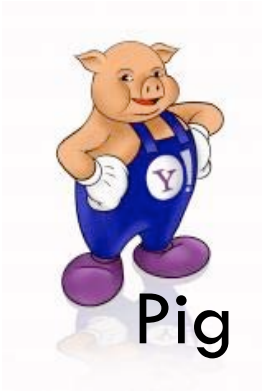
Mesos

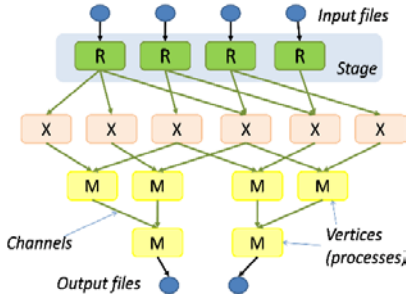
sharing the cluster.

Benjamin Hindman, **Andy Konwinski**, Matei Zaharia,
Ali Ghodsi, Anthony Joseph, Randy Katz, Scott Shenker, Ion Stoica

University of California, Berkeley

Google
Pregel





Dryad

CIEL

S4 distributed stream computing platform

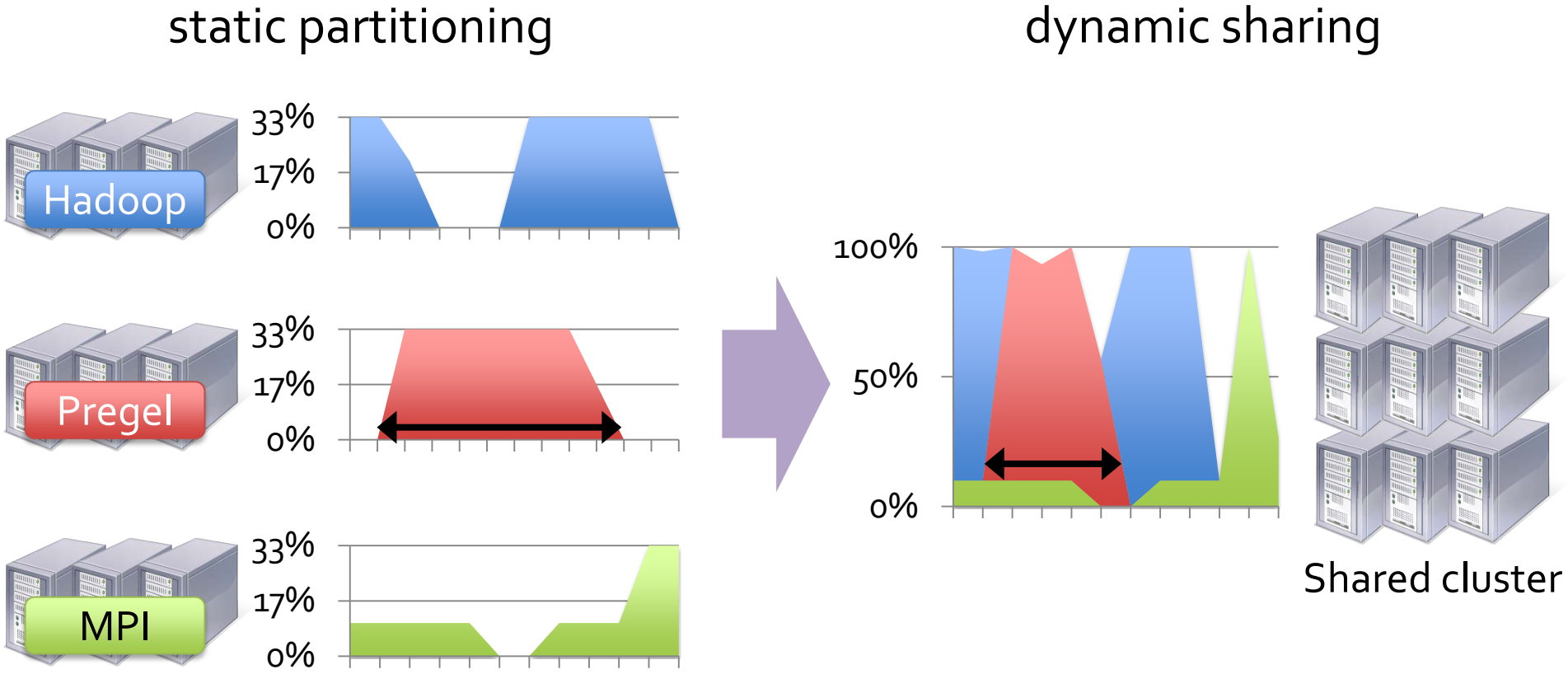
Google
Percolator

observation. Many distributed apps; no single one optimal for all use cases.

