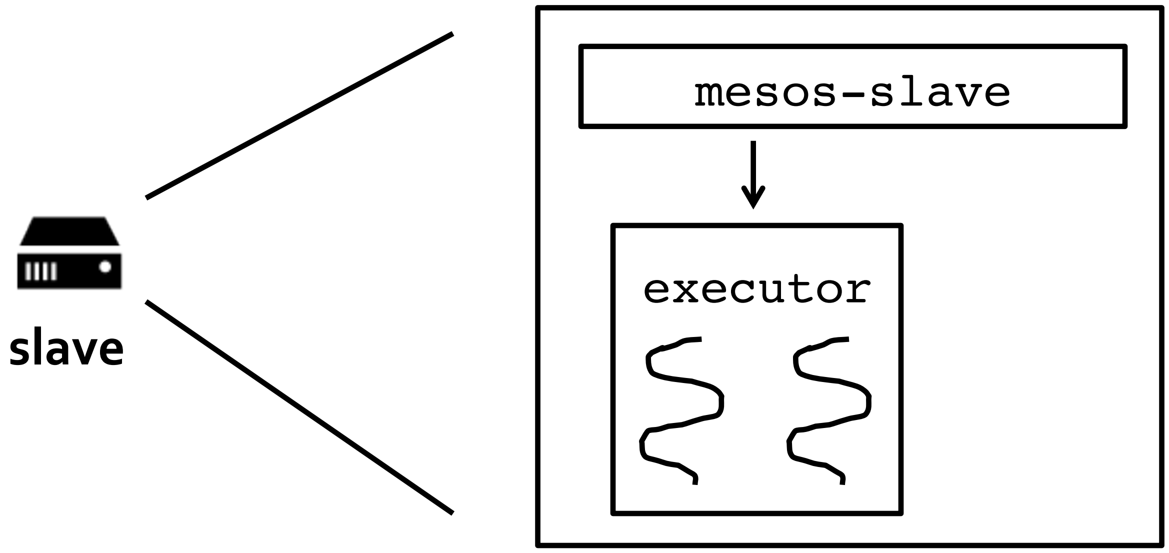
a task with an *executor*



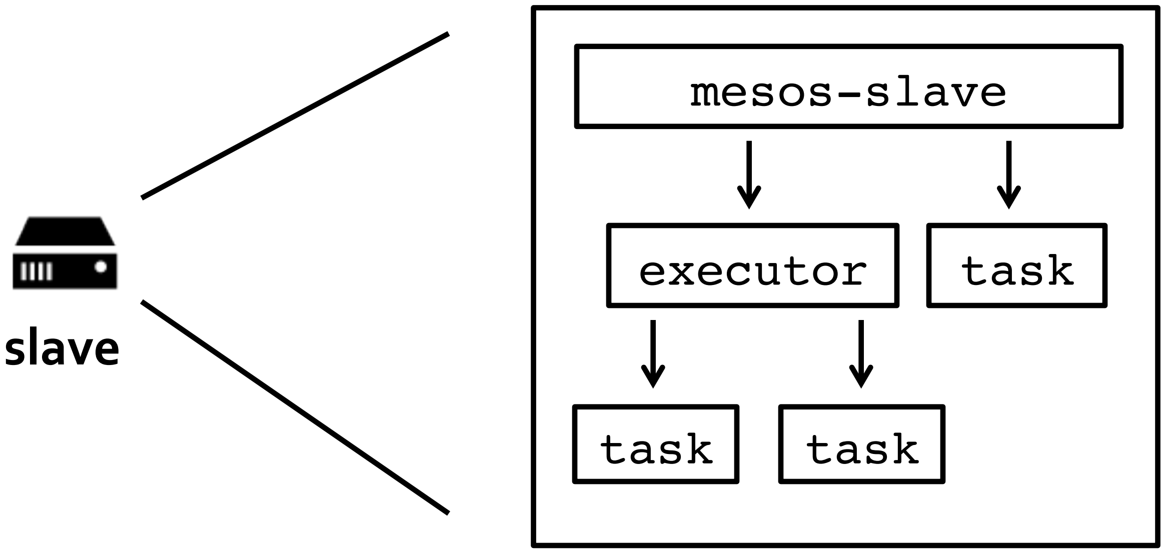
a task with an *executor*



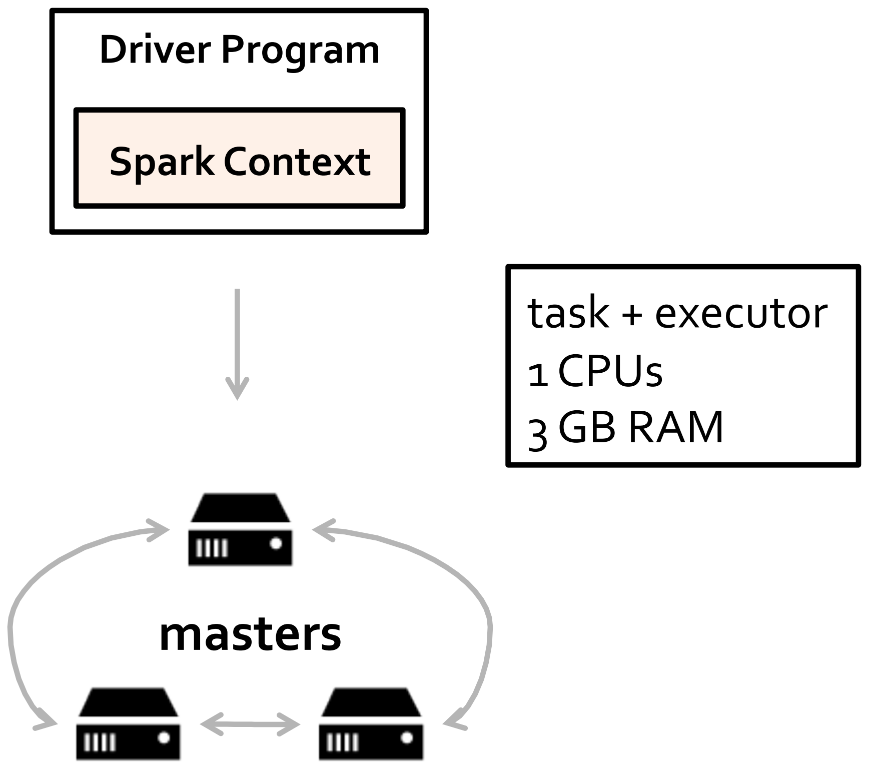
a task with an *executor*



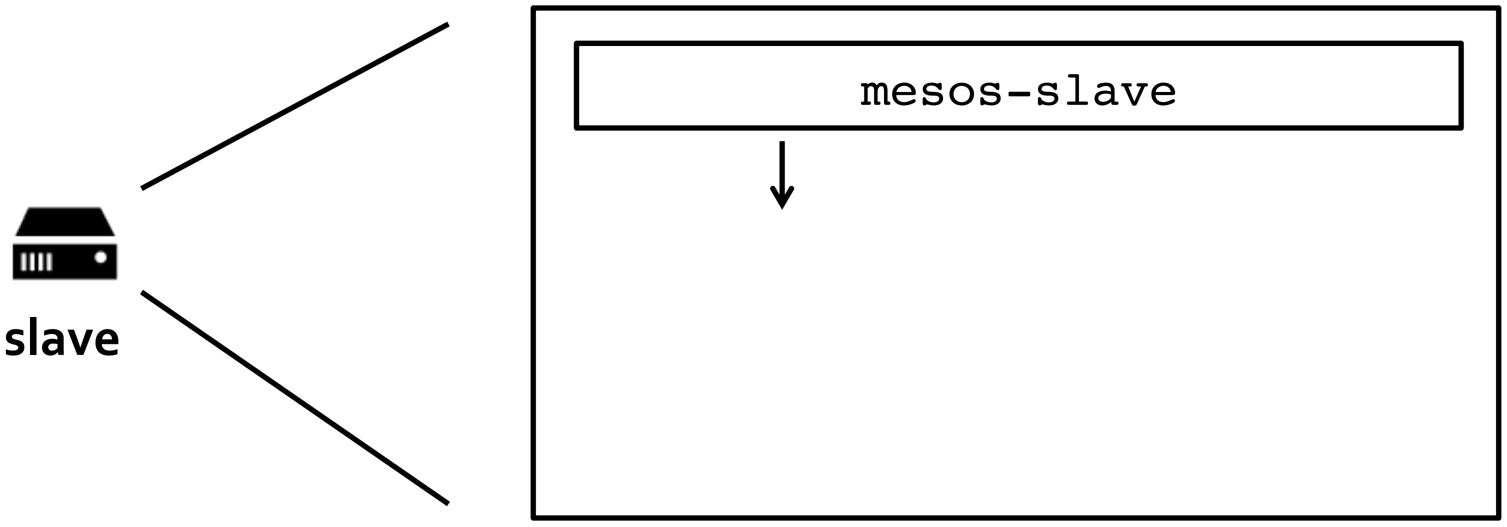
a task with an *executor*



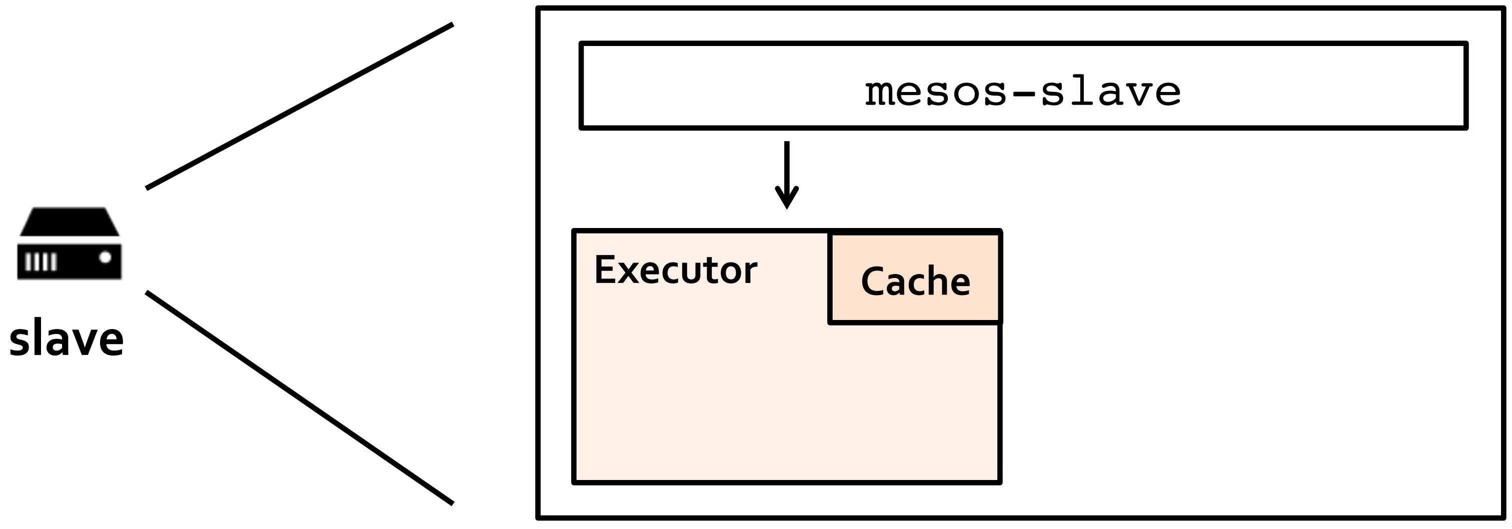
Spark execution



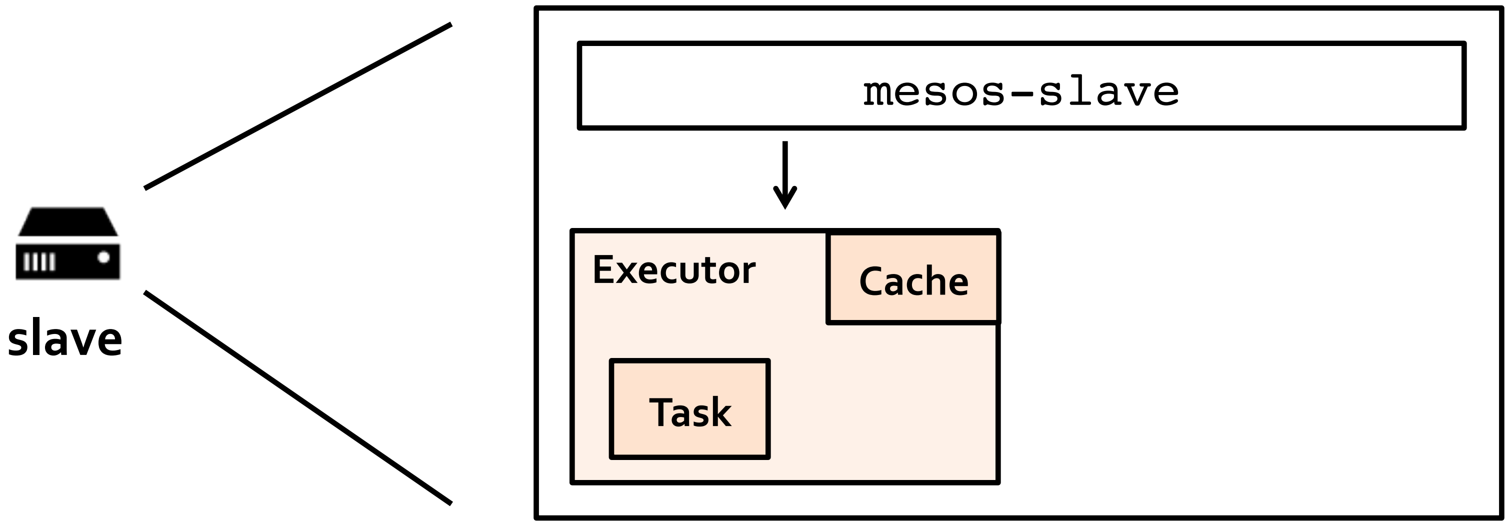
Spark execution



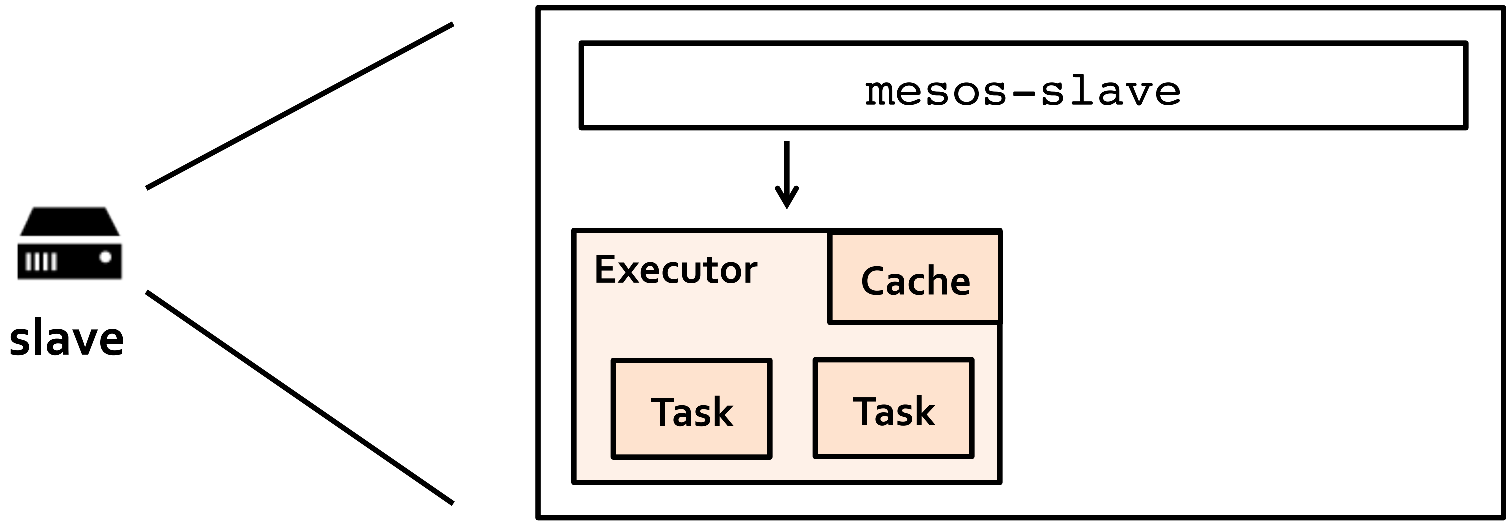
