



# Mesos: A Platform for Fine-Grained Resource Sharing in Data Centers (II)

Anthony D. Joseph

LASER Summer School  
September 2013

# My Talks at LASER 2013

1. AMP Lab introduction
2. The Datacenter Needs an Operating System
3. Mesos, part one
4. Dominant Resource Fairness
5. Mesos, part two
6. Spark

# Collaborators

- Matei Zaharia
- Benjamin Hindman
- Andy Konwinski
- Ali Ghodsi
- Randy Katz
- Scott Shenker
- Ion Stoica

# Apache Mesos

A common resource sharing layer for diverse frameworks



Run multiple instances of the *same* framework

- » Isolate production and experimental jobs
- » Run multiple versions of the framework concurrently

Support *specialized frameworks* for problem domains

# Implementation

20,000+ lines of C++

APIs in C, C++, Java, and Python

Master failover using ZooKeeper

Frameworks ported: Hadoop, MPI, Torque

New specialized frameworks: Spark, Apache/HaProxy

Open source Apache project

<http://mesos.apache.org/>

# Frameworks

Ported frameworks:

- » [Hadoop](#) (900 line patch)
- » [MPI](#) (160 line wrapper scripts)

New frameworks:

- » [Spark](#), Scala framework for iterative jobs (1300 lines)
- » [Apache+haproxy](#), elastic web server farm (200 lines)

# Isolation

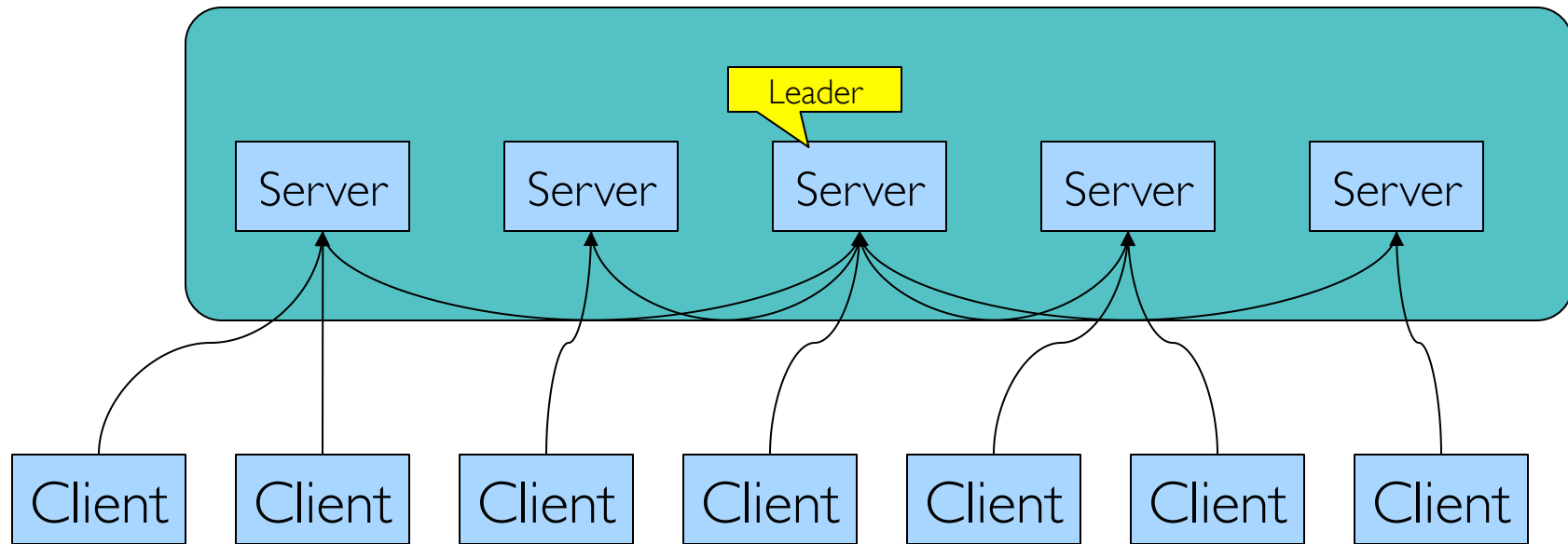
Mesos has pluggable *isolation modules* to isolate tasks sharing a node

Currently supports Linux Containers and Solaris projects

- » Can isolate memory, CPU, IO, network bandwidth

Could be a great place to use VMs

# Apache ZooKeeper



Multiple servers require coordination

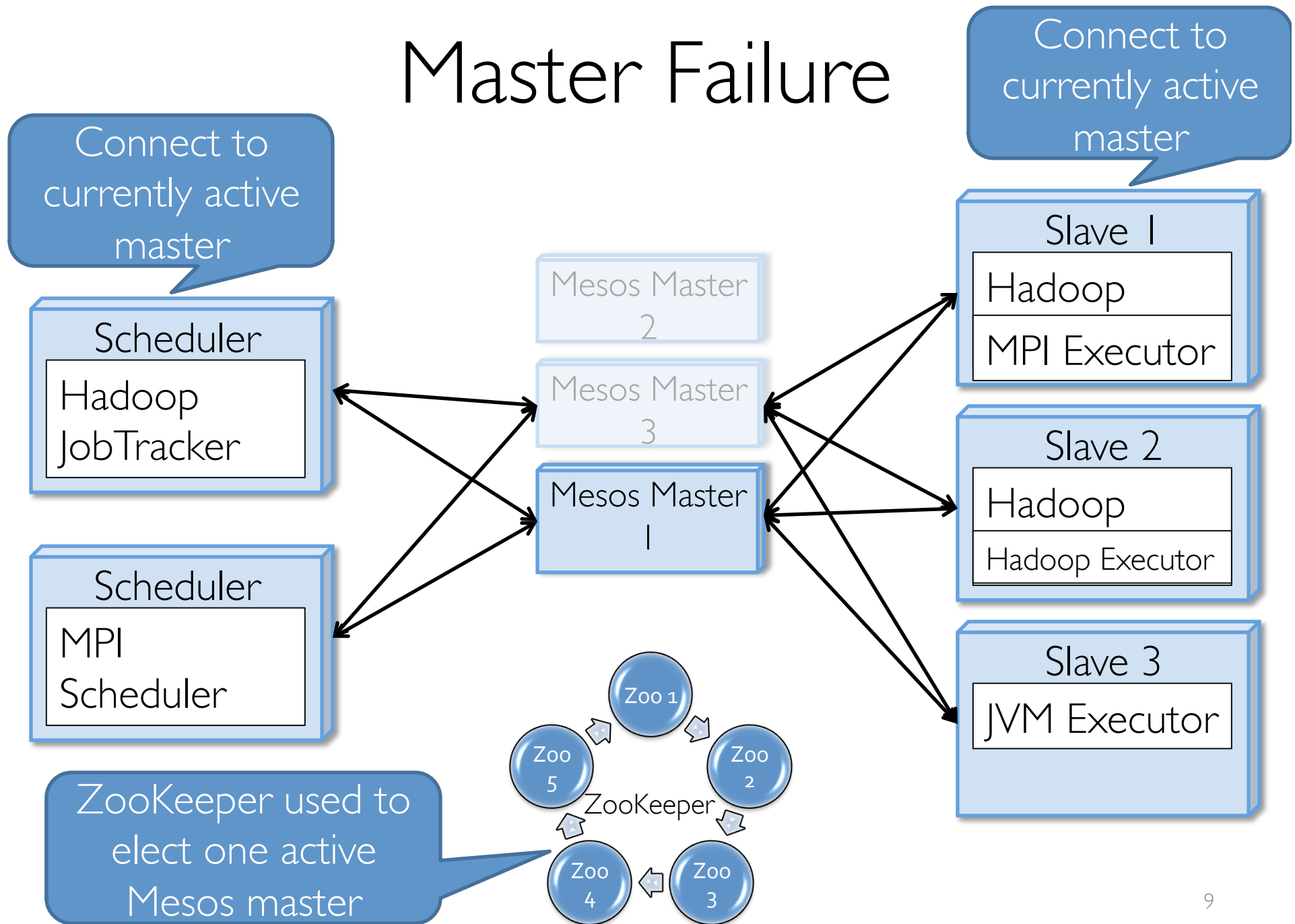
- » Leader Election, Group Membership, Work Queues, Data Sharding, Event Notifications, Configuration, and Cluster Management