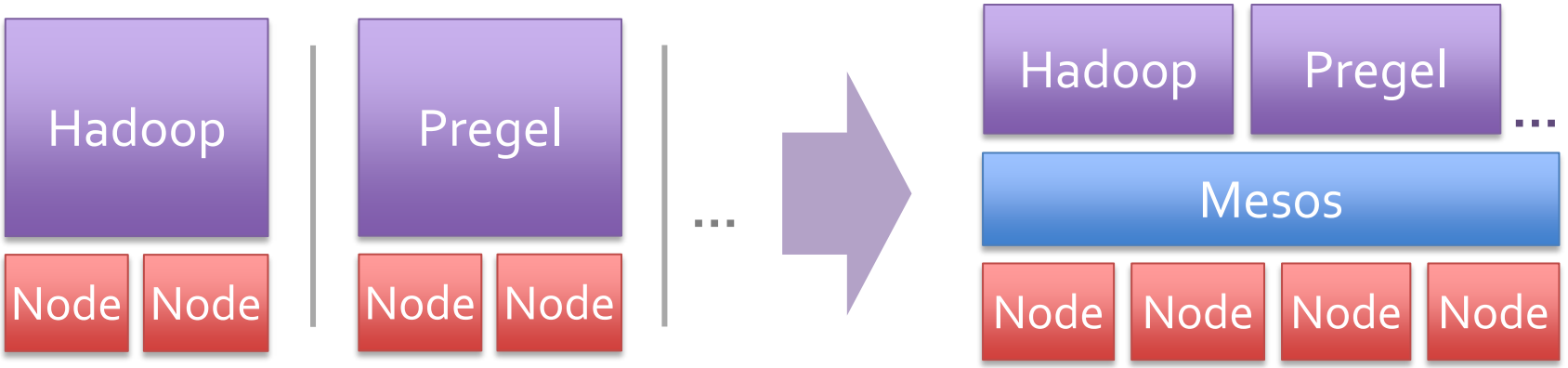
Want to run multiple applications in a single cluster

…to *maximize utilization*

…to *share data*


static partitioning
dynamic sharing
Hadoop
Pregel
MPI
33%
17%
0%
100%
50%
0%
Shared cluster

solution. Mesos, a common layer over which diverse applications can run.



Multiple instances of the *same* application

Build *specialized applications* targeting particular problem domains

Build *specialized applications*

goals.