Spark execution



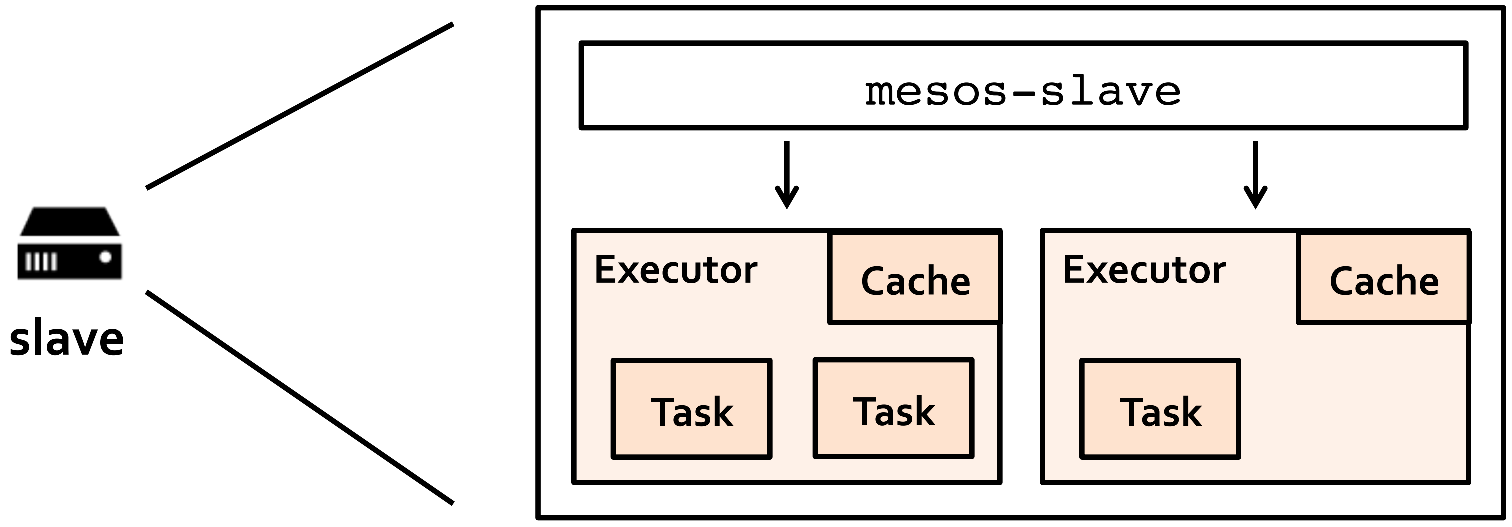
Spark execution



Spark execution



Spark execution



resource isolation

Mesos has containerization support on Linux
(built-in usage of cgroups and namespaces)

isolator modules:

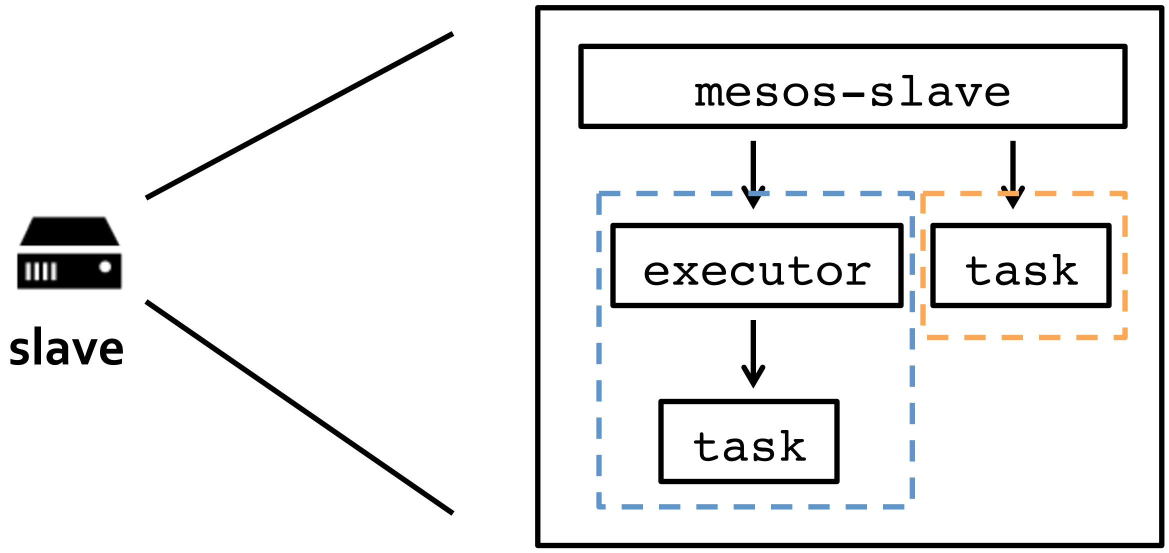
CPU (upper and lower bounds)

memory

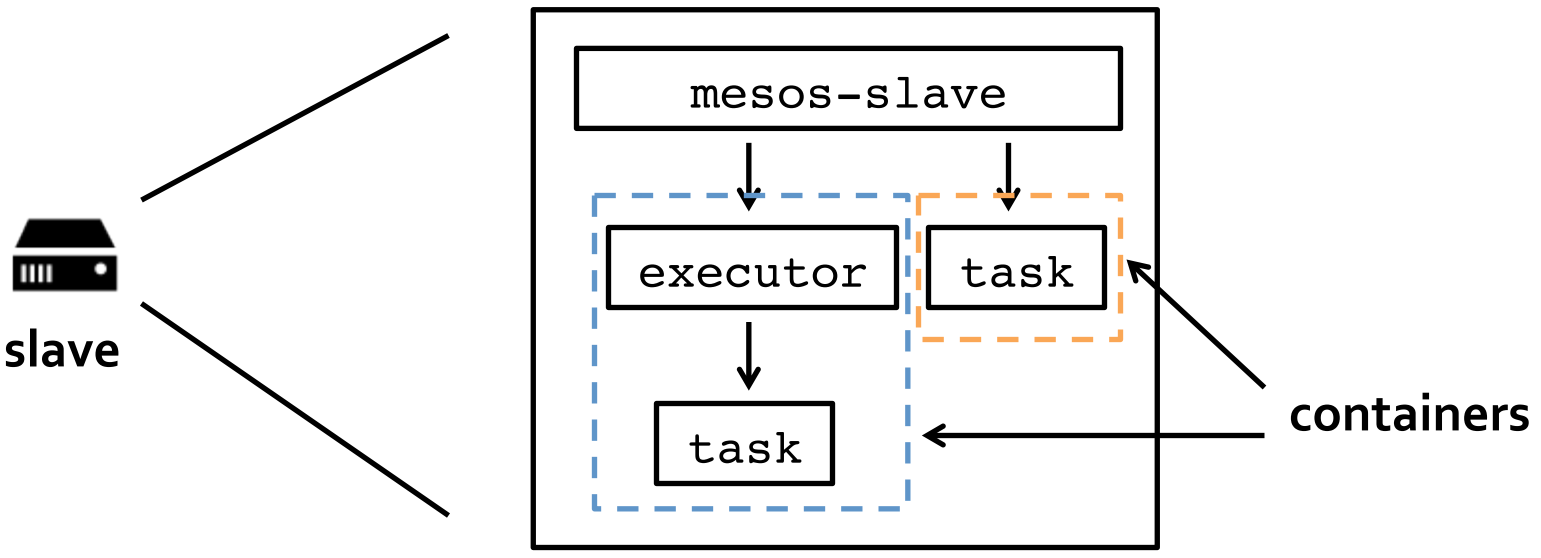
network I/O (in development)

filesystem (using LVM, planned)