Highly available, scalable, distributed coordination kernel

- » Ordered updates and strong persistence guarantees
- » Conditional updates (version), Watches for data changes



# Master Failure

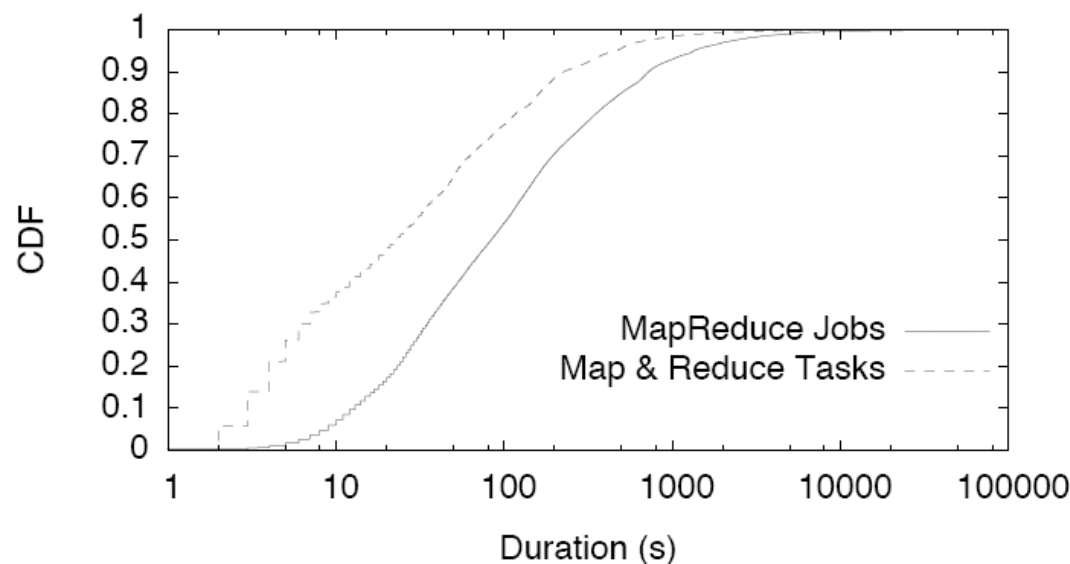


# Resource Revocation

Killing tasks to make room for other users

*Killing typically not needed for short tasks*

- » If avg task length is 2 min, a new framework gets 10% of all machines within 12 seconds on avg



Hadoop job and task durations at Facebook

# Resource Revocation (2)

*Not the normal case* because fine-grained tasks enable quick reallocation of resources

Sometimes necessary:

- » Long running tasks never relinquishing resources
- » Buggy job running forever
- » Greedy user who decides to makes his task long

*Safe allocation* lets frameworks have long running tasks defined by allocation policy

- » Users will get at least safe share within specified time
- » If stay below safe allocation, task won't be killed

# Resource Revocation (3)

Dealing with long tasks monopolizing nodes

- » Let slaves have *long slots* and *short slots*
- » Short slots killed if used too long by a task

Revoke only if a user is below its safe share and is interested in offers

- » Revoke tasks from users farthest above their safe share
- » Framework given a *grace period* before killing its tasks