High utilization of resources

Diverse applications

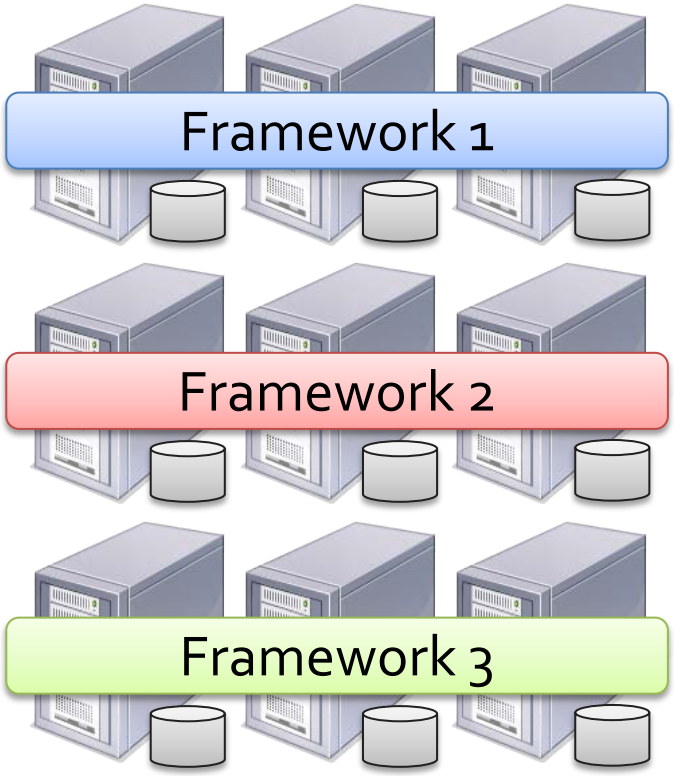
Scalability to 50-100K nodes

Fault tolerance

resulting design. Small microkernel-like core

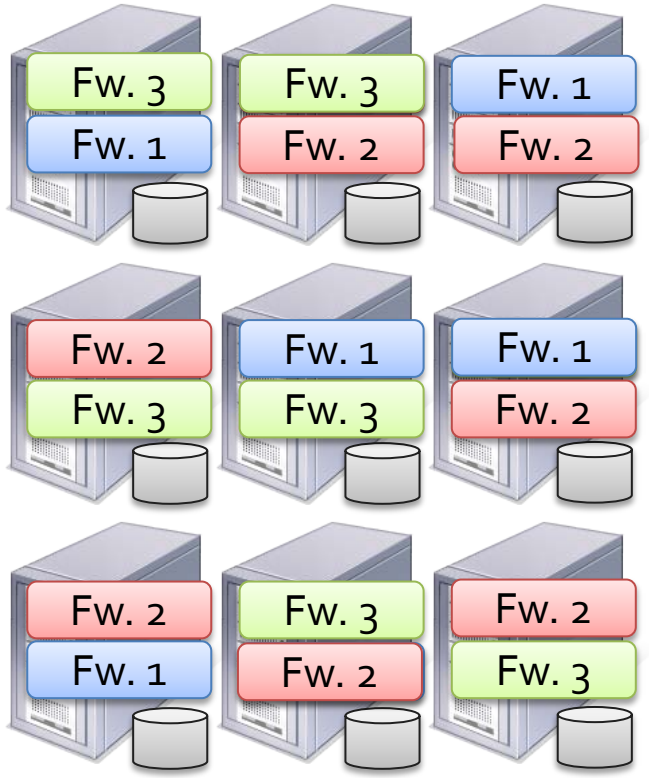
design element 1. Fine-grained sharing

Coarse-Grained Sharing (HPC):



Storage System (e.g. HDFS)

Fine-Grained Sharing (Mesos):



Storage System (e.g. HDFS)

+ Improved utilization, responsiveness, data locality

design element 2. Resource offers (vs. global scheduler)

global scheduler.

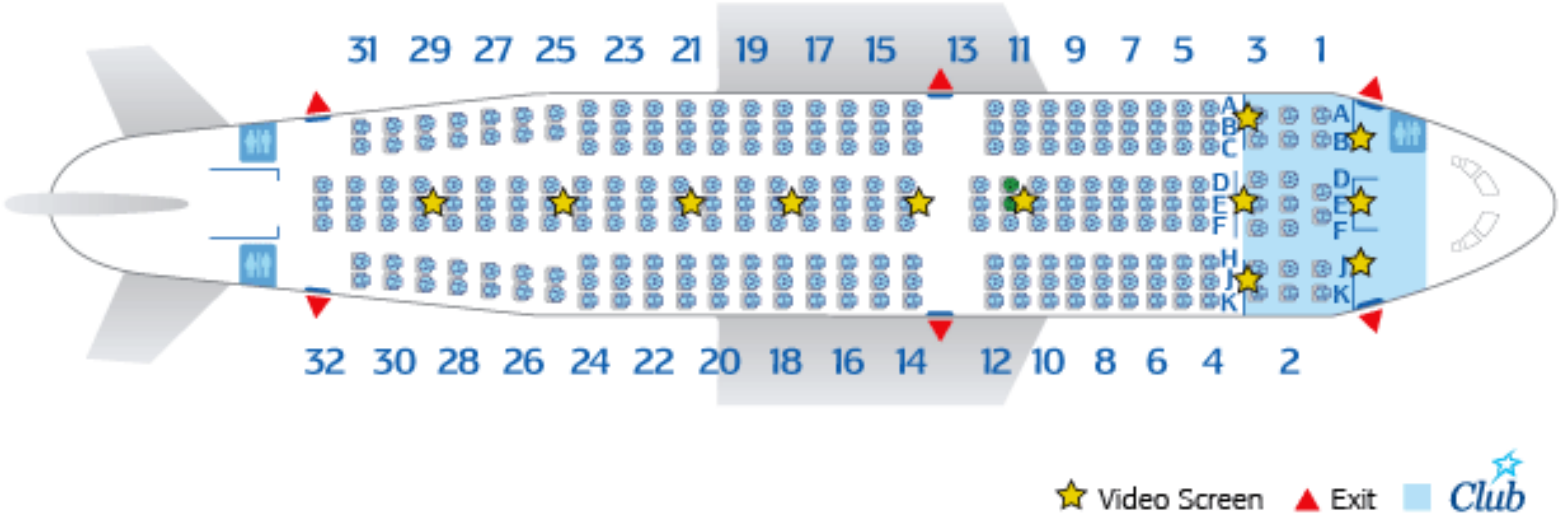
Frameworks express needs in a specification language, global scheduler matches them to resources

