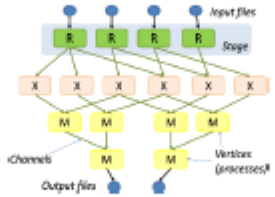
Mesos:
A Platform for Fine-Grained Resource Sharing in the Data Center

Jayashankar .T

Agenda

- Motivation & Problem Statement
- Design
- Architecture
- Scheduling Resource Offer
- Fault Tolerance
- Evaluation
- Comparison

Motivation



Dryad

Google
Pregel

- Many Cluster Compute Frameworks are available today
- Single framework do not suffice all applications

Cluster: a “Precious” Resource

One Cluster to Rule Them All !!

Typical Problem

- Facebook's Hadoop data warehouse
  - 2000 nodes cluster
  - Fair scheduler for Hadoop
  - Workloads are fine-grained, so task level resource allocation
  - Optimum data locality
- Only runs Hadoop ☹
- Can it run other frameworks fairly and efficiently ?

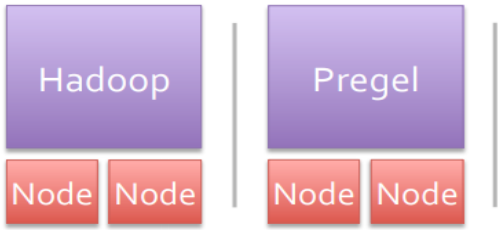
What do we want?

- We want to run multiple frameworks on our cluster

- Sharing improves cluster utilization:
  1. Applications share access to large datasets
  2. Costly to replicate across distinct nodes

Common Cluster Sharing Solutions

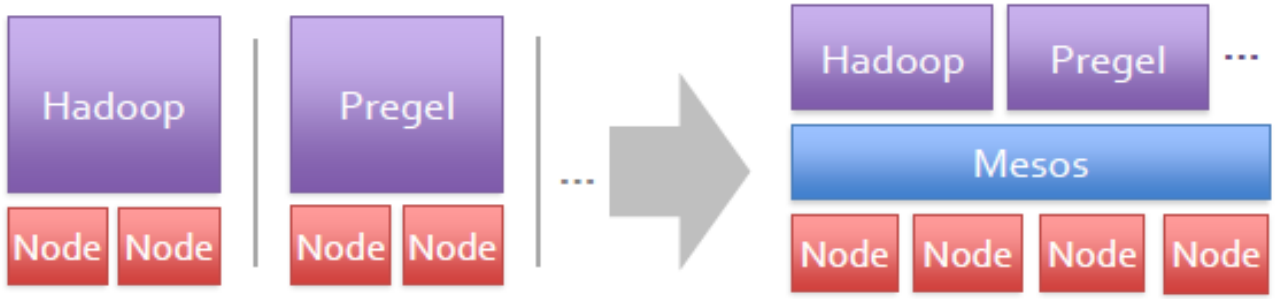
- Static Partitioning: run one framework per partition



- Assign VMs to each framework

- Concerns:
  - Non optimal cluster utilization
  - Inefficient data sharing (e.g. unnecessary replication)

