

Strata + Hadoop World London 2015
@deanwampler

Spark on Mesos

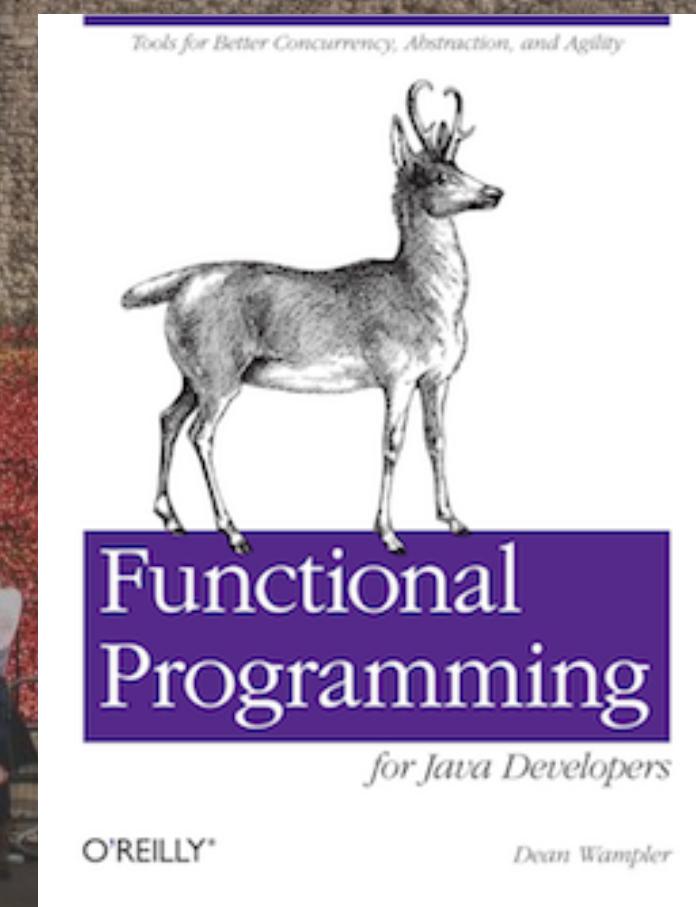
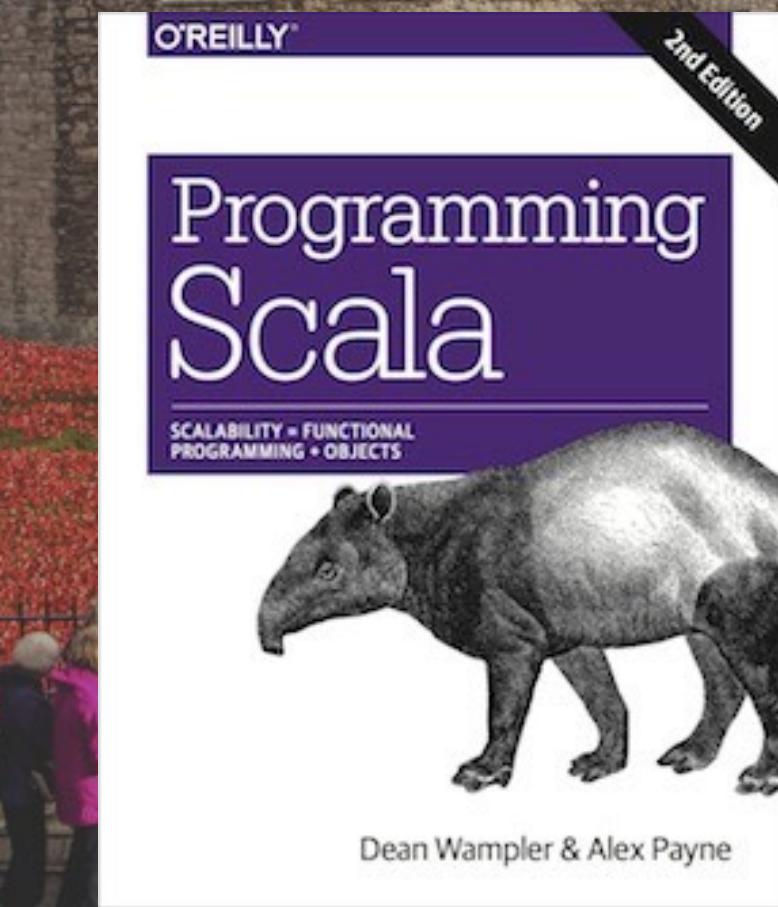


Saturday, May 9, 15

Photos, Copyright (c) Dean Wampler, 2014-2015, All Rights Reserved, unless otherwise noted. All photos were taken on Nov. 7, 2014 of "The Blood Swept Lands and Seas of Red" installation at the Tower of London, commemorating the 100th anniversary of the start of The Great War, a display of 888,246 ceramic poppies, one for each British military fatality in the war. (<http://poppies.hrp.org.uk/>)
Other content Copyright (c) 2015, Typesafe, but is free to use with attribution requested.
<http://creativecommons.org/licenses/by-nc-sa/2.0/legalcode>

Dean Wampler

dean.wampler@typesafe.com
polyglotprogramming.com/talks
@deanwampler



Saturday, May 9, 15

About me. You can find this presentation and others on Big Data and Scala at polyglotprogramming.com.

A wide-angle photograph of the Tower of London. In the foreground, a vast field of red poppies stretches across the ground. A crowd of people stands behind a metal fence, looking out at the poppies. The Tower of London's stone walls and towers rise in the background under a blue sky with white clouds.

Office Hour:
1525-1625 today

Saturday, May 9, 15

Check the schedule for the location.



Why Mesos?

Saturday, May 9, 15



5

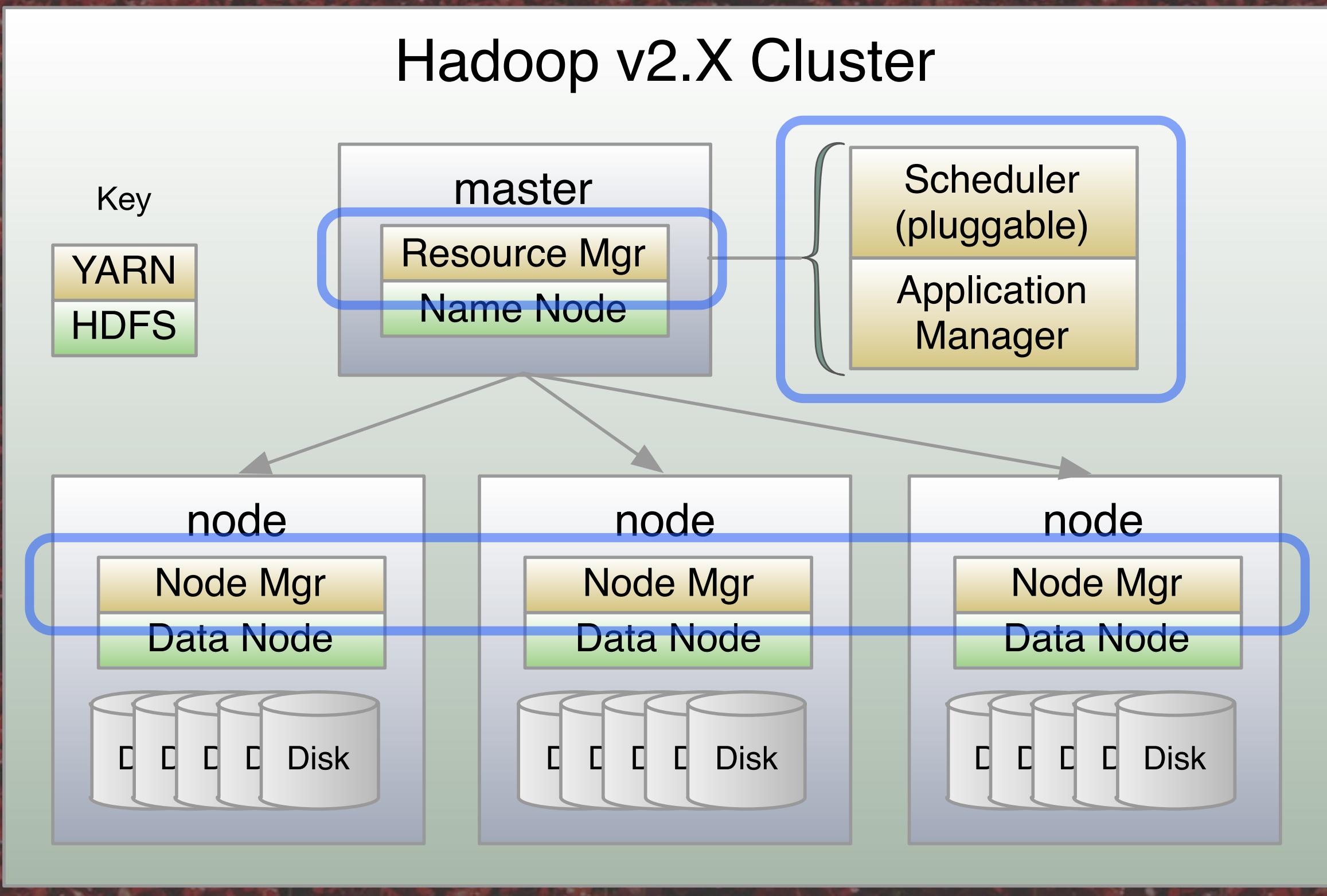
Saturday, May 9, 15

Something Familiar

YARN

(Yet Another Resource Negotiator)

Resource and Node Managers



7

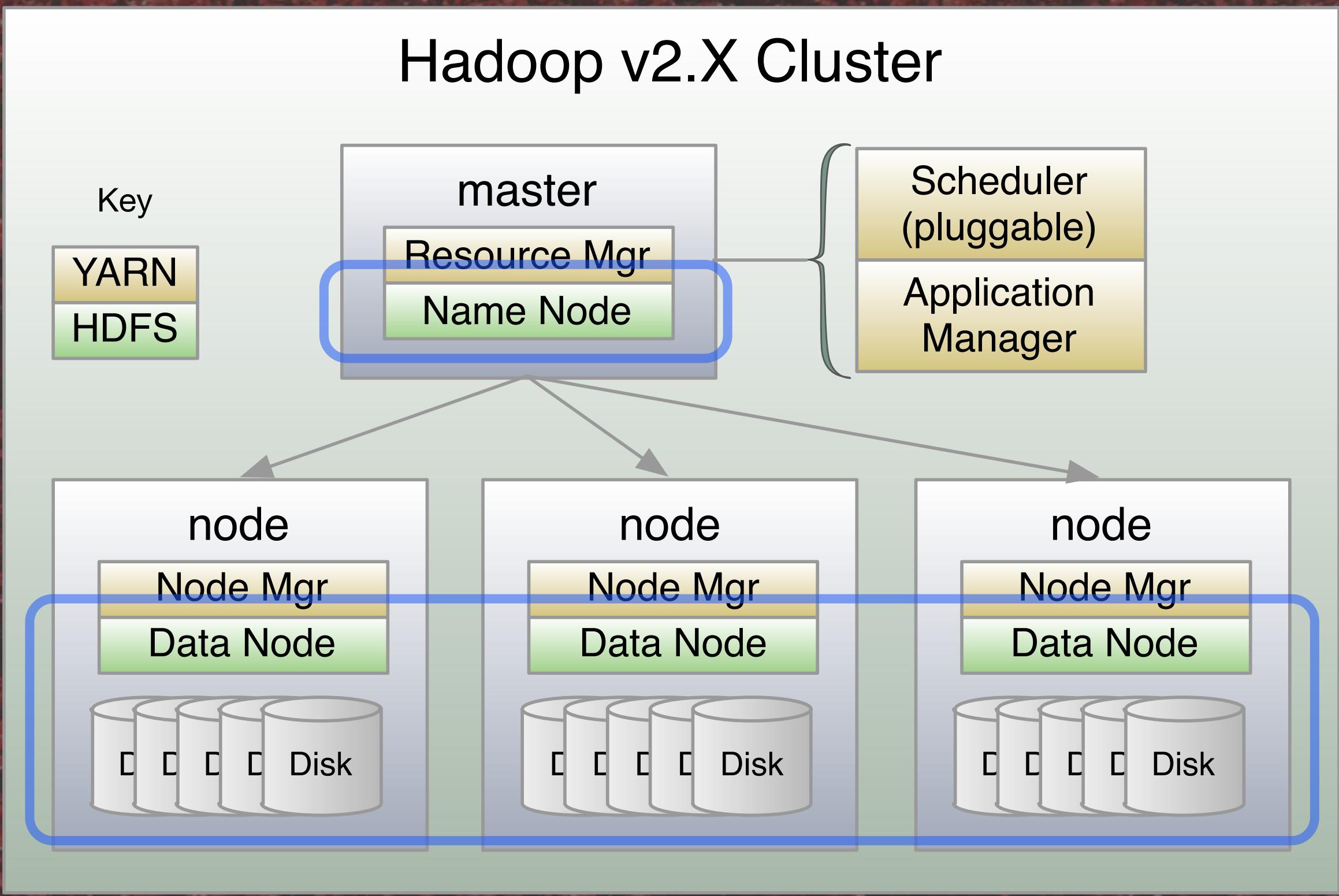
Saturday, May 9, 15

Hadoop 2 uses YARN to manage resources via the master Resource Manager, which includes a pluggable job Scheduler and an Application Manager (where user jobs reside). It coordinates with the Node Manager on each node to schedule jobs and provide resources.

See "Apache Hadoop YARN, Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2" (Addison-Wesley) and <http://hadoop.apache.org/docs/r2.3.0/hadoop-yarn/hadoop-yarn-site/YARN.html> for the gory details.

Master failover to standbys can be managed by Zookeeper (not shown).