# Example: Running MPI on Mesos

Users always told their safe share

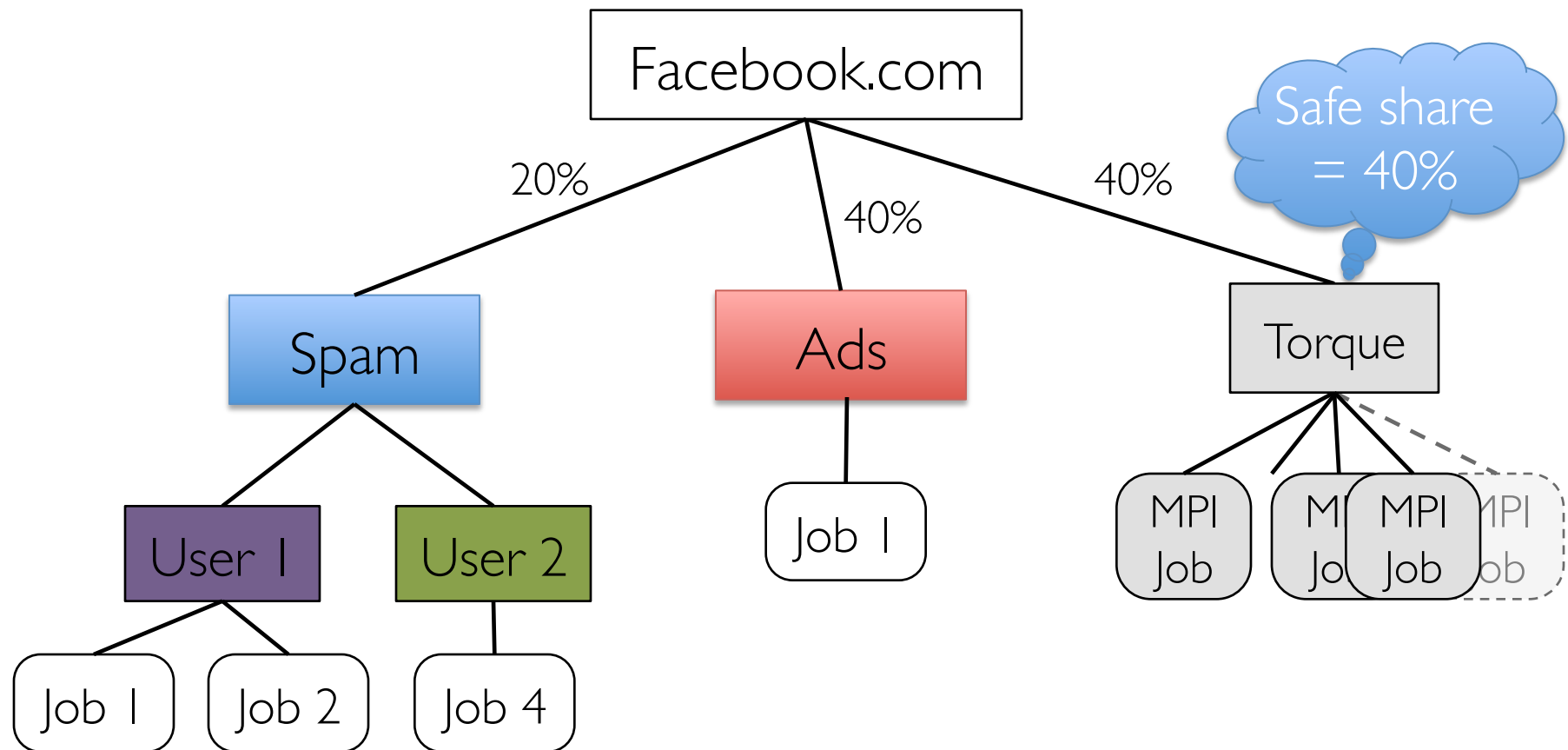
- » Avoid revocation by staying below it

Giving each user a small safe share may not be enough if jobs need many machines

Can run a traditional HPC scheduler as a user with a large safe share of the cluster, and have MPI jobs queue up on it

