Introduction to MESOS

By: **Morteza Zakeri**

Cluster and Grid Computing Course

Instructor: Dr. Mohsen Sharifi

School of Computer Engineering

Iran University of Science and Technology

Winter 2017

Outline

- Cluster Computing
  - Applications
  - Backgrounds and Terminology
  - Problems
- Sharing Frameworks
- Mesos
  - Goals to Design
  - Key Elements
  - Architecture

Outline

- Resource Offer
- Implementation and API
- Evaluation (Benchmarks)
- Limitations
- Related Works
- Conclusion
- References

Cluster computing main applications

- High Performance Computing (HPC)
  - ➢Large internet services
    - *E.g. social networks, online shopping, …*
  - ➢Data-Intensive scientific applications
    - *E.g. physics,* astronomy, *molecular sciences, …*

M4 globular cluster

Terminology

- Cluster
  - A collection of distributed machines that collaborate to present a single integrated computing resource to the user.
- Node
  - Each computing element (single machine itself) within cluster.
- Framework
  - A software system that manages and executes one or more jobs on a cluster.

Terminology

- Job
  - A unit of work run on the cluster nodes.
- *Slot*
  - *Nodes are subdivided into "slots".*
  - *(In some framework such as Hadoop and Dryad)*
- *Task*
  - *Jobs are composed of short tasks that are matched to slots.*
  - *(In some framework such as Hadoop and Dryad)*

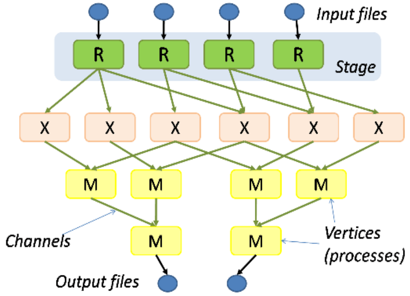
Terminology

- *Elastic* framework
  - Can scale its resources up and down during execution tasks (dynamically).
  - *Such as Hadoop and Dryad.*
- *Rigid* framework
  - Can start running its jobs only after it has acquired a fixed quantity of resources.
  - Can not scale (up/down) dynamically.
  - *Such as MPI.*

Background

- Diverse array of cluster computing frameworks:
  - Hadoop/ Map-Reduce (multi-terabyte data-sets)
  - Dryad (machine learning)
  - Pregel (graph computation)
  - MPI
  - …

Pregel



Dryad

Problem

- Rapid innovation in cluster computing frameworks.
  - ➢ There is no single framework that optimal for all applications.
  - ➢ Clusters typically run single framework at a time.

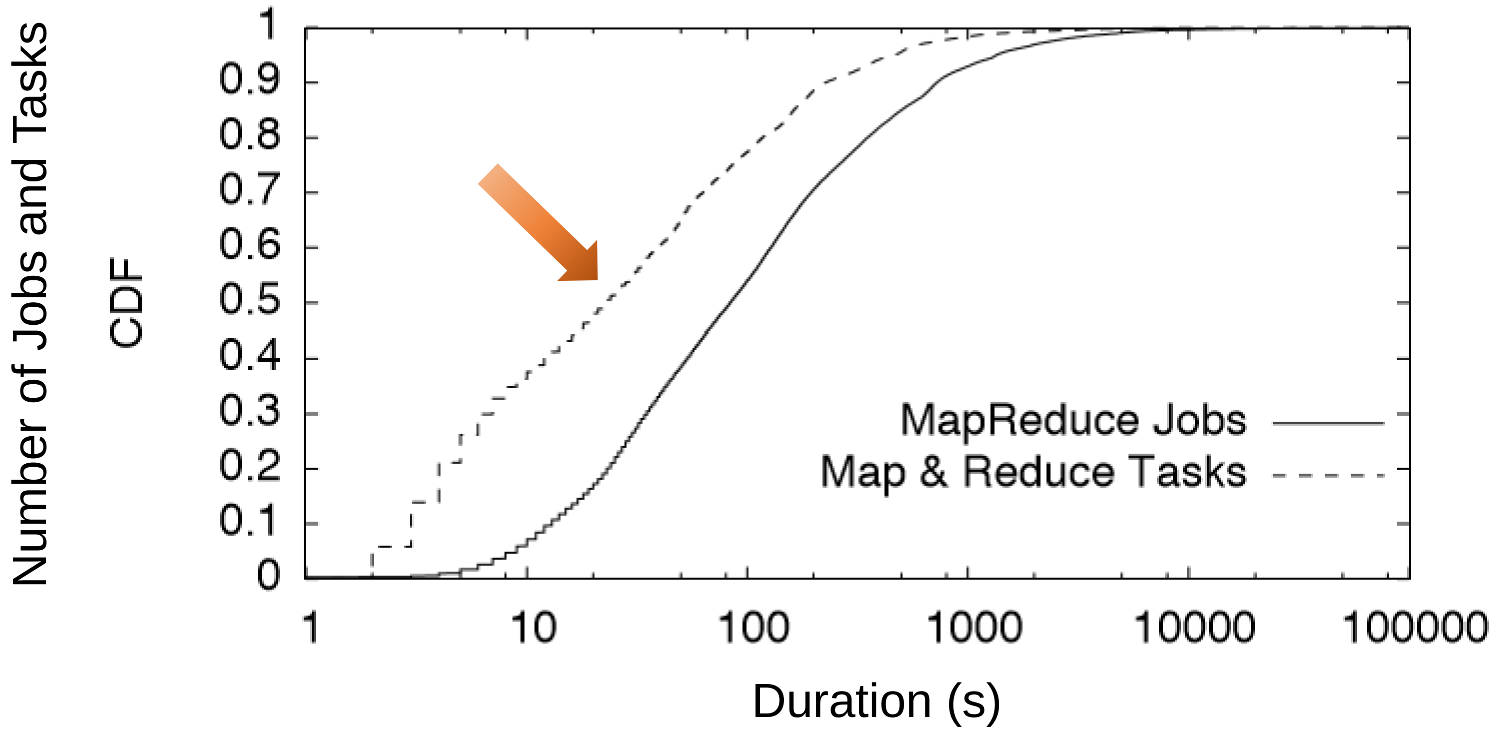
Problem

- Hadoop data warehouse at **Facebook:**
  - Include 2000-node Hadoop cluster,
  - Uses a **fair scheduler** for Hadoop,
  - Takes advantage of the **fine-grained** nature of the workload,
  - Allocate resources at the level of tasks and to optimize data locality.