HDFS: Name Node and Data Nodes



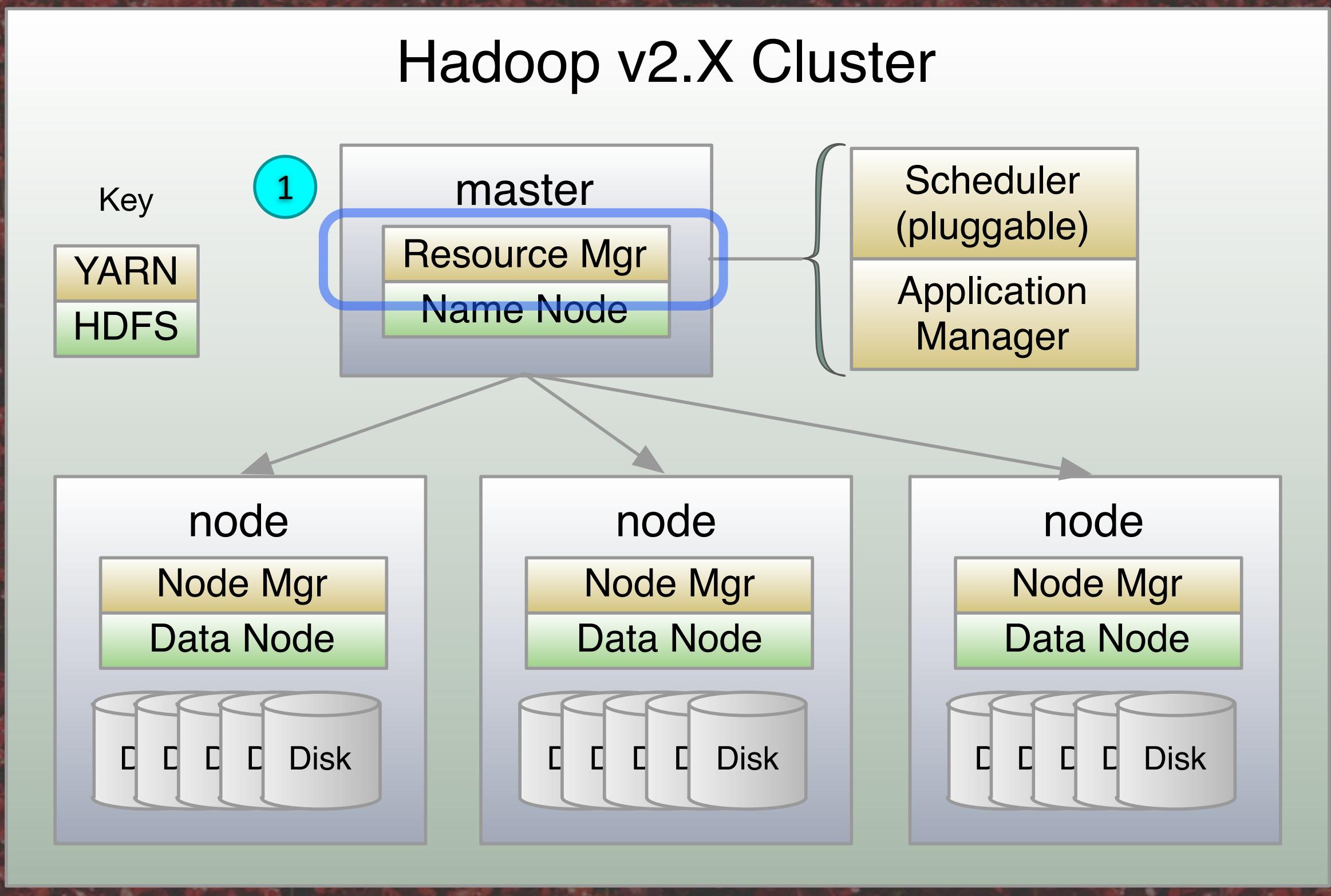
Job Submission

9

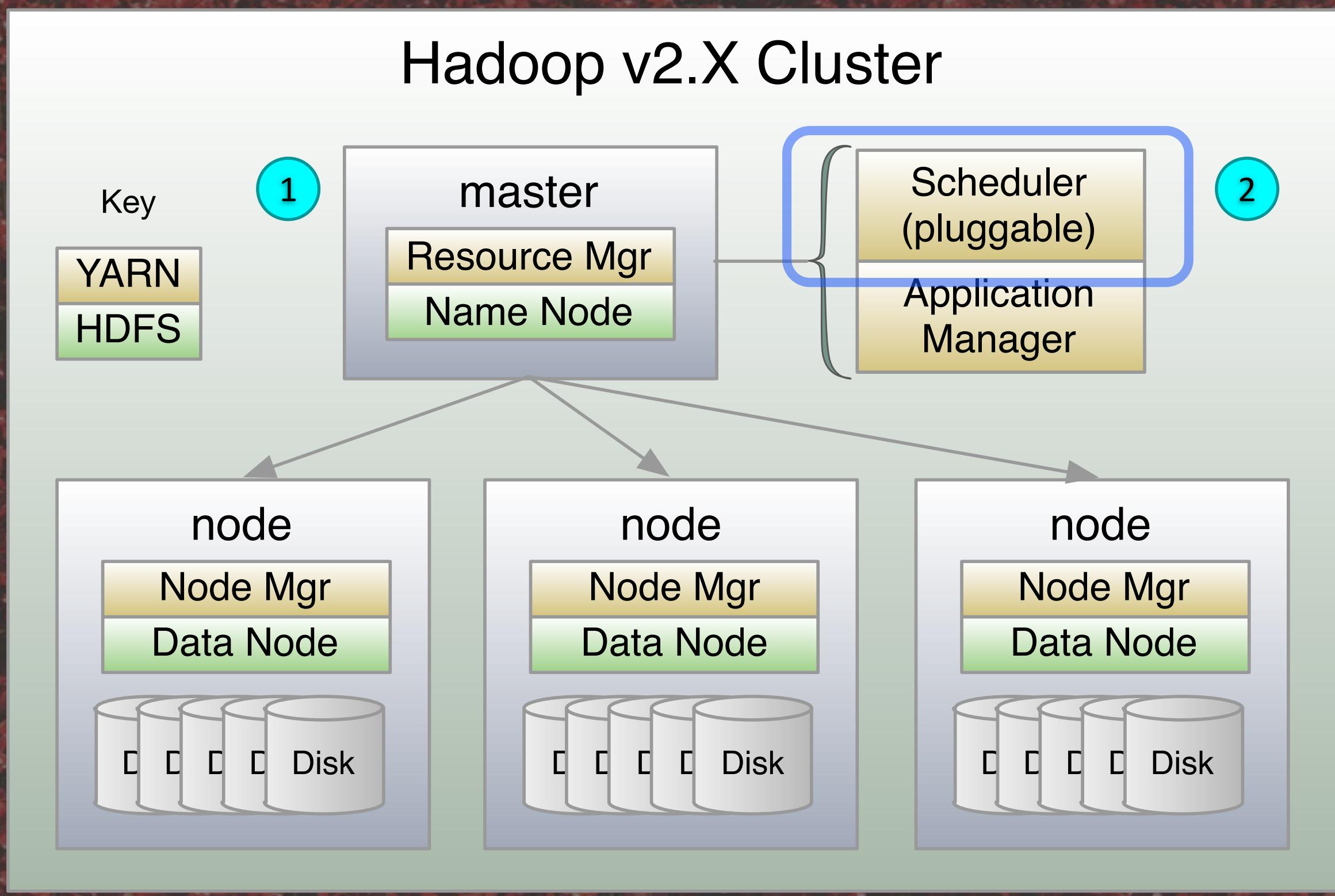
Saturday, May 9, 15

At a high-level, let's walk through the job submission process managed by YARN.

Resource Manager



Scheduler

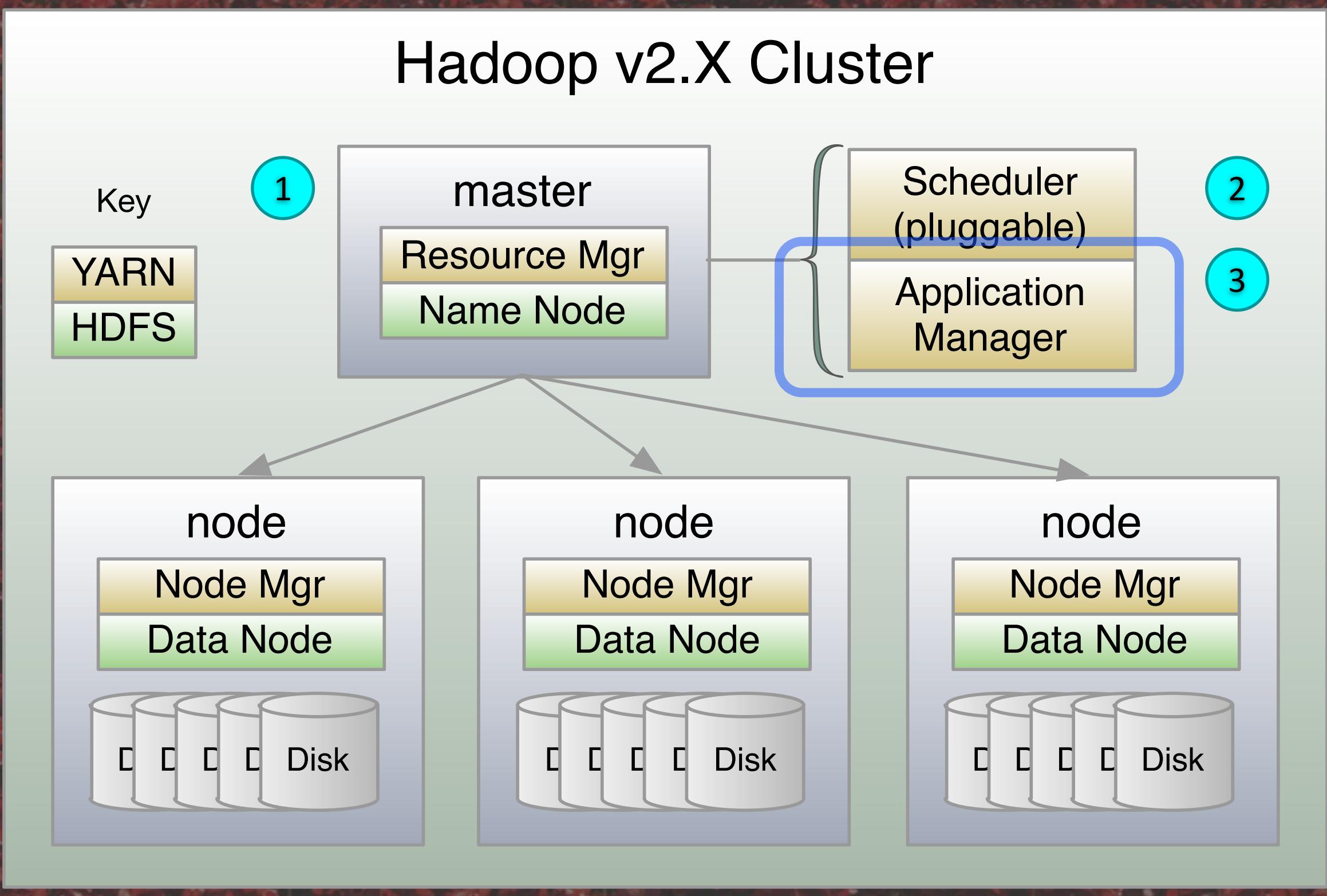


11

Saturday, May 9, 15

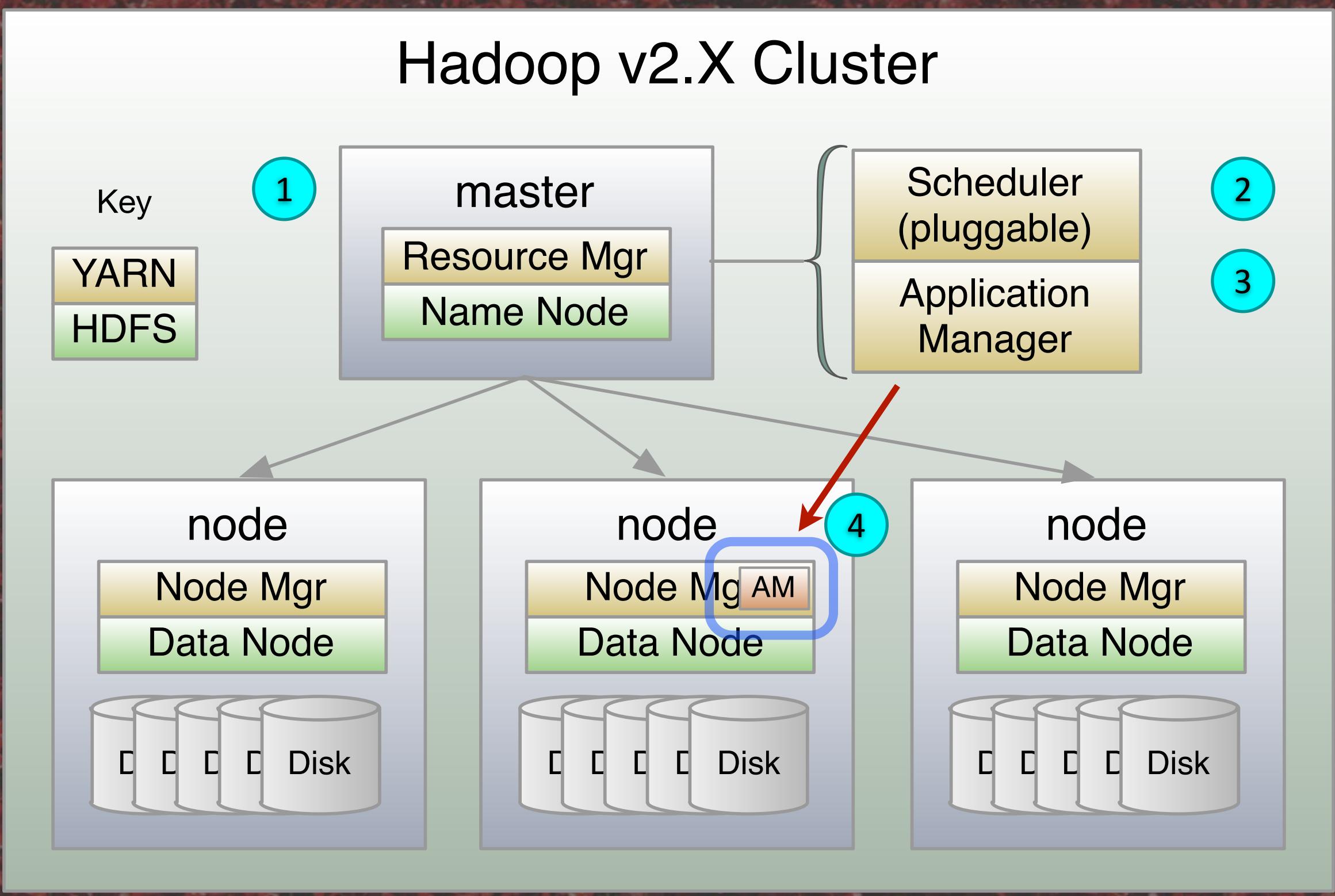
The Scheduler moves the app from “accepted” to “running” once enough resources are available for it. This is a global scheduler, the familiar FIFO, Capacity, and Fair schedulers. Note that they are global choices. Some variations of resource negotiation are possible in the Application Masters (see below).
The cluster is treated as a resource pool. In fact, in a busy cluster, resources can be requested back from the applications.

Application Manager



12

Application Master

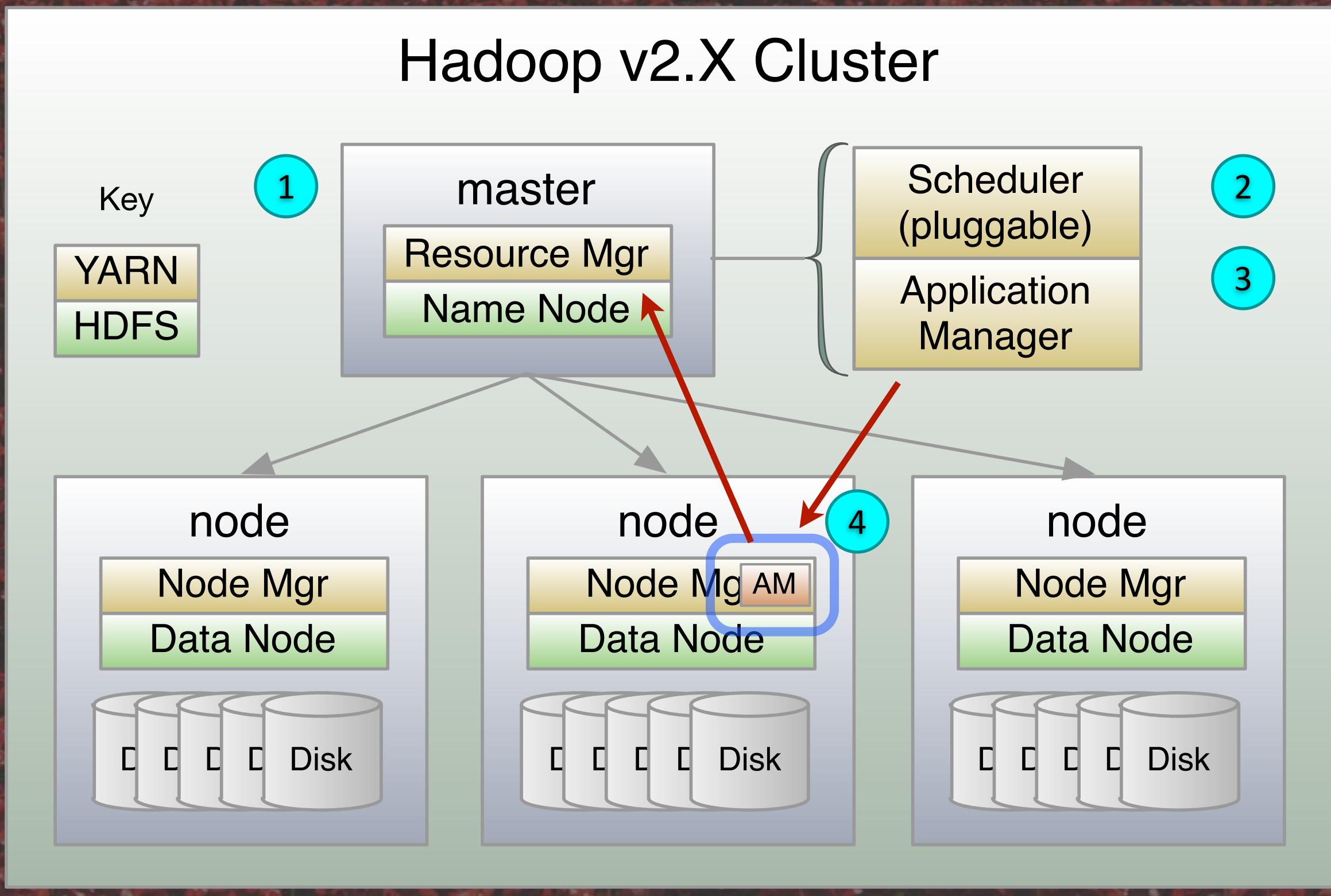


13

Saturday, May 9, 15

The Application Manager spawns one Application Master (AM) on one of the cluster nodes inside a container. This AM understands the compute engine, e.g., MapReduce, Spark.