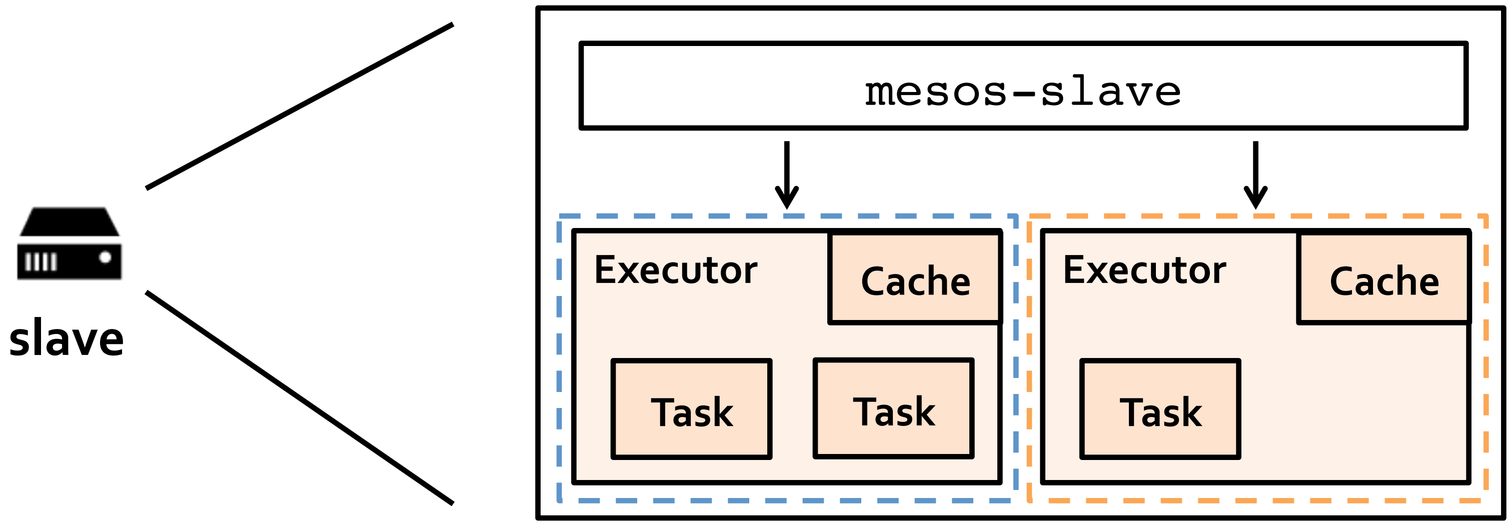
resource isolation



resource isolation



resource isolation



agenda

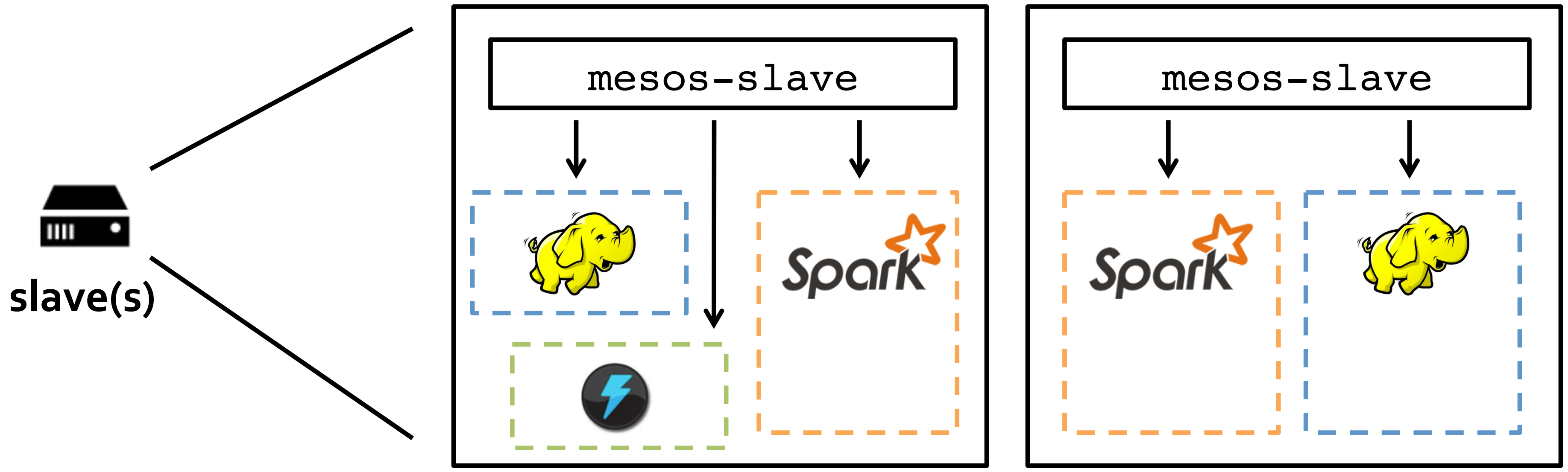
① Mesos

② Spark on Mesos

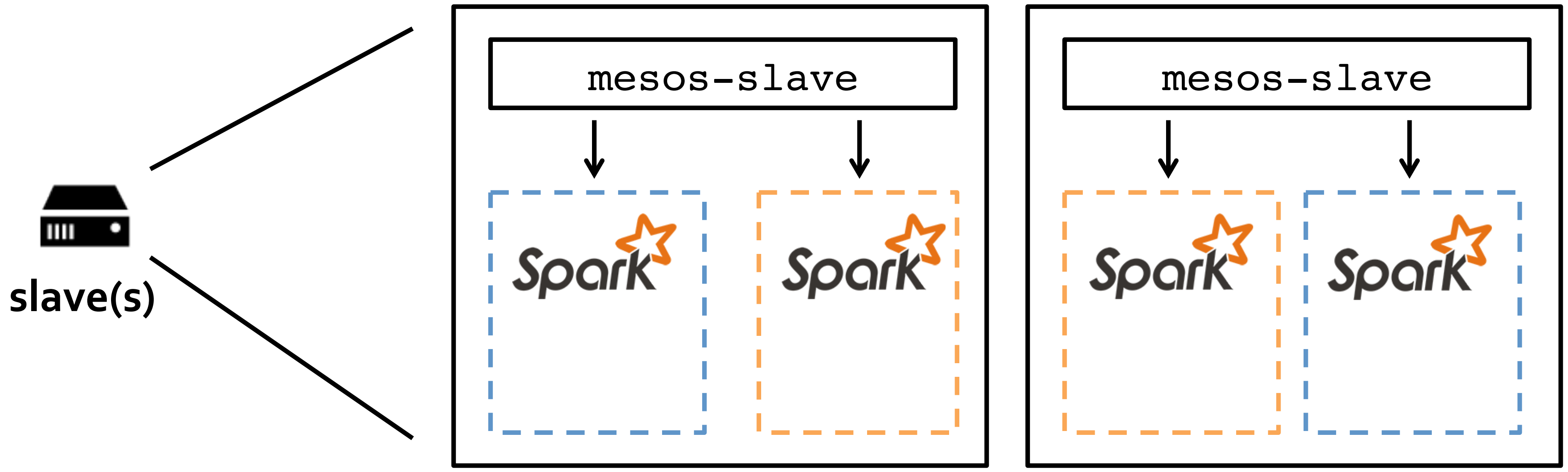
③ why Mesos?
  - ① multi-tenancy ⬅
  - ② fine-grained sharing
  - ③ why not?

④ long-lived services and other frameworks

multi-tenancy

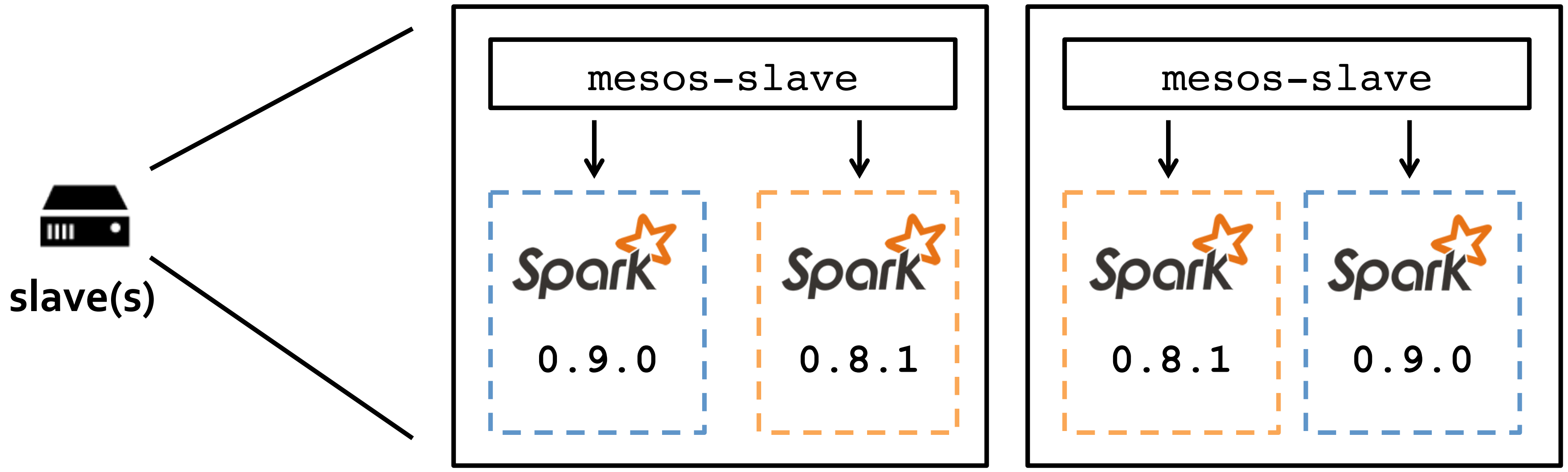


multi-tenancy (only Spark)



(can approximate w/ standalone mode by setting max # cores per application, otherwise get FIFO execution)

multi-tenancy (only Spark)



(run the tried and true and test out the new at the same time!)

