Open Source Datacenter Systems

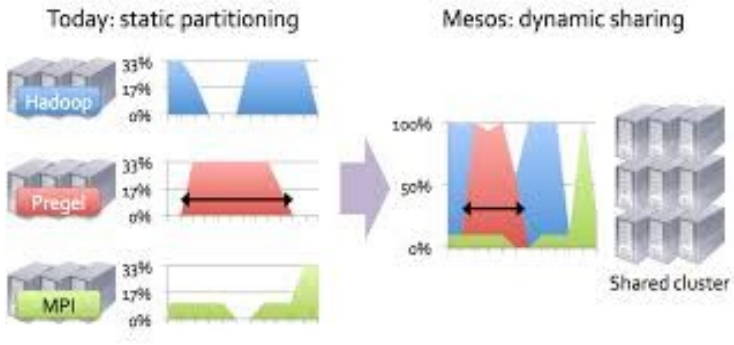
Apache Mesos

S*M*ACK Stack

- **Spark** – A general engine for large-scale data processing, enabling analytics from SQL queries to machine learning, graph analytics, and stream processing
- ***Mesos*** ***– Distributed systems kernel that provides resourcing and isolation across all the other SMACK stack components. Mesos is the foundation on which other SMACK stack components run.***
- **Akka** – A toolkit and runtime to easily create concurrent and distributed apps that are responsive to messages.
- **Cassandra** – Distributed database management system that can handle large amounts of data across servers with high availability.
- **Kafka** – A high throughput, low-latency platform for handling real-time data feeds with no data loss.

Mesos Introduction

- Developed at UC Berkeley by Benjamin Hindman et al.
- Operating system for a cluster, infact a distributed systems kernel
- 20,000 lines of C++
- Apache open source project
- Uses Linux containers for isolation
- Commercial support provided by D2IQ
- Provides dynamic sharing as opposed to static partitioning (see figure)



Mesos Goals

- High utilization of resources
- Support diverse frameworks
- Scalable to thousands of nodes
- Highly reliable

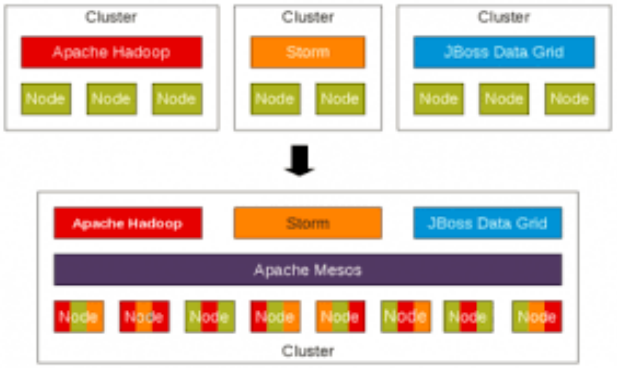
Resulting design has a small micro-kernel core that pushes scheduling logic to frameworks

