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

Ashvin Goel

Electrical and Computer Engineering
University of Toronto

ECE1724

Authors: Benjamin Hindman, Andy Konwinski, Matei Zaharia,
Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica

Many slides adapted from Ion Stoica

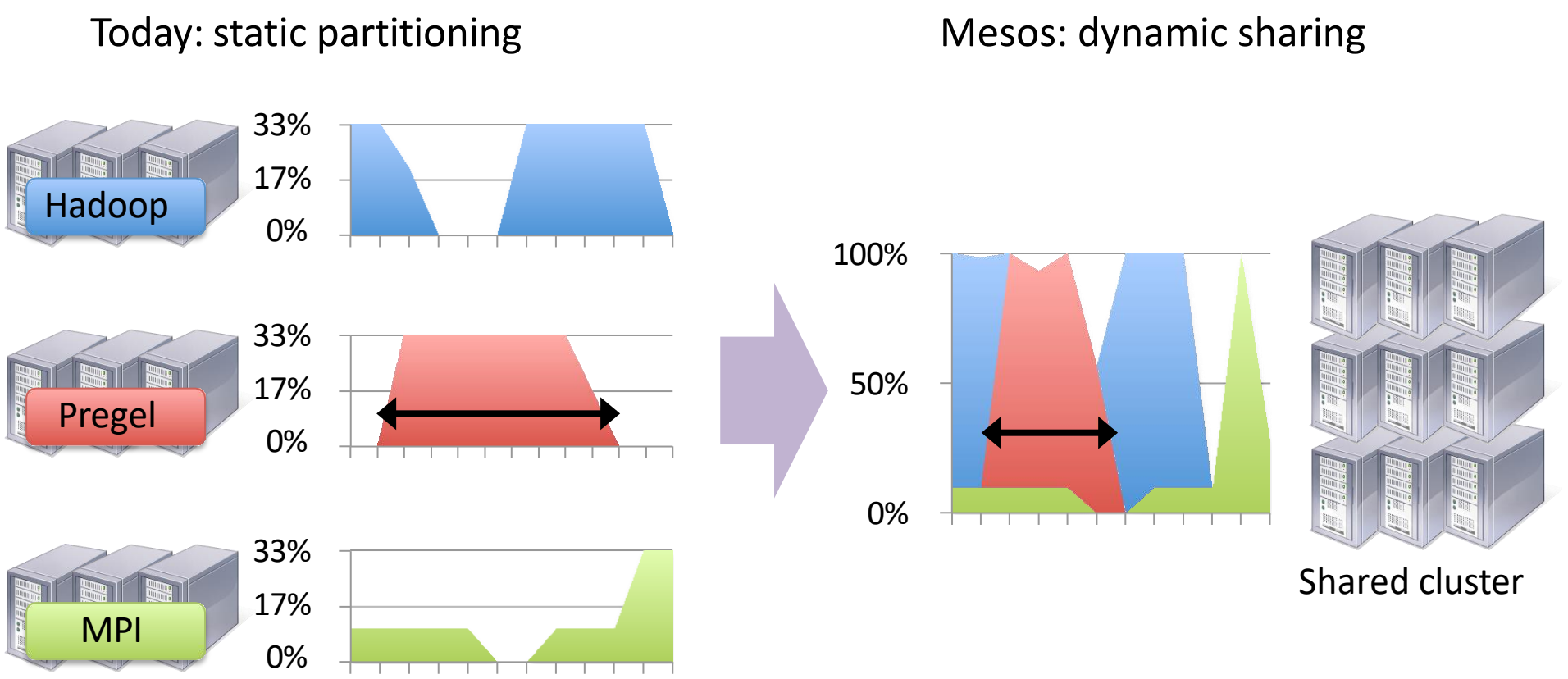
Goals of Mesos

- Fine-grained data sharing
- Support diverse frameworks within a cluster
- High resource utilization
- High scalability, reliability