agenda

① Mesos

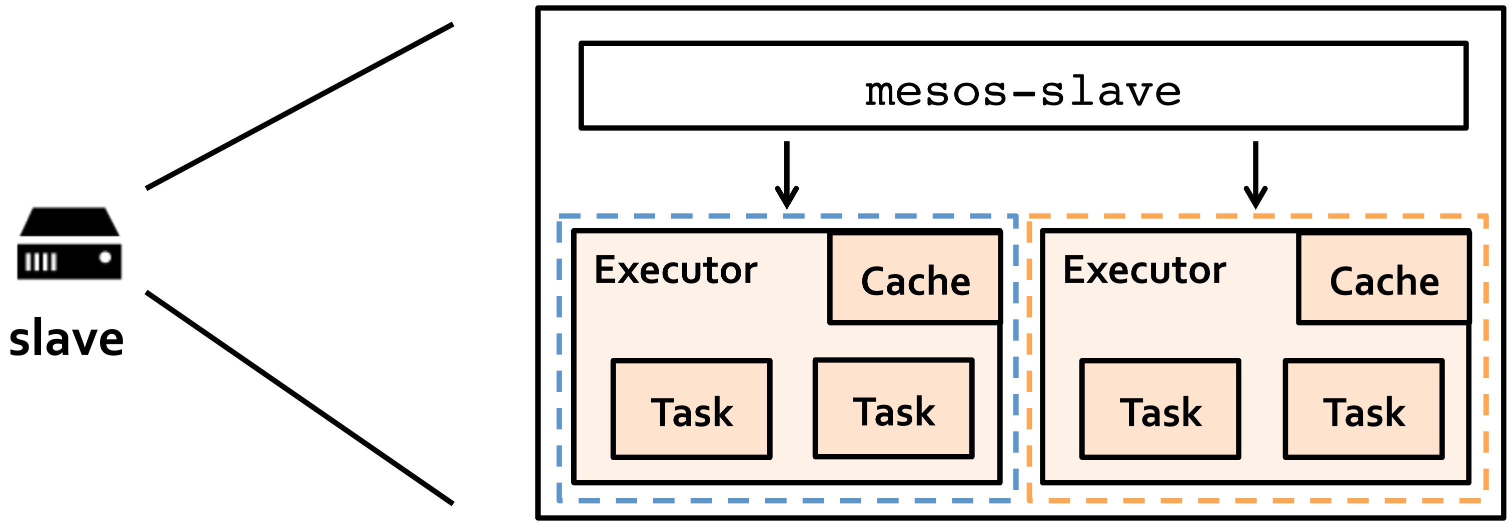
② Spark on Mesos

③ why Mesos?

- ① multi-tenancy
- ② fine-grained sharing ←
- ③ why not?

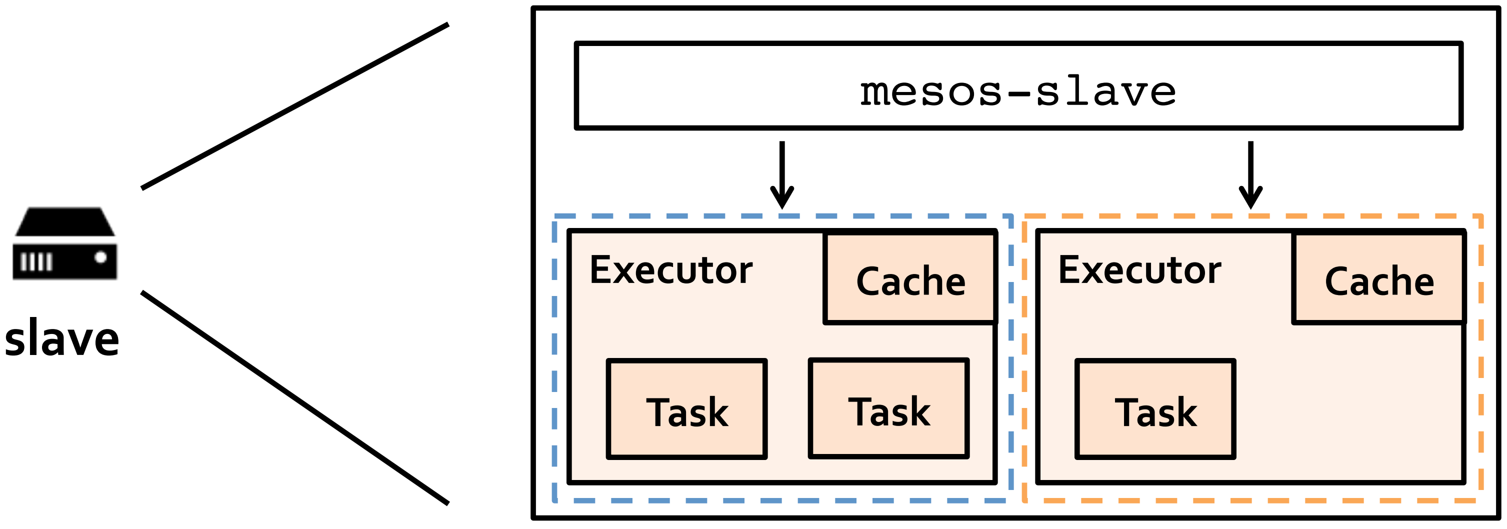
④ long-lived services and other frameworks

fine-grained sharing



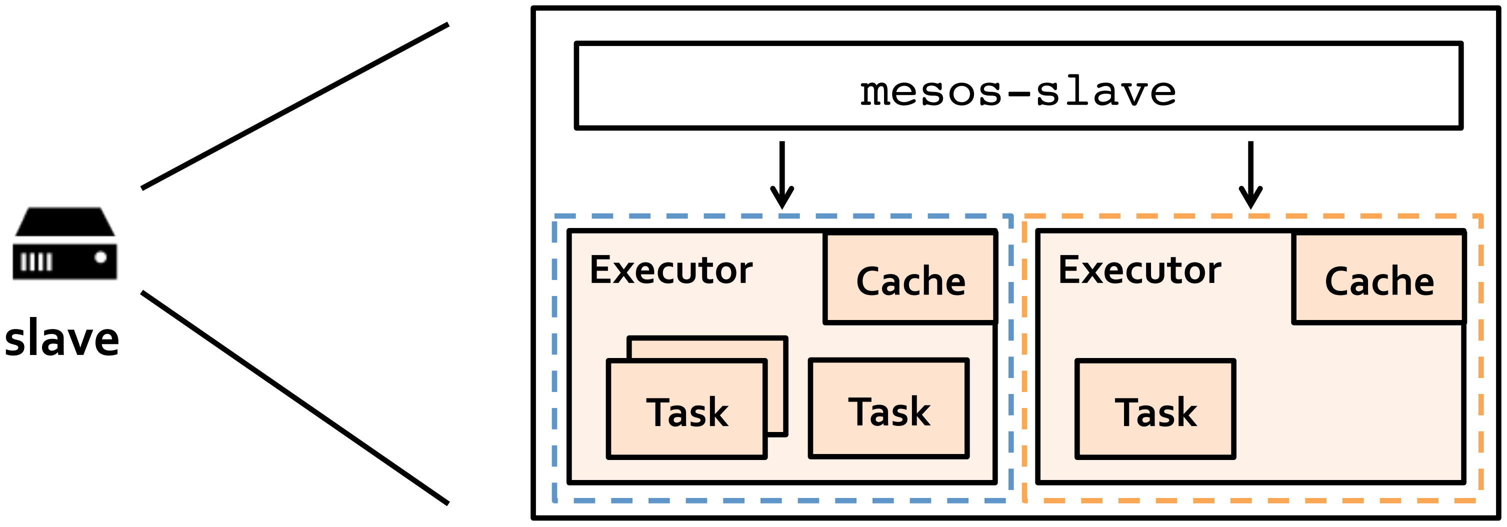
Spark executors only consume memory, can share CPU between

fine-grained sharing



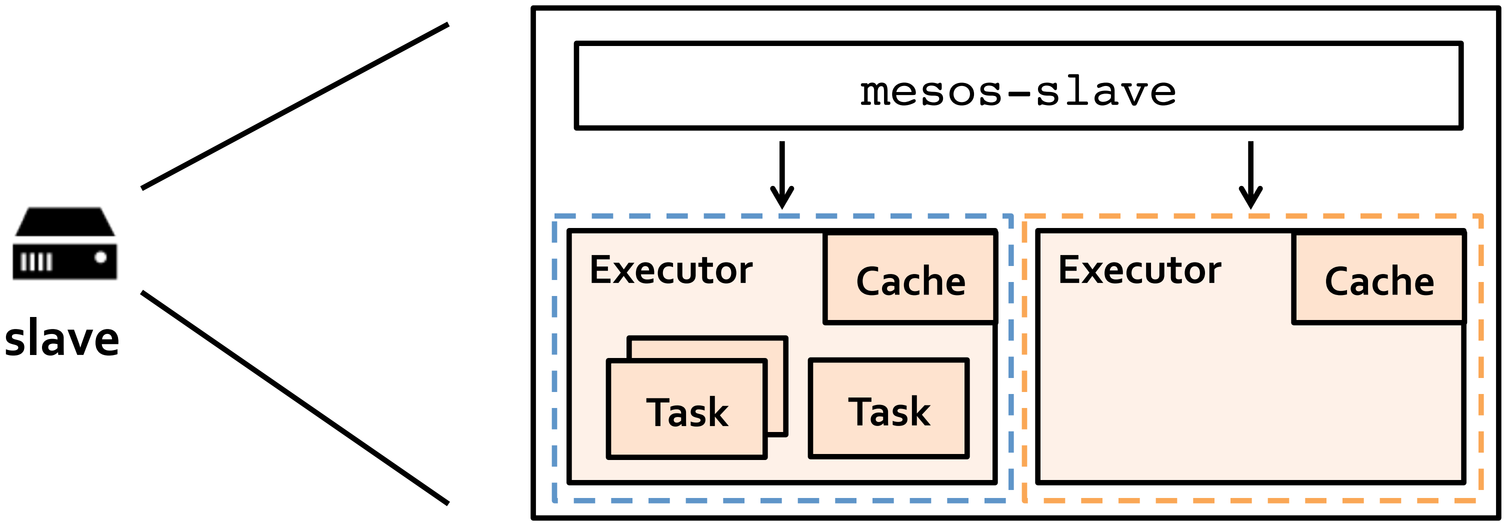
Spark executors only consume memory, can share CPU between

fine-grained sharing



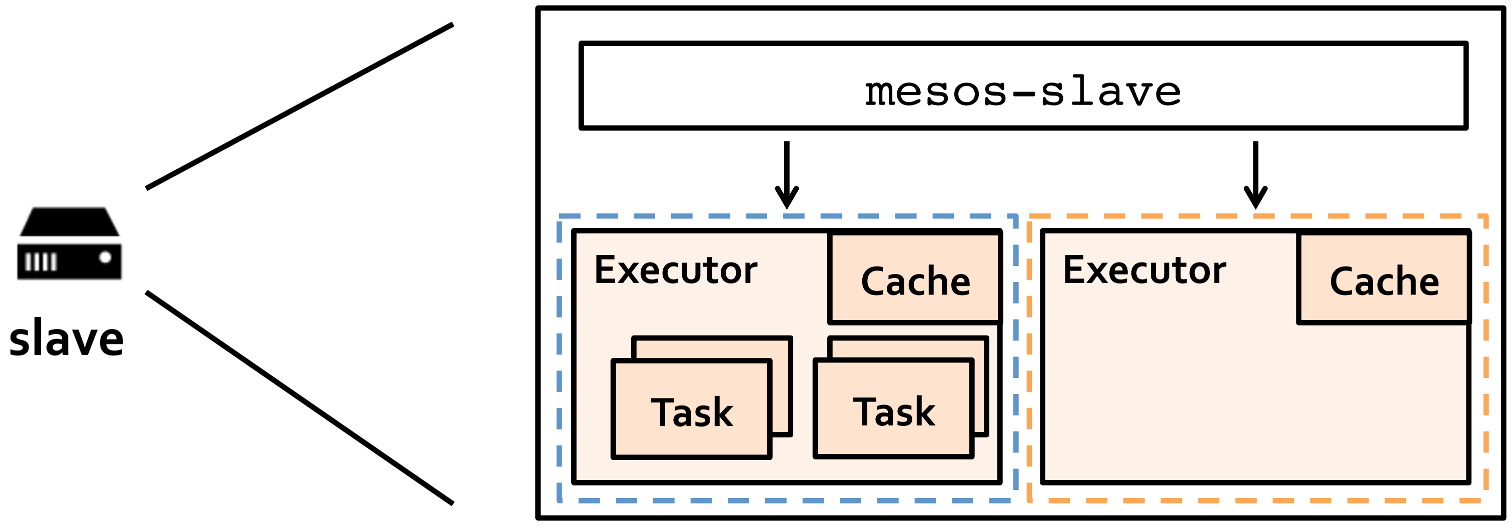
Spark executors only consume memory, can share CPU between