Mesos

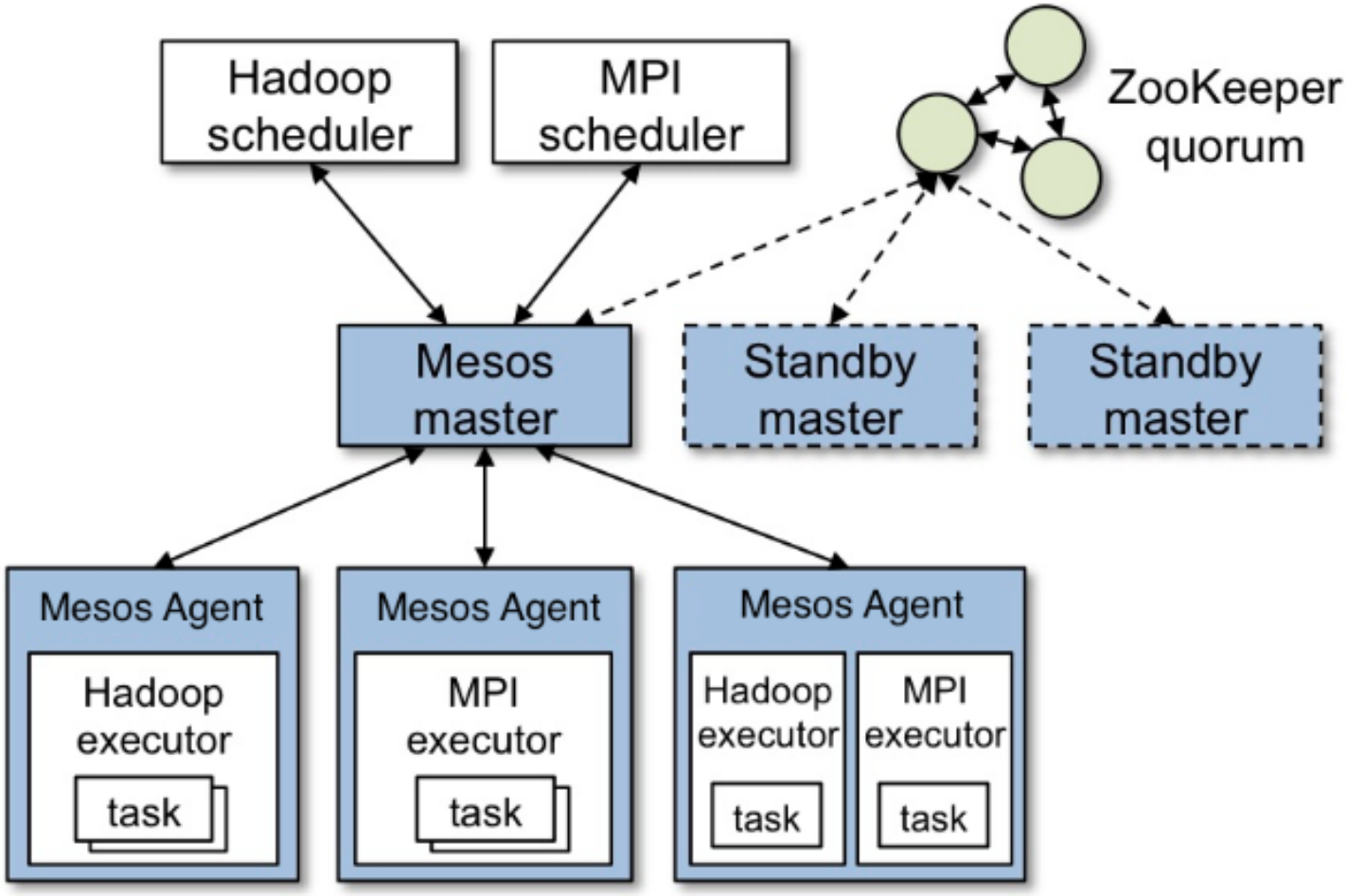


- Platform for sharing clusters between multiple computing frameworks
- Can run multiple instances of same framework
  - Provide isolation between production and development environment
  - Concurrently running several frameworks
- Support any new specialized frameworks
- Be scalable and reliable at the same time