Mesos As Cluster OS

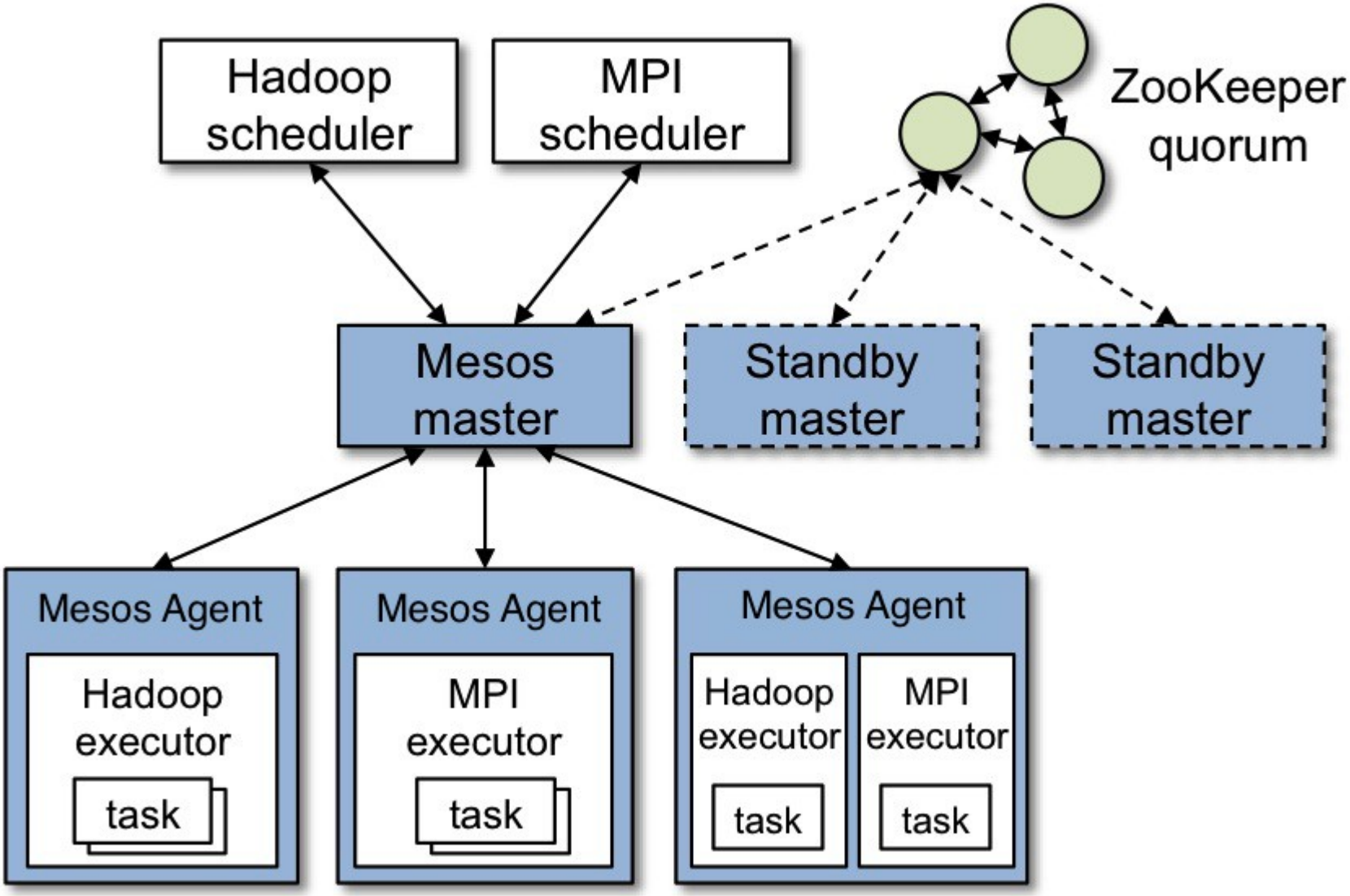
- No single cluster computing framework is optimal for all datacenter applications
- Example: Hadoop MapReduce is an example of a cluster computing framework
- Running multiple frameworks in a single cluster helps achieve following:
    - Maximum utilization of cluster resources (datacenters are expensive to build)
    - Ability to share data across frameworks (large datasets are expensive to replicate)

Mesos Features

- Performs inter-framework scheduling (fair share or priority)
- A common resource sharing layer over which diverse frameworks can run
- Level of abstraction is resources – CPU/Memory
- Delay scheduling
- Fine-grained sharing
- Data locality
- Resource offers – available resources seen by frameworks
- Uses Linux containers to isolate tasks (OS level)
- DC/OS -- open source datacenter OS built on Mesos



Mesos Architecture



Mesos Two-Level Scheduler

MESOS TWO-LEVEL SCHEDULER ARCHITECTURE

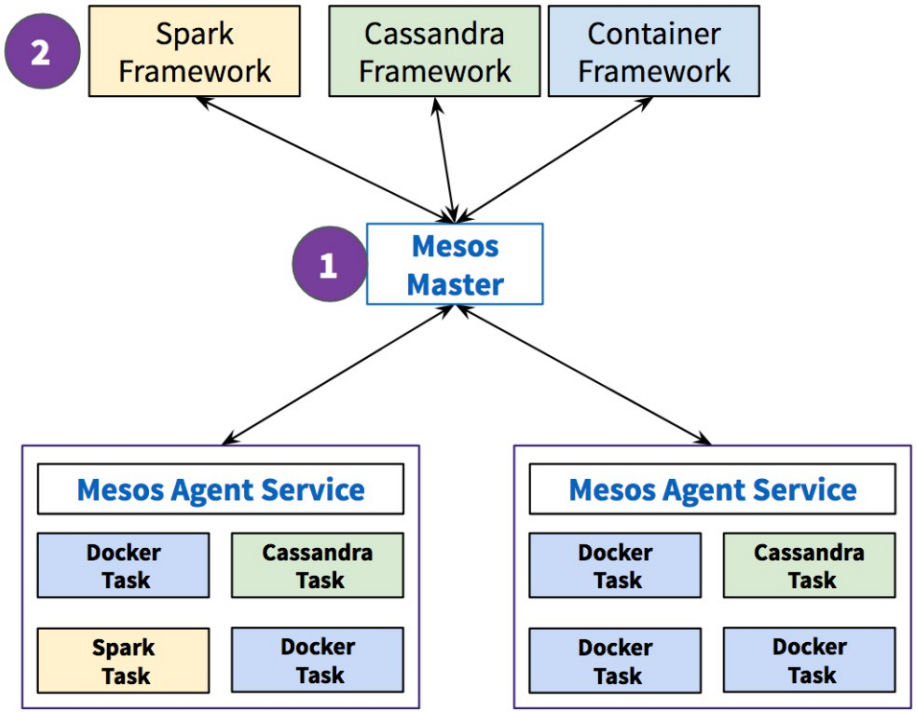
Two-level Scheduling

1 Mesos Master and Agent

- Abstracts data center resources into one pool
- Offers & tracks resources across all workloads
- Guarantees isolation
- Restarts workloads on node or task failure