fine-grained sharing



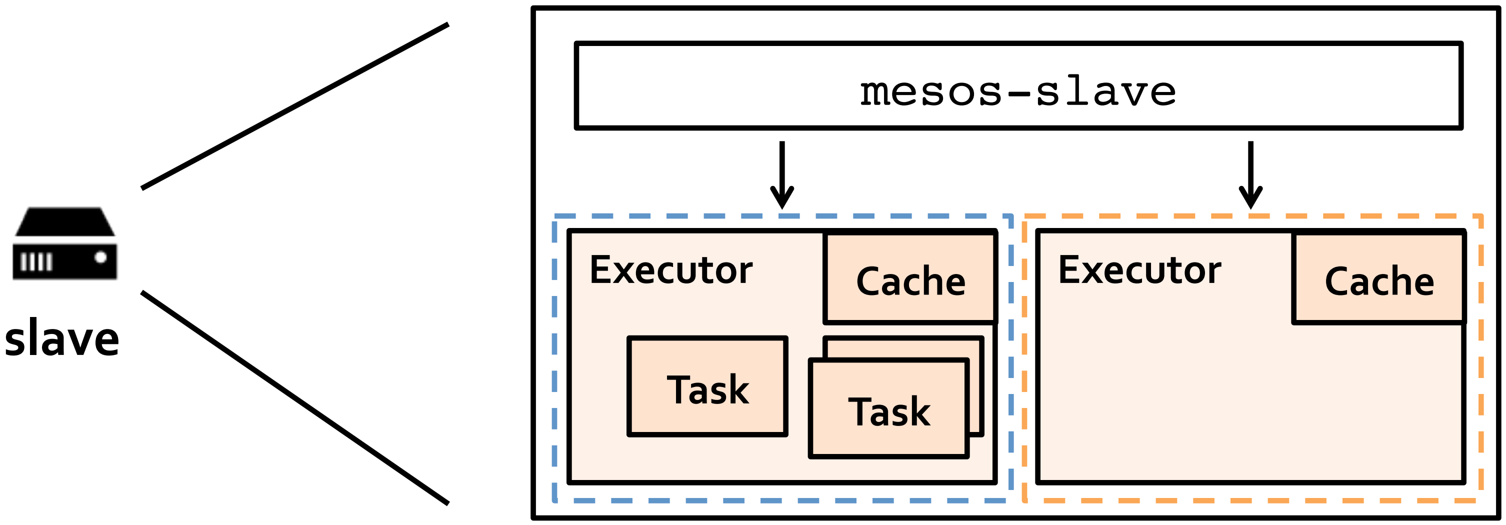
Spark executors only consume memory, can share CPU between

fine-grained sharing



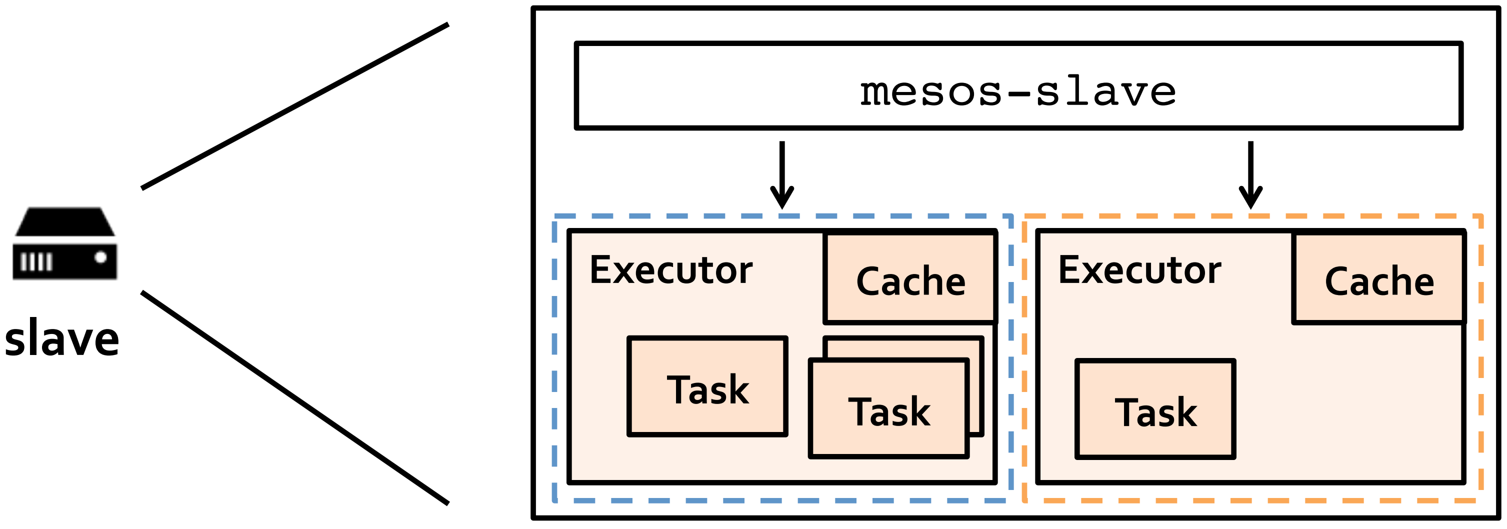
Spark executors only consume memory, can share CPU between

fine-grained sharing



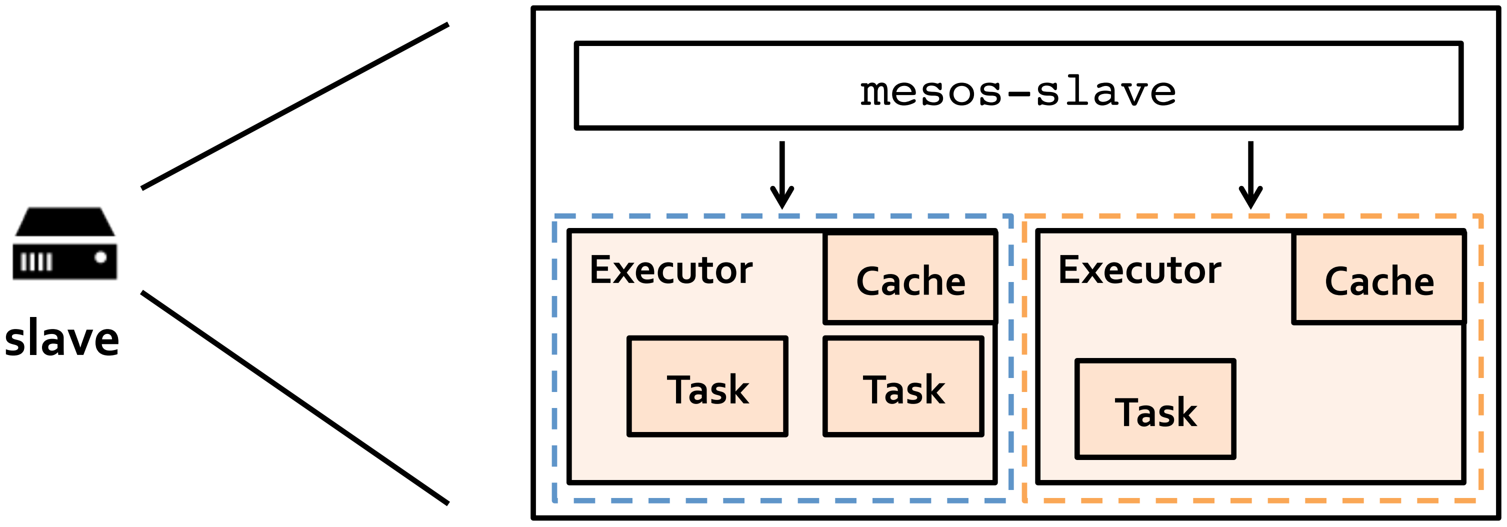
Spark executors only consume memory, can share CPU between

fine-grained sharing



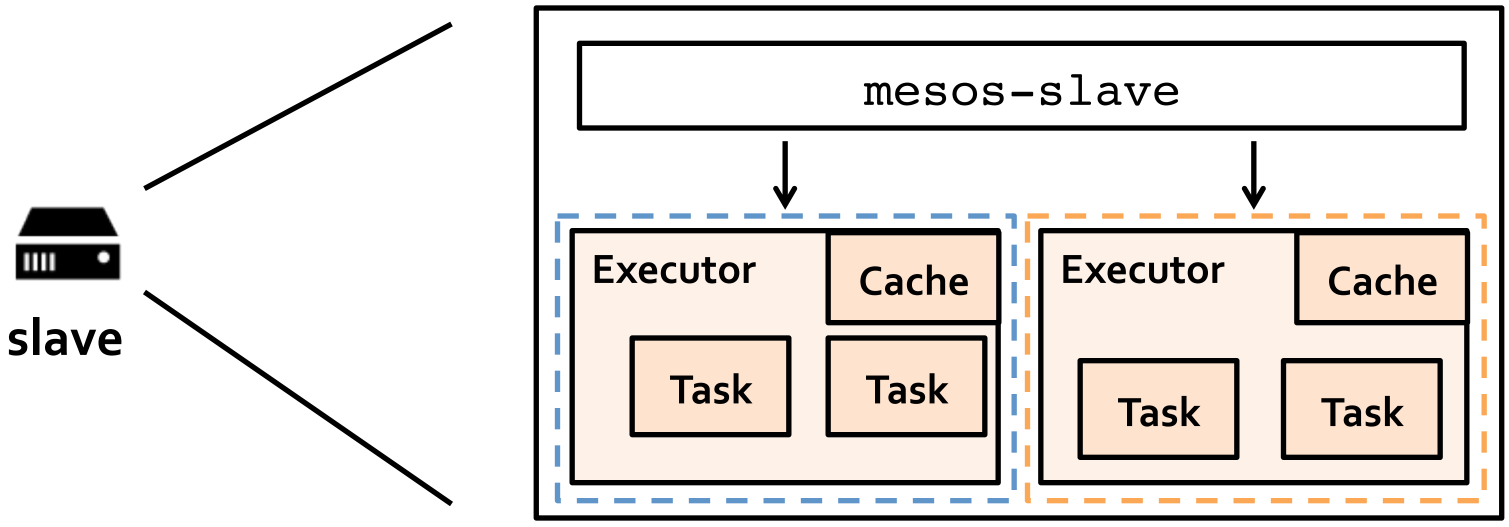
Spark executors only consume memory, can share CPU between