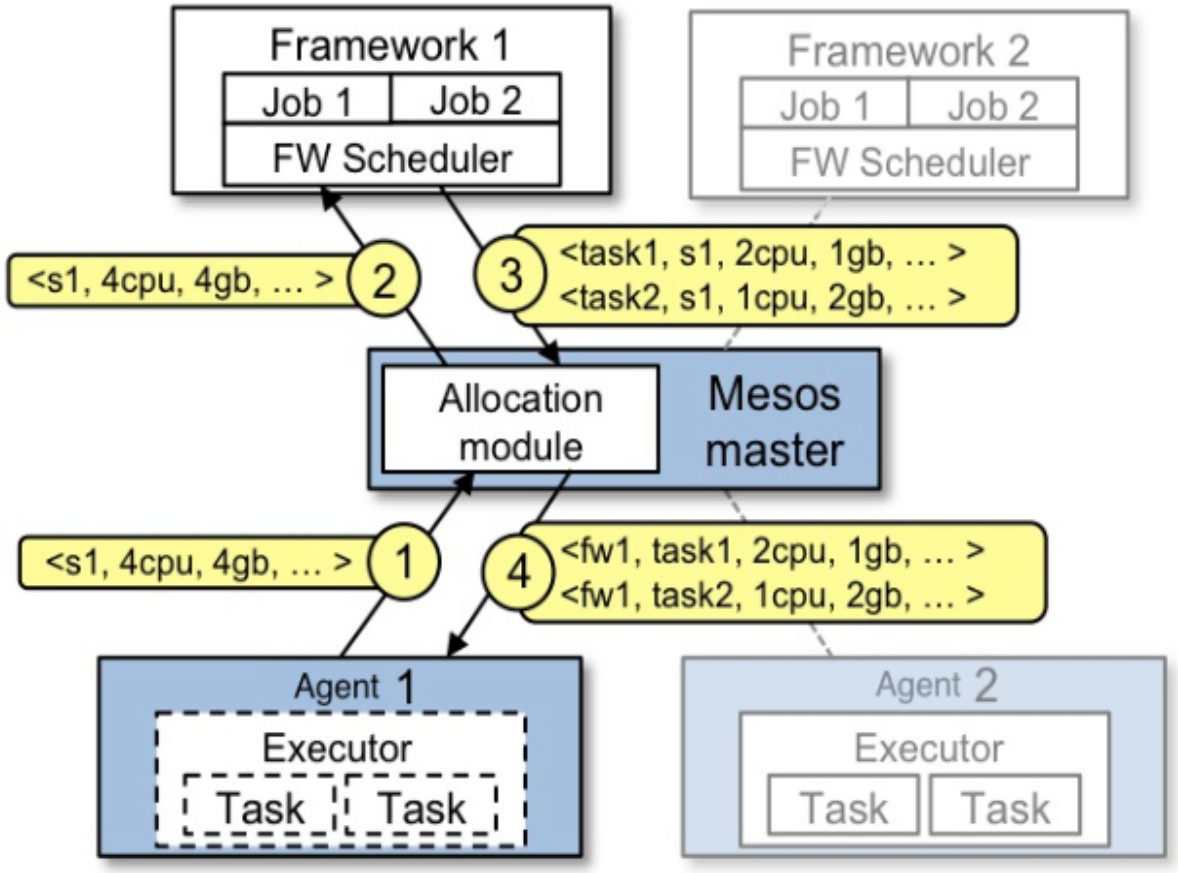
Mesos Design

- Provide minimal interface for resource sharing across frameworks
- Offload task scheduling and execution onto frameworks
- Thus,
  - Frameworks have the liberty to implement diverse solutions to problems
  - Keeping Mesos Simple, becomes robust, scalable, manageable and stable
- Although expectation is to have high-level libraries on top Mesos for fault tolerance (keeping Mesos small & flexible)

Mesos Architecture



Resource Offer



- Allocator on Master and Executor on Slave
- Step1: slave provide resource info
- Step2: offer made to framework
- Step3: Framework presents task
- Steps4: Master sends task to slaves

Resource Offer

- Mesos doesn't require frameworks to specify their requirements
- Frameworks can reject the offer, if it does not stratify constraints and can decide to wait
- To prevent framework from waiting too long, frameworks can set filters
  - Example: will never accept offer with less than 8G memory
- Filters optimize offer model