More Resource Negotiation



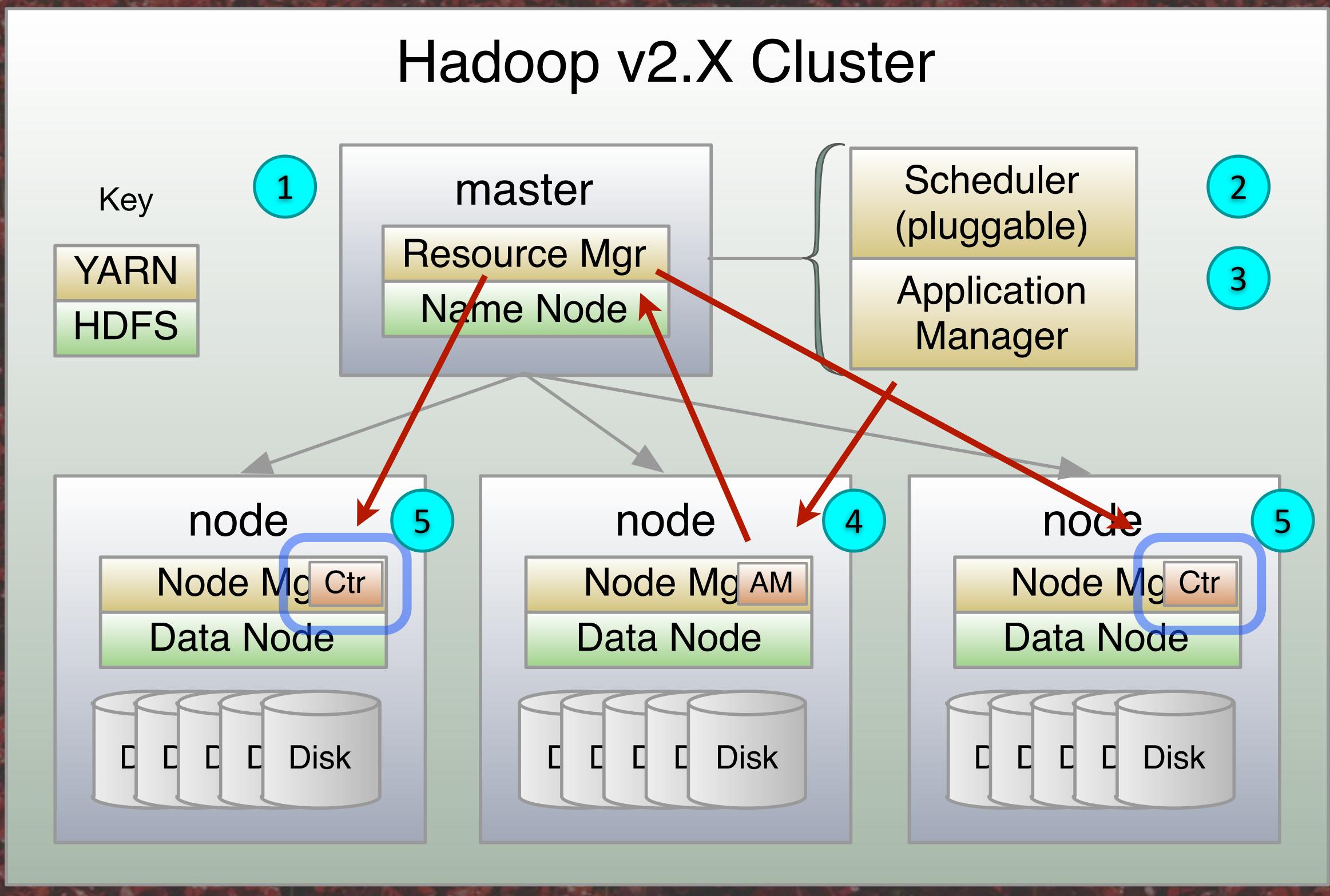
14

Saturday, May 9, 15

The AM manages the job lifecycle, makes resource requests of the RM for resources on the nodes, sometimes with locality preferences (e.g., where HDFS blocks are located) and other container properties, dynamically scales resource consumption through containers, manages the flow of execution (e.g., MR job “stages”) handles failures, etc. When a resource is scheduler for the AM by the RM, a lease is created that the AM will pick up on the next heartbeat.

The AM communicates with user code in containers through Google Protocol Buffers, so it’s agnostic to the code language, etc. Also, each AM supports application-specific logic for features like data locality and other optimizations.

Containers



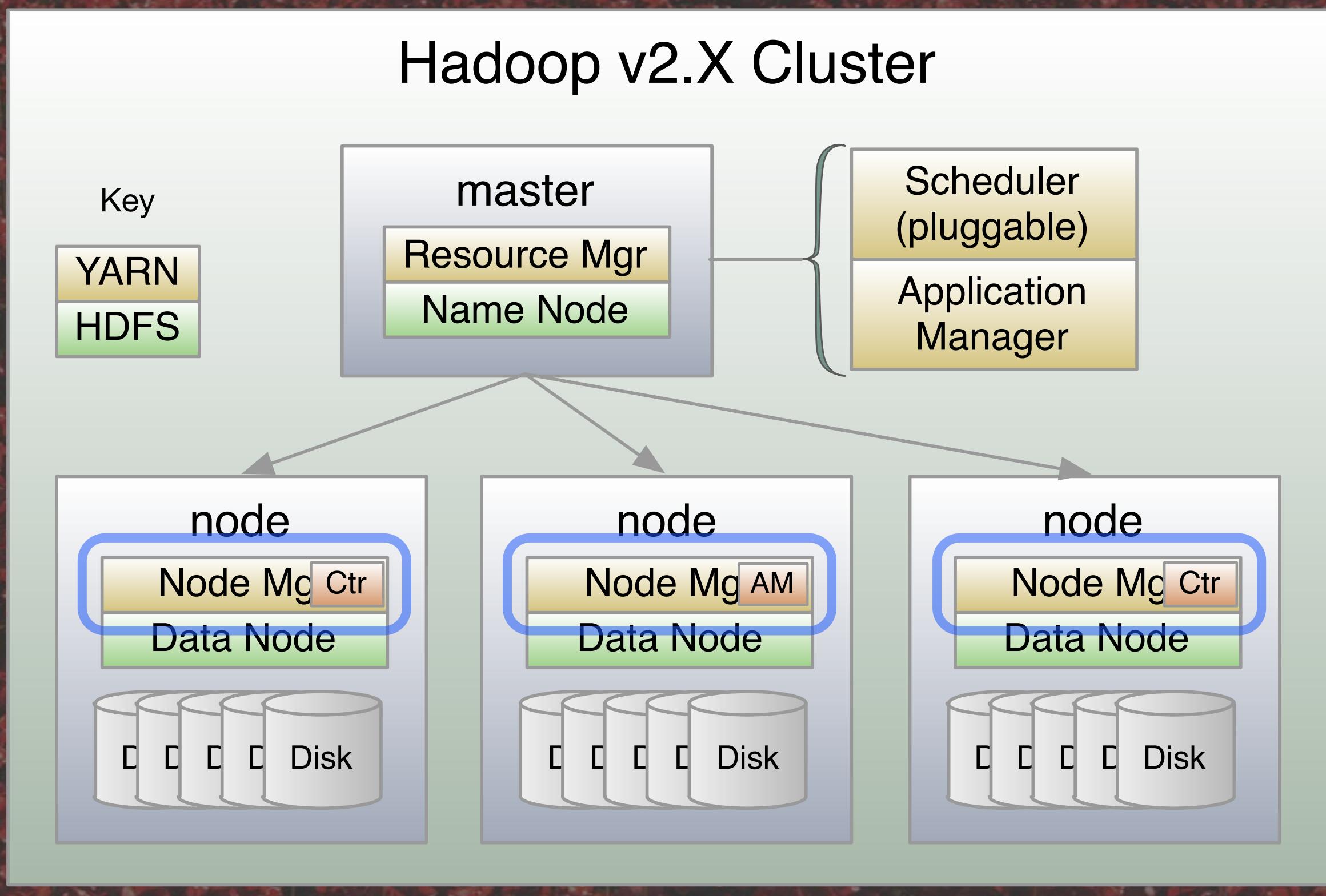
15

Saturday, May 9, 15

Resources are scheduled in new containers on the nodes.

This is a form of “two-level scheduling” with responsibilities split between the master processes and the AMs.

Node Manager

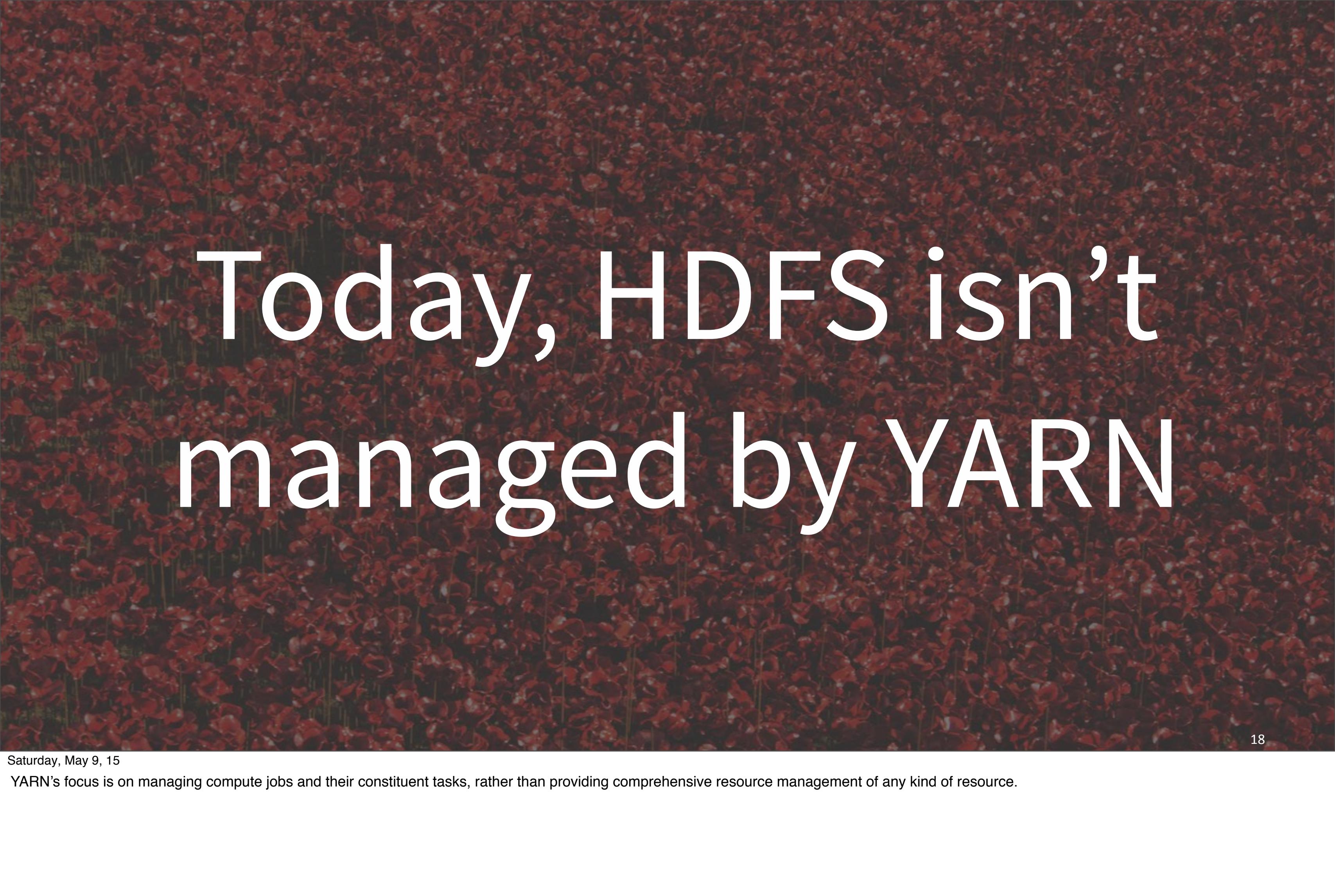


Saturday, May 9, 15

Finally, the containers are supervised locally by the Node Managers. They monitor resource usage by the containers, node health, communicate to the RM, etc.

A Few Key Points

- Resources are *dynamic*.
- Resources: CPU cores & memory.
- Scheduler is pluggable, but (mostly) global.
- Container model works best for “compute” jobs...

A close-up photograph of a vast field of red poppies. The flowers are densely packed, creating a rich, textured pattern of red against a dark, possibly black or very dark green background. The lighting is somewhat dim, suggesting an overcast day or a shaded area.

Today, HDFS isn't managed by YARN

18

Saturday, May 9, 15

YARN's focus is on managing compute jobs and their constituent tasks, rather than providing comprehensive resource management of any kind of resource.

A close-up photograph of a field of red poppies, their vibrant flowers contrasting with dark green leaves and stems. The perspective is slightly elevated, looking down into the dense cluster of flowers.

But greater
flexibility is planned.

[http://hortonworks.com/blog/evolving-
apache-hadoop-yarn-provide-resource-
workload-management-services/](http://hortonworks.com/blog/evolving-apache-hadoop-yarn-provide-resource-workload-management-services/)

19

Saturday, May 9, 15

This greater flexibility is a so-called “container delegation” model that makes app container handling more flexible in ways that are necessary to support a wider variety of services, such as those for HDFS. Today you can’t use YARN to manage a Cassandra ring on the same cluster, for example.



Saturday, May 9, 15

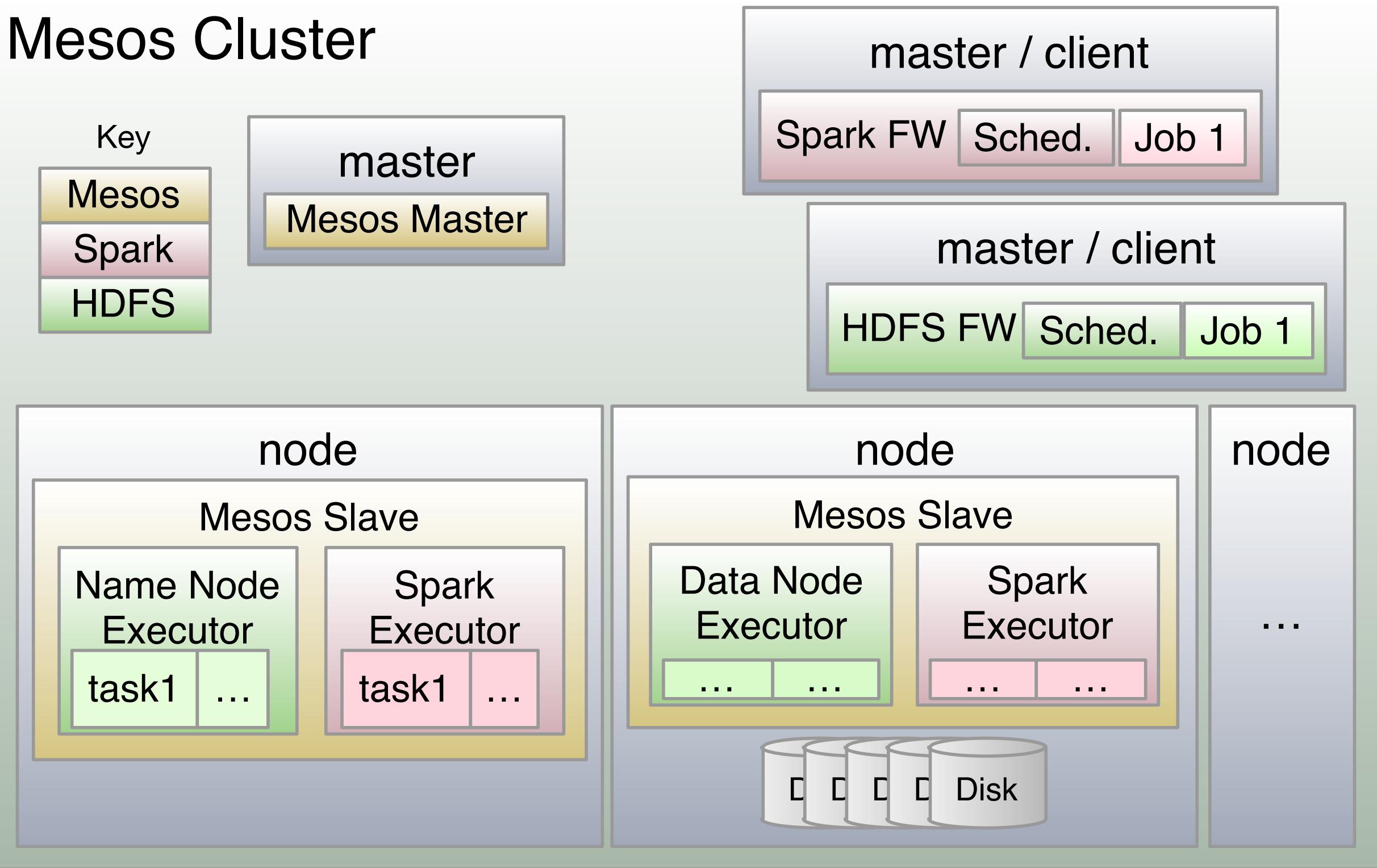
A photograph of a vast field of red poppies in the foreground, leading up to a dark, textured stone wall in the background. The scene is overexposed, with the red flowers appearing as a solid red mass.