fine-grained sharing



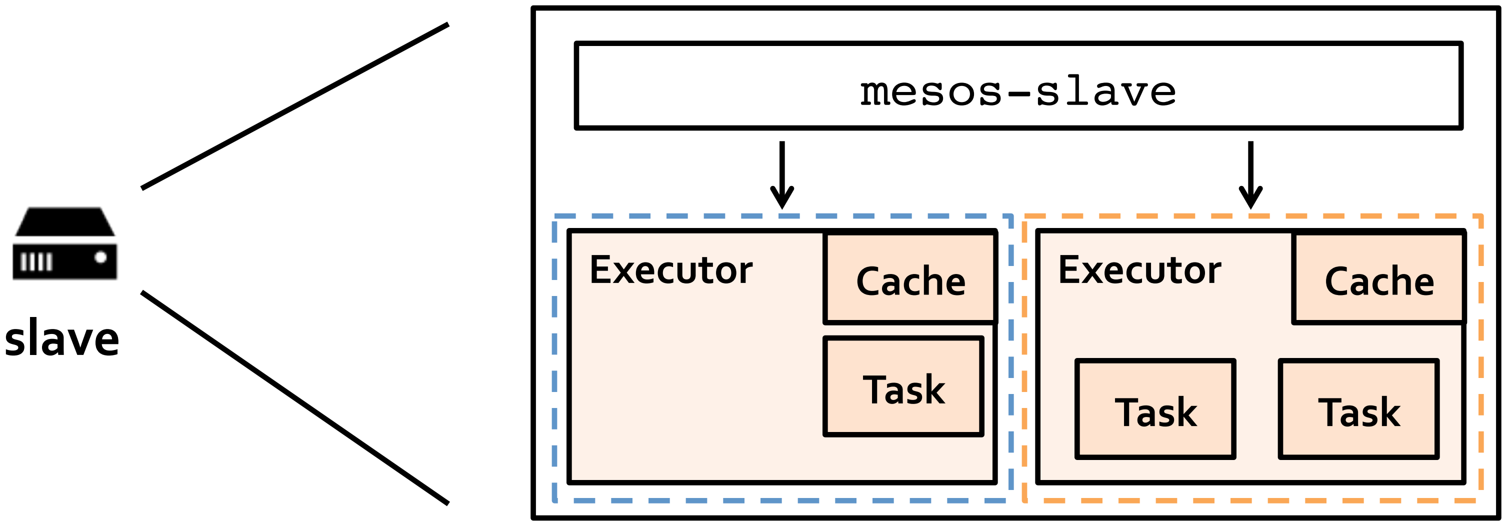
Spark executors only consume memory, can share CPU between

fine-grained sharing



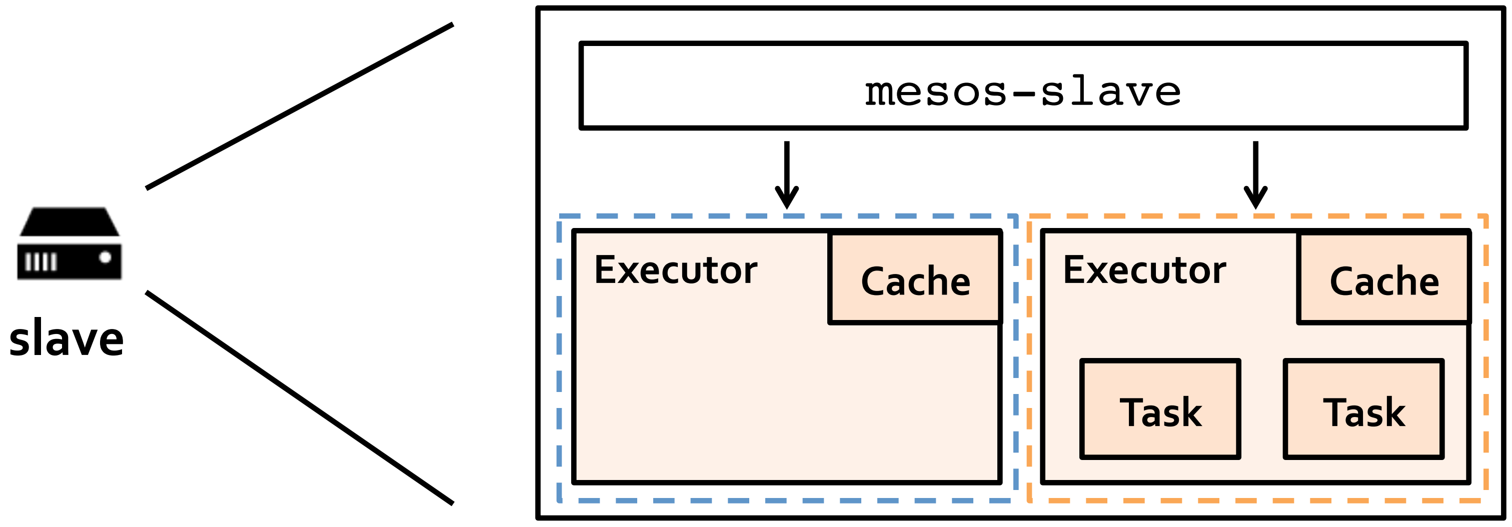
Spark executors only consume memory, can share CPU between

fine-grained sharing



Spark executors only consume memory, can share CPU between

fine-grained sharing



Spark executors only consume memory, can share CPU between

agenda

① Mesos

② Spark on Mesos

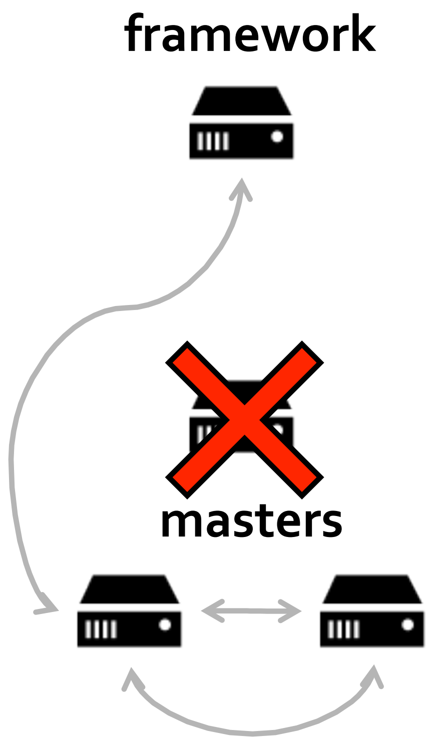
③ why Mesos?
   ① multi-tenancy
   ② fine-grained sharing
   ③ why not? ←

④ long-lived services and other frameworks

why not?

more moving pieces means more things to learn and more things that can fail …

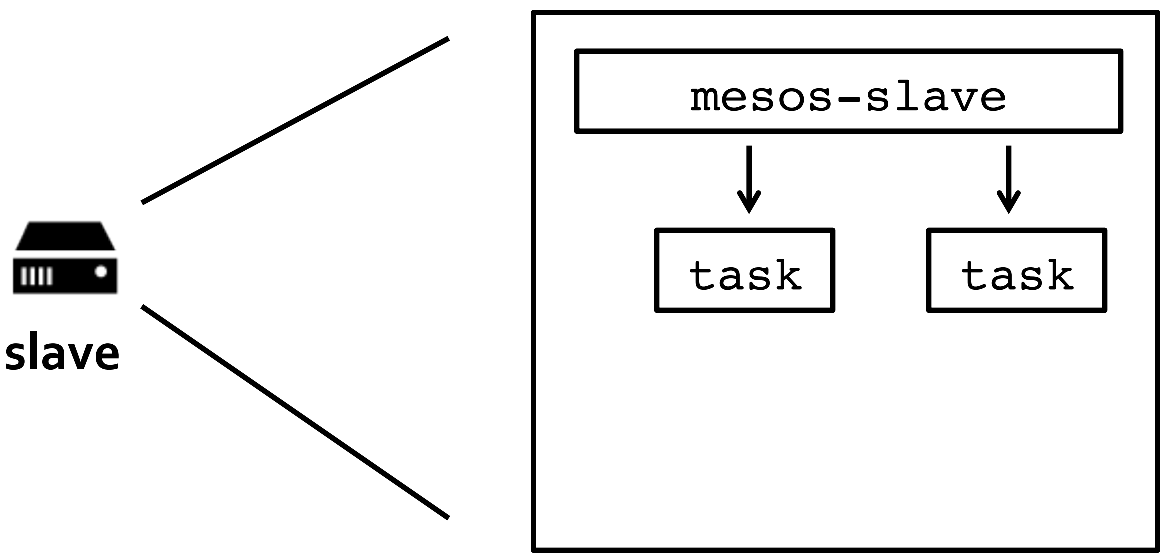
master failover



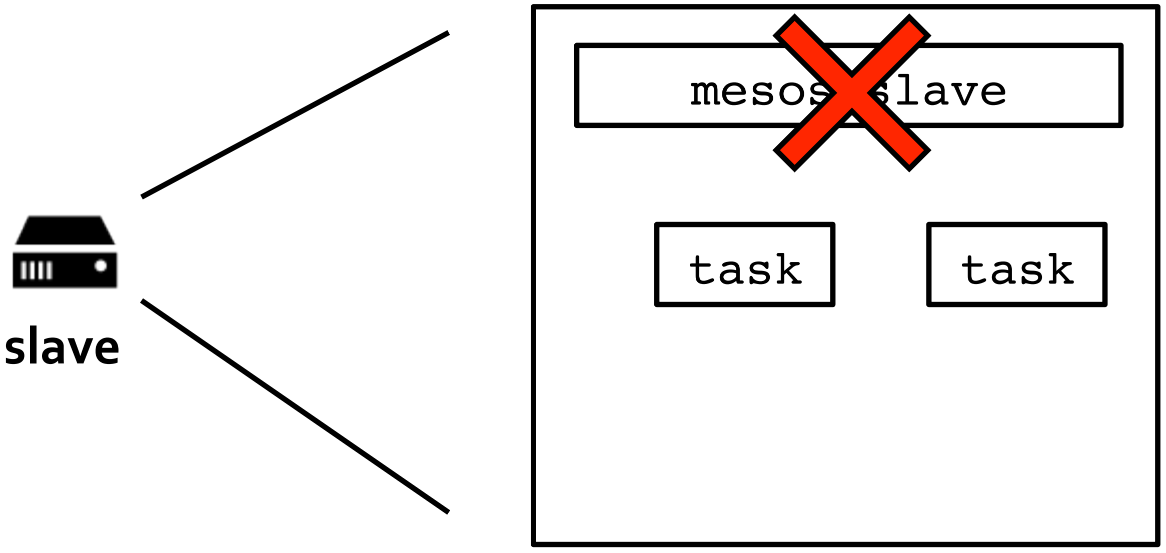
after a new master is elected all frameworks and slaves connect to the new master

all tasks keep running across master failover!