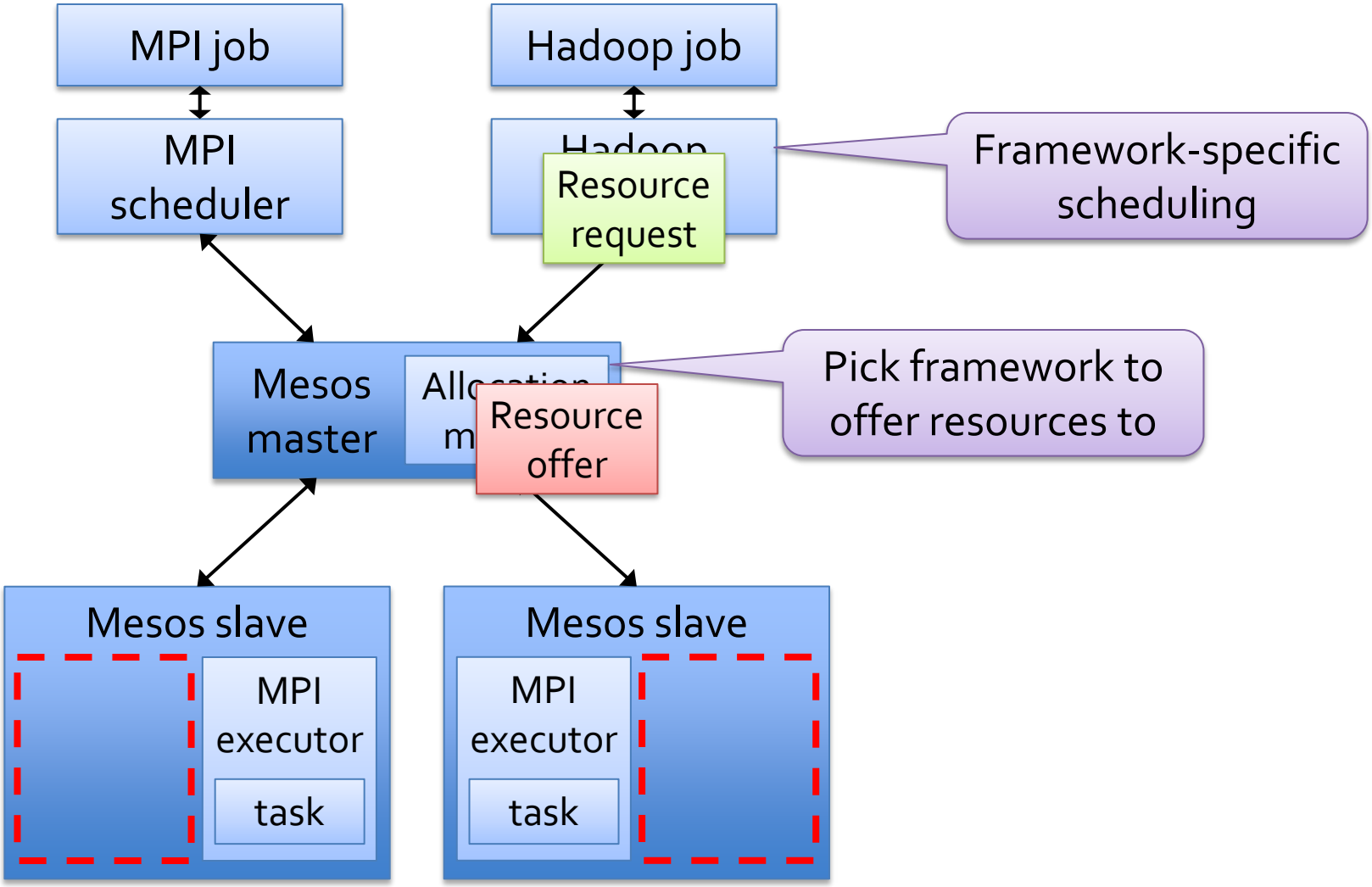
+ Can make optimal decisions
– Complex: language must support all framework needs
– Difficult to scale and to make robust
– Future frameworks may have unanticipated needs

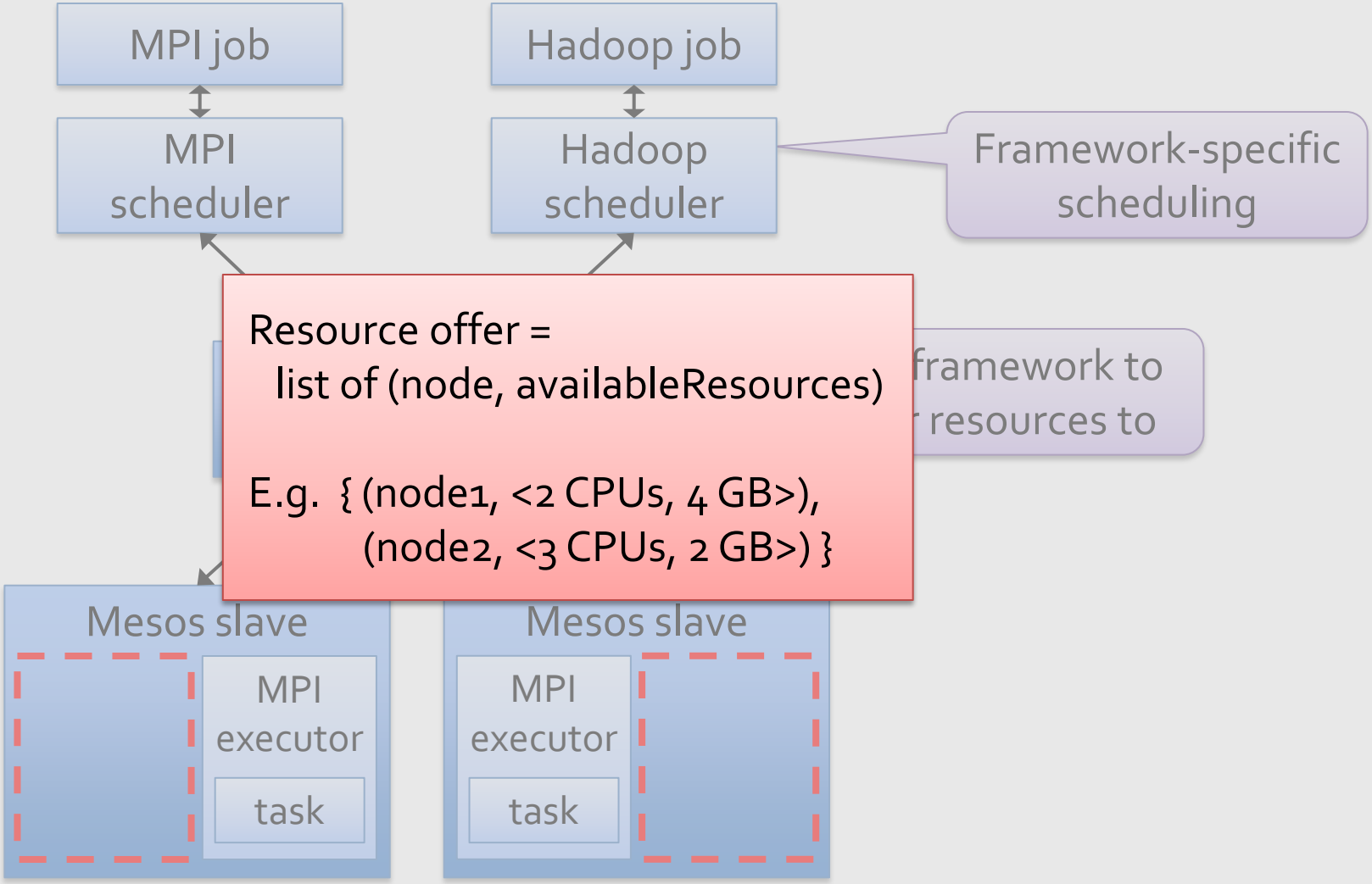
mesos: resource offers.