Something New

Mesos

mesos.apache.org

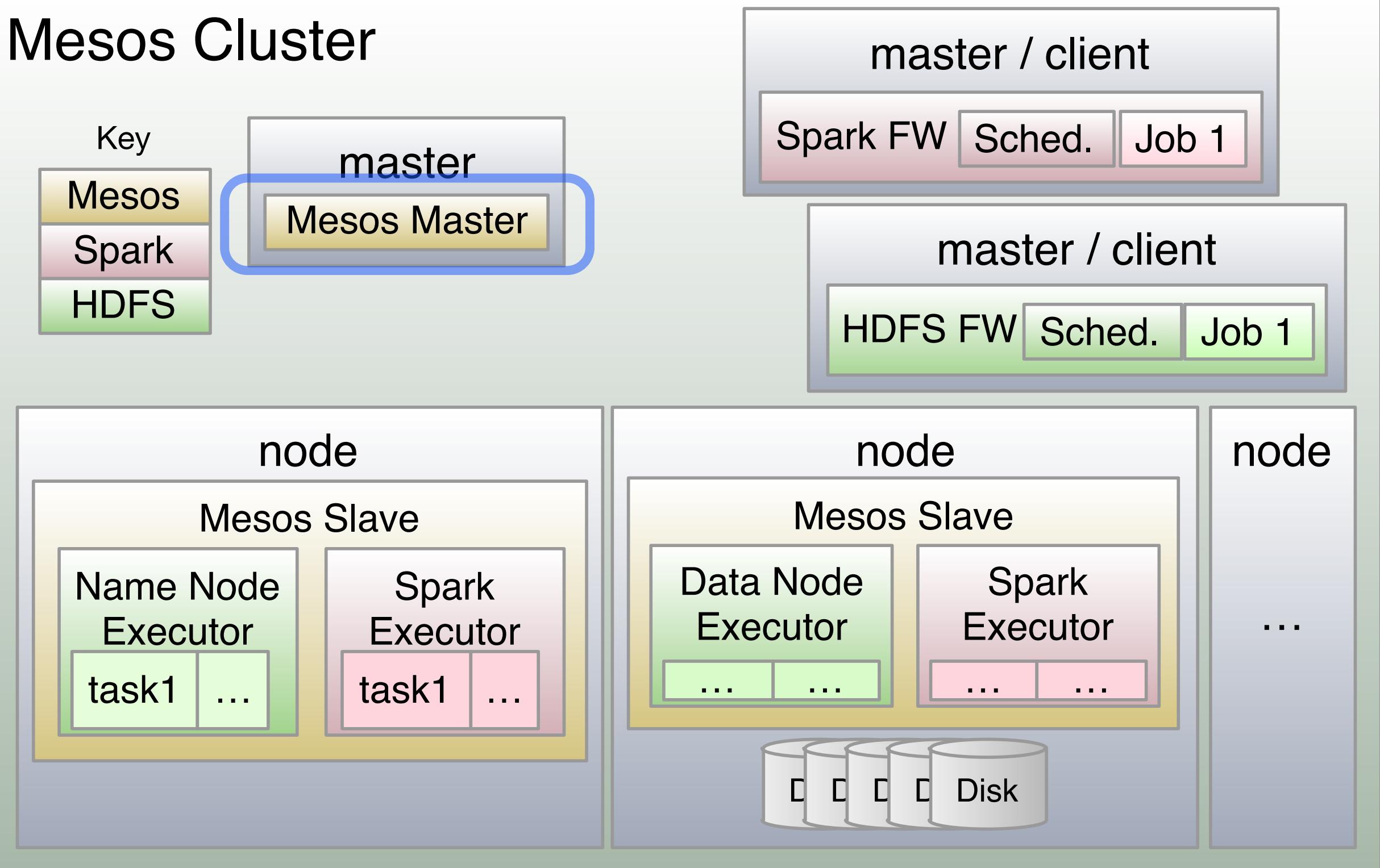
Mesos Cluster



22

Mesos Master

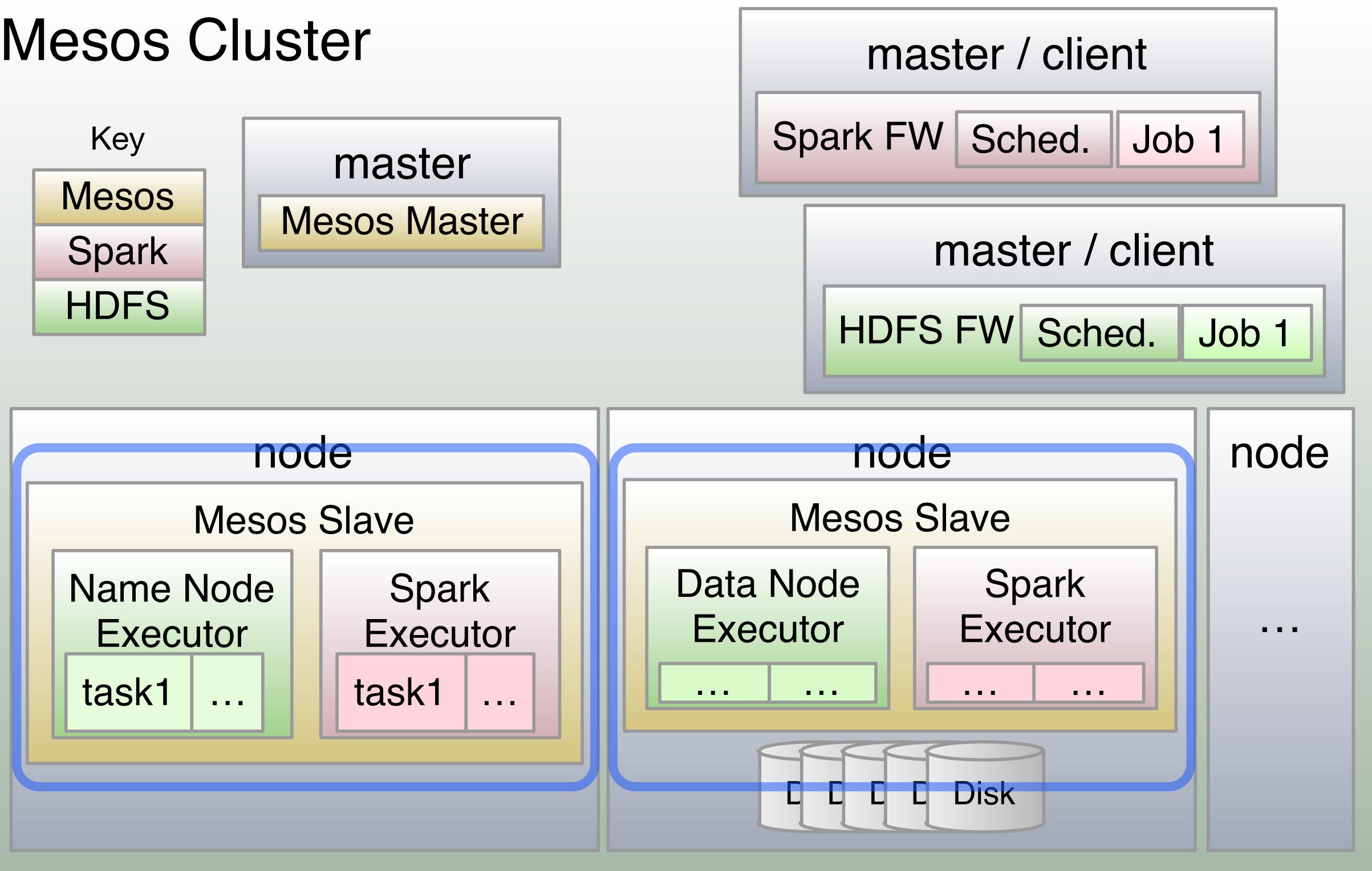
Mesos Cluster



23

Mesos Slaves

Mesos Cluster



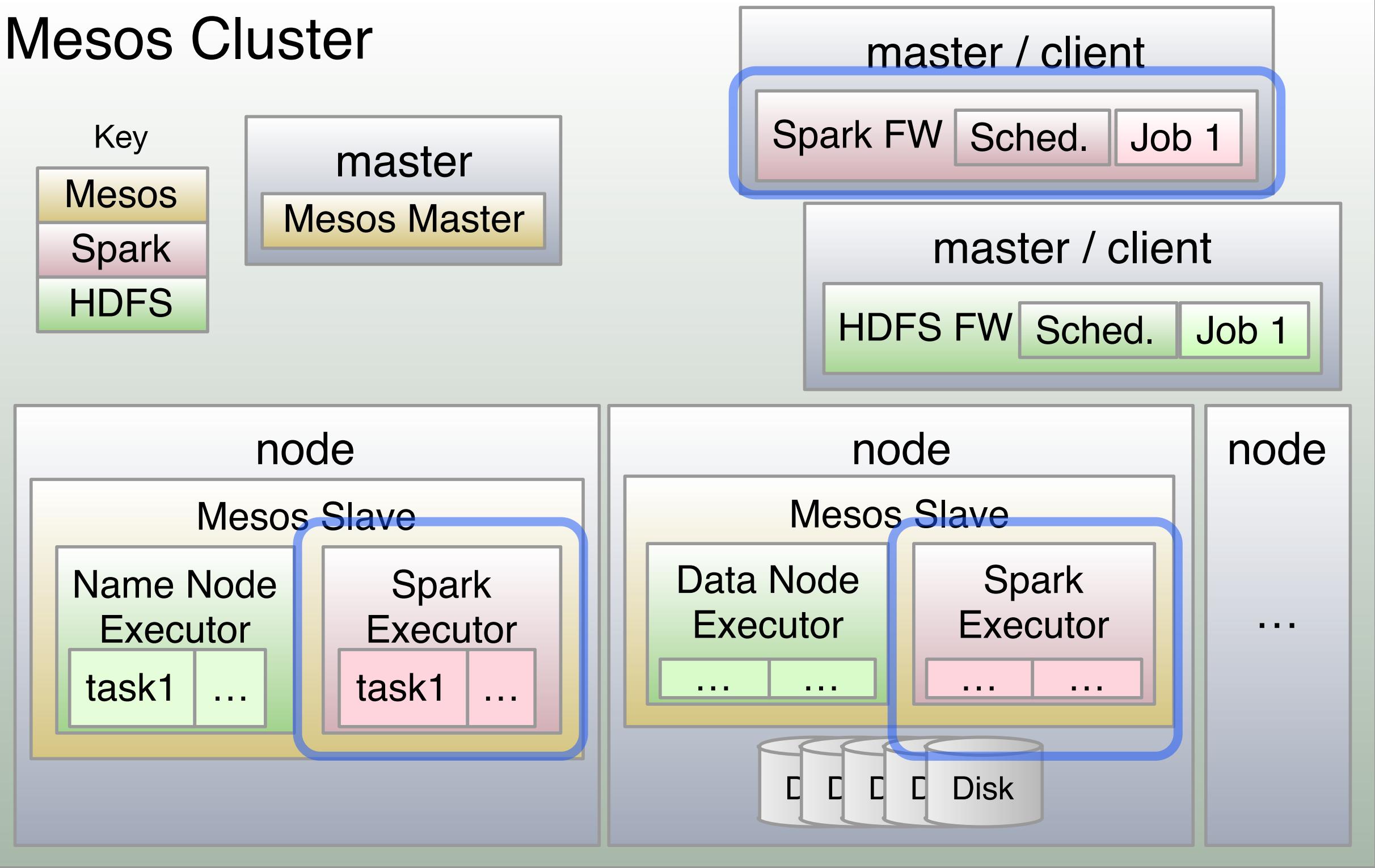
24

Saturday, May 9, 15

The Master controls a Slave daemon on each node, which controls the application Executors running on that node.

Frameworks

Mesos Cluster



25

Saturday, May 9, 15

Each application is a Framework (FW), with a Scheduler. Normally, this will run on the cluster somewhere, not necessarily the the master node, but it could run on some “client” node. It communicates with the Master (using a Scheduler Driver) to negotiate scheduling and resources, etc. Each node has an Executor under which all processes for the Framework are executed. Here I’ve circled just those for Spark.

A framework could manage many “jobs”, such as submitted Spark batch jobs, as we’ll see.

For HDFS, note how the master processes, Name Node and Data Node are run inside executors.

A photograph of a vast field of red poppies in the foreground, leading up to a dark, textured stone wall. The scene is overexposed, with the red flowers appearing as a solid red field.

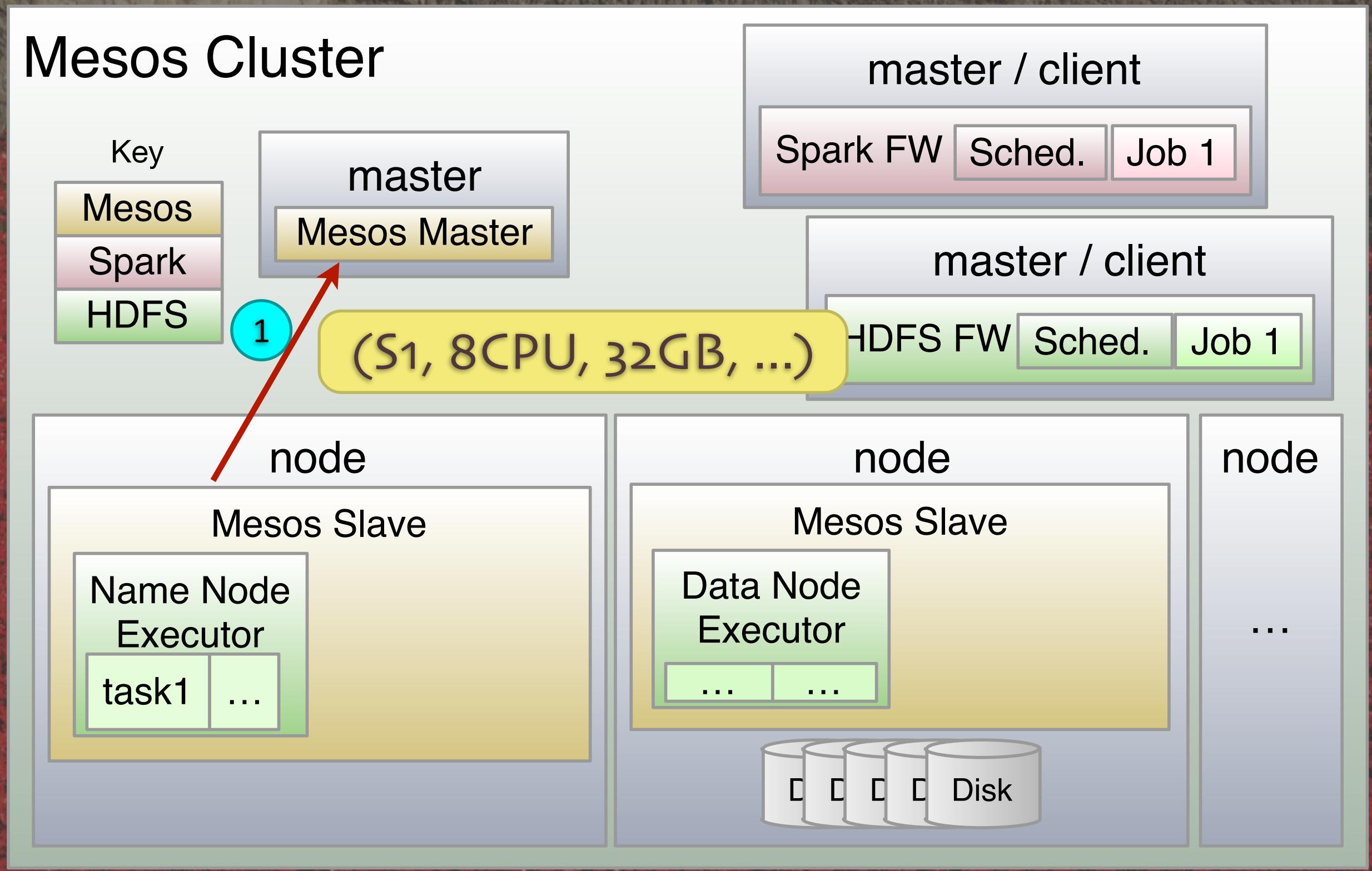
Resources are offered.
They can be refused.

26

Saturday, May 9, 15

A key strategy in Spark is to offer resources to frameworks, which can chose to accept or reject them. Why reject them? The offer may not be sufficient for the need, but it's also a technique for informally imposing policies of interest to the application, such as data locality.