  ↓

  - This cluster can only run Hadoop jobs!
- Can it run an MPI program efficiently?

Problem



CDF of job and task durations in Facebook’s Hadoop data warehouse

Motivation to Sharing

- We want to run multiple frameworks in a single cluster.
- Sharing improves cluster utilization,
- Allows applications to share access to large datasets.
  - may be too costly to replicate across distinct clusters!

Common solutions for sharing a cluster

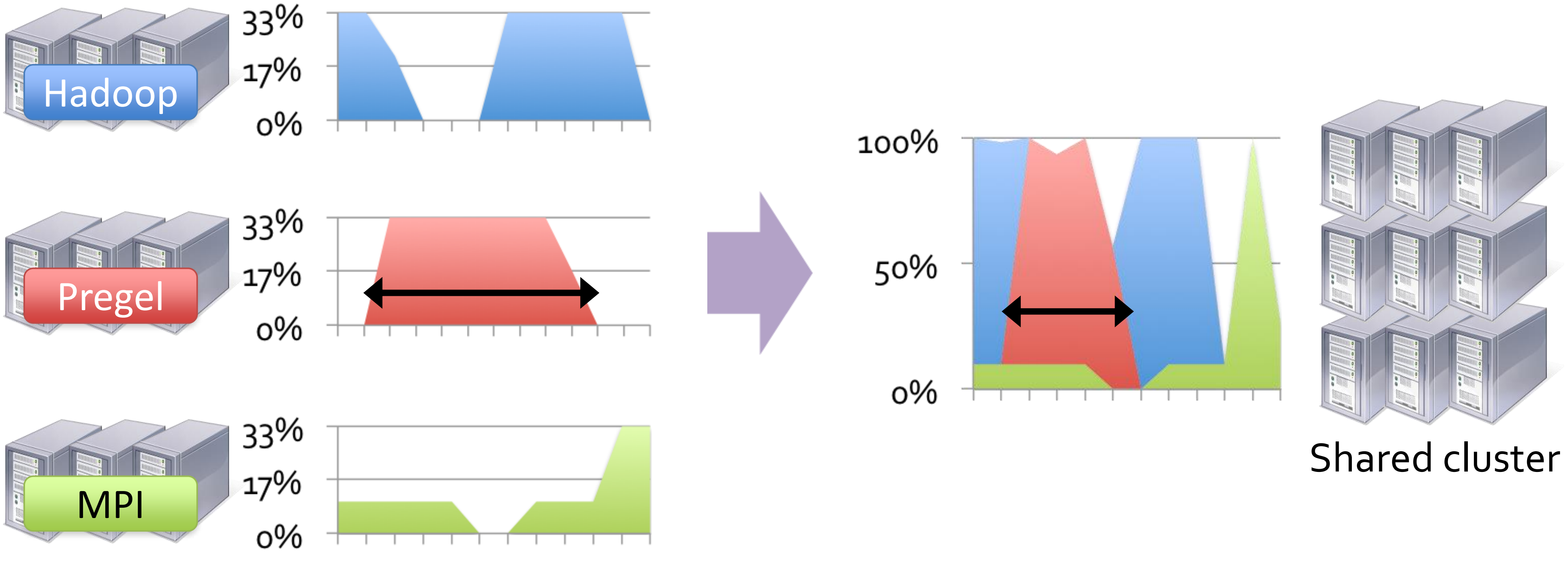
1. **Statically partition** the cluster and run one framework per partition.
2. **Allocate a set of VMs** to each framework.

- Disadvantages:
  - No high utilization,
  - No efficient data sharing.

Better solution: Mesos!

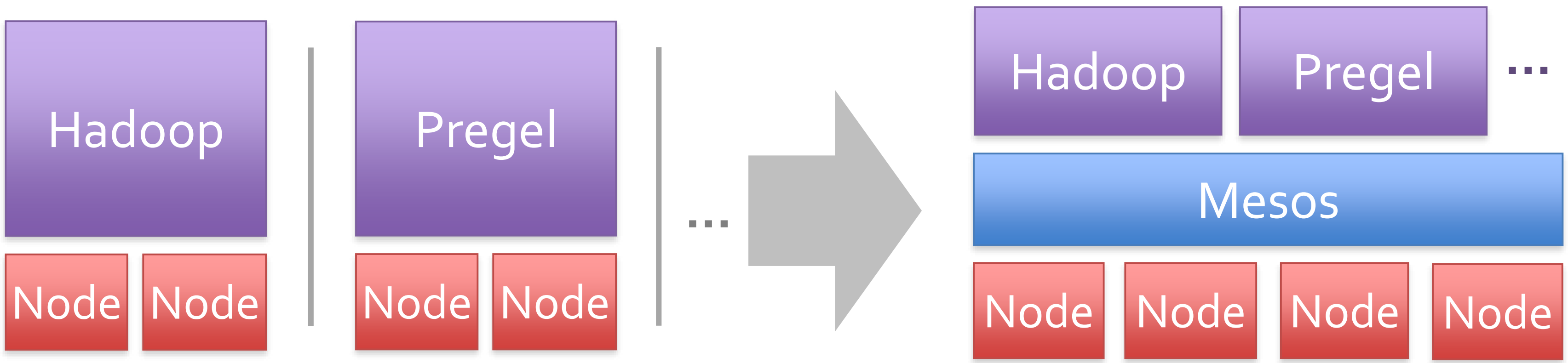
Today: static partitioning

Mesos: dynamic sharing



What is Mesos?