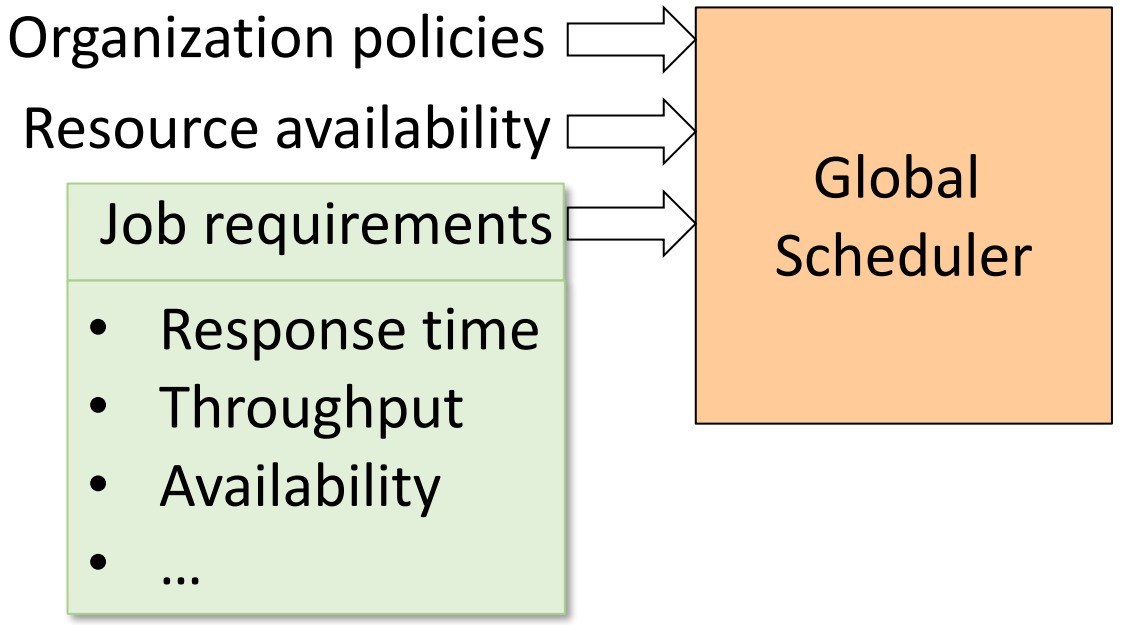
Fine-Grained Data Sharing

- Multiplex tasks on physical nodes
  - Tasks share CPU, memory, disk of physical node
  - Typical tasks are 10s of seconds to minutes
- Tasks may belong to different jobs
- Jobs may run on different frameworks