Example: Spark



27

Saturday, May 9, 15

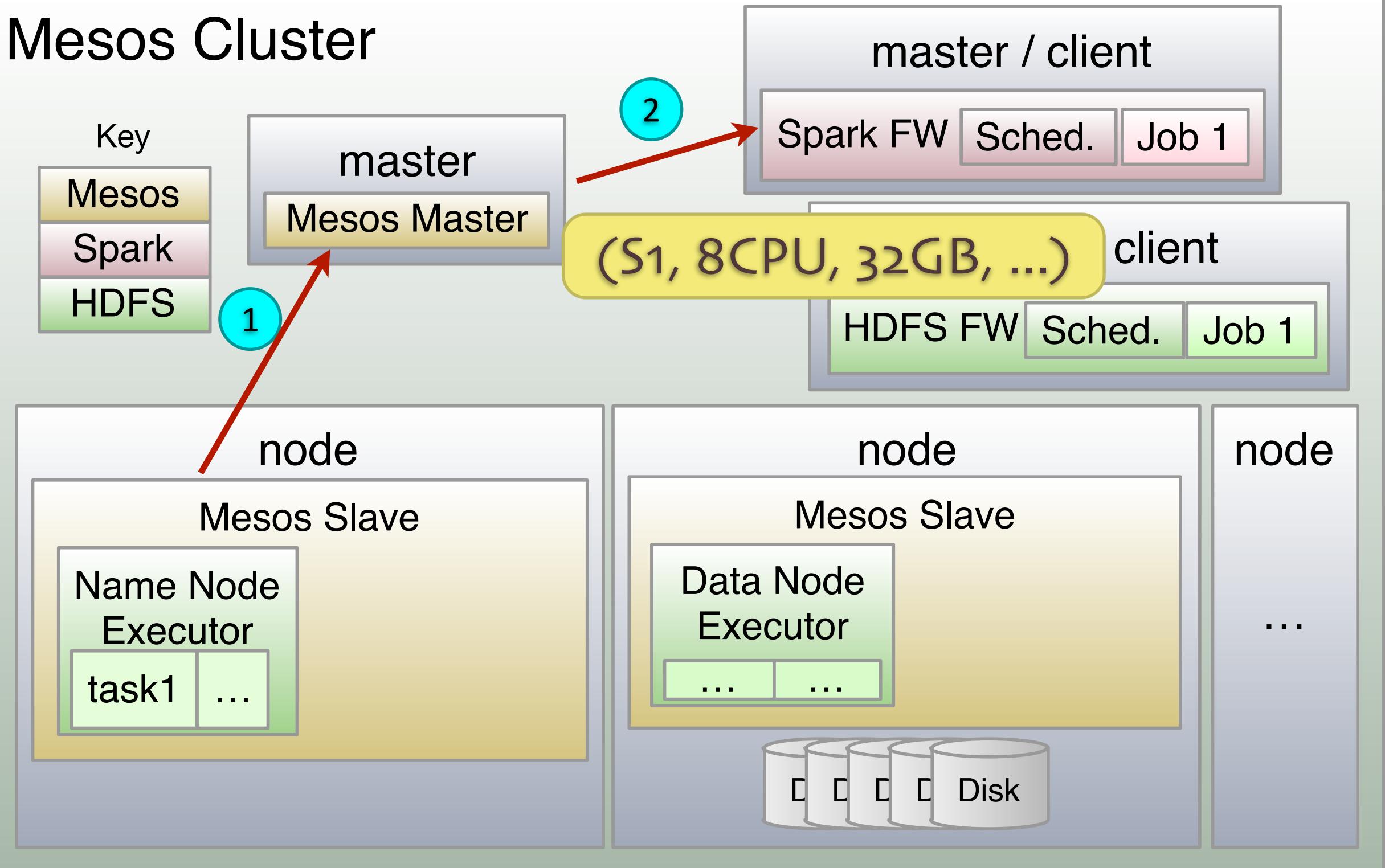
Assume that HDFS is already running and we want to allocate resources and start executors running for Spark.

1. A slave (#1) tells the Master (actually the Allocation policy module embedded within it) that it has 8 CPUs, 32GB Memory. (Mesos can also manage ports and disk space.)

Adapted from <http://mesos.apache.org/documentation/latest/mesos-architecture/>

Example

Mesos Cluster



28

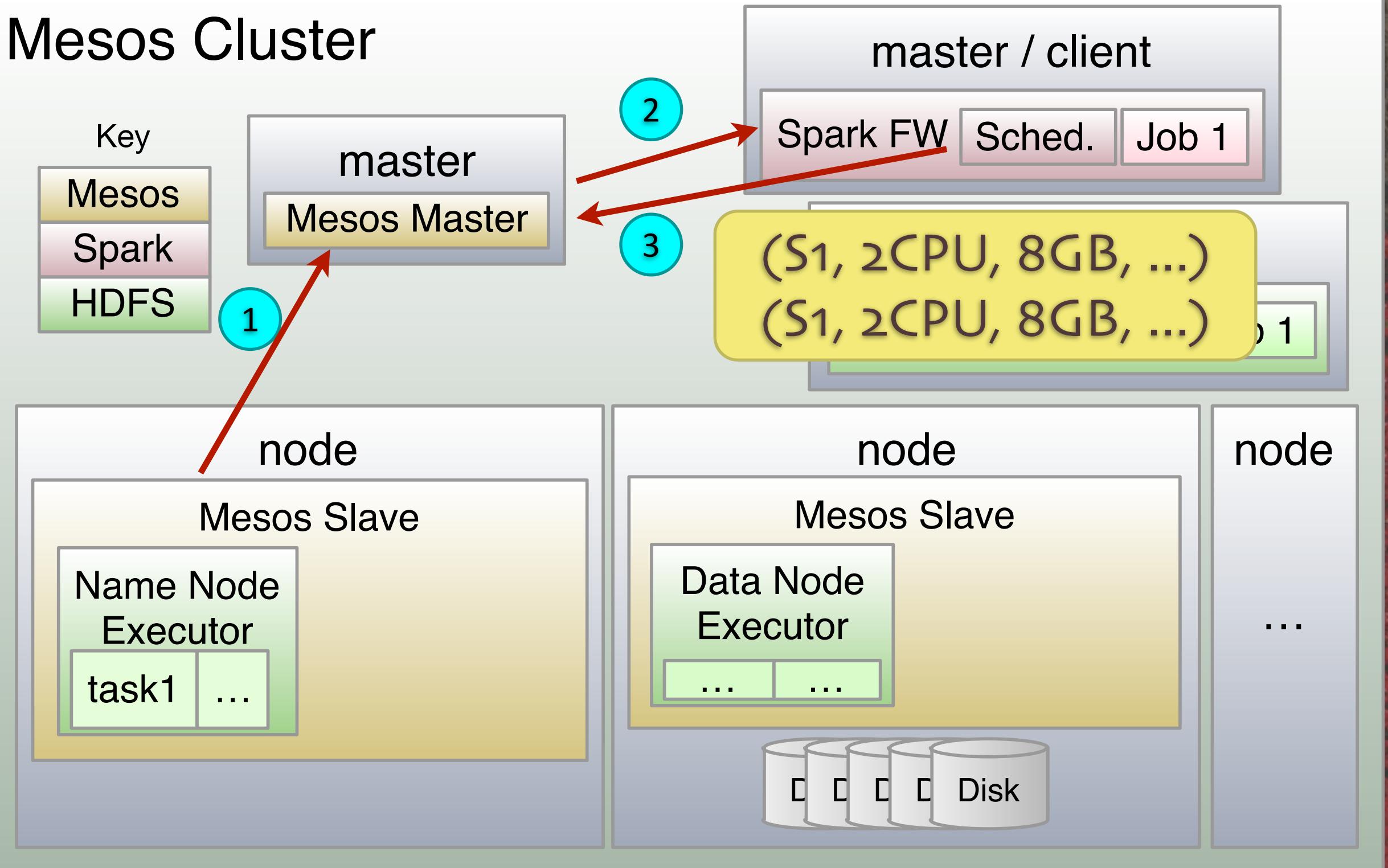
Saturday, May 9, 15

A resource offer example. Adapted from <http://mesos.apache.org/documentation/latest/mesos-architecture/>

2. The Allocation module in the Master says that all the resources should be offered to the Spark Framework.

Example

Mesos Cluster



29

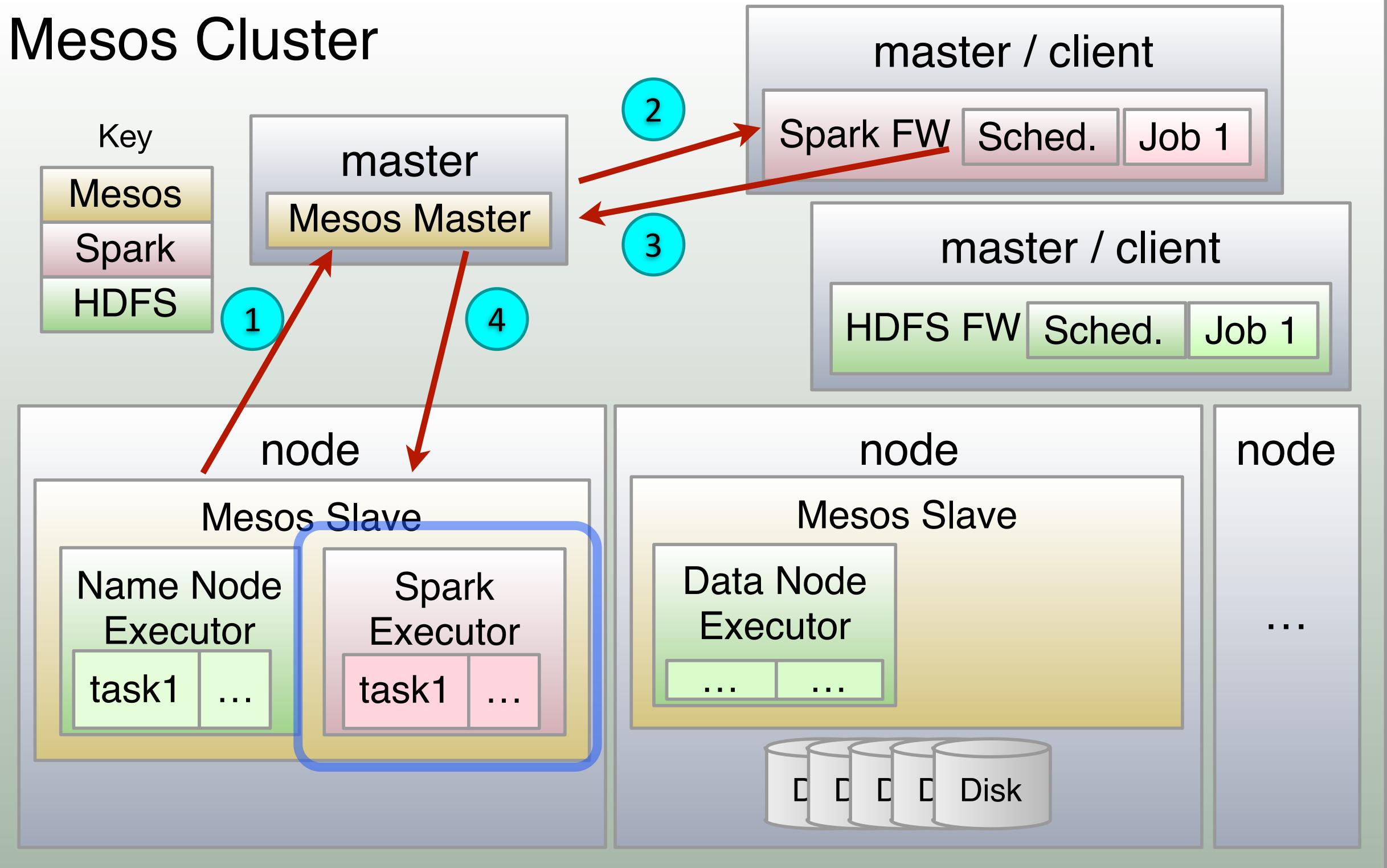
Saturday, May 9, 15

A resource offer example. Adapted from <http://mesos.apache.org/documentation/latest/mesos-architecture/>

3. The Spark Framework Scheduler replies to the Master to run two tasks on the node, each with 2 CPU cores and 8GB of memory. The Master can then offer the rest of the resources to other Frameworks.

Example

Mesos Cluster



30

Saturday, May 9, 15

A resource offer example. Adapted from <http://mesos.apache.org/documentation/latest/mesos-architecture/>

4. The master spans the executor (if not already running - we'll dive into this bubble!!) and the subordinate tasks.

A Few Key Points

- Resources are *dynamic*.
- Resources: CPU cores, memory, *plus* disk, ports.
- Scheduling, resource negotiation fine grained and per-framework.

[http://mesos.berkeley.edu/
mesos_tech_report.pdf](http://mesos.berkeley.edu/mesos_tech_report.pdf)

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, Ion Stoica

University of California, Berkeley

Thursday 30th September, 2010, 12:57

Abstract

We present Mesos, a platform for sharing commodity clusters between multiple diverse cluster computing frameworks, such as Hadoop and MPI. Sharing improves cluster utilization and avoids per-framework data replication. Mesos shares resources in a fine-grained manner, allowing frameworks to achieve data locality by taking turns reading data stored on each machine. To

The solutions of choice to share a cluster today are either to statically partition the cluster and run one framework per partition, or allocate a set of VMs to each framework. Unfortunately, these solutions achieve neither high utilization nor efficient data sharing. The main problem is the mismatch between the allocation granularities of these solutions and of existing frameworks. Many frameworks, such as Hadoop and Dryad, employ a fine-

[http://mesos.berkeley.edu/
mesos_tech_report.pdf](http://mesos.berkeley.edu/mesos_tech_report.pdf)

Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center
*“Our results show that Mesos can achieve
near-optimal data locality when sharing
the cluster among diverse frameworks,
can scale to 50,000 (emulated) nodes,
and is resilient to failures.”*

We present Mesos, a platform for sharing commodity clusters between multiple diverse cluster computing frameworks, such as Hadoop and MapReduce. It improves cluster utilization and avoids per-framework data replication. Mesos shares resources in a fine-grained manner, allowing frameworks to achieve data locality by taking turns reading data stored on each machine. To

The solutions to share a cluster today are either to statically partition the cluster and run one framework per partition, or allocate a set of VMs to each framework. Unfortunately, these solutions achieve neither high utilization nor efficient data sharing. The main problem is the mismatch between the allocation granularities of these solutions and of existing frameworks. Many frameworks, such as Hadoop and Dryad, employ a fine-

[http://mesos.berkeley.edu/
mesos_tech_report.pdf](http://mesos.berkeley.edu/mesos_tech_report.pdf)

Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center

*“To validate our hypothesis ...,
we have also built a new framework
on top of Mesos called Spark...”*

Abstract

We present Mesos, a platform for sharing commodity clusters between multiple diverse cluster computing frameworks, such as Hadoop and MPI. Sharing improves cluster utilization and avoids per-framework data replication. Mesos shares resources in a fine-grained manner, allowing frameworks to achieve data locality by taking turns reading data stored on each machine. To

The solutions of choice to share a cluster today are either to statically partition the cluster and run one framework per partition, or allocate a set of VMs to each framework. Unfortunately, these solutions achieve neither high utilization nor efficient data sharing. The main problem is the mismatch between the allocation granularities of these solutions and of existing frameworks. Many frameworks, such as Hadoop and Dryad, employ a fine-

Containers

- Mesos Containerizer:
 - Lightweight, Linux cgroups, ...
 - Can compose with extra file, disk, PID isolators.

Containers

- Docker Containerizer:
 - Launch docker images as tasks or executors.
 - More complete isolation.