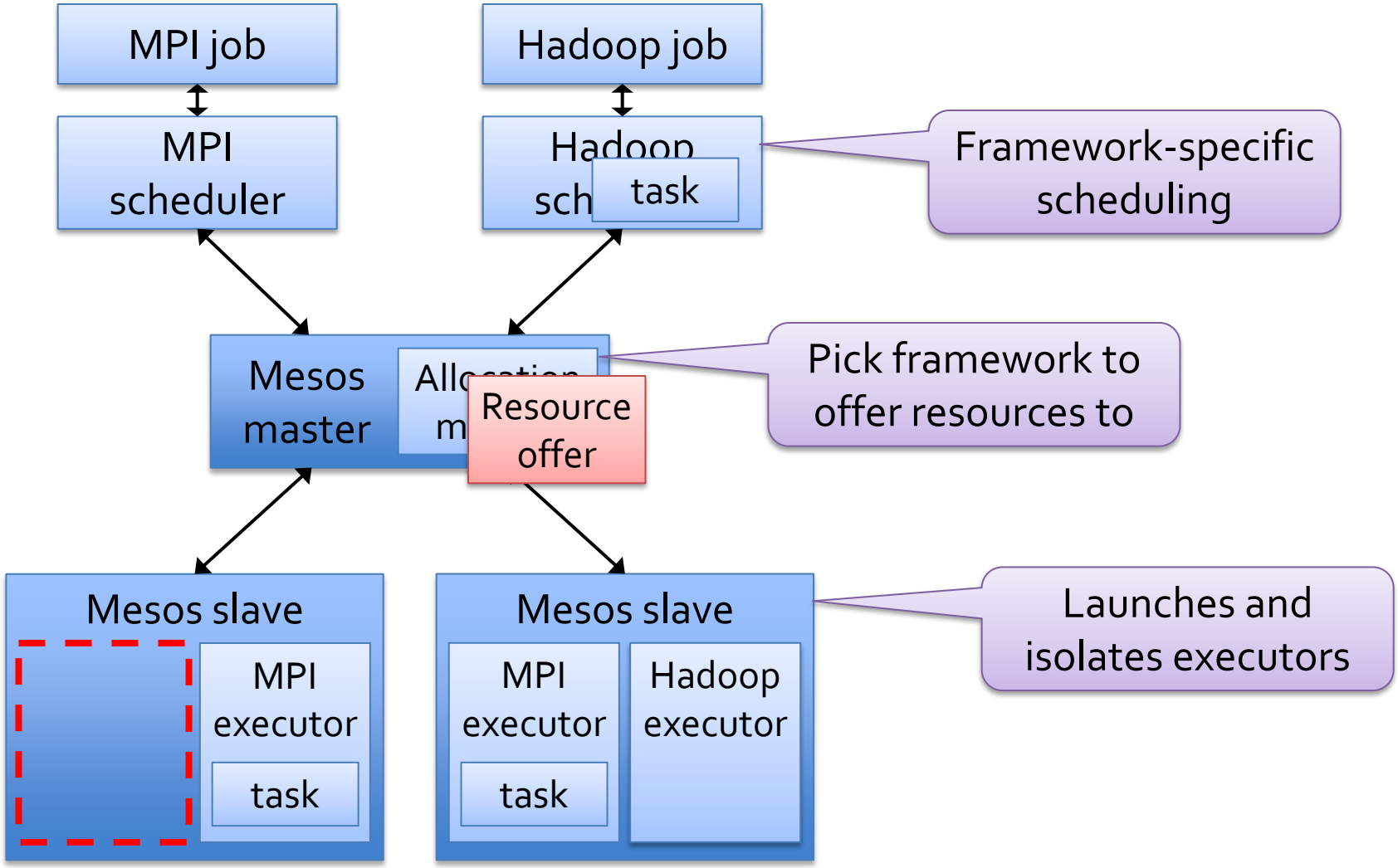
Offer available resources to frameworks, let them pick which resources to use and which tasks to launch