- A platform for sharing clusters between multiple diverse cluster computing frameworks.
  - E.g. Hadoop + Pregel + MPI + …



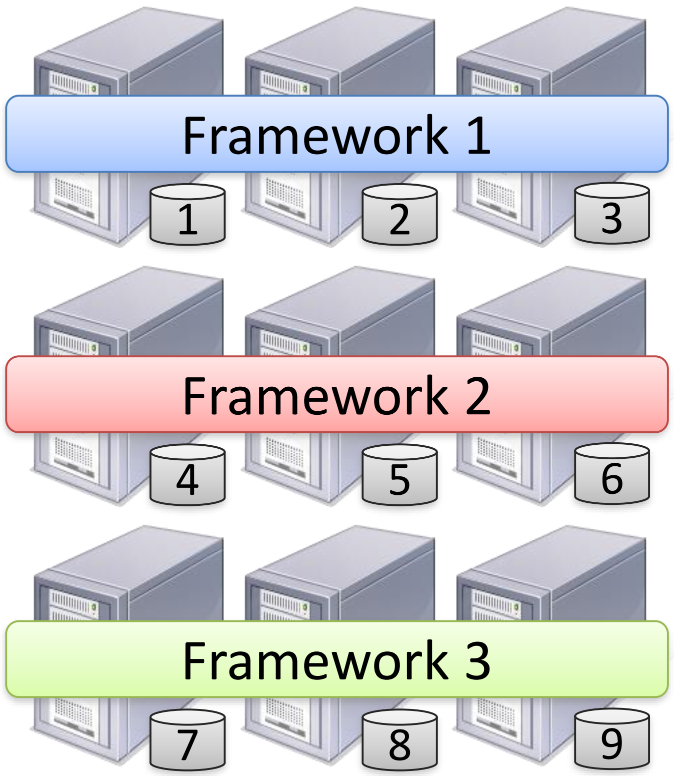
Goals to Design Mesos

- **High utilization**
- **Efficient data sharing**
- **Support diverse frameworks** (current and future)
- **Scalability** to 10,000's of nodes
- **Reliability** in face of failures (robust to failures)

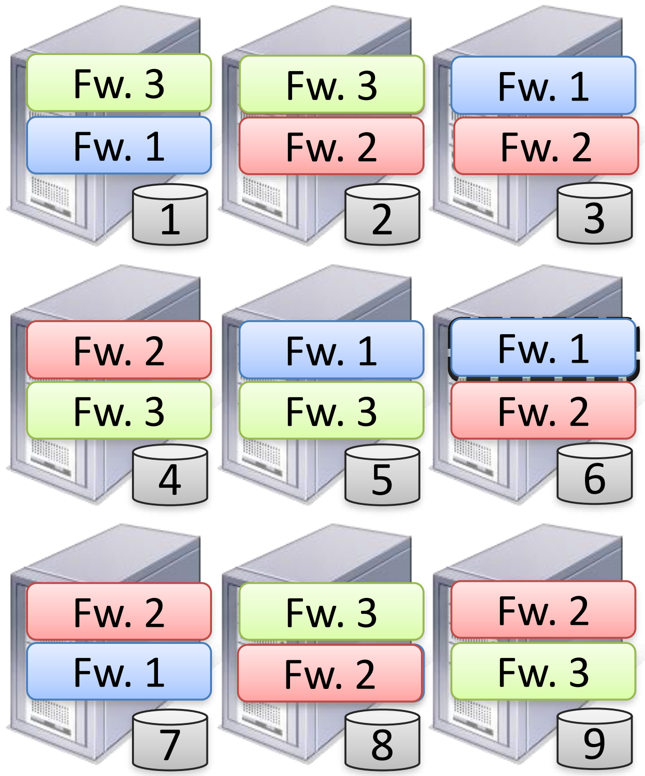
Design Philosophy

- Small microkernel-like core
- Implements **fine-grained sharing**
  - Allocation at the level of tasks within a job
- Simple, scalable application-controlled **scheduling mechanism**.

Fine-Grained Sharing



Storage System (e.g. HDFS)



Storage System (e.g. HDFS)

Scheduling Mechanism: Challenges

- How to build a **scalable** and **efficient** system that supports a wide array of both **current** and **future** frameworks?

1. Each framework have different scheduling needs.
2. Scheduling system must scale to large number of cluster nodes.
   - Running hundreds of jobs with millions of tasks.
3. System must be fault-tolerant and highly available.