- » E.g. Torque gets 40% of cluster

# Example: Torque on Mesos



# Some Mesos Deployments



1,000's of nodes running over a dozen production services



Genomics researchers using Hadoop and Spark on Mesos



Spark in use by Yahoo! Research



Spark for analytics

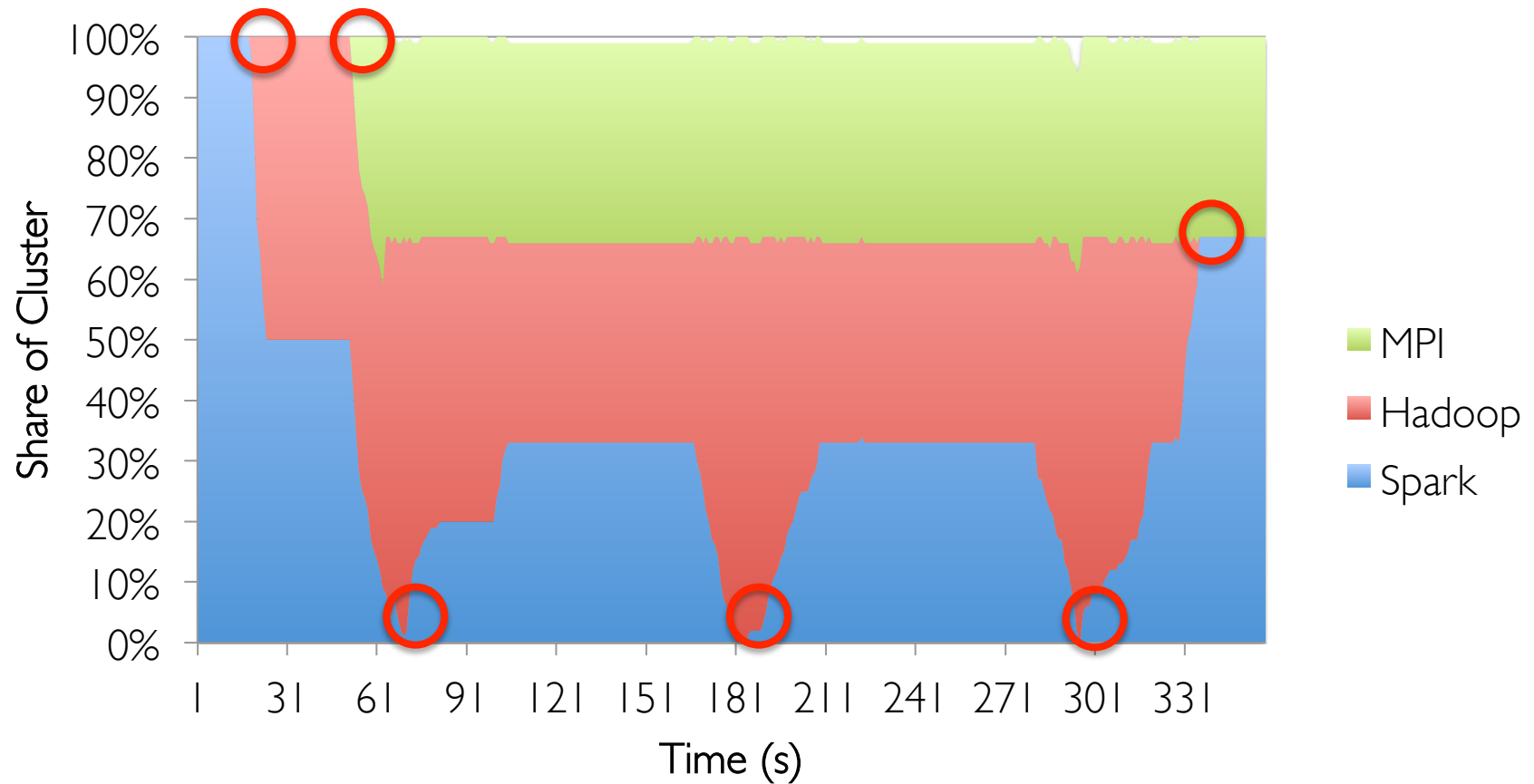


Hadoop and Spark used by machine learning researchers

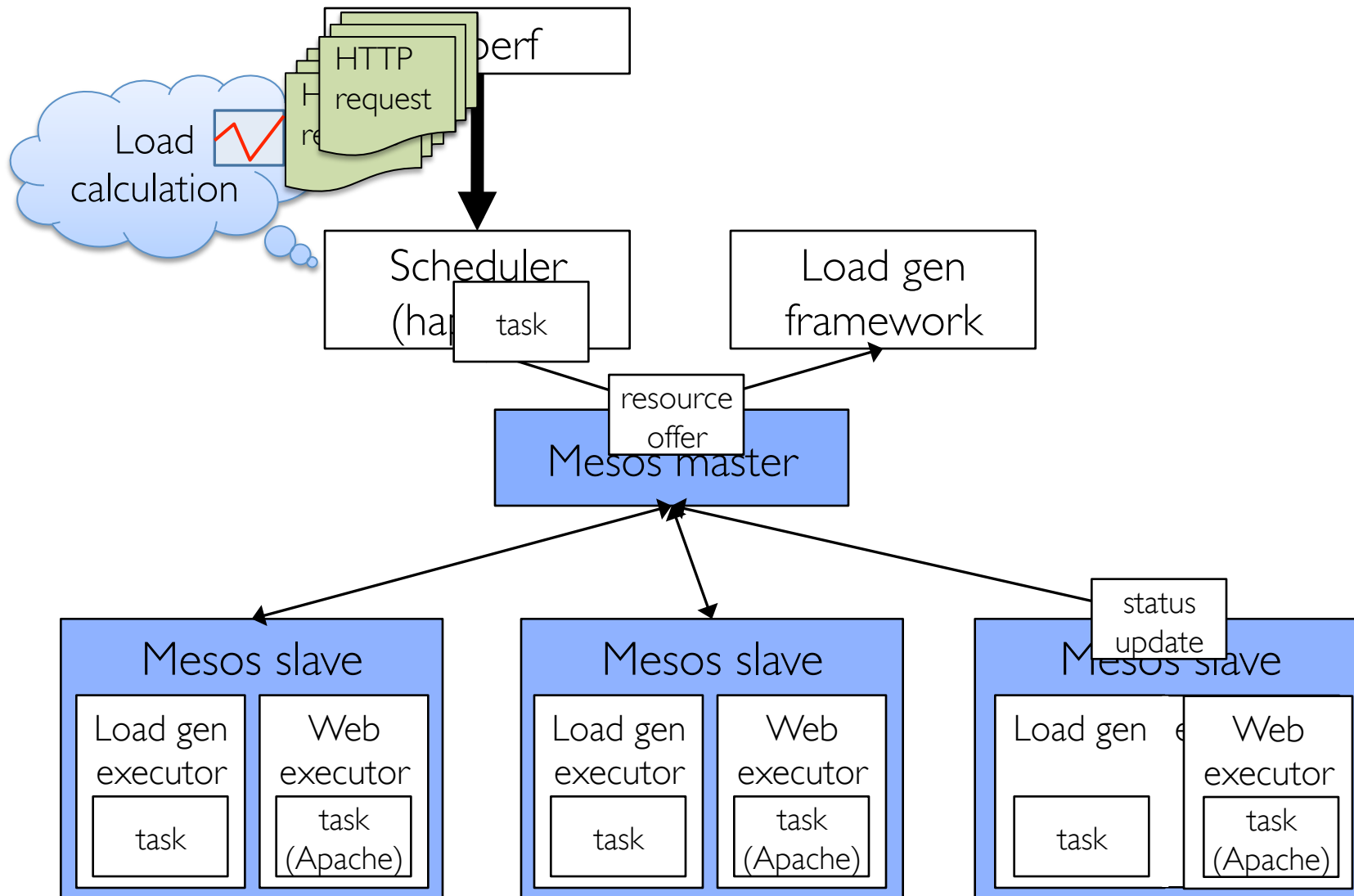
# Results



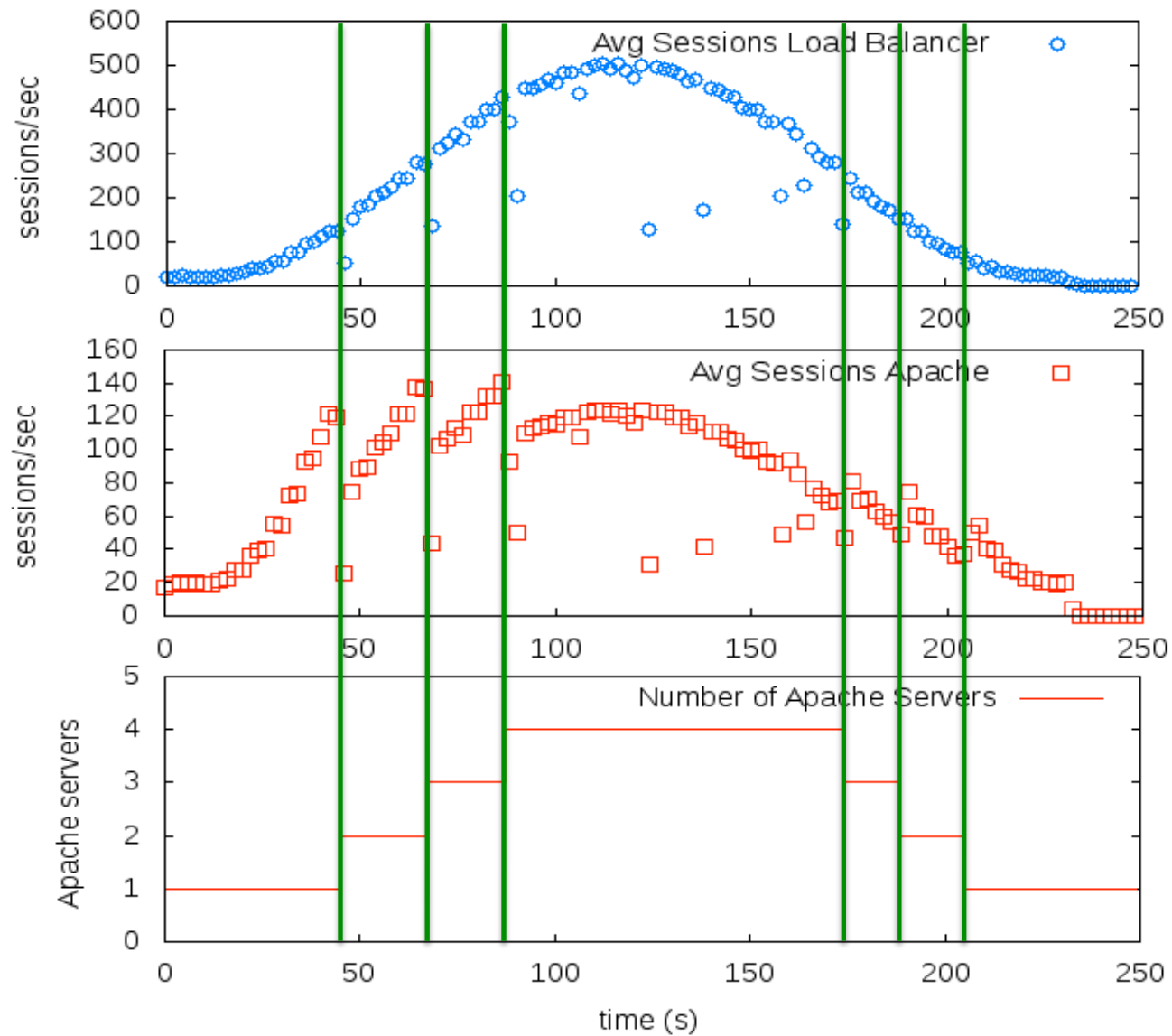
# Dynamic Resource Sharing



# Elastic Web Server Farm

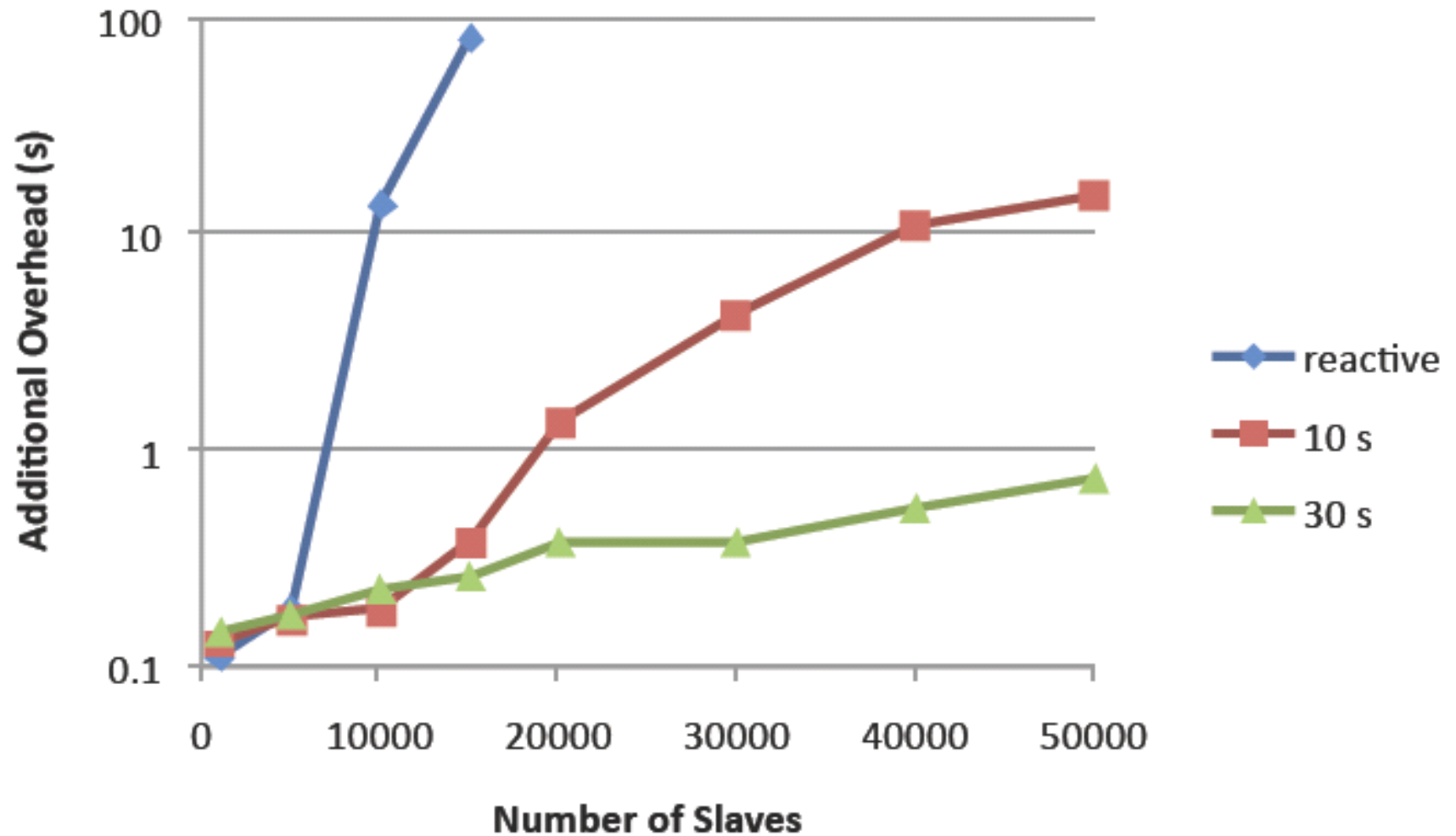


# Web Framework Results



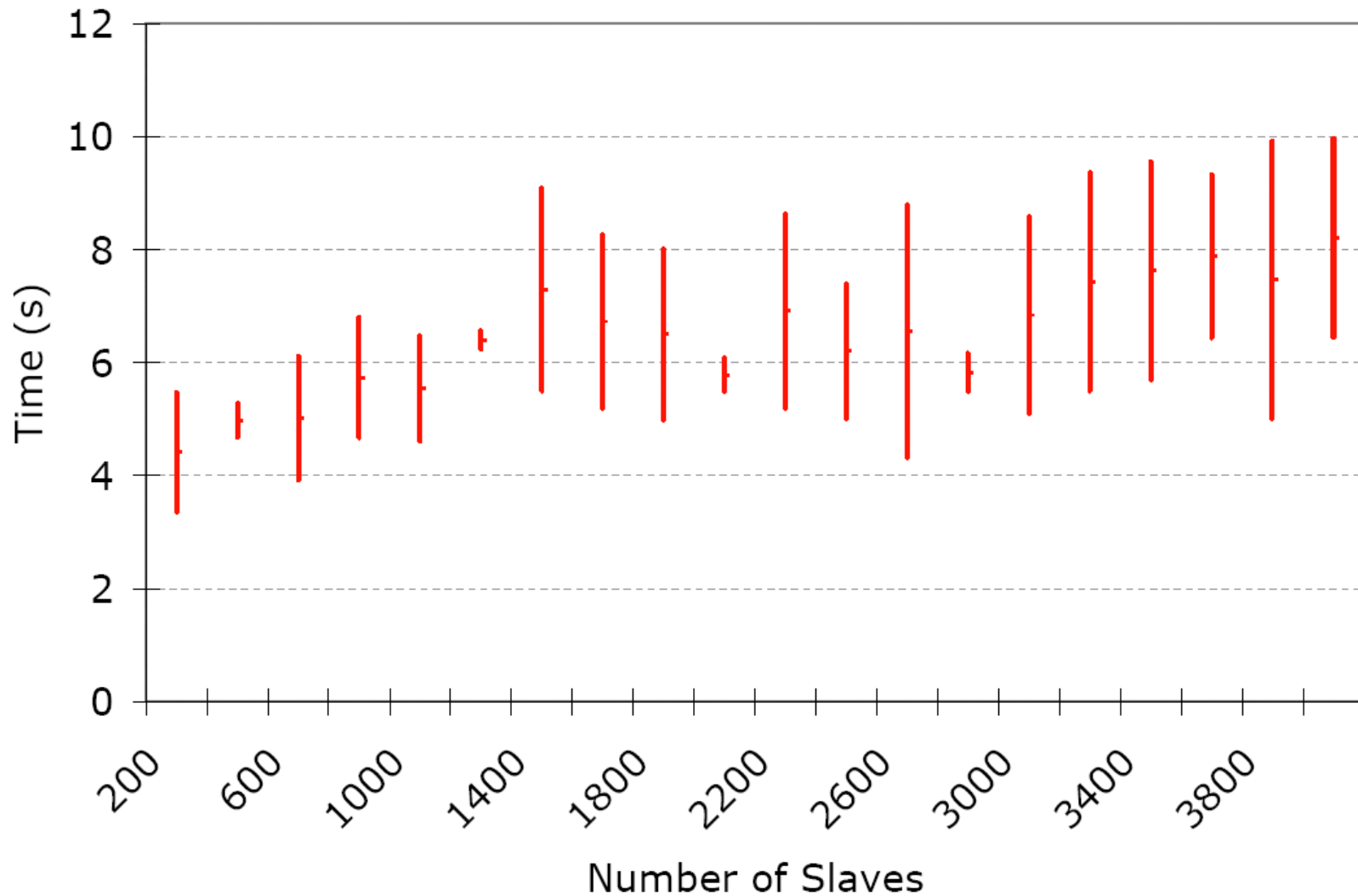