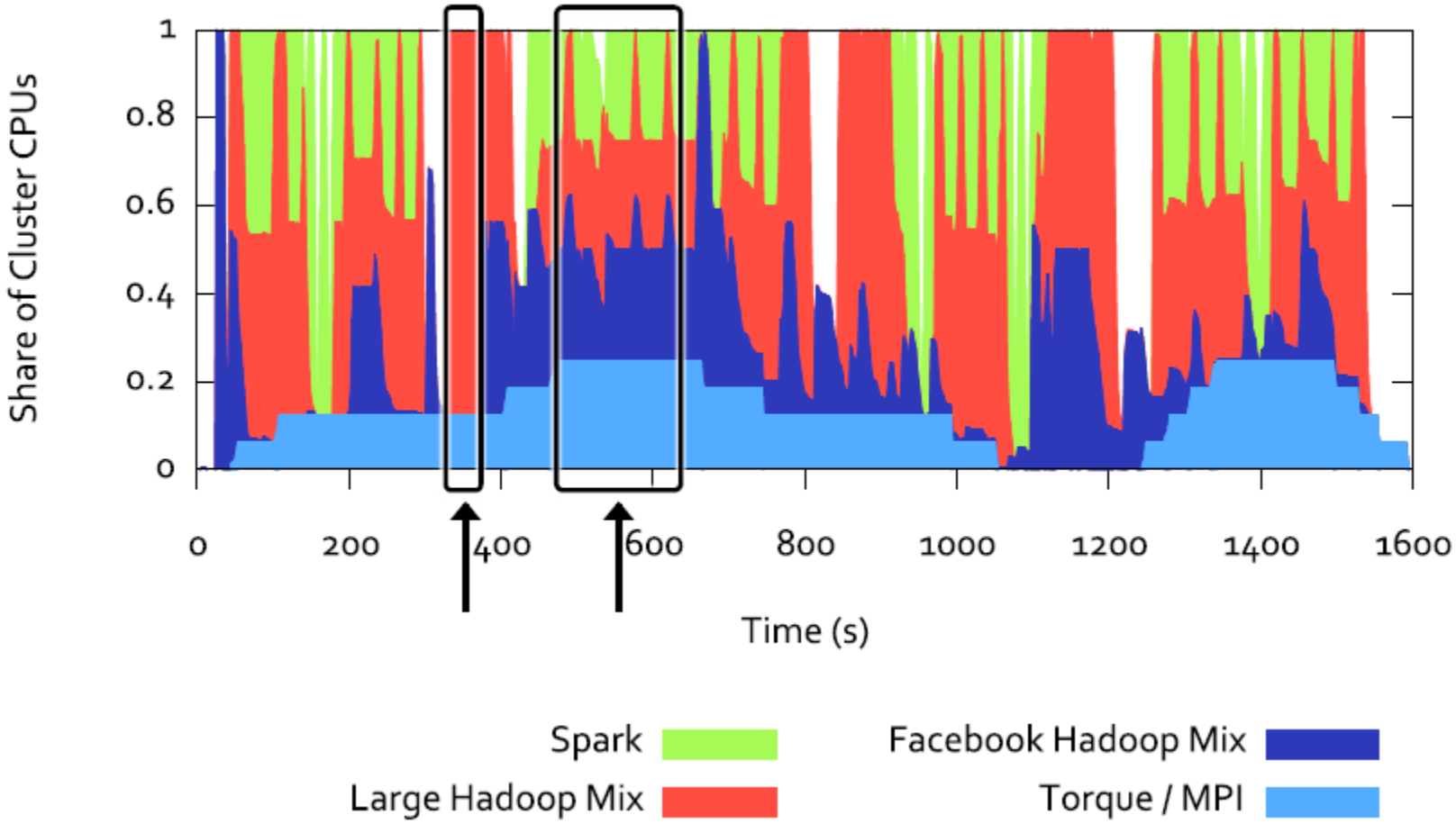
Mesos Characteristics

- Filter can be directly provided at master to short circuit offer process
  - Resource offered is Resource allocated
  - Every offer has timeout for acceptance – Master rescinds the offer after that
- Pluggable Allocation Module, support for flexible allocation policy
  - Fair sharing policy: Frameworks with Small Tasks wait less
  - Strict Priorities
  - Guaranteed Allocation: task revocation wont happen for certain frameworks (interdependent like MPI)
- Isolation is achieved through OS container

Fault Tolerance

- Master has to be fault tolerant:
  - Master is designed to be soft state, new master can reconstruct internal state from slaves and framework schedulers
  - Master stores: active slaves, active frameworks and running tasks
- Multiple masters run in hot standby and Zookeepers is used for leader election
- Node and executor failure are reported to framework, to be taken care
- Scheduler failure is overcome with framework registering multiple schedulers for redundancy