+ Keeps Mesos simple, lets it support future frameworks
- Decentralized decisions might not be optimal




MPI job
Hadoop job
MPI scheduler
Resource request
Framework-specific scheduling
Mesos master
Resource offer
Pick framework to offer resources to
Mesos slave
MPI executor
task
Mesos slave
MPI executor
task


MPI job
Hadoop job
MPI scheduler
Hadoop scheduler
Framework-specific scheduling
framework to
resources to
Resource offer =
list of (node, availableResources)
E.g. { (node1, <2 CPUs, 4 GB>),
(node2, <3 CPUs, 2 GB>) }
Mesos slave
MPI executor
task
Mesos slave
MPI executor
task


MPI job
Hadoop job
MPI scheduler
Hadoop sch
task
Framework-specific scheduling
Mesos master
Allocation m
Resource offer
Pick framework to offer resources to
Mesos slave
MPI executor
task
Mesos slave
MPI executor
task
Hadoop executor
Launches and isolates executors

200+ nodes running ~12 production services (stream processing)

Genomics researchers using Hadoop and Spark on Mesos

Spark in use by Y! research

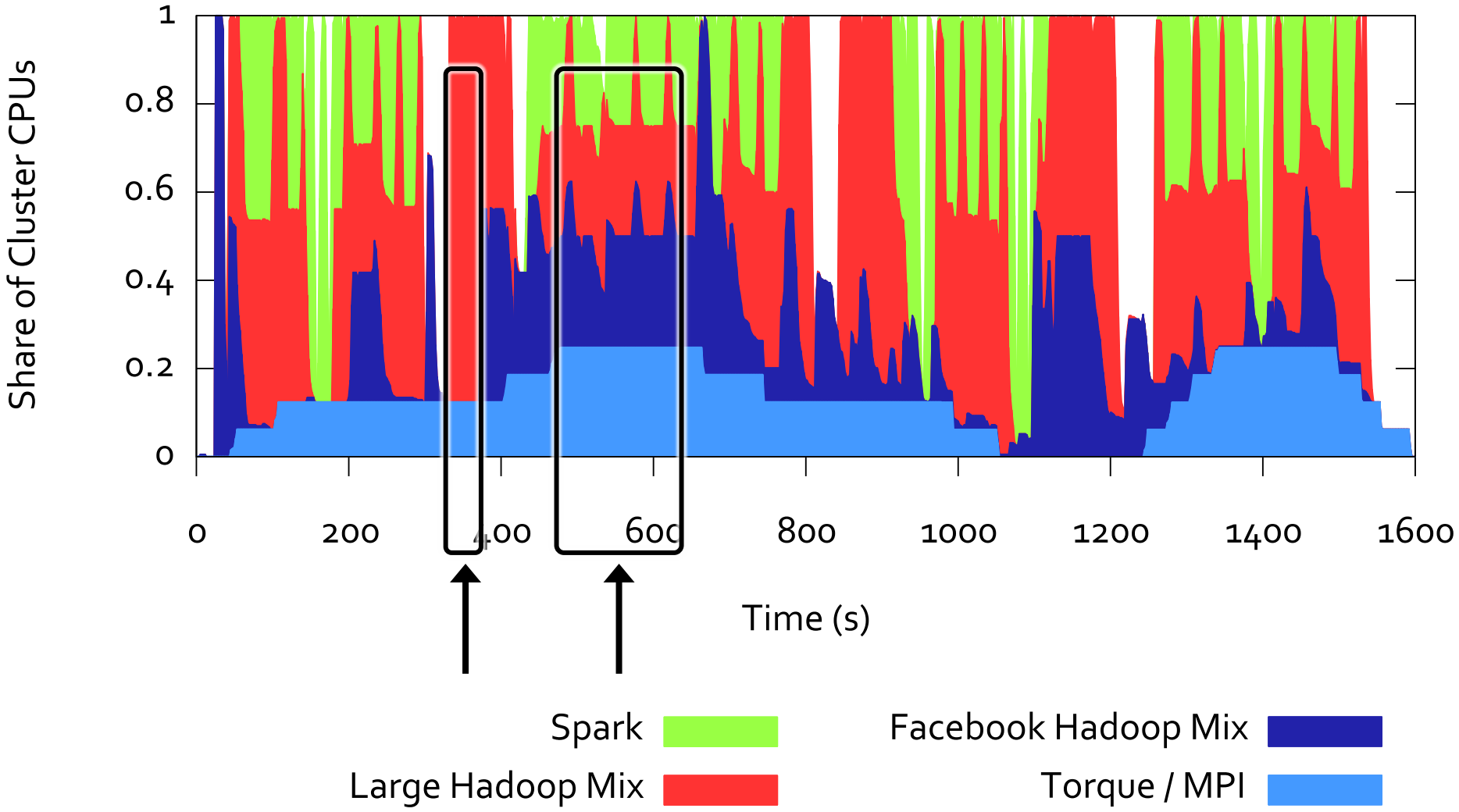
Spark for Analytics

Hadoop, running Experiments, Spark for ML Grad students for research

Your Name Here

Cutting Edge Research!