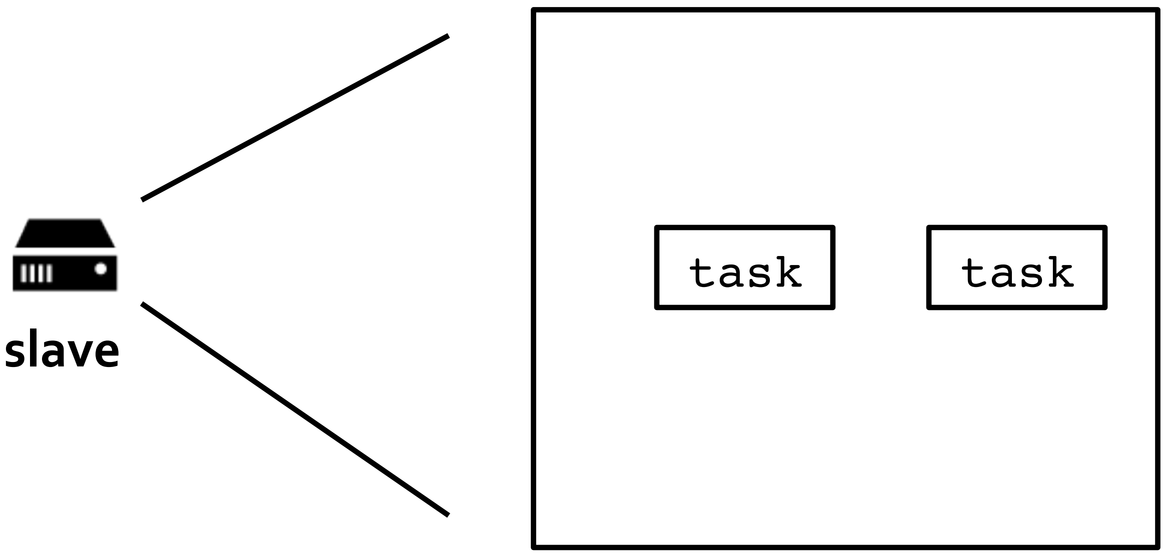
slave failover



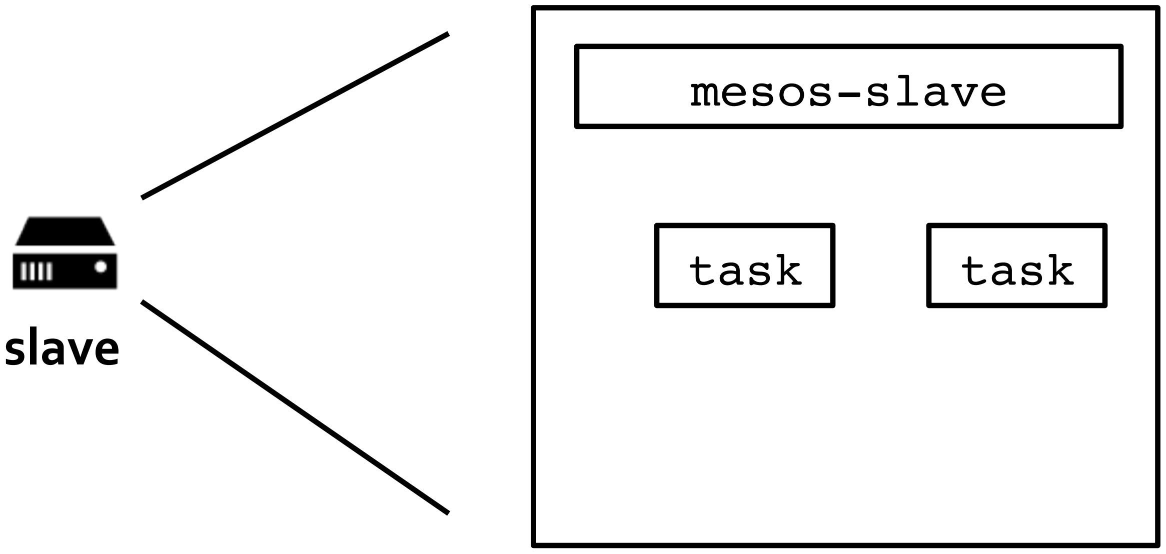
slave failover



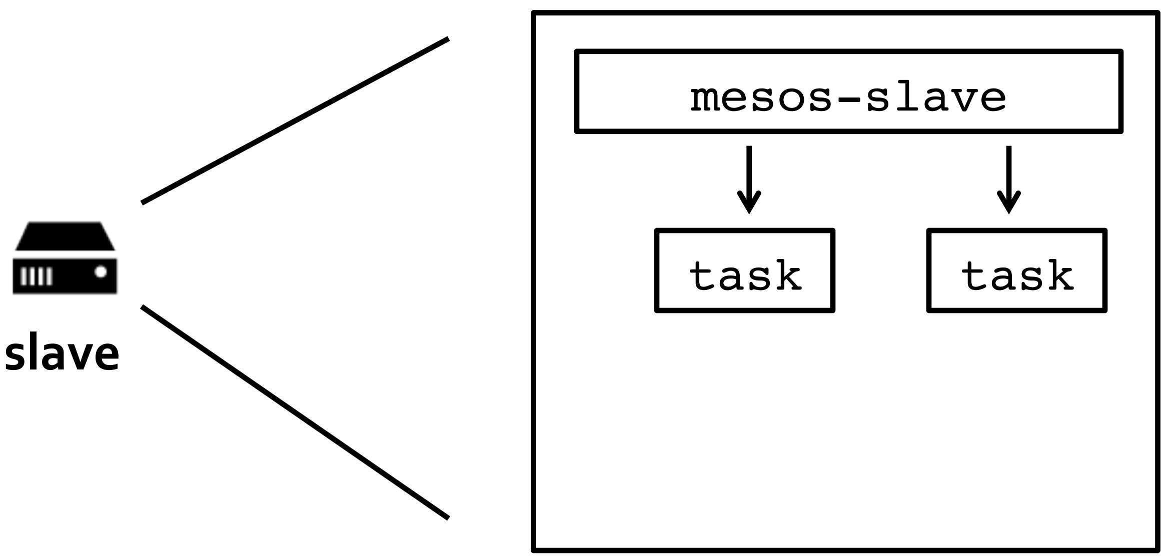
slave failover



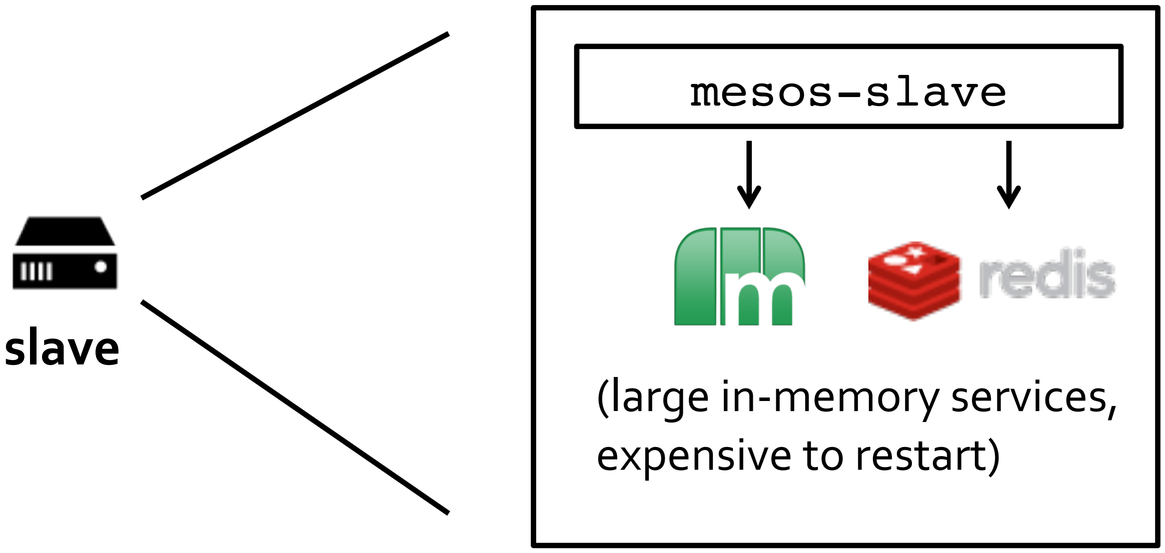
slave failover



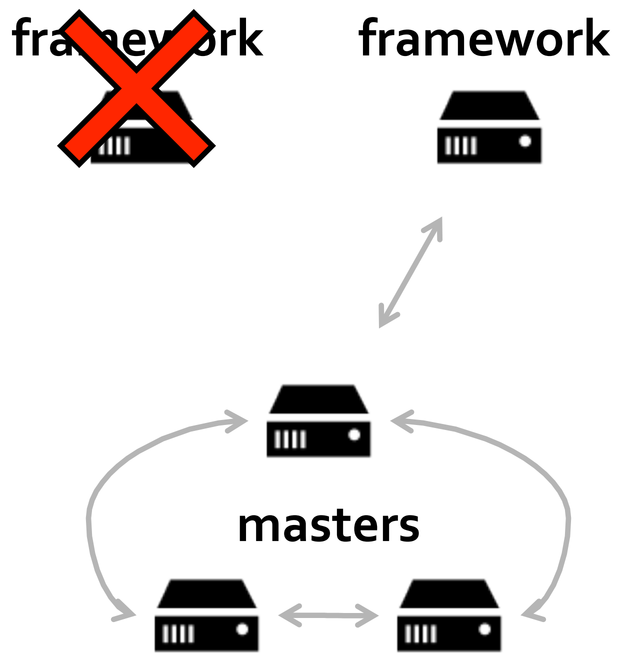
slave failover



slave failover @twitter



framework failover



framework re-registers with master and resumes operation

all tasks keep running across framework failover!

agenda

① Mesos

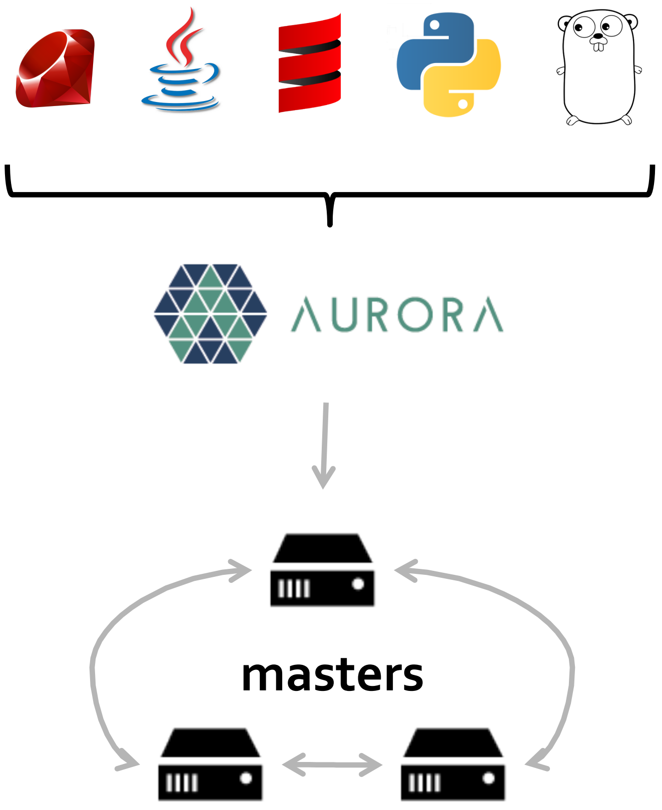
② Spark on Mesos

③ why Mesos?

  ① multi-tenancy
  ② fine-grained sharing
  ③ why not?