Resource Sharing

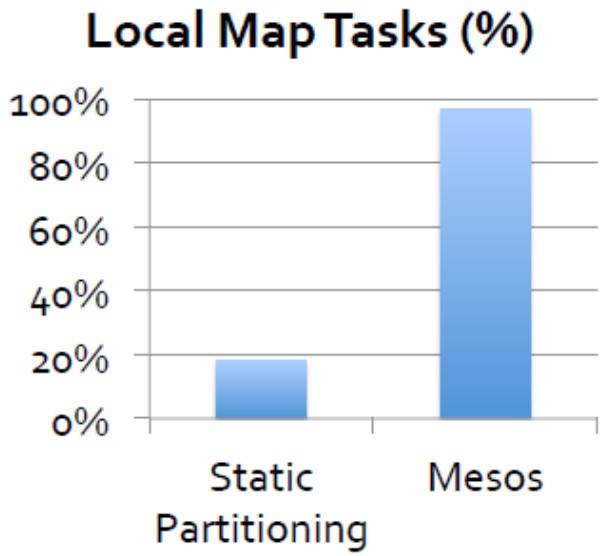


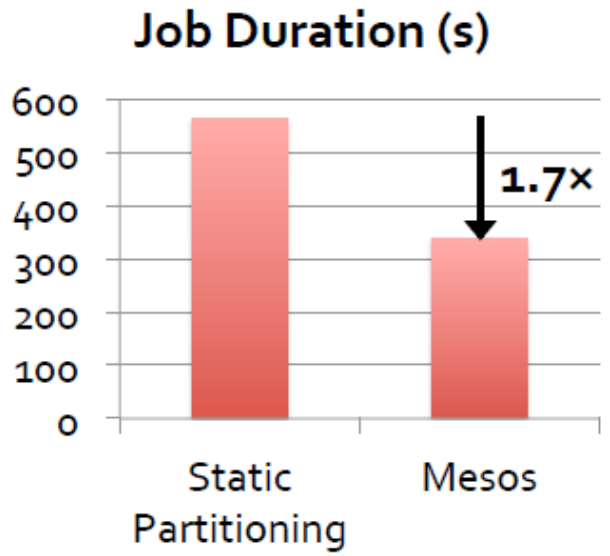
Data Locality with Resource Offers

- Mesos use *"delay scheduling"*: wait for limited time for specific local nodes else continue

Ran 16 instances of Hadoop on a shared HDFS cluster

Used delay scheduling [EuroSys '10] in Hadoop to get locality (wait a short time to acquire data-local nodes)





Scalability

Mesos only performs *inter-framework* scheduling (e.g. fair sharing), which is easier than intra-framework scheduling

Result:
Scaled to 50,000 emulated slaves,
200 frameworks,
100K tasks (30s len)

Limitations and Overcoming them

- Starvation of large tasked frameworks
  - Allocation modules support a minimum offer size on each slave, and abstain from offering resources on the slave until this amount is free
- Interdependent Frameworks: framework using data generated by other
  - Such scenarios are rare in practice.
  - frameworks only have preferences over which nodes they use, and can have filters for specific nodes
- Complex Frameworks: schedulers have to be smart to judge resource offers
  - Job type and time can not be predicted to have a centralized scheduler

Mesos v Borg

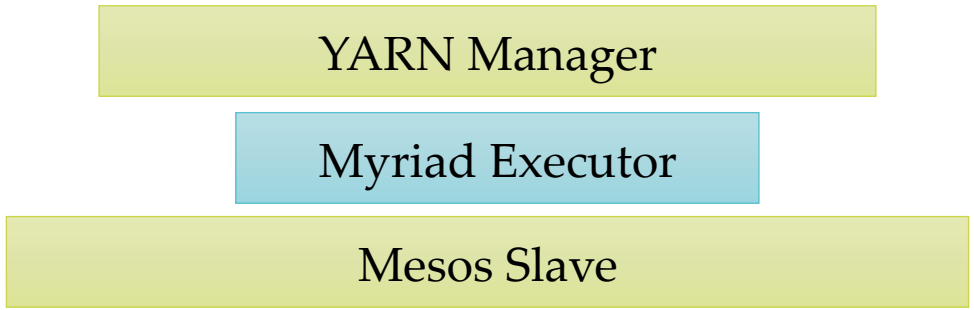
- Less Control and Simple
- Very less start up overhead
- Frameworks have to be modified to support Mesos

- Complex but Better Control
- More Start up Latency
- Framework/Applications need be changed much

"Mesos = Borg – Scheduling"

Mesos v YARN

- YARN makes the decision where jobs should go,
- Thus it is modeled as a monolithic scheduler.
- Running YARN over Mesos: Project 