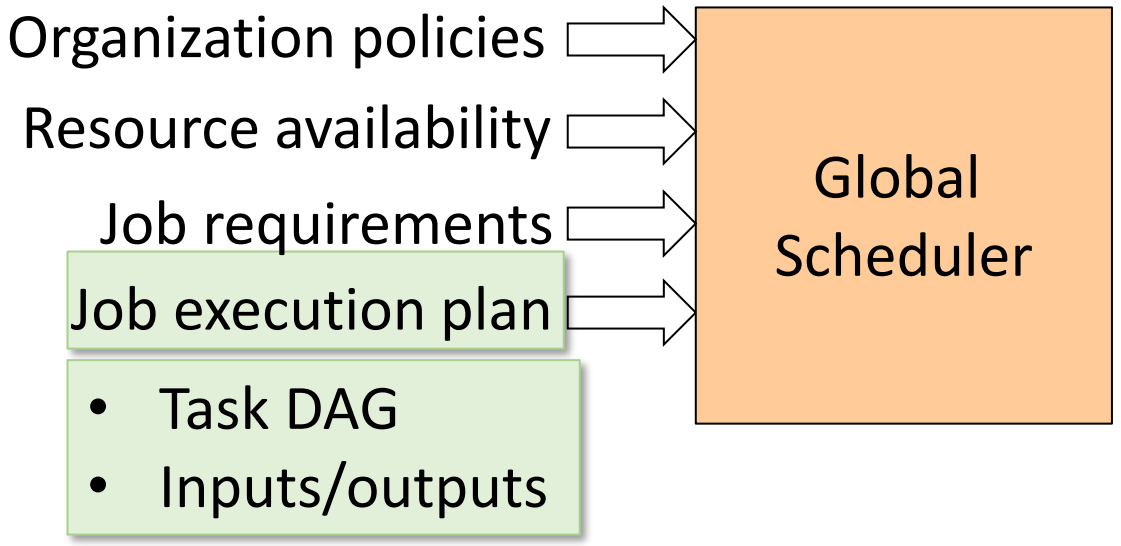
Fine-Grained Data Sharing Example



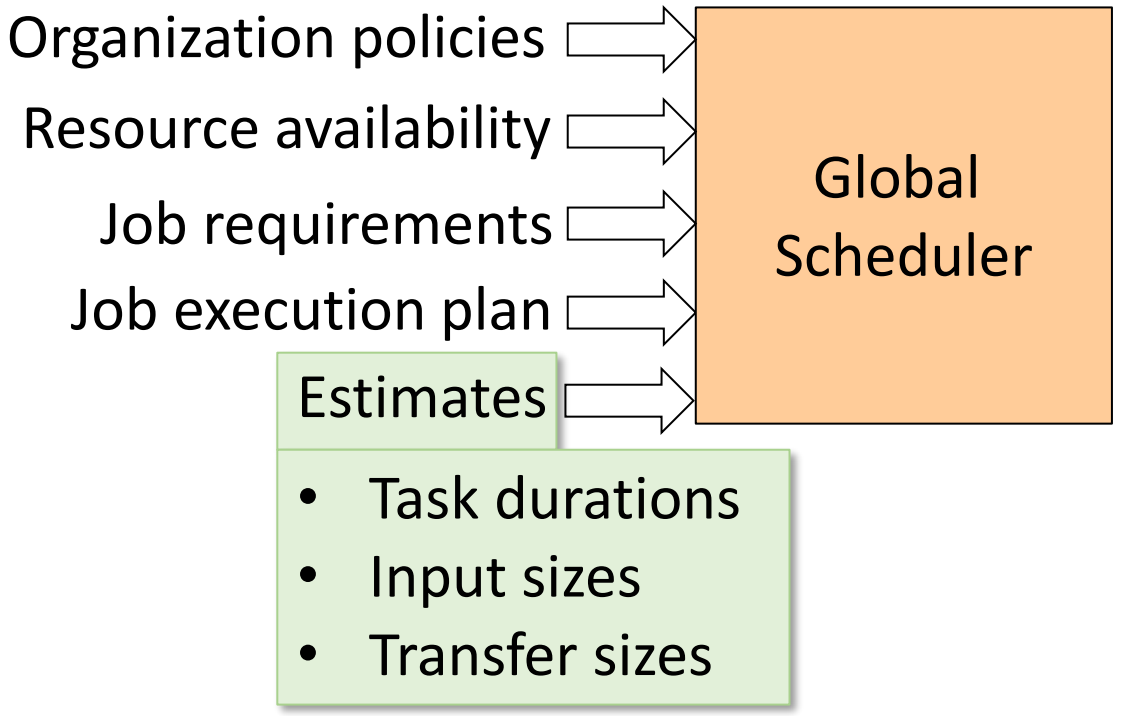
Baseline Approach: Global Scheduler



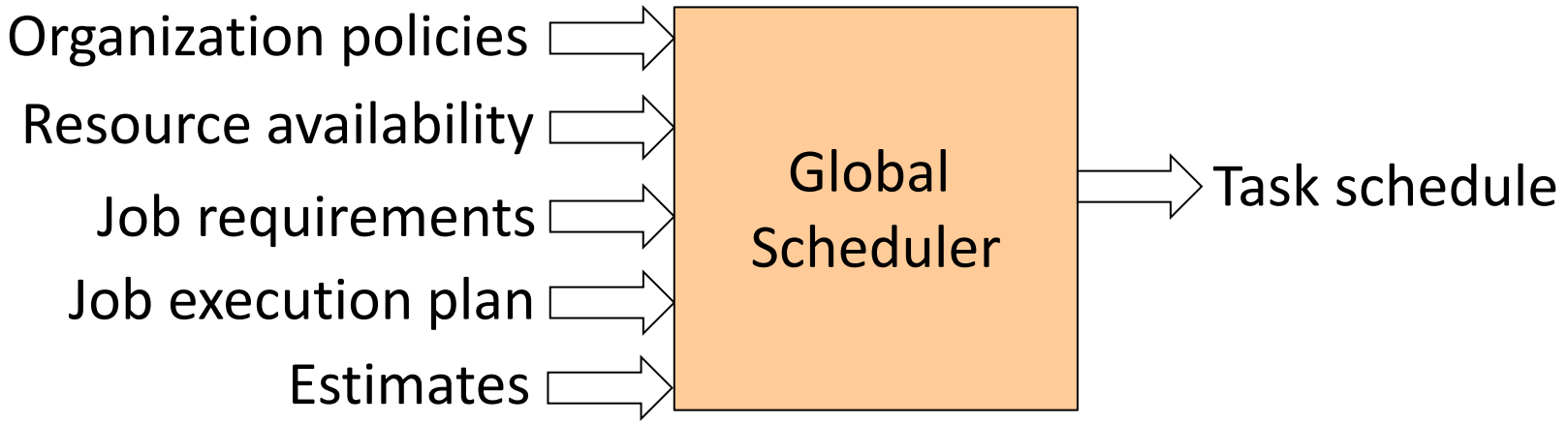
Baseline Approach: Global Scheduler



Baseline Approach: Global Scheduler