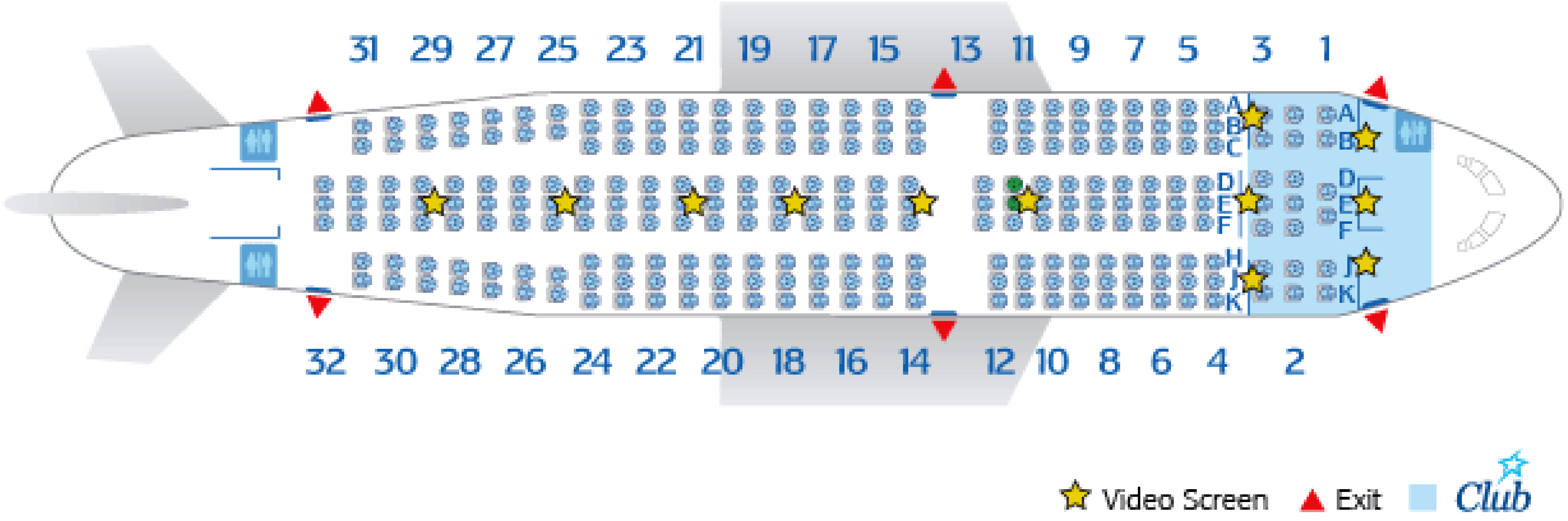
Scheduling Mechanism: Approaches

- **Centralized Scheduling**
  - Frameworks express needs in a specification language,
  - Global scheduler matches them to resources.
- \+ Can make optimal decisions.
- – Complexity,
- – Difficult to scale and to make robust,
- – Future frameworks may have unanticipated needs.

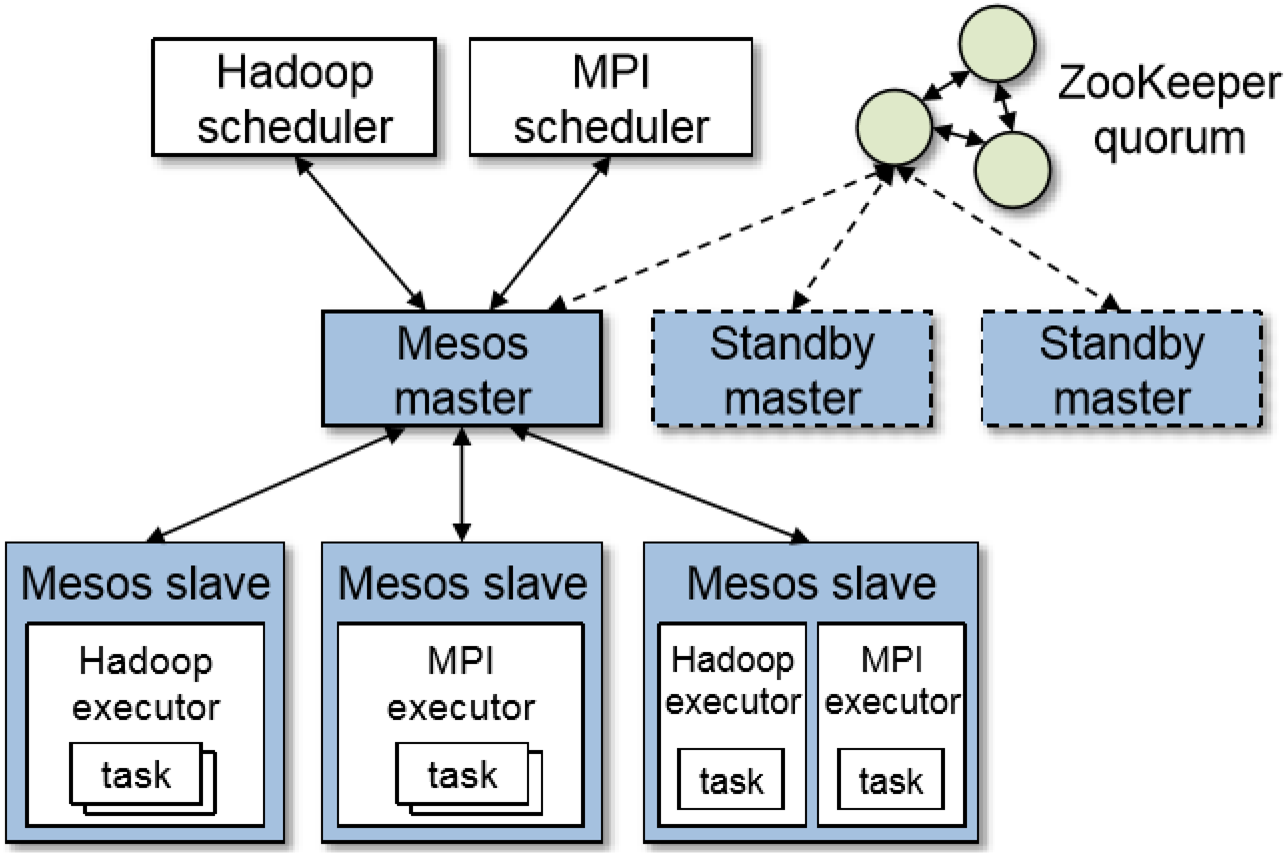
Scheduling Mechanism: Approaches

- Mesos Approach: **Resource Offer**
  - Offer available resources to frameworks,
  - Delegating control over scheduling to them.
- \+ Keeps Mesos simple, lets it support future frameworks,
- \- Decentralized decisions might not be optimal!

Resource Offer



Mesos Architecture



Mesos architecture diagram, showing two running frameworks (Hadoop and MPI).