Containers

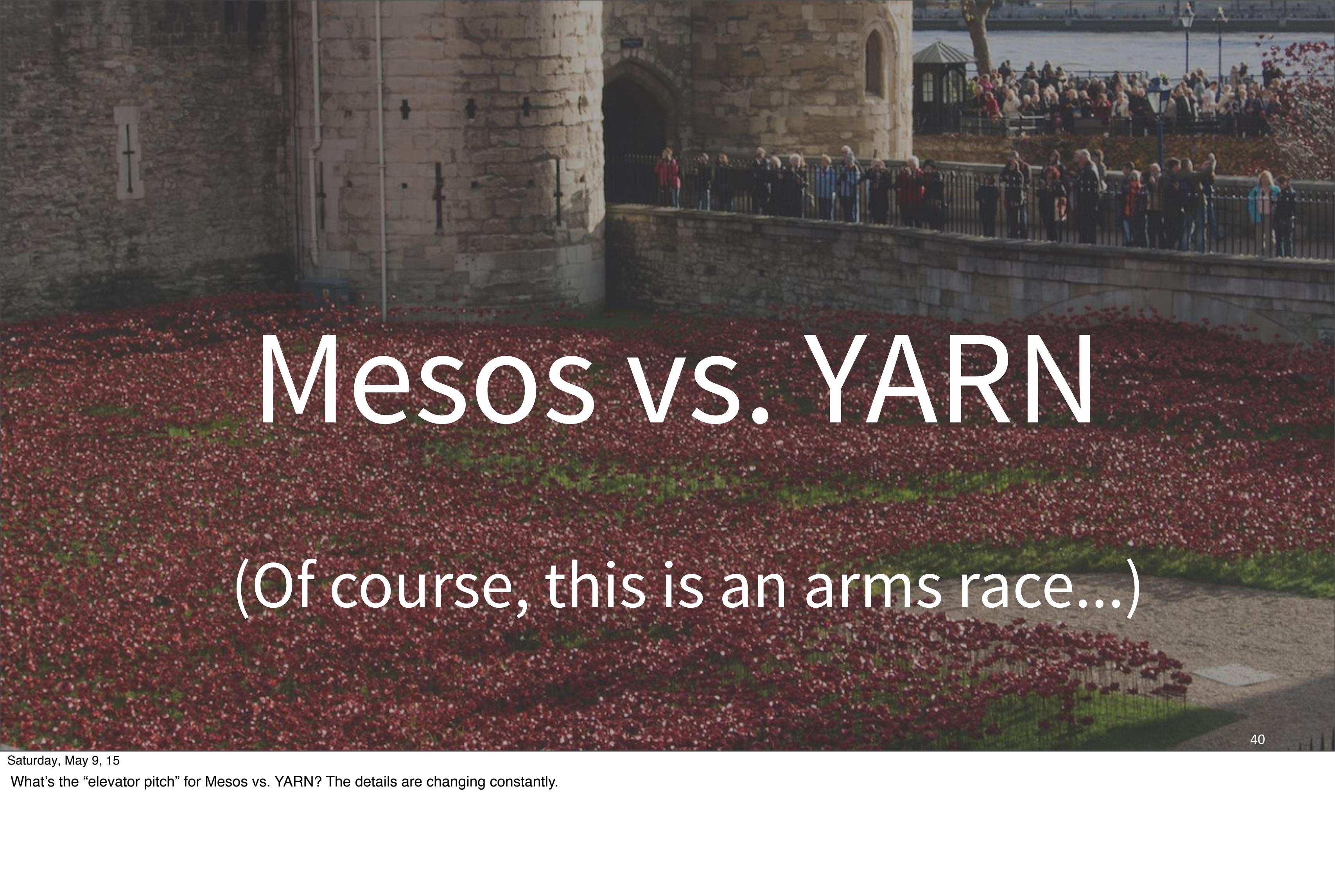
- External Containerizer:
 - Protocol for custom containerizers.

Apps on Mesos

- Apple's Siri
- MySQL - Mysos
- Cassandra
- HDFS
- YARN! - Myriad
- others...



Saturday, May 9, 15



MESOS vs. YARN

(Of course, this is an arms race...)

YARN

- Mature.
- Large developer community.
- Large ecosystem of 3rd-party apps.
- Supported by Hadoop vendors.
- Improving support for long-running apps.

YARN

- ❑ Limits of scheduler & container models:
 - ❑ Limited to “~compute” jobs.
 - ❑ Hadoop-centric.
 - ❑ Other resources: disk, network, ...?

Mesos

- Next generation resource model.
- More resource “kinds”.
- Flexible containerization.
- Wider class of applications, even all of Hadoop!
- More fine-grained scheduling.

Mesos

- Easier programming model for 3rd-party apps.

Mesos

- Smaller, less mature ecosystem.
- Doesn't fill all Enterprise “niches”.
- Smaller developer community.
- Fewer support vendors.



Spark on Mesos



Saturday, May 9, 15



Saturday, May 9, 15

Both Born at UC Berkeley

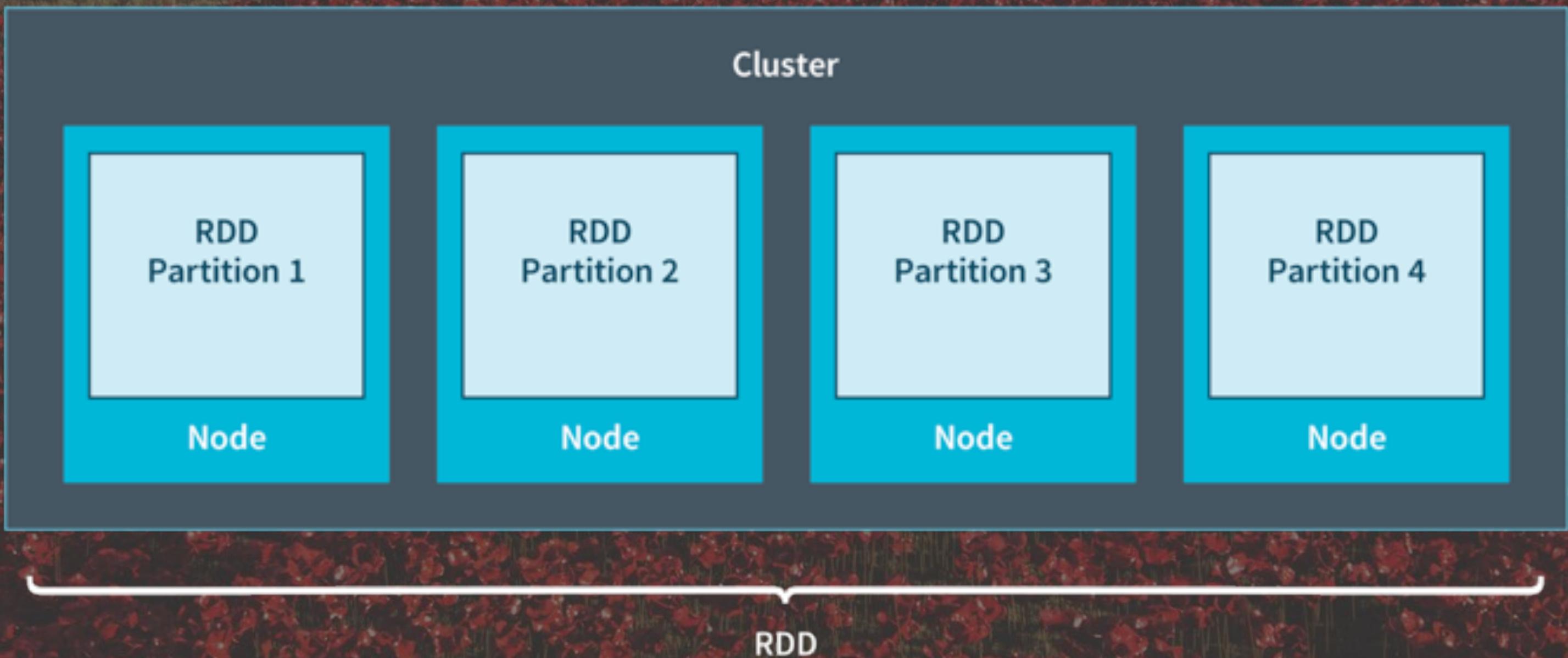
- Benjamin Hindman & Matei Zaharia
- ~2007-2009
- Both now top-level Apache projects

Spark Is Cluster Agnostic

- YARN
- Mesos
- Standalone mode
- Local (1 machine)

Compute Model

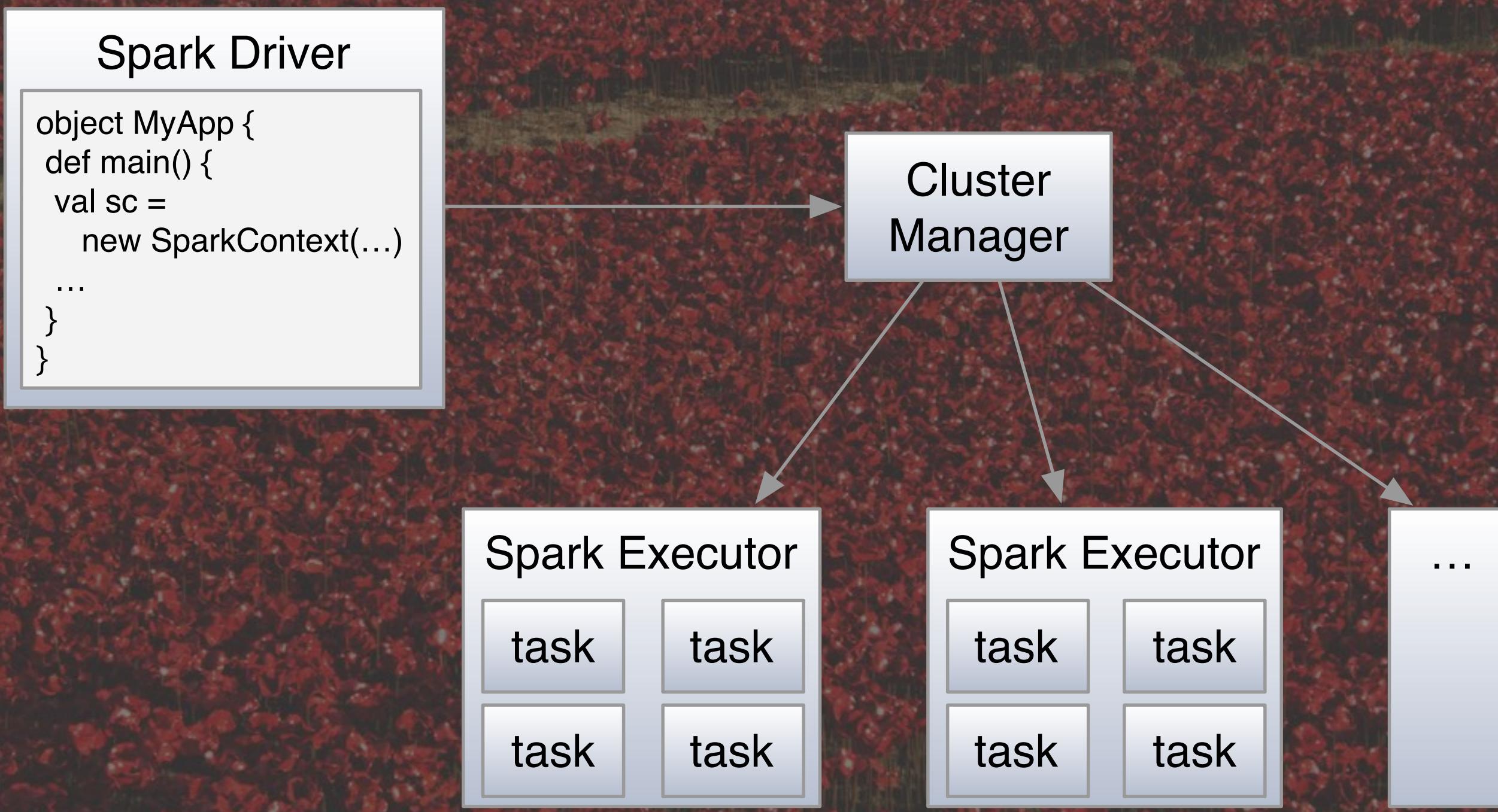
- RDDs: Resilient Distributed Datasets



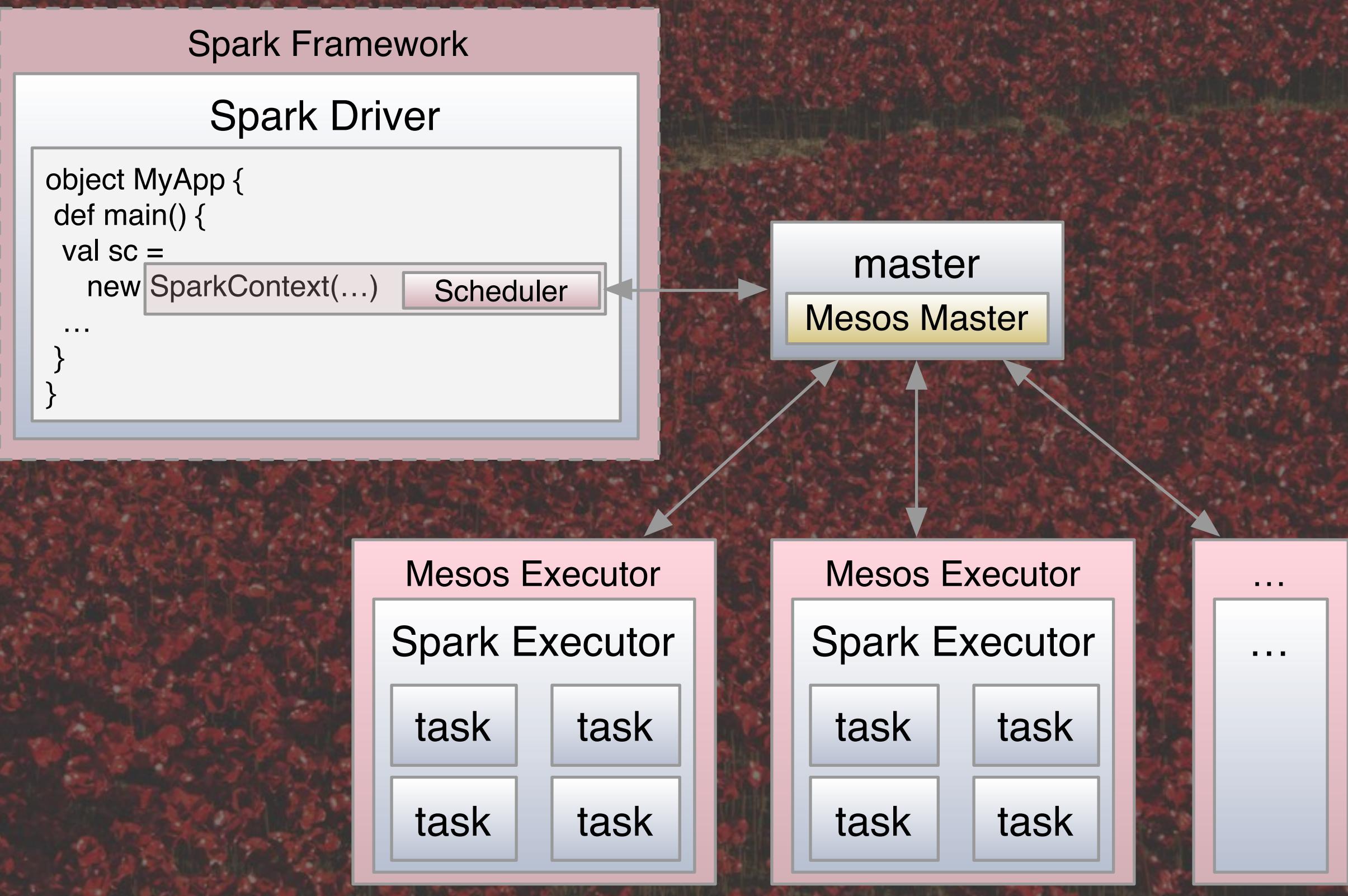
Saturday, May 9, 15

Your data is partitioned across the cluster. The overall data structure behaves like a collection with functional/SQL operations on it. It's resilient because if one partition is lost (say the node crashes), Spark can reconstruct it from earlier RDDs in the processing pipeline, all the way back to the input.