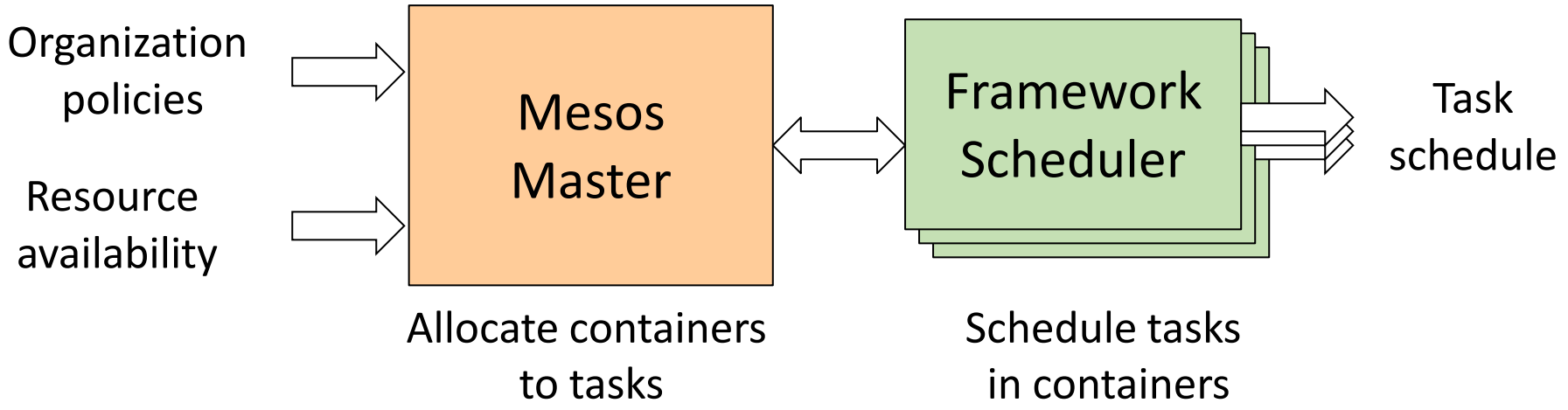


Baseline Approach: Global Scheduler



- Advantages:
  - Can achieve optimal schedule
- Disadvantages:
  - Complex, hard to scale and ensure resilience
  - Hard to anticipate future framework requirements
  - Need to refactor existing frameworks