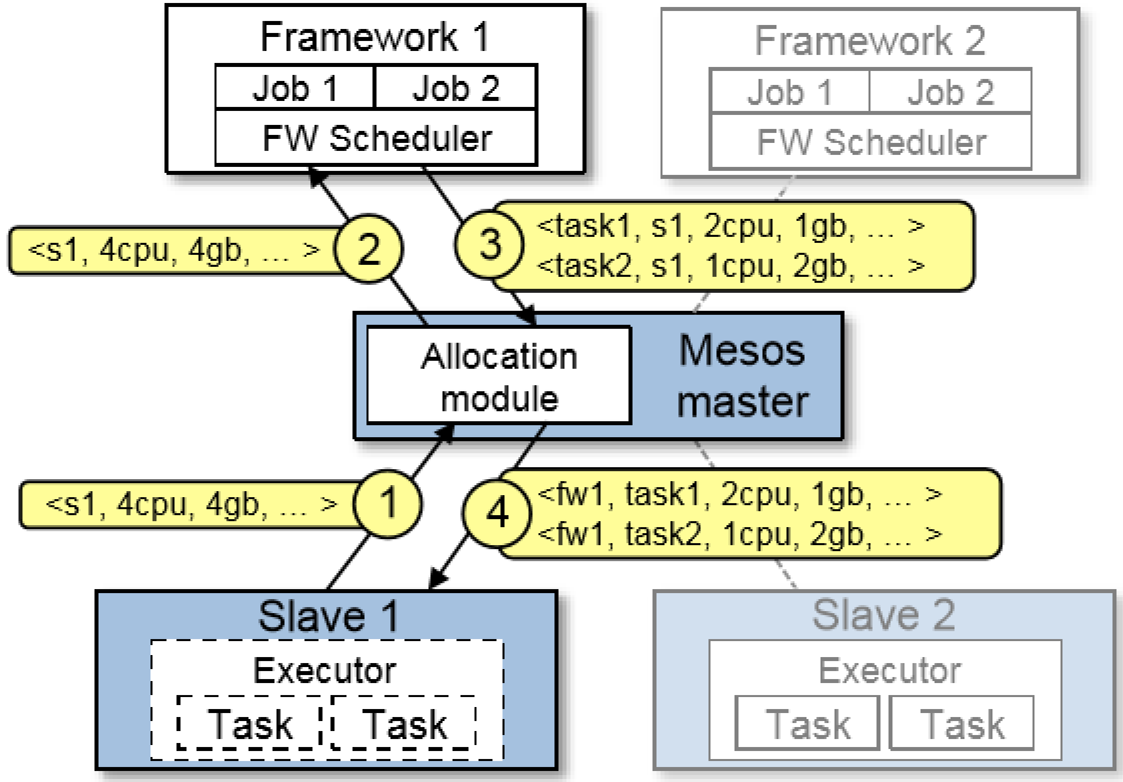
Mesos Architecture: Resource Offer

- The **master** implements fine-grained sharing across frameworks using resource offers.
- Each resource offer is a list of free resources on multiple slaves.
- The master decides **how many** resources to offer to each framework according to an organizational policy.
- Mesos lets organizations define their own policies via a pluggable allocation module.

Mesos Architecture: Resource Offer

- Each framework running on Mesos consists of two components:
  - A scheduler that registers with the **master** to be offered resources.
  - An executor process that is launched on slave nodes to run the framework's tasks.
- **Frameworks' schedulers** select which of the offered resources to use.

Resource Offer: Example



Optimization: Filters

- Let frameworks short-circuit rejection by providing a predicate on resources to be offered
  - E.g. “nodes from list L” or “nodes with > 8 GB RAM”
  - Could generalize to other hints as well

Isolation

- Isolate resources using OS container technologies
  - Linux Containers (LXC)
  - Solaris Projects
- Limit the CPU, memory, network bandwidth, and (in new Linux kernels) I/O usage of a process tree.
- Not perfect, but much better than no isolation!

Fault Tolerance

- Master’s only state is the list of active slaves, active frameworks, and running tasks.
- Run multiple masters in a **hot-standby** configuration
  - Using ZooKeeper.

- To deal with scheduler failures, Mesos allows a framework to register multiple schedulers.
  - when one fails, another one is notified by the Mesos master to take over.

Implementation

- About 10,000 lines of C++ code.
- Runs on Linux, Solaris and OS X.
- Frameworks ported: **Apache Hadoop**, **MPICH2**, **Torque**
- New specialized framework: Spark
  - for iterative jobs
  - up to **20×** faster than Hadoop
- & Open Source :)

Organizations using Mesos

- **Twitter** uses Mesos on > 100 nodes to run ~12 production services (mostly stream processing).
- **UCSF** medical researchers are using Mesos to run Hadoop and eventually non-Hadoop apps.
- **Berkeley** machine learning researchers are running several algorithms at scale on Spark.
- Read more at:
  - http://mesos.apache.org/documentation/latest/powered-by-mesos/

Mesos API

Scheduler Callbacks

resourceOffer(offerId, offers)
offerRescinded(offerId)
statusUpdate(taskId, status)
slaveLost(slaveId)

Scheduler Actions

replyToOffer(offerId, tasks)
setNeedsOffers(bool)
setFilters(filters)
getGuaranteedShare()
killTask(taskId)

Executor Callbacks

launchTask(taskDescriptor)
killTask(taskId)

Executor Actions

sendStatus(taskId, status)