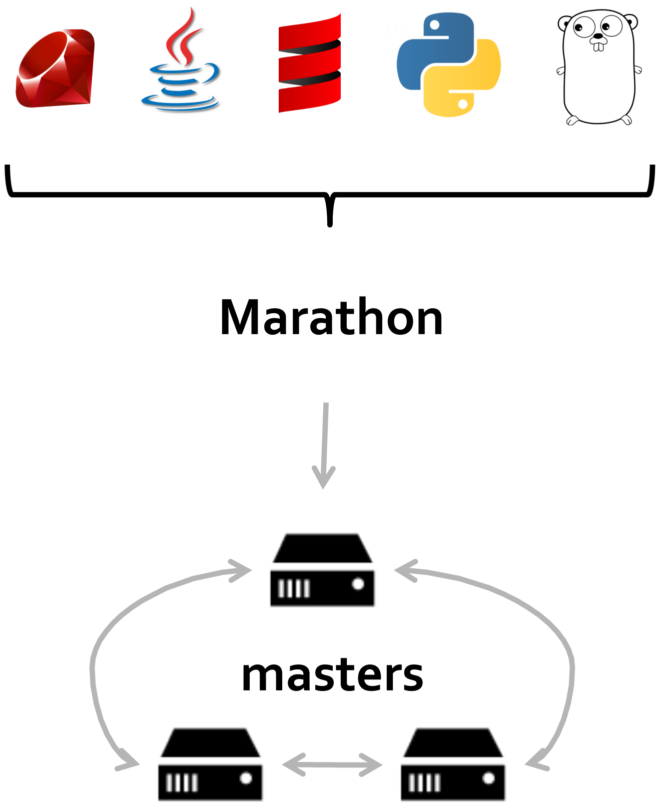
④ long-lived services and other frameworks ←

Apache Aurora (incubating)



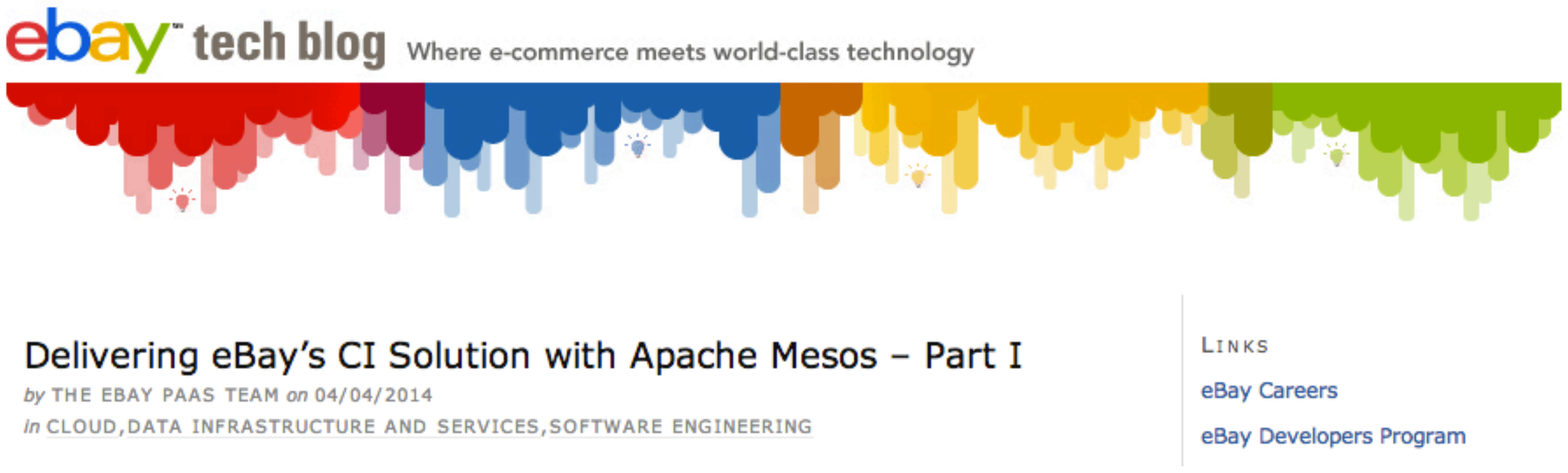
Aurora is a Mesos framework that makes it easy to launch services written in Ruby, Java, Scala, Python, Go, etc!

Marathon (from Mesosphere)



Marathon is a Mesos framework that makes it easy to launch services written in Ruby, Java, Scala, Python, Go, etc!

Jenkins on Mesos



ebay tech blog Where e-commerce meets world-class technology

Delivering eBay's CI Solution with Apache Mesos – Part I

by THE EBAY PAAS TEAM *on* 04/04/2014

in CLOUD, DATA INFRASTRUCTURE AND SERVICES, SOFTWARE ENGINEERING

LINKS

eBay Careers

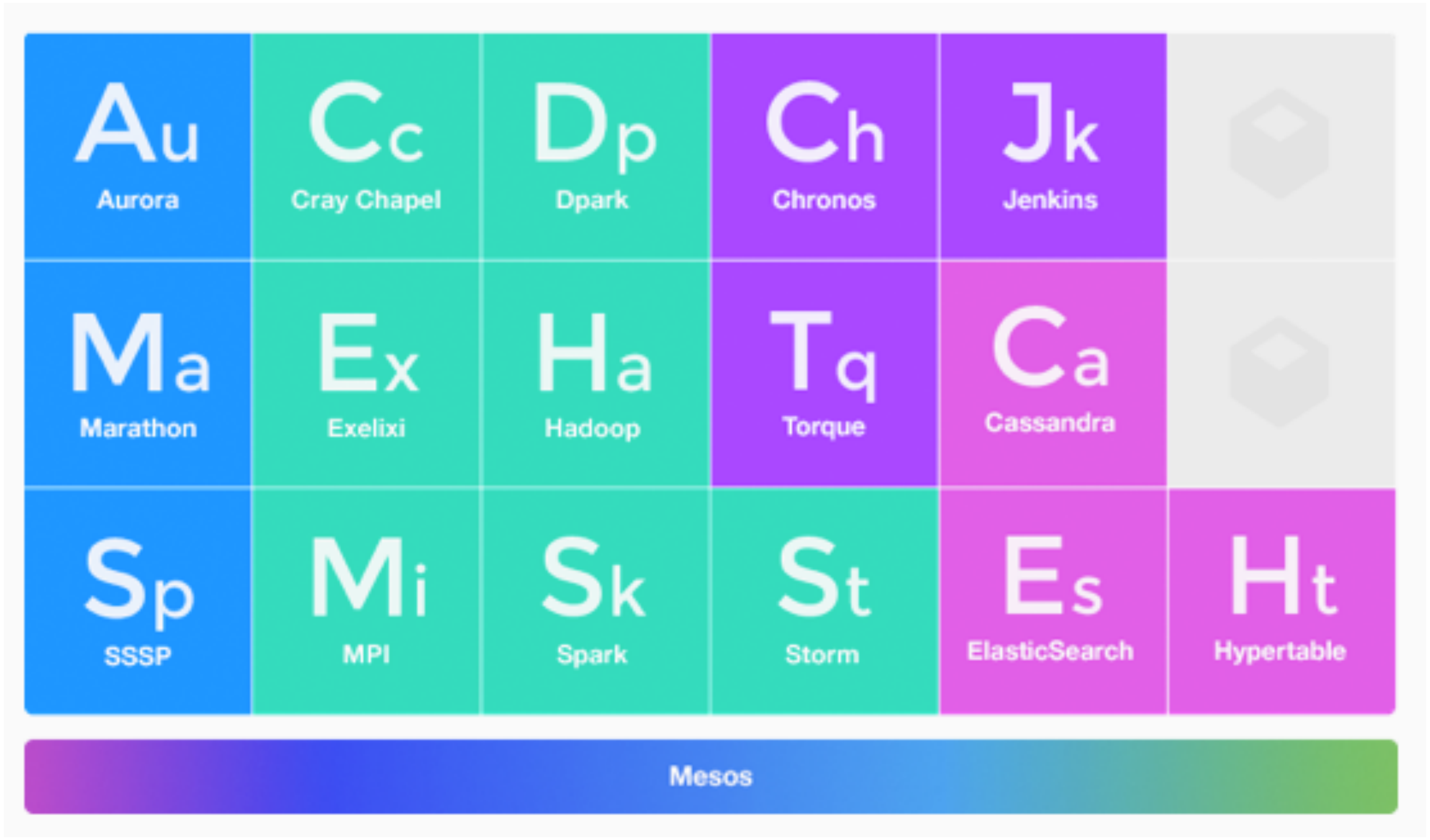
eBay Developers Program

(http://bit.ly/1frLrLf)

CloudBees

Apache Mesos and Jenkins - elastic build slaves

(http://bit.ly/1nHwM3r)

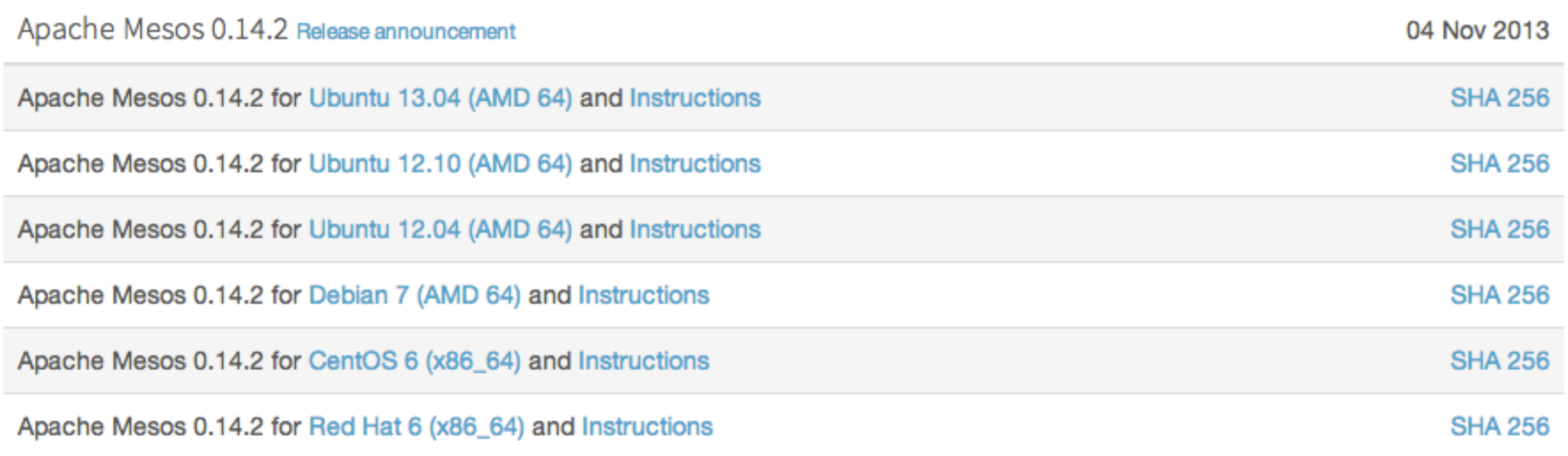

Au
Aurora
Cc
Cray Chapel
Dp
Dpark
Ch
Chronos
Jk
Jenkins
Ma
Marathon
Ex
Exelixi
Ha
Hadoop
Tq
Torque
Ca
Cassandra
Sp
SSSP
Mi
MPI
Sk
Spark
St
Storm
Es
ElasticSearch
Ht
Hypertable
Mesos

mesosphere

Elastic Mesos: elastic.mesosphere.io

Launch an Apache Mesos Cluster in ③ ② ①

Packages:



Apache Mesos 0.14.2 Release announcement	04 Nov 2013
Apache Mesos 0.14.2 for Ubuntu 13.04 (AMD 64) and Instructions	SHA 256
Apache Mesos 0.14.2 for Ubuntu 12.10 (AMD 64) and Instructions	SHA 256
Apache Mesos 0.14.2 for Ubuntu 12.04 (AMD 64) and Instructions	SHA 256
Apache Mesos 0.14.2 for Debian 7 (AMD 64) and Instructions	SHA 256
Apache Mesos 0.14.2 for CentOS 6 (x86_64) and Instructions	SHA 256
Apache Mesos 0.14.2 for Red Hat 6 (x86_64) and Instructions	SHA 256

Thank You!

mesos.apache.org

mesos.apache.org/blog

@ApacheMesos