References

- MESOS Project

  http://mesos.apache.org/documentation/latest/

- USENIX Video

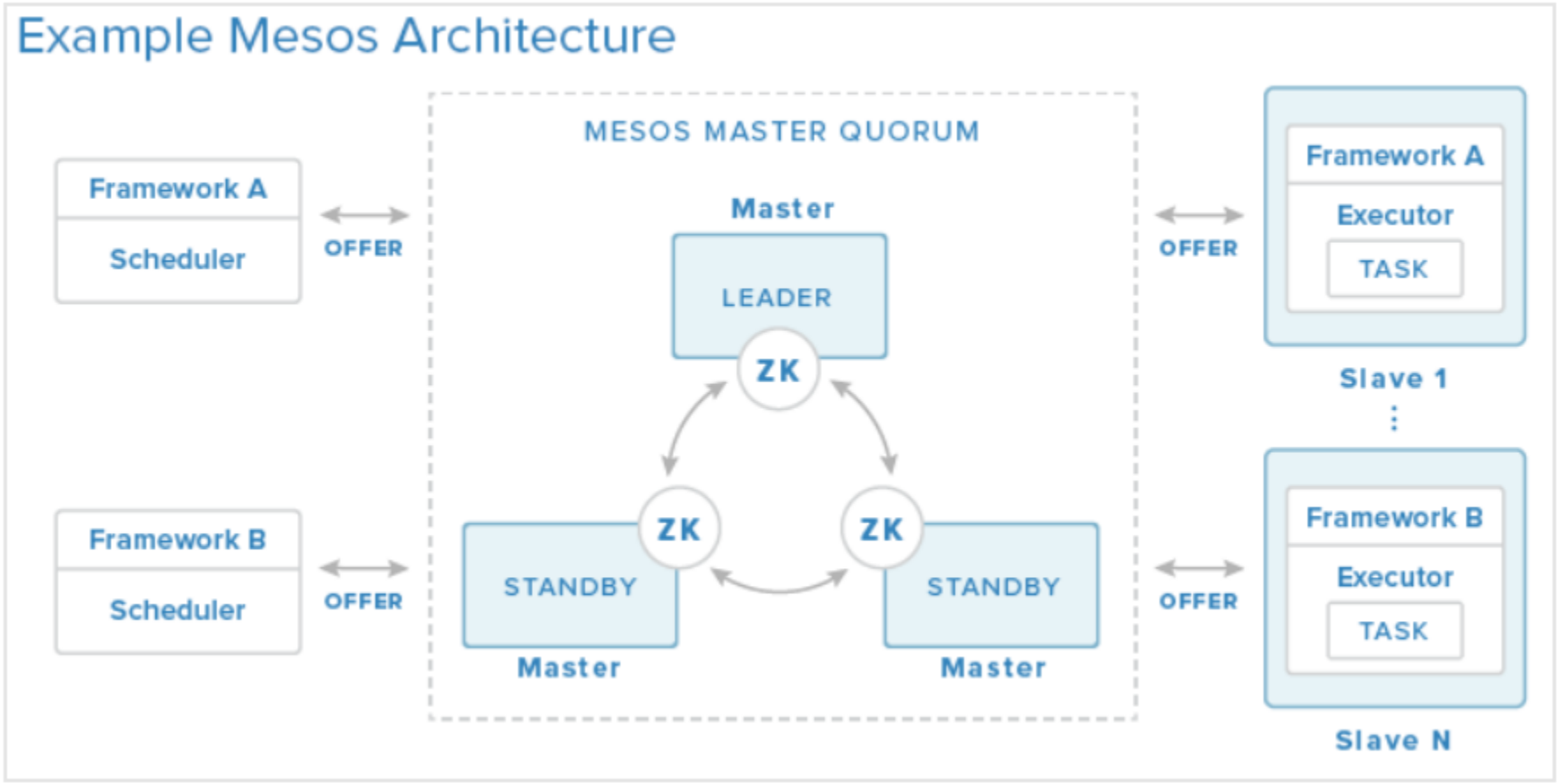
  https://www.usenix.org/conference/nsdi11/mesos-platform-fine-grained-resource-sharing-data-center

Additional slides

Centralized v Distributed Scheduling

	Centralized	Distributed
Workload heterogeneity	✓	
Task placement	✓	
Enforcing scheduling invariants	✓	
Allocation latency		✓
Slot utilization		✓
Scalability		✓