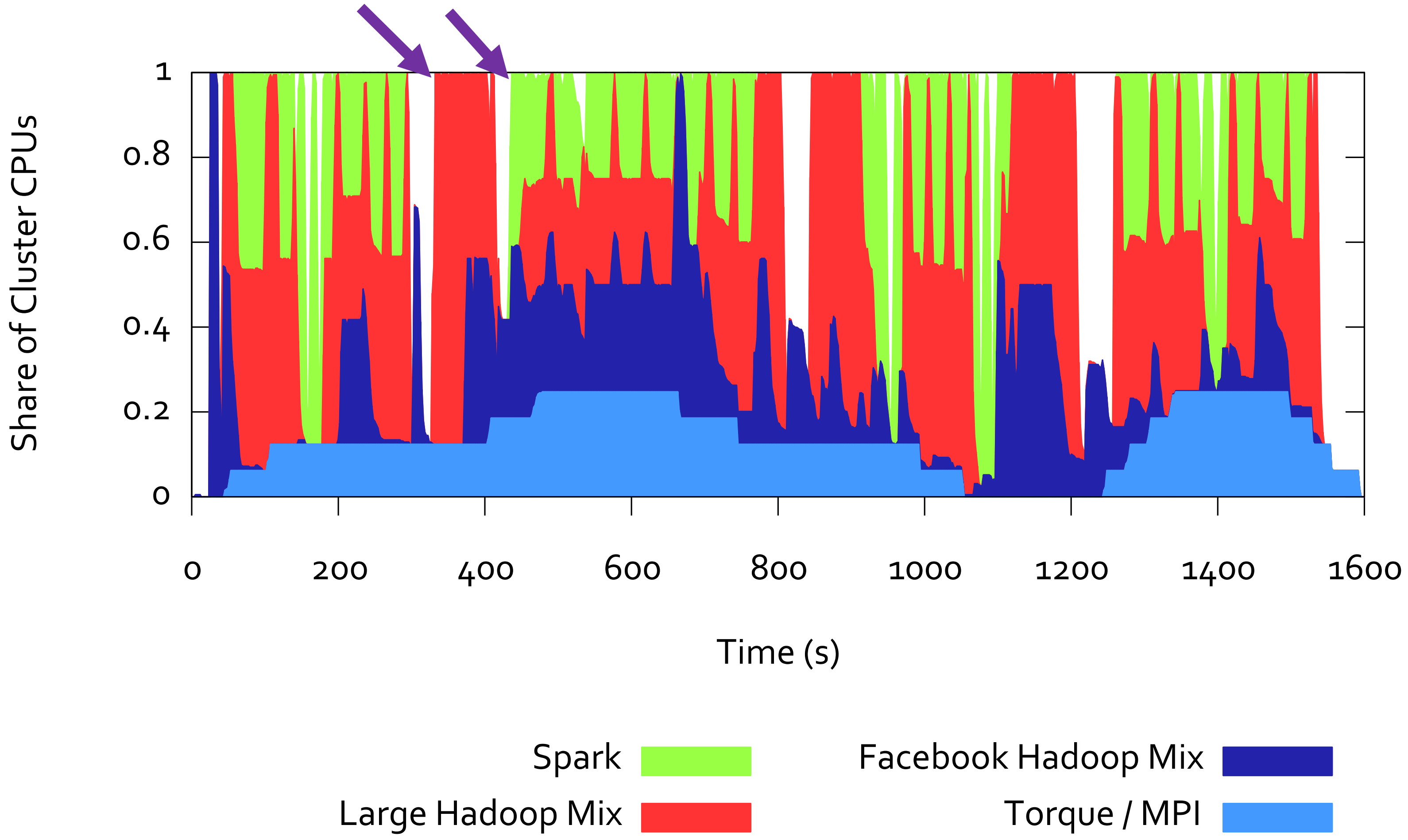
Evaluation

- Mesos is evaluated through a series of experiments on the Amazon Elastic Compute Cloud (EC2).
  - 96 Nodes
  - Nodes with 4 CPU cores and 15 GB of RAM.
- Four workloads in two scenarios:
  - ➢ 96-node Mesos cluster using fair sharing

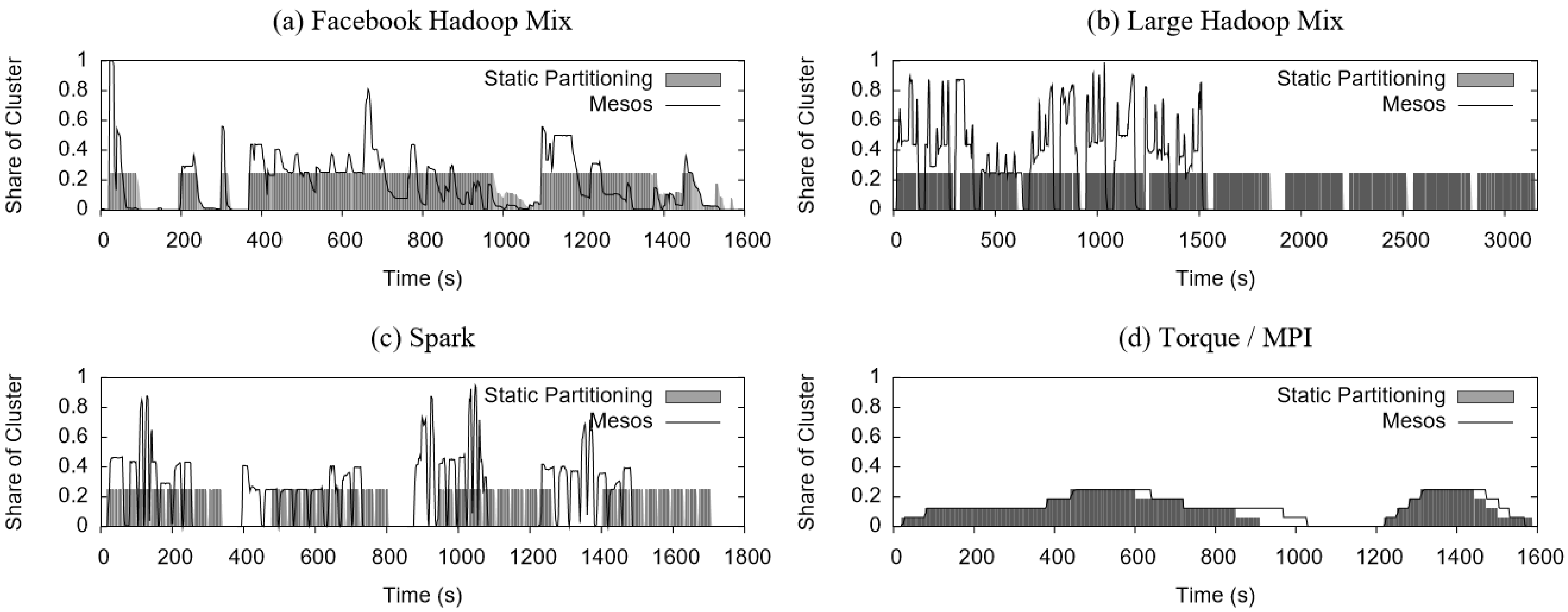
  V.S.