evidence that sharing helps.


Share of Cluster CPUs
1
0.8
0.6
0.4
0.2
0
0
200
400
600
800
1000
1200
1400
1600
Time (s)
Spark
Large Hadoop Mix
Facebook Hadoop Mix
Torque / MPI

vs. static allocation (each app gets 25% of nodes)

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

let's talk!

(actually we have been)

about…

1. Lessons learned: ours, Tashi's, others
2. Long running services
3. Socialized services (HDFS, HBase)
4. Priority scheduling

mesos-project.org

Backup Slides

Deployments


RAD Lab

What's running?

Stream processing (tweets from the "firehose")

- » Detecting spam and malicious users
  - Duplicate detection
  - Repeat offender detection
  - Spam account creation banning
  - … more
- » Trend identification

Conviva

35% of all analytics jobs in Spark on Mesos

Spark jobs running on data in Hive

Porting Hive on top of Spark (Shark)

UCSF

Goal: run Hadoop and Spark to analyze genomic data for UCSF Medical Center

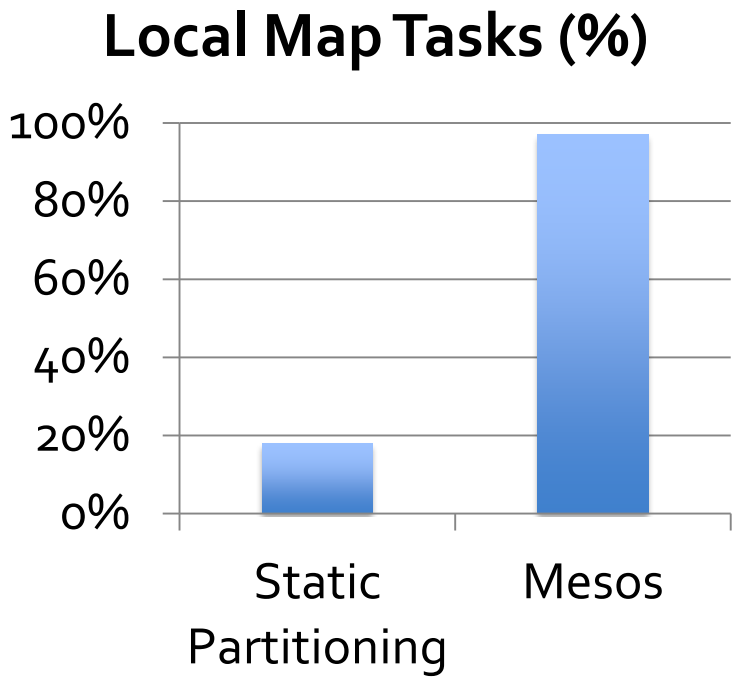
Scale out from 10 dedicated nodes onto ~1000 node HPC cluster running Sun Grid Engine

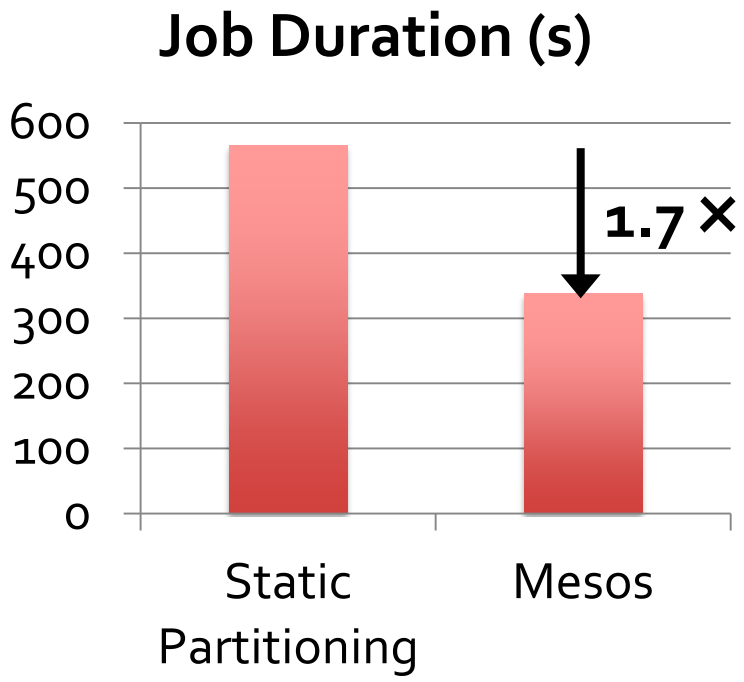
- » Nodes allocated through SGE join Mesos cluster