Mesos: Hierarchical Scheduler

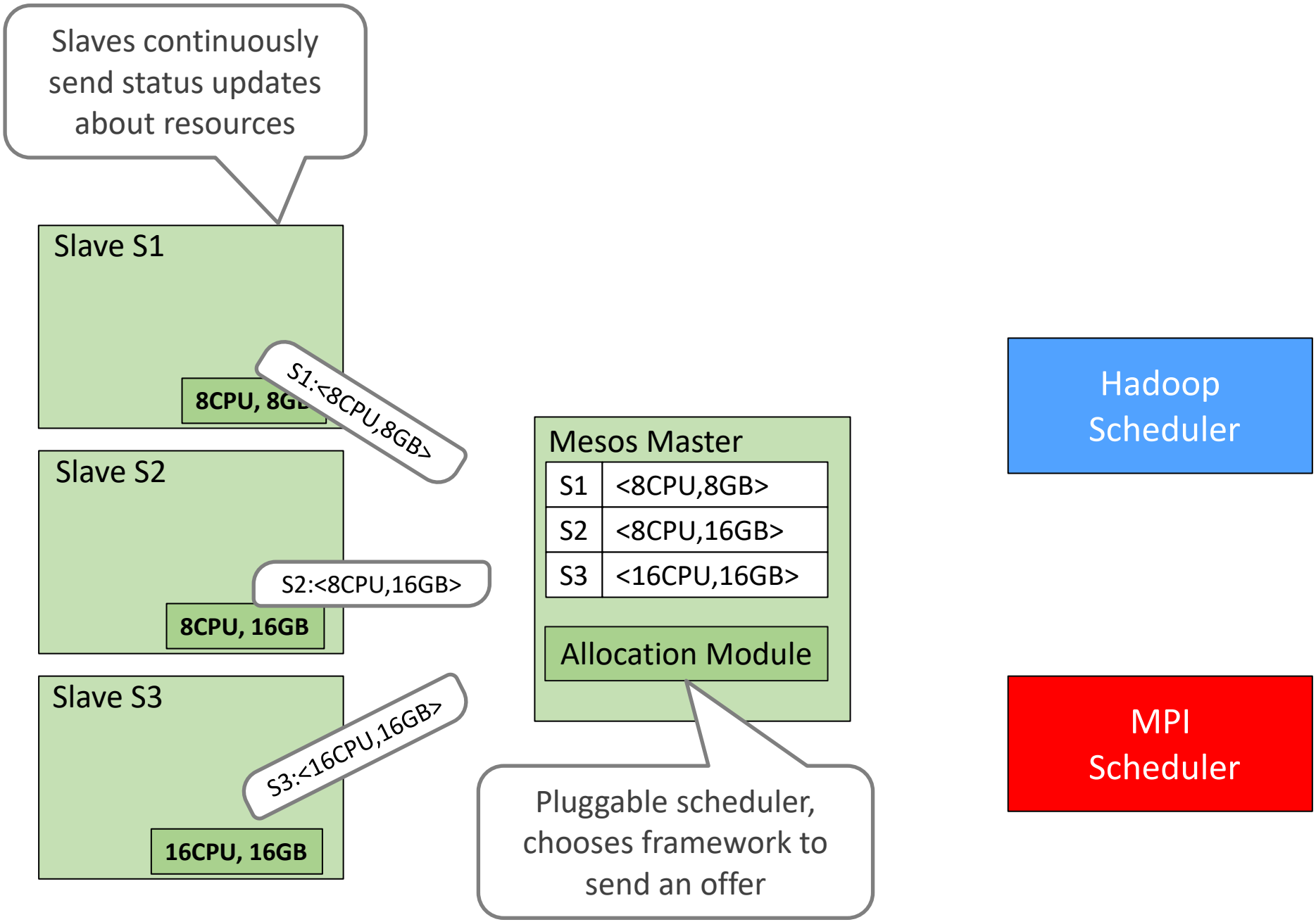


- Advantages:
  - Simpler, easier to scale and make resilient
  - Easy to port existing frameworks, support new ones
- Disadvantages:
  - Distributed scheduling decision may not be optimal