  - ➢ 4 static partitions of the cluster (24 nodes per partition)

Dynamic Resource Sharing



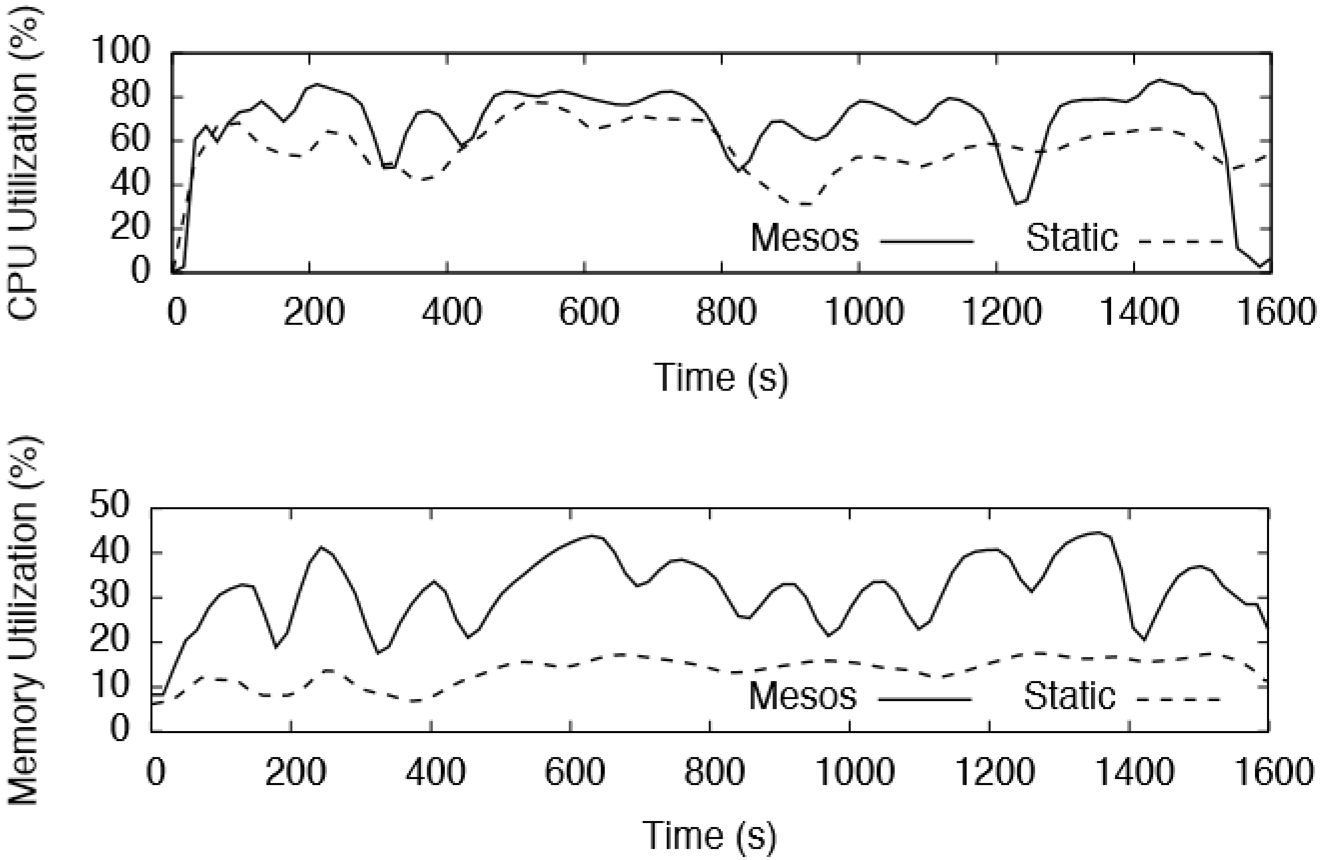
Static vs Mesos: Resource Allocation



Static vs Mesos: Speedup

Framework	Speedup on Mesos
Facebook Hadoop Mix	1.14 ×
Large Hadoop Mix	2.10 ×
Spark	1.26 ×
Torque / MPI	0.96 × ←

Static vs Mesos: Utilization



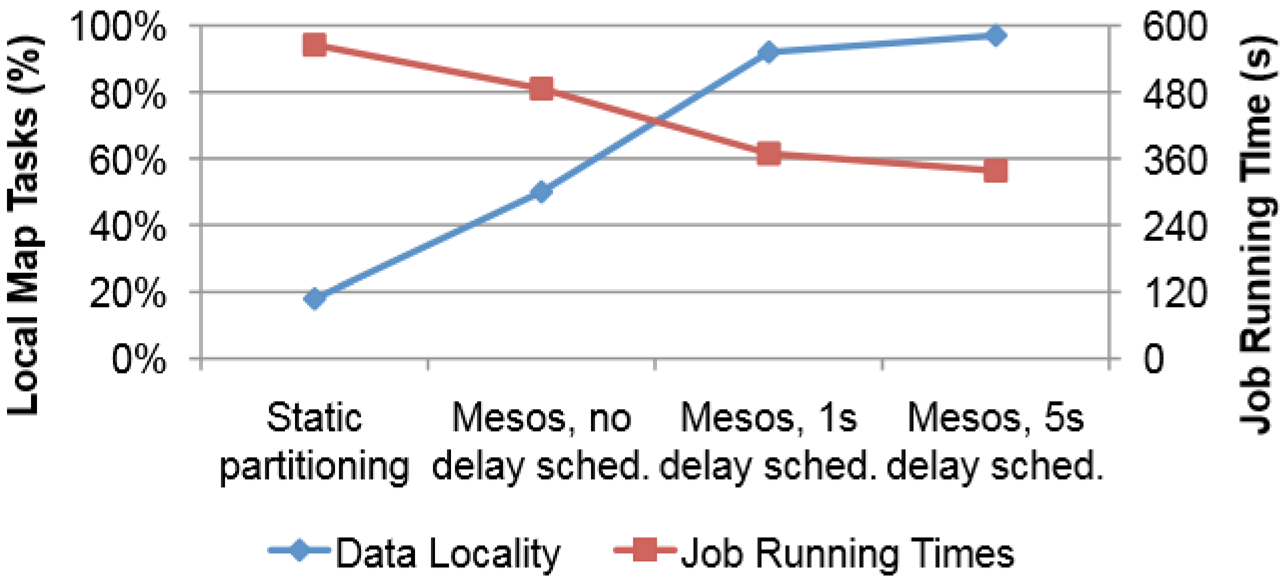
Mesos Overhead

- Ran two benchmarks using MPI and Hadoop on an EC2 cluster with **50 nodes**.
  - 2 CPU cores and 6.5 GB RAM
- The MPI job (LINPACK benchmark):
  - ~50.9s without Mesos → ~51.8s with Mesos
- Hadoop job (WordCount job):
  - ~160s without Mesos → ~166s with Mesos
- Result
  - In both cases, the overhead of using Mesos was **less than 4%**