Data Locality with Resource Offers

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)

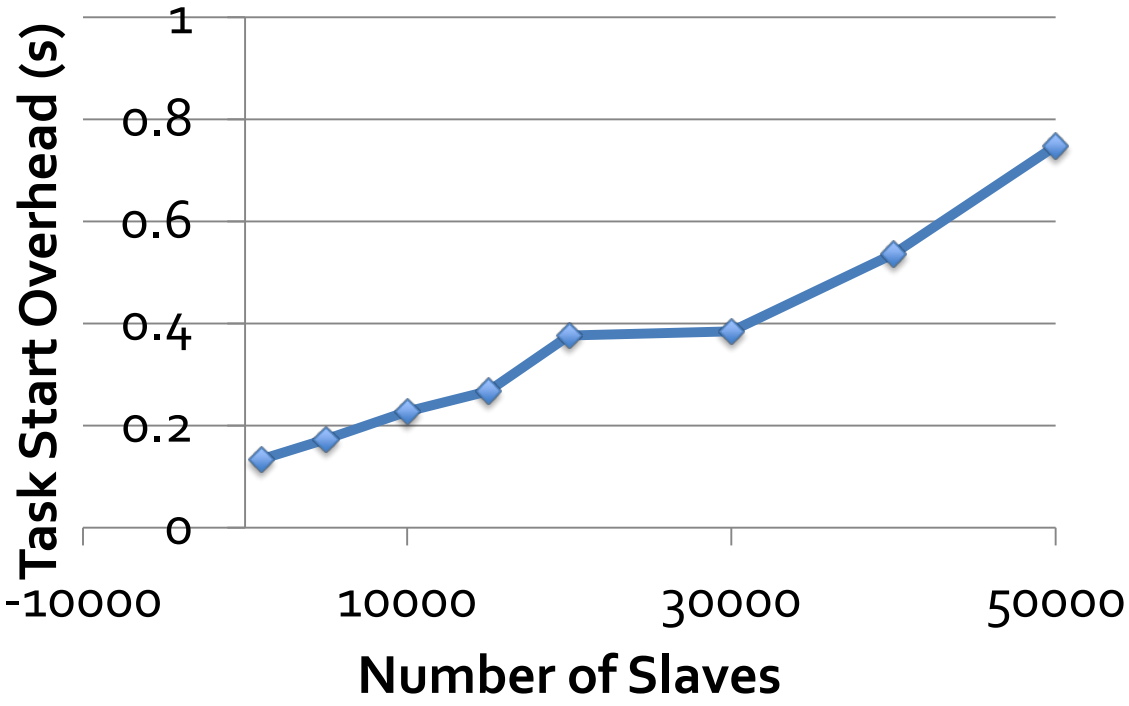




Scalability

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)



Fault Tolerance

Mesos master has only *soft state*: list of currently running frameworks and tasks

Rebuild when frameworks and slaves re-register with new master after a failure

Result: fault detection and recovery in ~10 sec

Related Work

HPC schedulers (e.g. Torque, LSF, Sun Grid Engine)
- » Coarse-grained sharing for inelastic jobs (e.g. MPI)

Virtual machine clouds
- » Coarse-grained sharing similar to HPC

Condor
- » Centralized scheduler based on matchmaking

Parallel work: Next-Generation Hadoop
- » Redesign of Hadoop to have per-application masters
- » Also aims to support non-MapReduce jobs
- » Based on resource request language with locality prefs

Conclusion

Mesos shares clusters efficiently among diverse frameworks thanks to 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, Conviva and UCSF

Framework Isolation

Mesos uses OS isolation mechanisms, such as Linux containers and Solaris projects

Containers currently support CPU, memory, IO and network bandwidth isolation

Not perfect, but much better than no isolation

Analysis

Resource offers work well when:

- Frameworks can scale up and down elastically
- Task durations are homogeneous
- Frameworks have many preferred nodes

These conditions hold in current data analytics frameworks (MapReduce, Dryad, …)

- Work divided into short tasks to facilitate load balancing and fault recovery
- Data replicated across multiple nodes

Revocation

Mesos allocation modules can revoke (kill) tasks to meet organizational SLOs

Framework given a grace period to clean up

"Guaranteed share" API lets frameworks avoid revocation by staying below a certain share

Mesos API

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)