Mesos Architecture



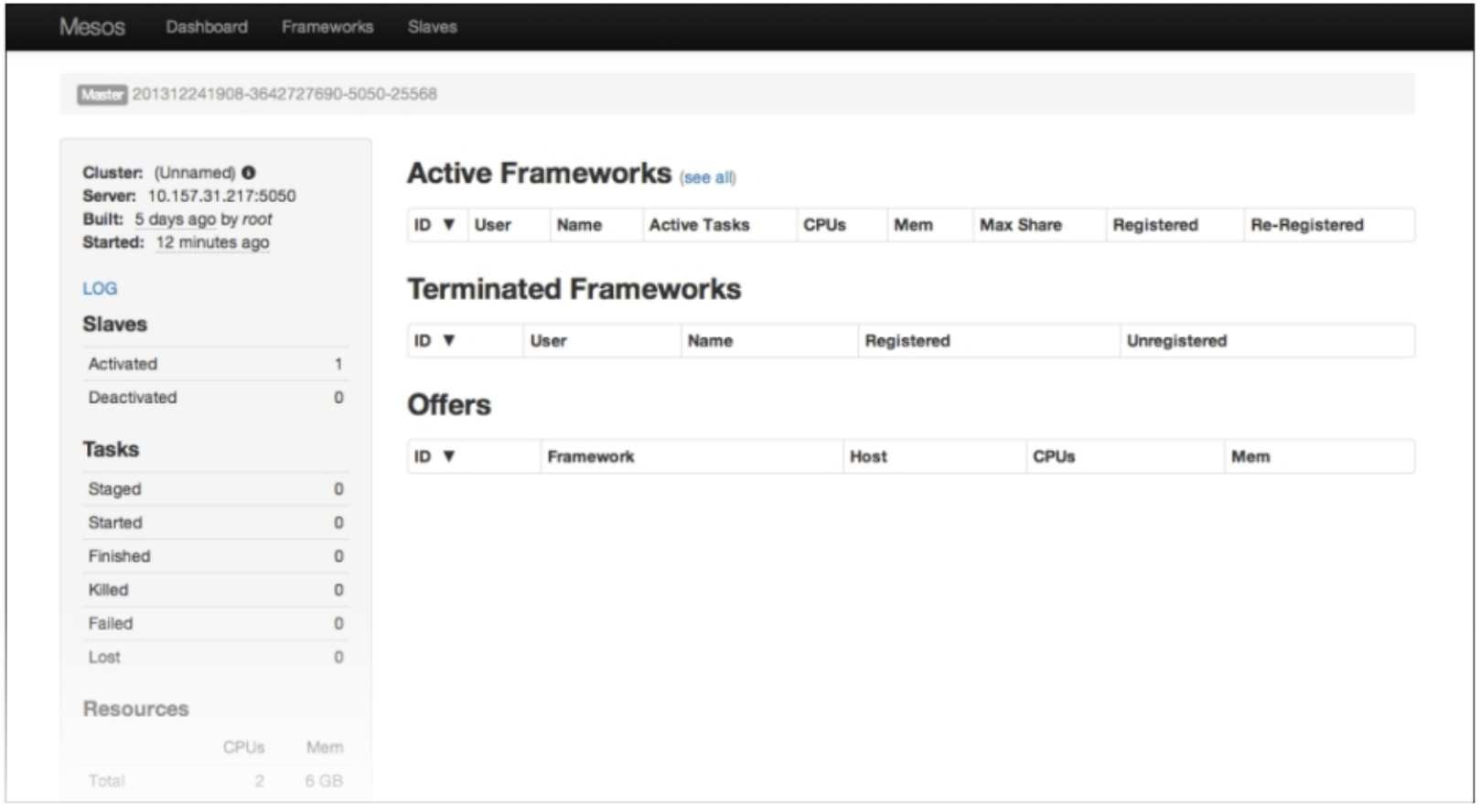
Mesos APIs

Scheduler Callbacks	Scheduler Actions
resourceOffer(offerId, offers) offerRescinded(offerId) statusUpdate(taskId, status) slaveLost(slaveId)	replyToOffer(offerId, tasks) setNeedsOffers(bool) setFilters(filters) getGuaranteedShare() killTask(taskId)

Executor Callbacks	Executor Actions
launchTask(taskDescriptor) killTask(taskId)	sendStatus(taskId, status)

Mesos Web UI

Mesos provides a web UI for reporting information about the Mesos cluster. It can be accessed from `<master-host>:<port>` ; in our case, this will be `http://master:5050` . This includes the slaves, aggregated resources, frameworks, and so on. Here is the screenshot of the web interface:



Mesos web interface

Mesos Ecosystem

- Mesosphere – DC/OS: datacenter operating system
- Mesosphere – Marathon: container management system
- Airbnb -- Chronos: scheduler for Mesos, eases the orchestration of jobs