Cluster Abstraction



Mesos Coarse Grained Mode

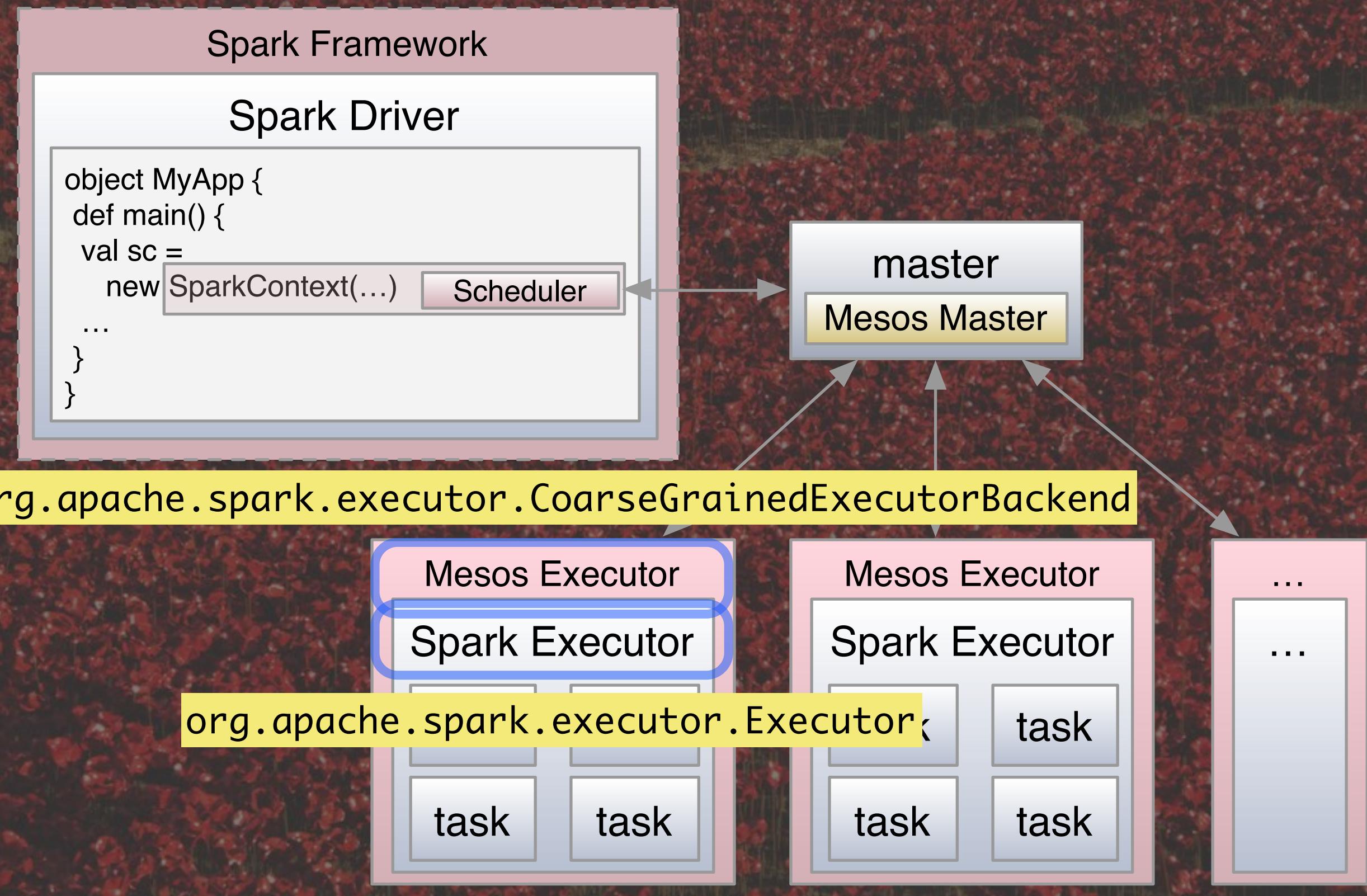


52

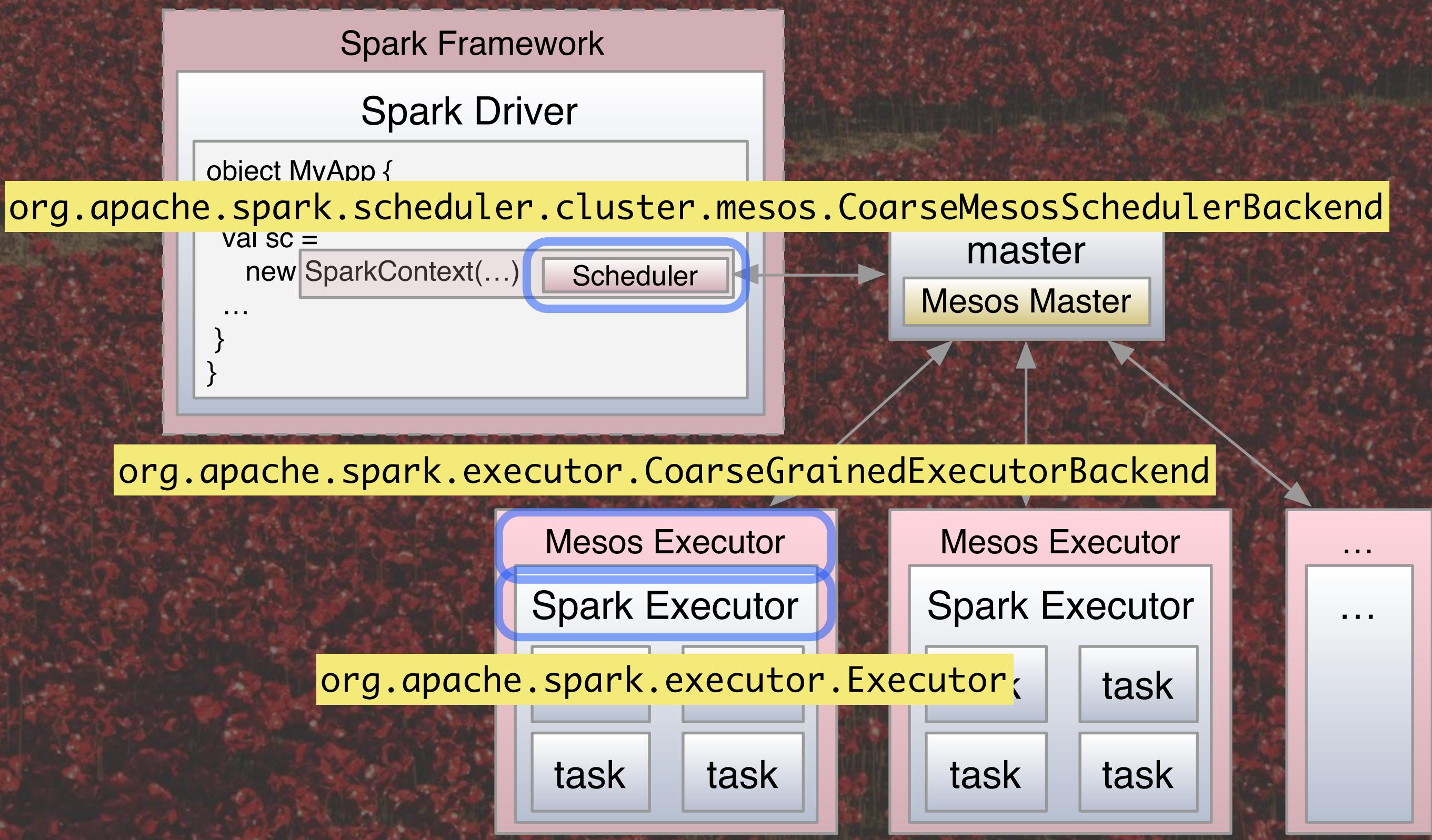
Saturday, May 9, 15

Unfortunately, because Spark and Mesos “grew up together”, each uses the same terms for concepts that have diverged. The “Mesos Executors” shown were called “Spark Executors” in previous slides. The actual Scala class name is `org.apache.spark.executor.CoarseGrainedExecutorBackend`. It has a “main” and a process. It encapsulates a cluster-agnostic instance of Scala class `org.apache.spark.executor.Executor`, which manages the Spark tasks. The nested “Spark Executor” is called `CoarseGrainedExecutorBackend`. Note that the Spark “Framework” is an abstraction, not an actual service or process. Those are the constituents, like the scheduler, executors, etc.

Mesos Coarse Grained Mode

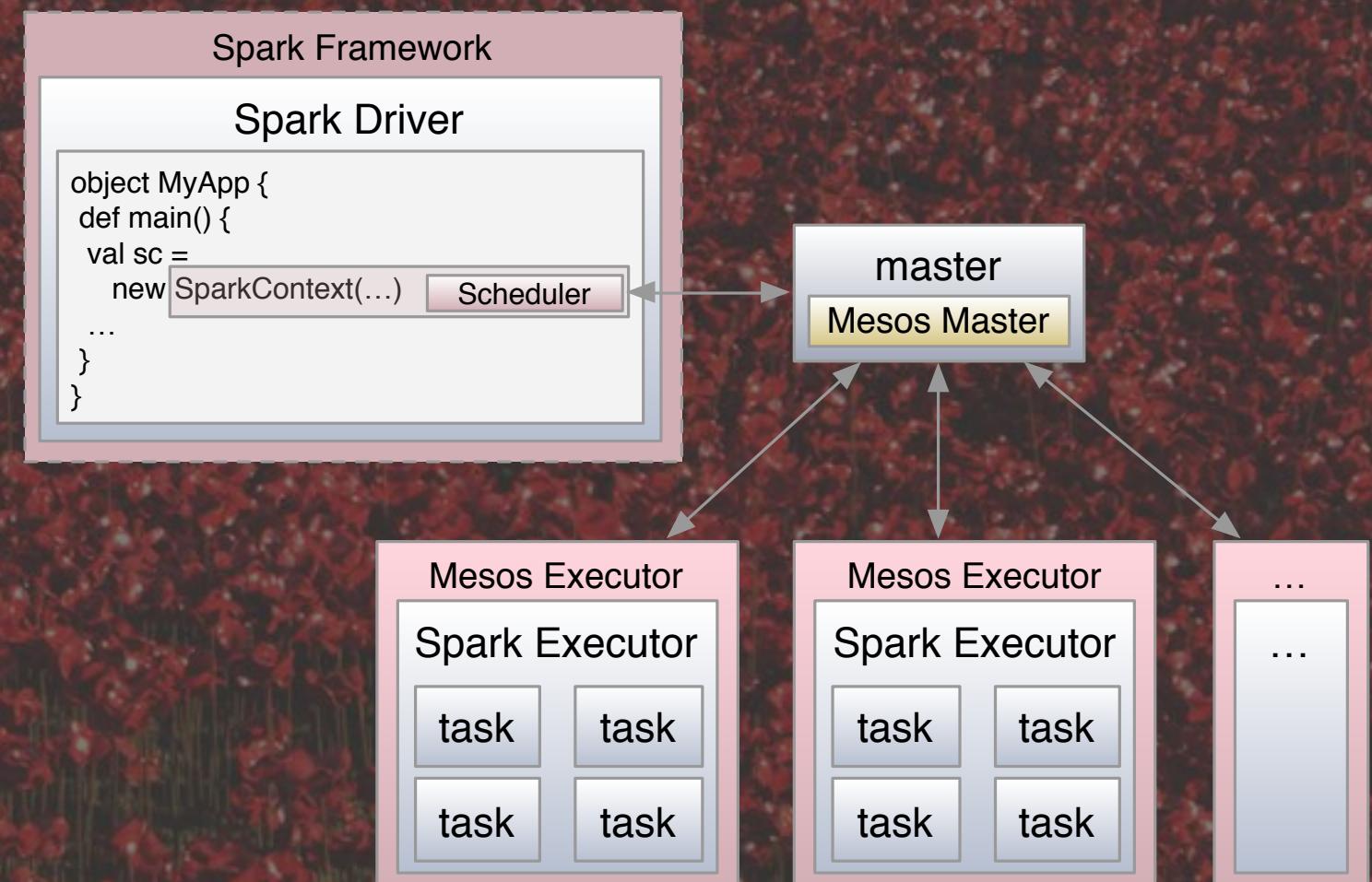


Mesos Coarse Grained Mode



Mesos Coarse Grained Mode

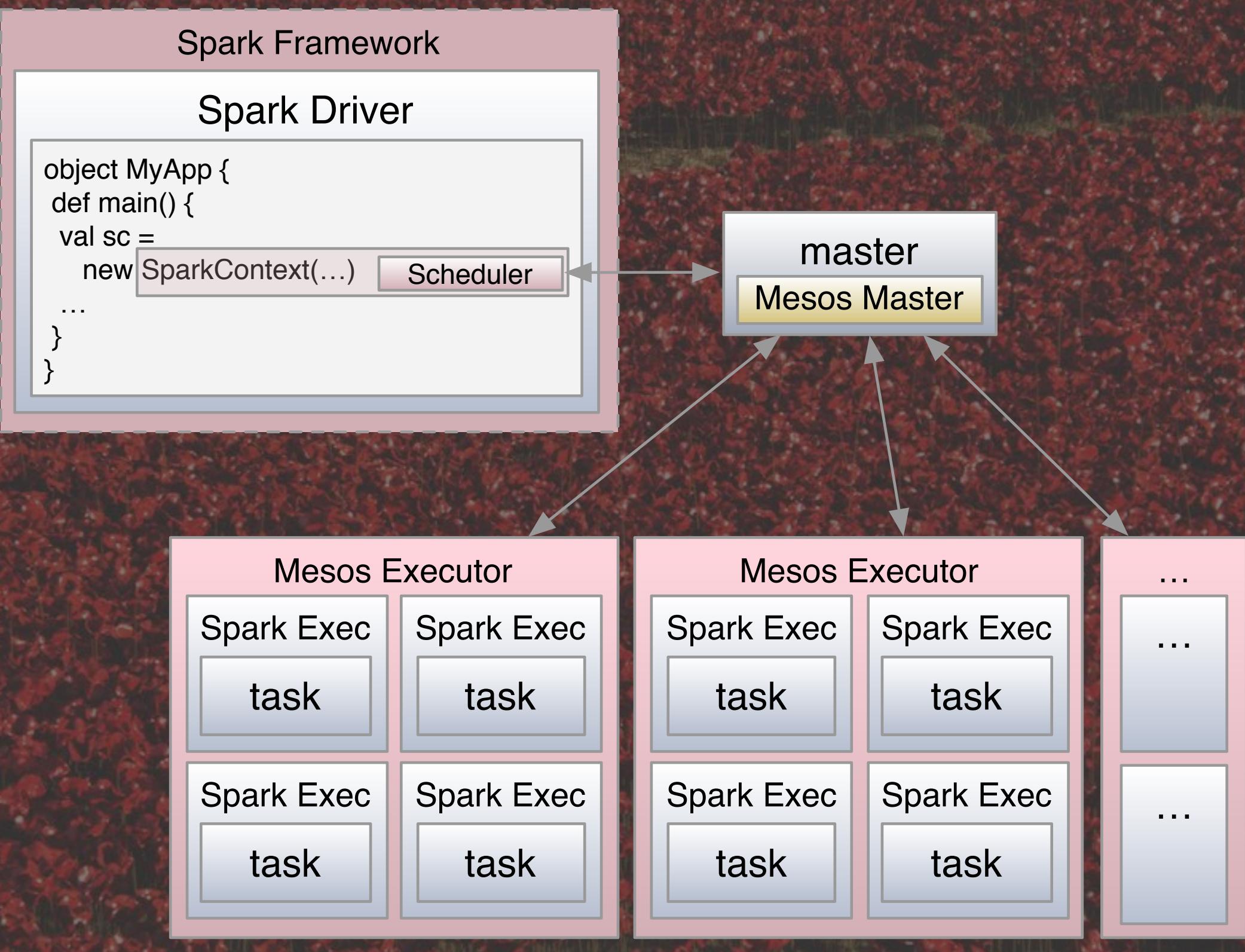
- One Mesos and one Spark executor for the job's lifetime.
- Tasks are spawned by Spark itself.



Mesos Coarse Grained Mode

- Fast startup for tasks:
Better for interactive sessions.
- Resources locked up in larger
Mesos task.
 - But dynamic allocation coming...
- One task per slave!

Mesos Fine Grained Mode

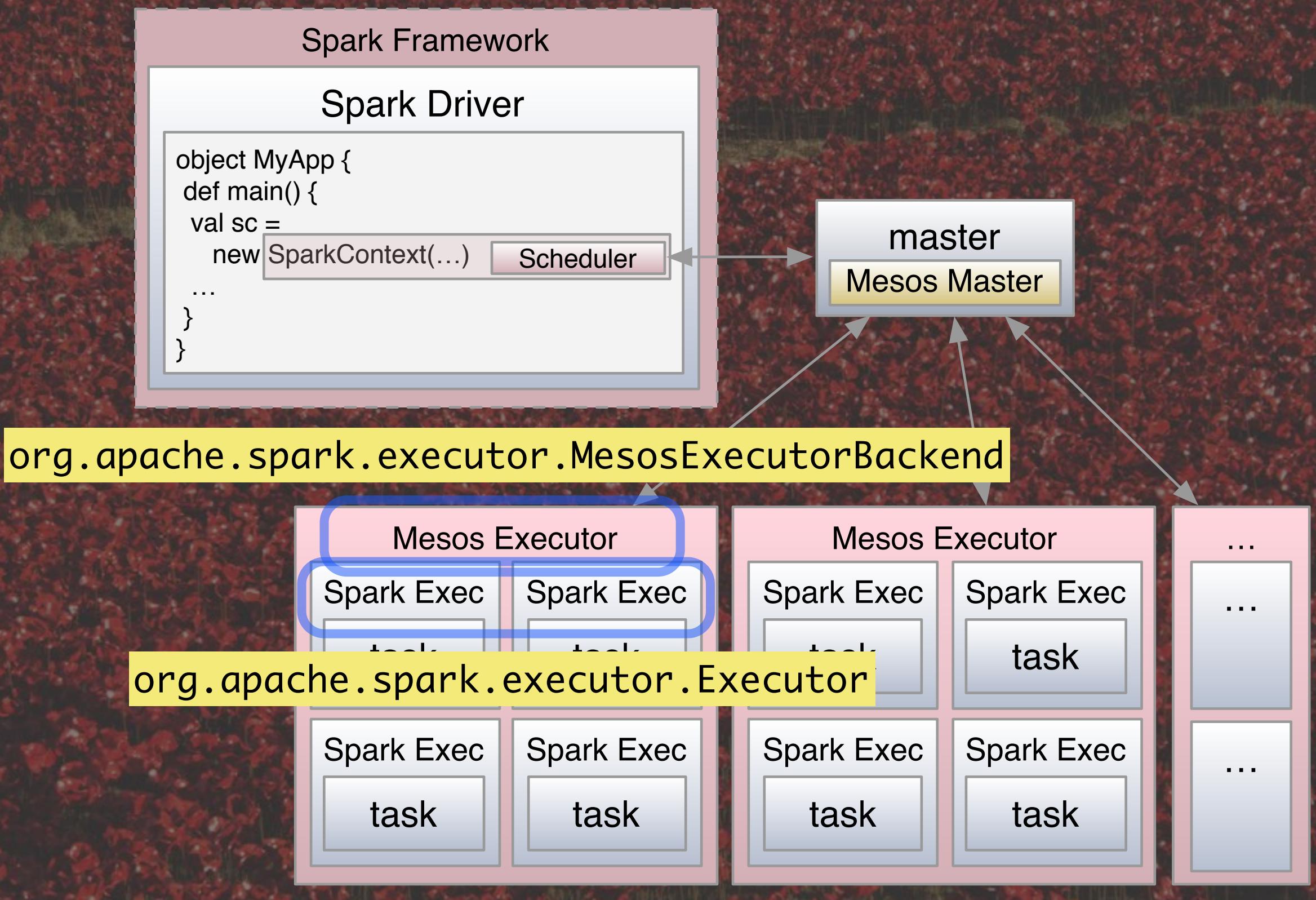


57

Saturday, May 9, 15

There is still one Mesos executor. The actual Scala class name is now `org.apache.spark.executor.MesosExecutorBackend`. The nested “Spark Executor” is still the Mesos agnostic `org.apache.spark.executor.Executor`. The scheduler (a `org.apache.spark.scheduler.cluster.mesos.MesosSchedulerBackend`) is instantiated as a field in the `SparkContext`.