Data Locality with Resource Offers

- Ran 16 instances of Hadoop on a shared HDFS cluster
  - Using 93 EC2 nodes
  - 4 CPU cores and 15 GB RAM
- Used **delay scheduling** in Hadoop to get locality (wait a short time to acquire data-local nodes)
- Running the Hadoop instances on Mesos improves data locality.

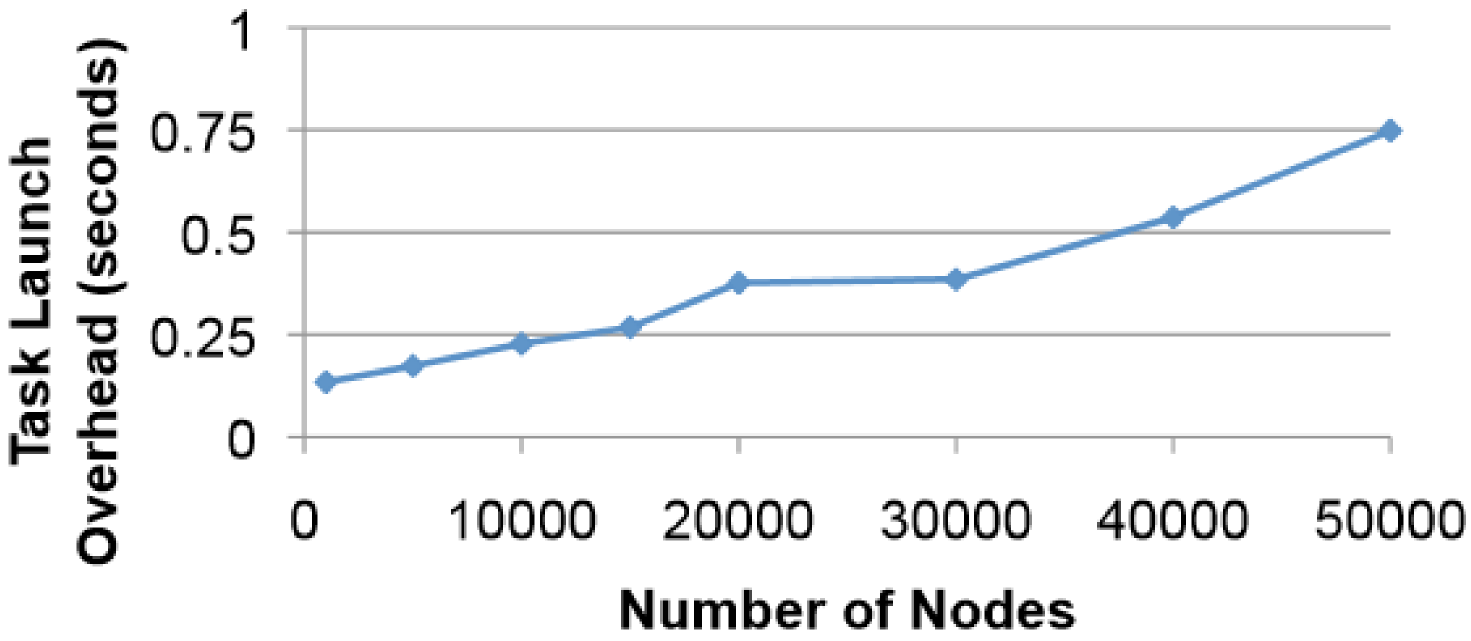
Data Locality with Resource Offers



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)

Scalability



Results Analysis

- **Resource offer** works well when:
  - Frameworks can scale up and down elastically
  - Task durations are homogeneous
  - Frameworks have many preferred nodes
- Otherwise Mesos may not be useful.
  - E.g. in the case of Torque / MPI
- Programming against Mesos is not easy :)

Related Work

- **HPC schedulers** (e.g. Sun Grid Engine, Torque, …)
  - Coarse-grained sharing for inelastic jobs (e.g. MPI jobs).
- **Virtual machine clouds**
  - Coarse-grained sharing similar to HPC schedulers.
- **Condor**
  - Centralized scheduler based on matchmaking
- …

Conclusion

- Mesos **shares clusters efficiently** among diverse frameworks.
- Two design elements:
  - **Fine-grained sharing,** at the level of tasks,
  - **Resource offers,** a scalable mechanism for application-controlled scheduling.
- Enables co-existence of current frameworks and development of new specialized ones.
- In use at Twitter, UC Berkeley, …

Conclusion

- What do you think about this?

☺

References

- [1] Hindman, B., Konwinski, A., Matei, Z., Ghodsi, A., D.Joseph, A., Katz, R., … Stoica, I. (2011). Mesos: ***A platform for fine-grained resource sharing in the data center***. *Proceedings of the …*, 32. https://doi.org/10.1109/TIM.2009.2038002
- [2] ***Apache Mesos***. (n.d.). Retrieved from http://mesos.apache.org/ ,2017
- [3] ***Apache Hadoop***. (n.d.). Retrieved from https://hadoop.apache.org/, 2017
- [4] Zaharia, M., Borthakur, D., Sen Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010). ***Delay scheduling***. Proceedings of the 5th European Conference on Computer Systems - EuroSys ’10, 265. https://doi.org/10.1145/1755913.1755940

Thank you for your attention!

Any QUESTIONS?

- *m-zakeri@live.com*