2 Mesos Framework

- Consume resources
- Deploys tasks
- Provide application specific logic for deployment, recovery, upgrade..etc



Mesos Design

The master enables fine-grained sharing of resources (CPU, RAM, …) across frameworks by making them *resource offers*.

Each resource offer contains a list of <agent ID, resource1: amount1, resource2: amount2, ...>

The master decides *how many* resources to offer to each framework according to a given organizational policy, such as fair-sharing or strict priority.

To support a diverse set of policies, the master employs a modular architecture that makes it easy to add new allocation modules via a plugin mechanism.

Mesos Design

A framework running on top of Mesos consists of two components:
a *scheduler* that registers with the master to be offered resources, and
an *executor* process that is launched on agent nodes to run the framework's tasks

While the master determines **how many** resources are offered to each framework, the frameworks' schedulers select **which** of the offered resources to use.

When a framework accepts offered resources, it passes to Mesos a description of the tasks it wants to run on them. In turn, Mesos launches the tasks on the corresponding agents.

Mesos Scheduling and Allocation

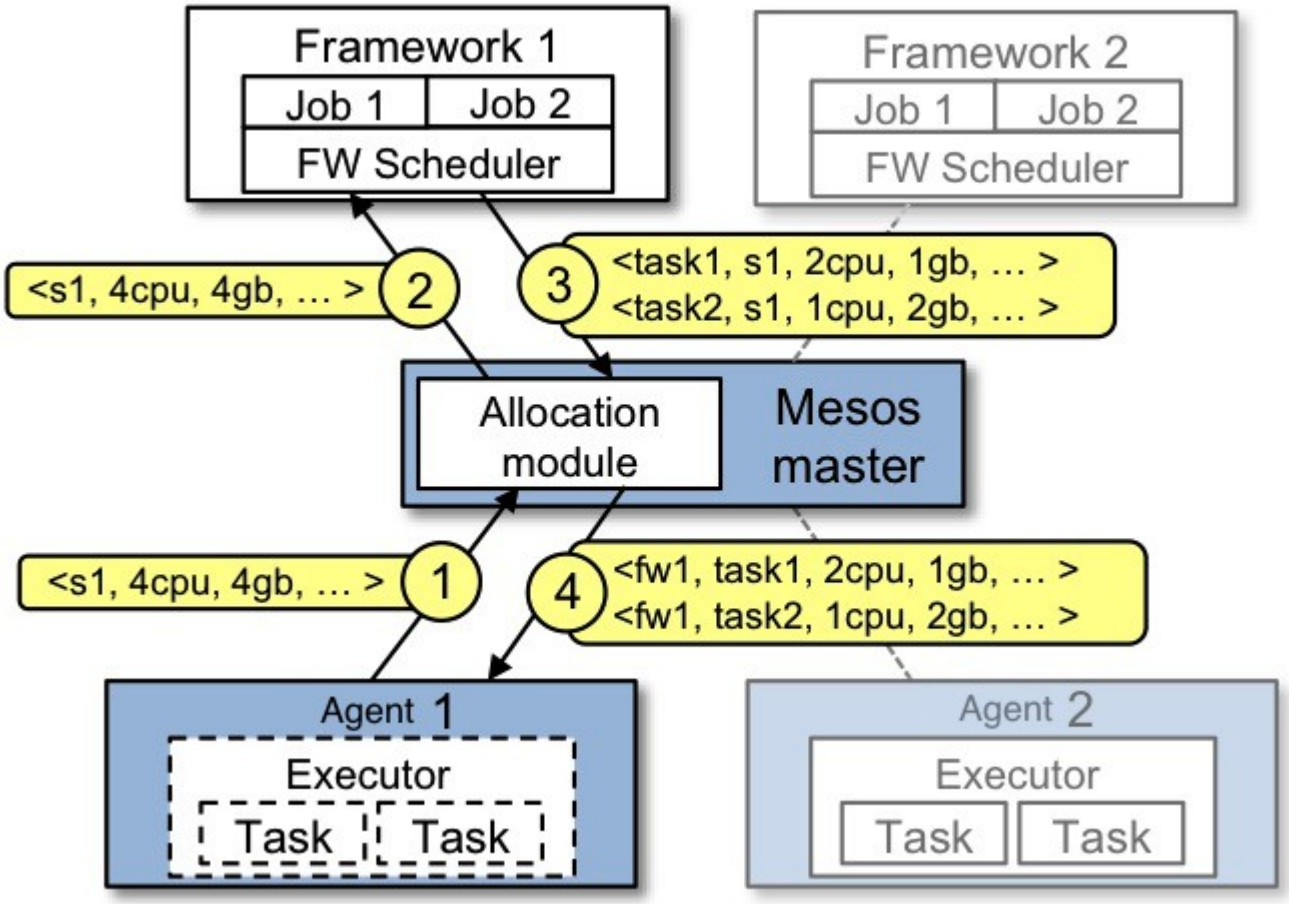
Agent 1 reports to the master that it has 4 CPUs and 4 GB of memory free. The master then invokes the allocation policy module, which tells it that framework 1 should be offered all available resources.