# Scalability

Task startup overhead with 200 frameworks



# Fault Tolerance

Mean time to recovery, 95% confidence



# Deep Dive Experiments

## Macrobenchmark experiment

- » Test the benefits of using Mesos to multiplex a cluster between multiple diverse frameworks

## High level goals of experiment

- » Demonstrate increased CPU/memory utilization due to multiplexing available resources
- » Demonstrate job runtime speedups

# Macrobenchmark setup

100 Extra Large EC2 instances (4 cores/15GB ram per machine)

Experiment length: ~25 minutes

Realistic workload

1. A Hadoop instance running a mix of small and large jobs based on the workload at Facebook
2. A Hadoop instance running a set of large batch jobs
3. Spark running a series of machine learning jobs
4. Torque running a series of MPI jobs

# Goal of experiment

Run the four frameworks and corresponding workloads...

- » 1<sup>st</sup> on a cluster that is shared via Mesos
- » 2<sup>nd</sup> on 4 partitioned clusters, each  $\frac{1}{4}$  the size of the shared cluster

Compare resource utilization and workload performance (i.e., job run times) on *static partitioning* vs. *sharing with Mesos*



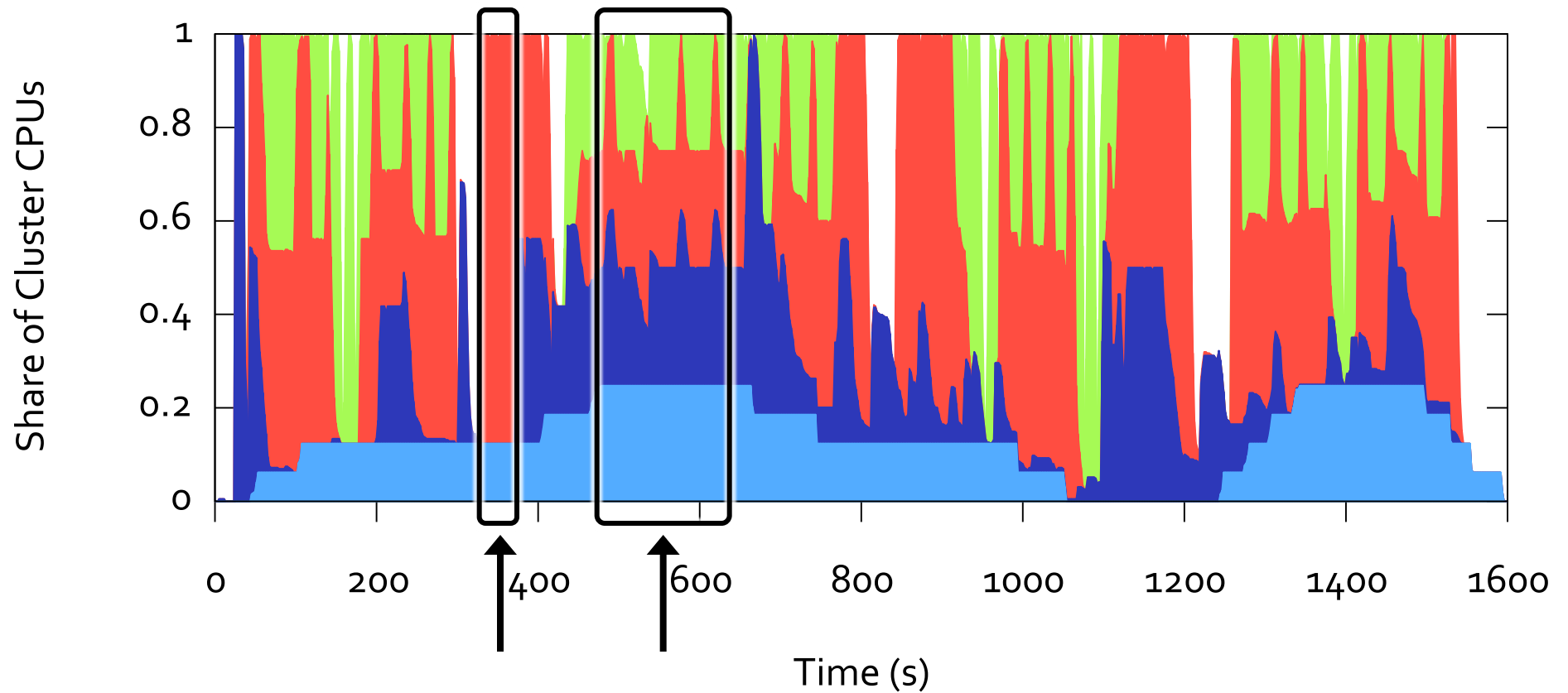
# Macrobenchmark Details:





## Breakdown of the Facebook Hive (Hadoop) Workload mix

Bin	Job Type	Map Tasks	Reduce Tasks	Jobs Run
1	Selection	1	NA	38
2	Text search	2	NA	18
3	Aggregation	10	2	14
4	Selection	50	NA	12
5	Aggregation	100	10	6
6	Selection	200	NA	6
7	Text Search	400	NA	4
8	Join	400	30	2

# Results: CPU Allocation

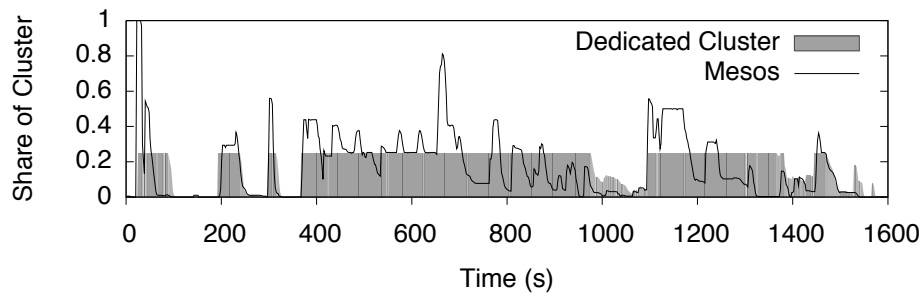
100 node cluster



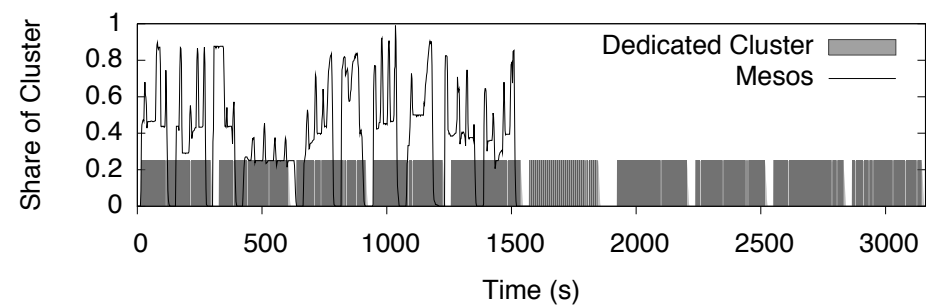
Spark		Hadoop (Facebook Mix)	
Hadoop (Batch Jobs)		Torque / MPI	