Mesos Fine Grained Mode

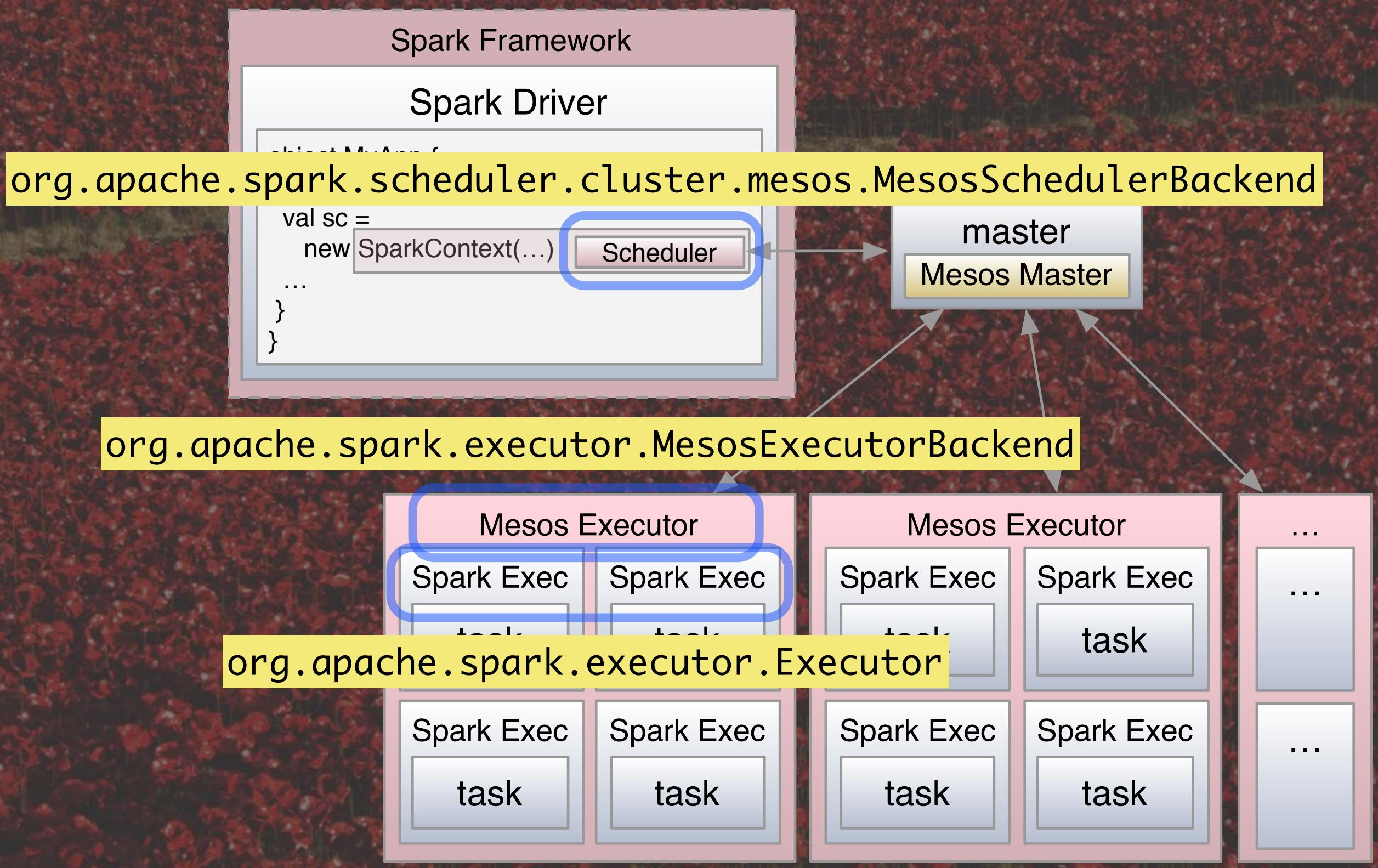


58

Saturday, May 9, 15

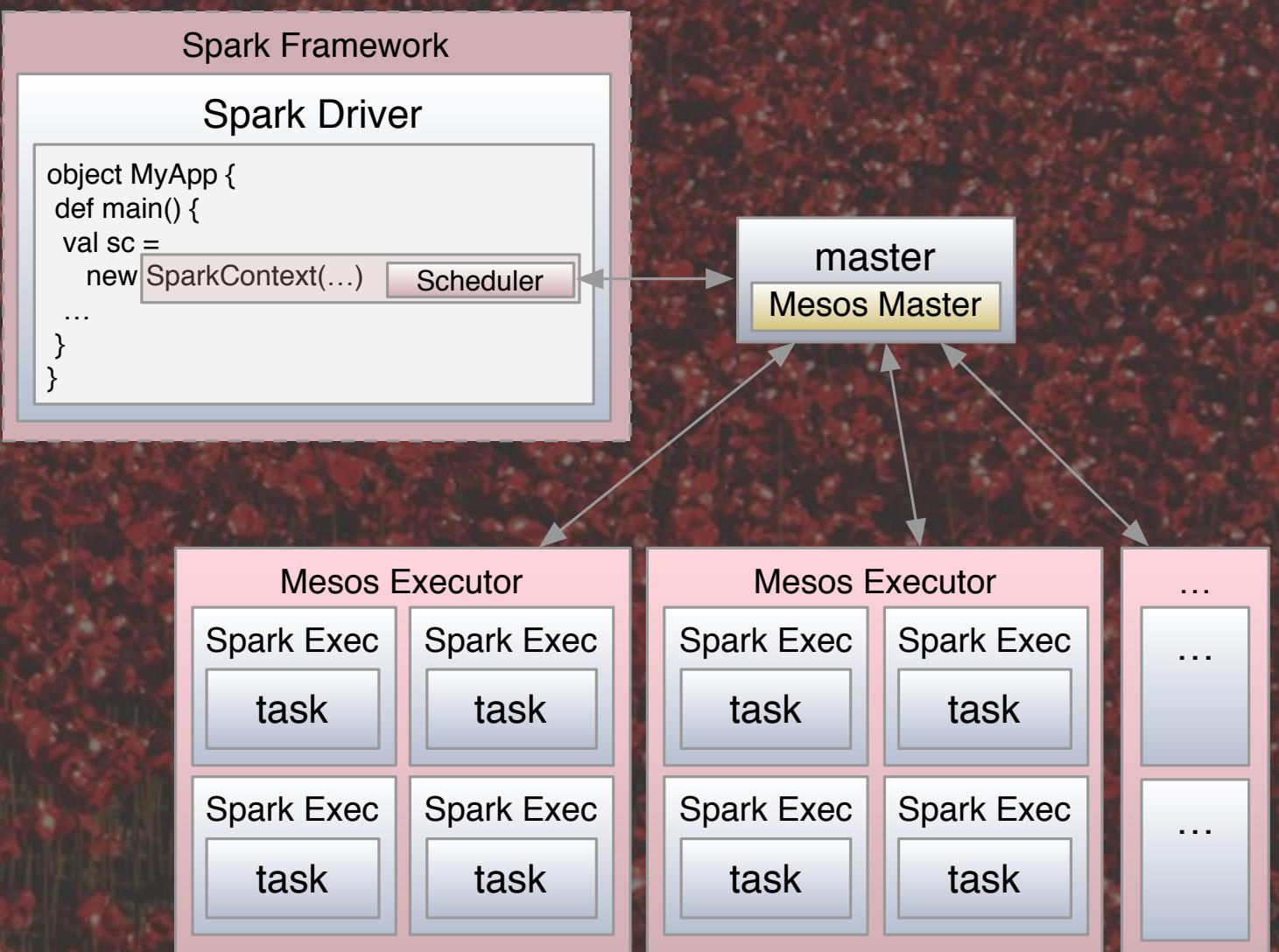
There is still one Mesos executor. The actual Scala class name is now `org.apache.spark.executor.MesosExecutorBackend` (no “FineGrained” prefix), which is now Mesos-aware. The nested “Spark Executor” is still the Mesos-agnostic `org.apache.spark.executor.Executor`, but there will be one created per task now. The scheduler (a `org.apache.spark.scheduler.cluster.mesos.MesosSchedulerBackend`) is instantiated as a field in the `SparkContext`.

Mesos Fine Grained Mode



Mesos Fine Grained Mode

- One Mesos task per Spark executor
- Spark tasks are spawned as threads.



60

Mesos Fine Grained Mode

- Slower startup for tasks:
Fine for batch and relatively static
streaming.
- Better resource utilization.

A photograph of the Tower of London's moat filled with thousands of red poppies, a memorial to the fallen. The medieval stone walls of the tower rise behind the flowers, and a modern building is visible in the background.

“Deployment” Mode

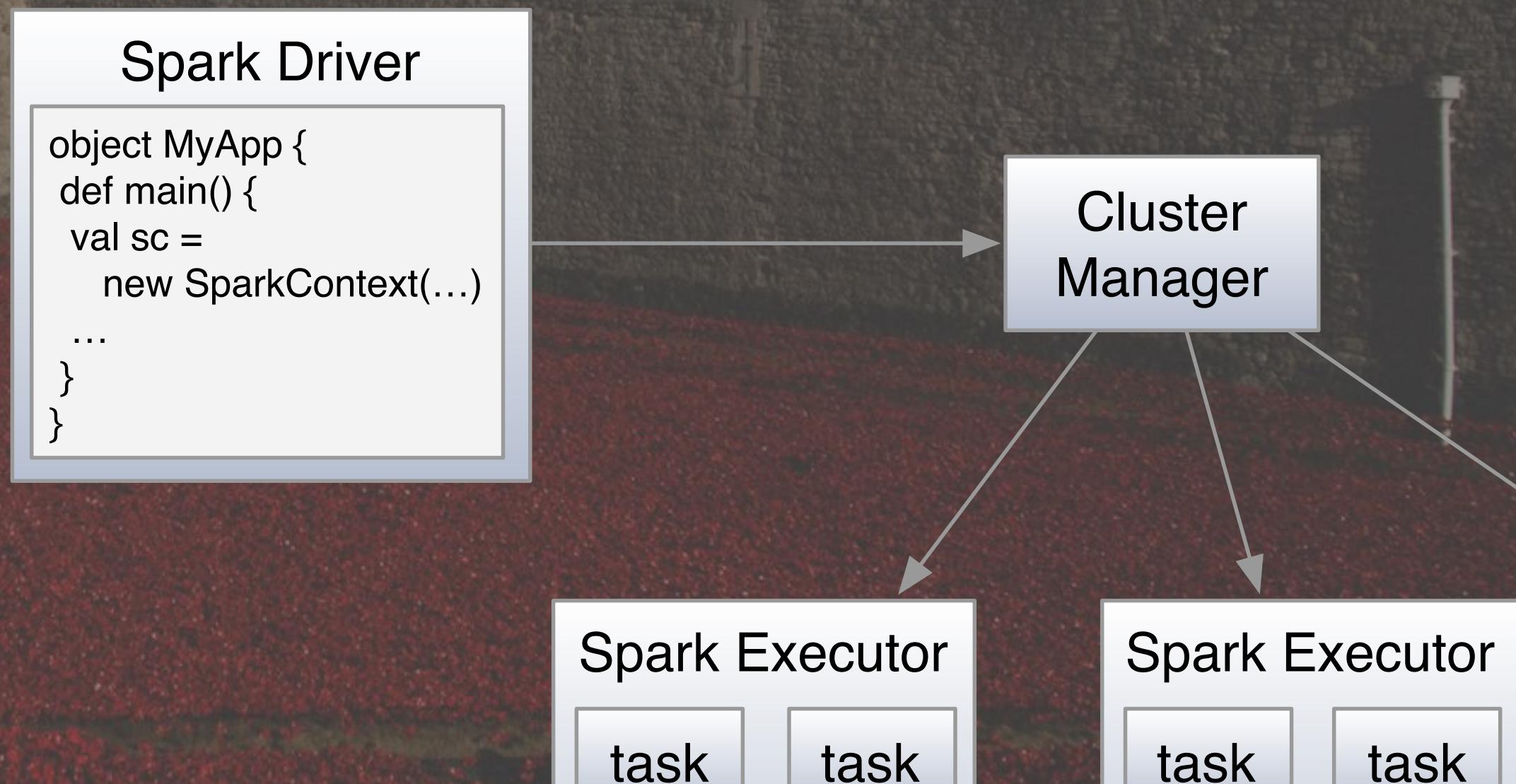
62

Saturday, May 9, 15

This is the Spark term for a how you run your driver program. It's one last axis of variability...

Client vs. Cluster Mode

- Where does the *Driver* actually run?



Client Mode

- Driver runs in a separate program.
 - E.g., spark shell on your laptop.
 - Could also be a running process on the cluster.
- Submitting process waits for exit.

Cluster Mode

- Driver managed by Mesos.
 - E.g., long-running streaming job.
- Submitting process exits immediately.



Saturday, May 9, 15

Language?

- Java
- Scala

Language?

Java
 Scala

Compute Engine?

- MapReduce
- Spark

Compute Engine?

- MapReduce
- Spark

Cluster Platform?

- YARN
- Mesos

Cluster Platform?



YARN
 Mesos

72

Saturday, May 9, 15

This is cheating a bit, as I said they aren't exactly targeting the same problem and you can even run YARN on Mesos, but just as the Hadoop community replaced Spark after five or so years of widespread public use, I think five years from now Mesos could be the "baseline" for Big Data systems, including those labeled "Hadoop". Two reasons:

1. The compelling efficiency that Mesos brings and the way it lets you use one cluster, not several.
2. The relative ease with which developers are implement new services and porting existing services on top of Mesos.

os-production
172.03.12.1

Dashboard

Services

Datacenter

The Mesosphere Datacenter Operating System

Put your datacenter and cloud on autopilot with the Mesosphere datacenter operating system. Save time, save money, and deliver software faster.

[Get the Public Beta](#)

Saturday, May 9, 15

Typesafe now offers commercial support for development teams using Spark on Mesosphere DCOS, in partnership with Mesosphere. Typesafe also offers developer support for teams getting started with Spark, but with plans to deploy to other platforms, like YARN.

Typesafe is working with Mesosphere on Spark + Mesos

Put your datacenter and cloud on autopilot with the Mesosphere datacenter operating system. Save time, save money, and deliver software faster.



Spark is a fast and general engine for large-scale
data processing built in Scala

*The Spark logo is the property of the Apache foundation.

[SCROLL DOWN TO LEARN MORE](#)

typesafe.com/reactive-big-data



Spark is a fast and general engine for large-scale
data processing built in Scala

*The Spark logo is the property of the Apache foundation.

[SCROLL DOWN TO LEARN MORE](#)

Spark on Mesos

dean.wampler@typesafe.com
polyglotprogramming.com/talks
[@deanwampler](https://twitter.com/deanwampler)



Saturday, May 9, 15

Photos, Copyright (c) Dean Wampler, 2014-2015, All Rights Reserved, unless otherwise noted. All photos were taken on Nov. 7, 2014 of "The Tower of London Remembers", commemorating the Armistice Remembrance display of 888,246 ceramic poppies, one for each British soldier who died in The Great War, to mark the 100th anniversary of the start of the war. (<http://poppies.hrp.org.uk/>)
Other content Copyright (c) 2015, Typesafe, but is free to use with attribution requested.
<http://creativecommons.org/licenses/by-nc-sa/2.0/legalcode>