The master sends a resource offer describing what is available on agent 1 to framework 1.

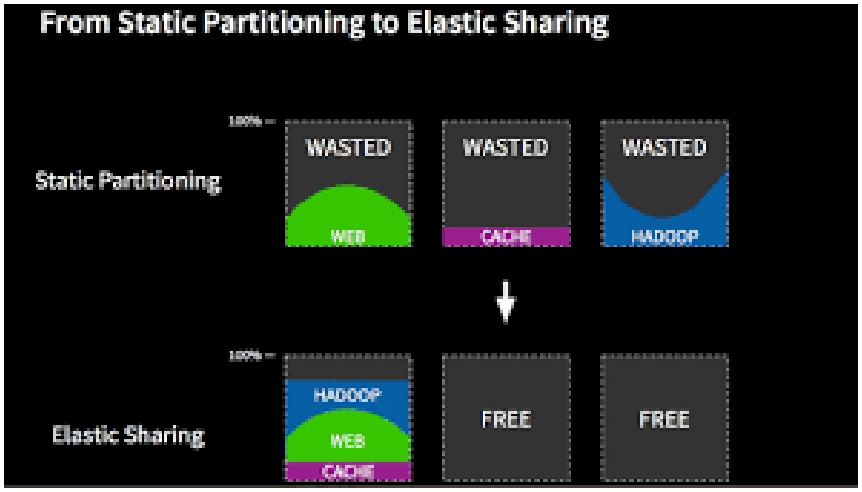
The framework's scheduler replies to the master with information about two tasks to run on the agent, using <2 CPUs, 1 GB RAM> for the first task, and <1 CPUs, 2 GB RAM> for the second task.

Finally, the master sends the tasks to the agent, which allocates appropriate resources to the framework's executor, which in turn launches the two tasks (depicted with dotted-line borders in the figure). Because 1 CPU and 1 GB of RAM are still unallocated, the allocation module may now offer them to framework 2.

Mesos Allocation Design



Dynamic Sharing vs Static



Mesos Allocation

The thin interface provided by Mesos allows it to scale and allows the frameworks to evolve independently, one question remains: how can the constraints of a framework be satisfied without Mesos knowing about these constraints?

For example, how can a framework achieve data locality without Mesos knowing which nodes store the data required by the framework? Mesos answers these questions by simply giving frameworks the ability to **reject** offers.

A framework will reject the offers that do not satisfy its constraints and accept the ones that do. In particular, a simple policy called delay scheduling, in which frameworks wait for a limited time to acquire nodes storing the input data, yields nearly optimal data locality.

<u>Offer Optimization:</u>

- Optimization to ensure offers are not rejected all the time
- Frameworks can indicate filters to say “offer nodes with > 4GB RAM” or “run on nodes with GPU” or similar

Key Takeaways

- Mesos provides dynamic sharing and allocation of resources
- Proven scalability to thousands of nodes (50,000 node cluster at Mesophere)
- Frameworks execute faster as they schedule their own tasks
- Mesos provides capability to run mixed frameworks on same cluster
- Mesos is designed to be lightweight and resilient
- Master failover using ZooKeeper (odd # of nodes recommended for availability and reliability)
- Frameworks ported: Hadoop, MPI, Torque
- Mesos has fast recovery after master failure
- Mesos supports launching tasks built on Docker
- Specialized framework: Spark for iterative jobs (up to 20x faster than Hadoop)

Resources/Further Reading

- Mesos.apache.org
- https://dcos.io/
- https://mesosphere.github.io/marathon/ (A container orchestration platform for Mesos and DC/OS)
- http://mesos.apache.org/documentation/latest/ (Documentation)
- Official white paper: https://people.eecs.berkeley.edu/~alig/papers/mesos.pdf
- https://d2iq.com/products/dcos (Commercial support)