# Sharing With Mesos vs. No-Sharing (Dedicated Cluster)

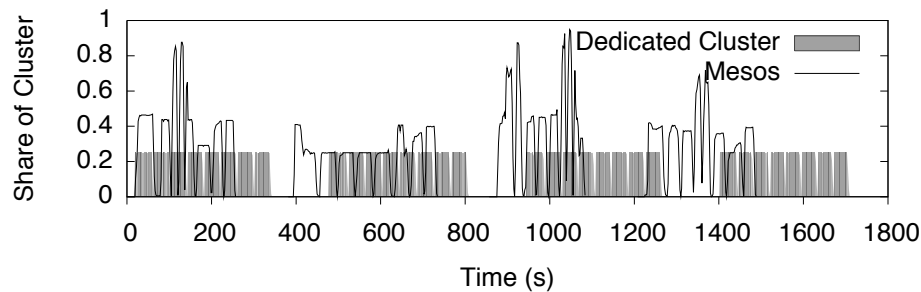
(a) Facebook Hadoop Mix



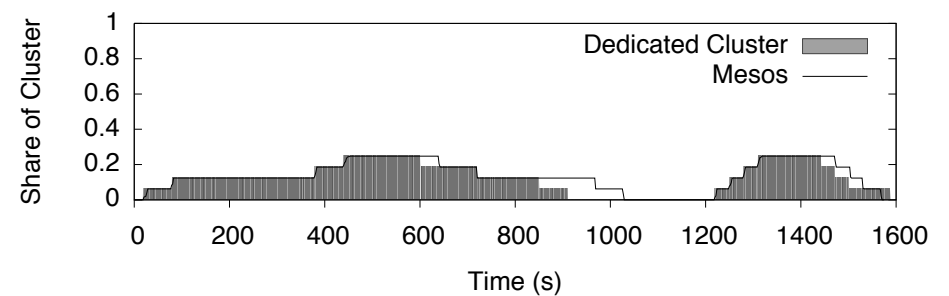
(b) Large Hadoop Mix



(c) Spark

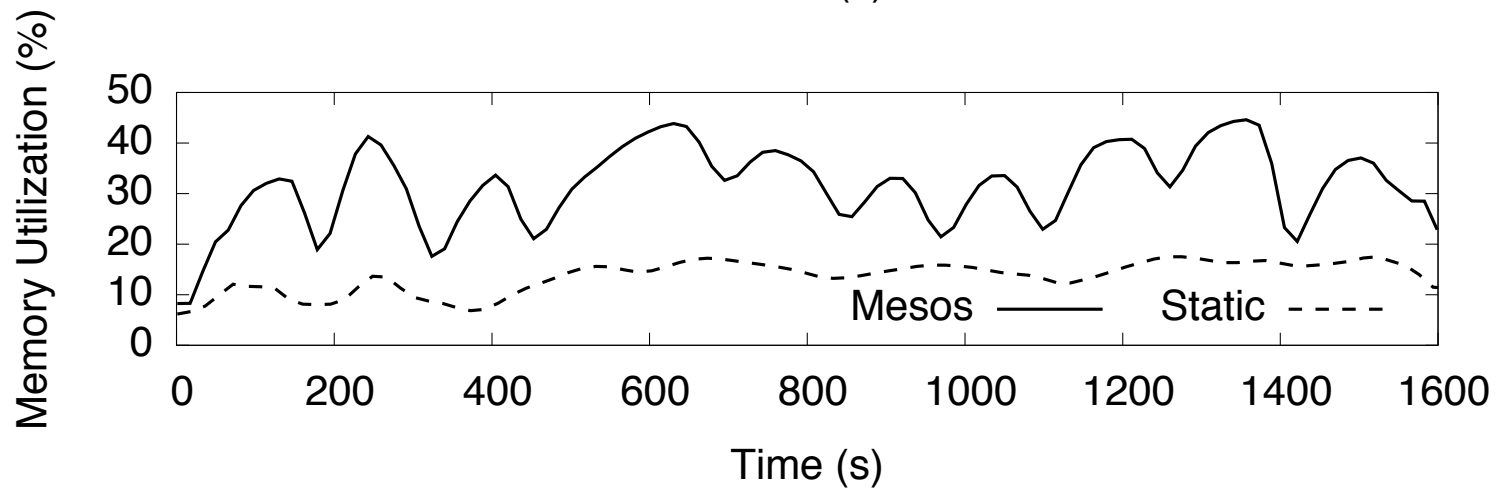
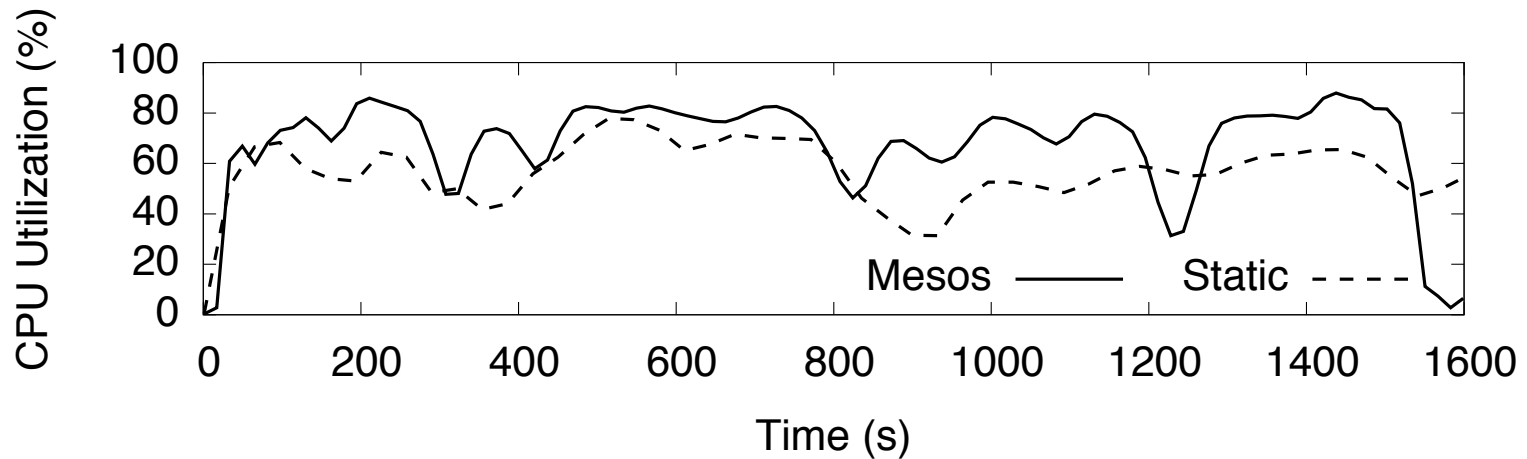


(d) Torque / MPI



# Cluster Utilization

## Mesos vs. Dedicated Clusters



# Job Run Times (and Speedup) Grouped By Framework

Framework	Sum of Exec times on Dedicated Cluster (s)	Sum of Exec Times on Mesos (s)	Speedup
Facebook Hadoop Mix	7235	6319	1.14
Large Hadoop Mix	3143	1494	2.10
Spark	1684	1338	1.26
Torque / MPI	3210	3352	0.96



2x speedup for  
Large Hadoop Mix

# Job Run Times (and Speedup)

## Grouped by Job Type

Framework	Job Type	Time on Dedicated Cluster (s)	Avg. Speedup on Mesos
Facebook Hadoop Mix	selection (1)	24	0.84
	text search (2)	31	0.90
	aggregation (3)	82	0.94
	selection (4)	65	1.40
	aggregation (5)	192	1.26
	selection (6)	136	1.71
	text search (7)	137	2.14
	join (8)	662	1.35
Large Hadoop Mix	text search	314	2.21
Spark	ALS	337	1.36
Torque / MPI	small tachyon	261	0.91
	large tachyon	822	0.88

# Discussion: Facebook Hadoop Mix Results

Smaller jobs perform worse on Mesos:

- » Side effect of interaction between fair sharing performed by Hadoop framework (among its jobs) and performed by Mesos (among frameworks)
- » When Hadoop has more than 1/4 of the cluster, Mesos allocates freed up resources to framework farthest below its share
- » Significant effect on any small Hadoop job submitted during this time (long delay relative to its length)
- » In contrast, Hadoop running alone can assign resources to the new job as soon as any of its tasks finishes

# Discussion: Facebook Hadoop Mix Results

Similar problem with hierarchical fair sharing  
appears in networks

- » Mitigation #1: run small jobs on a separate framework, or
- » Mitigation #2: use lottery scheduling as the Mesos allocation policy



# Discussion: Torque Results

Torque is the only framework that performed worse, on average, on Mesos

- » Large **tachyon** jobs took on average 2 minutes longer
- » Small ones took 20s longer

# Discussion: Torque Results

## Causes of delay

- » Partially due to Torque having to wait to launch 24 tasks on Mesos before starting each job – average delay is 12s
- » Rest of the delay may be due to stragglers (slow nodes)
- » In standalone Torque run, two jobs each took ~60s longer to run than others
- » Both jobs used a node that performed slower on single-node benchmarks than the others (Linux reported a 40% lower bogomips value on the node)
- » Since **tachyon** hands out equal amounts of work to each node, it runs as slowly as the slowest node

# Macrobenchmark Summary

Evaluated performance of diverse set of frameworks representing realistic workloads running on Mesos versus a statically partitioned cluster

Showed 10% increase in CPU utilization, 18% increase in memory utilization

Some frameworks show significant speed ups in job run time

Some frameworks show minor slowdowns in job run time due to experimental/environmental artifacts

# Summary

Mesos is a platform for sharing data centers among diverse cluster computing frameworks

- » Enables efficient fine-grained sharing
- » Gives frameworks control over scheduling

Mesos is

- » Scalable (50,000 slaves)
- » Fault-tolerant (MTTR 6 sec)
- » Flexible enough to support a variety of frameworks (MPI, Hadoop, Spark, Apache, ...)

# My Talks at LASER 2013

1. AMP Lab introduction
2. The Datacenter Needs an Operating System
3. Mesos, part one
4. Dominant Resource Fairness
5. Mesos, part two
6. Spark