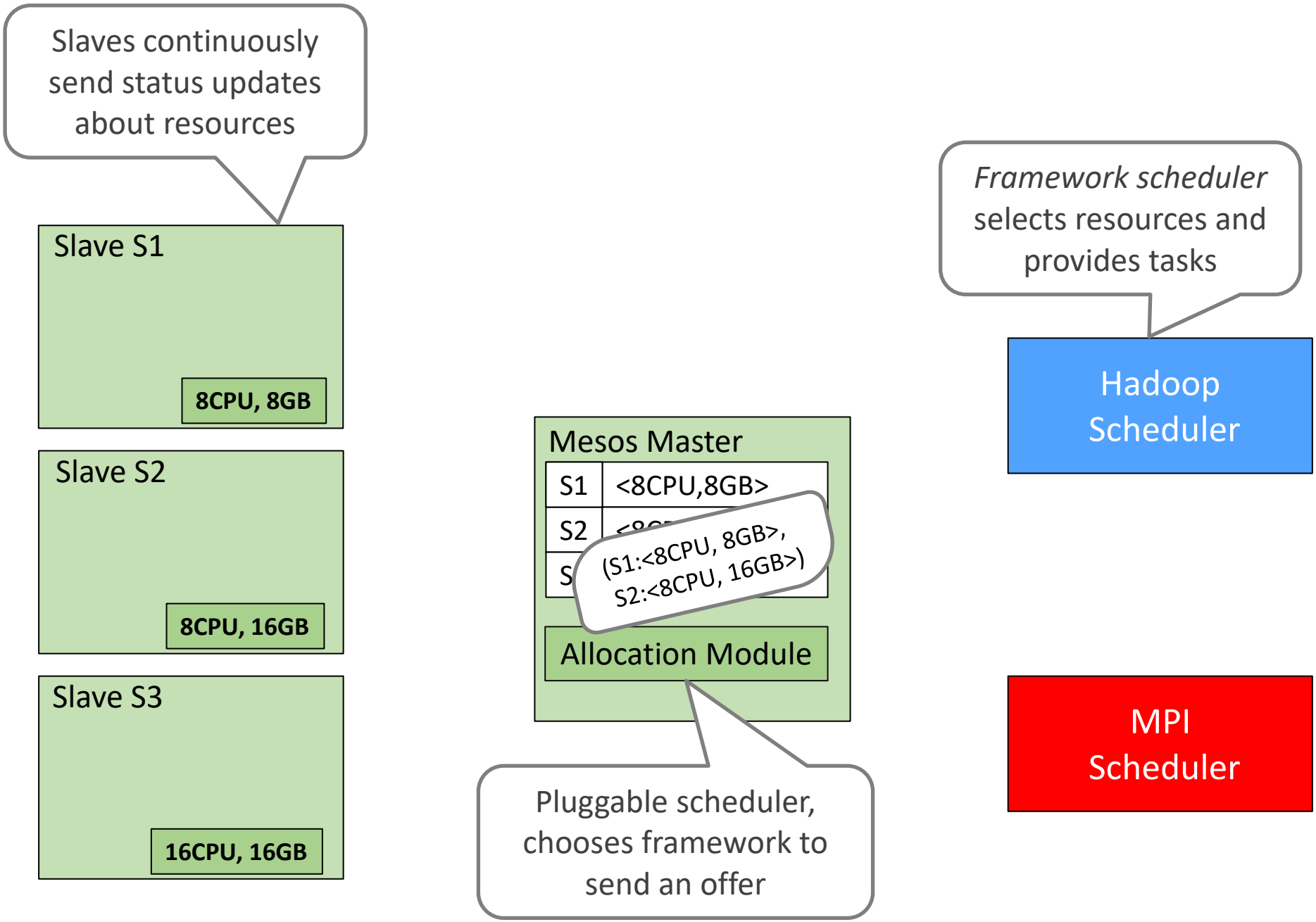
Mesos Approach: Resource Offers

- Master sends resource offers to frameworks
  - A resource offer is a vector of available resources on a node
    - E.g., node1: <1CPU, 1GB>, node2: <4CPU, 16GB>
- Frameworks choose whether to accept offer or not
  - On accepting offer, framework decides which tasks to run
  - Approach pushes task scheduling to frameworks

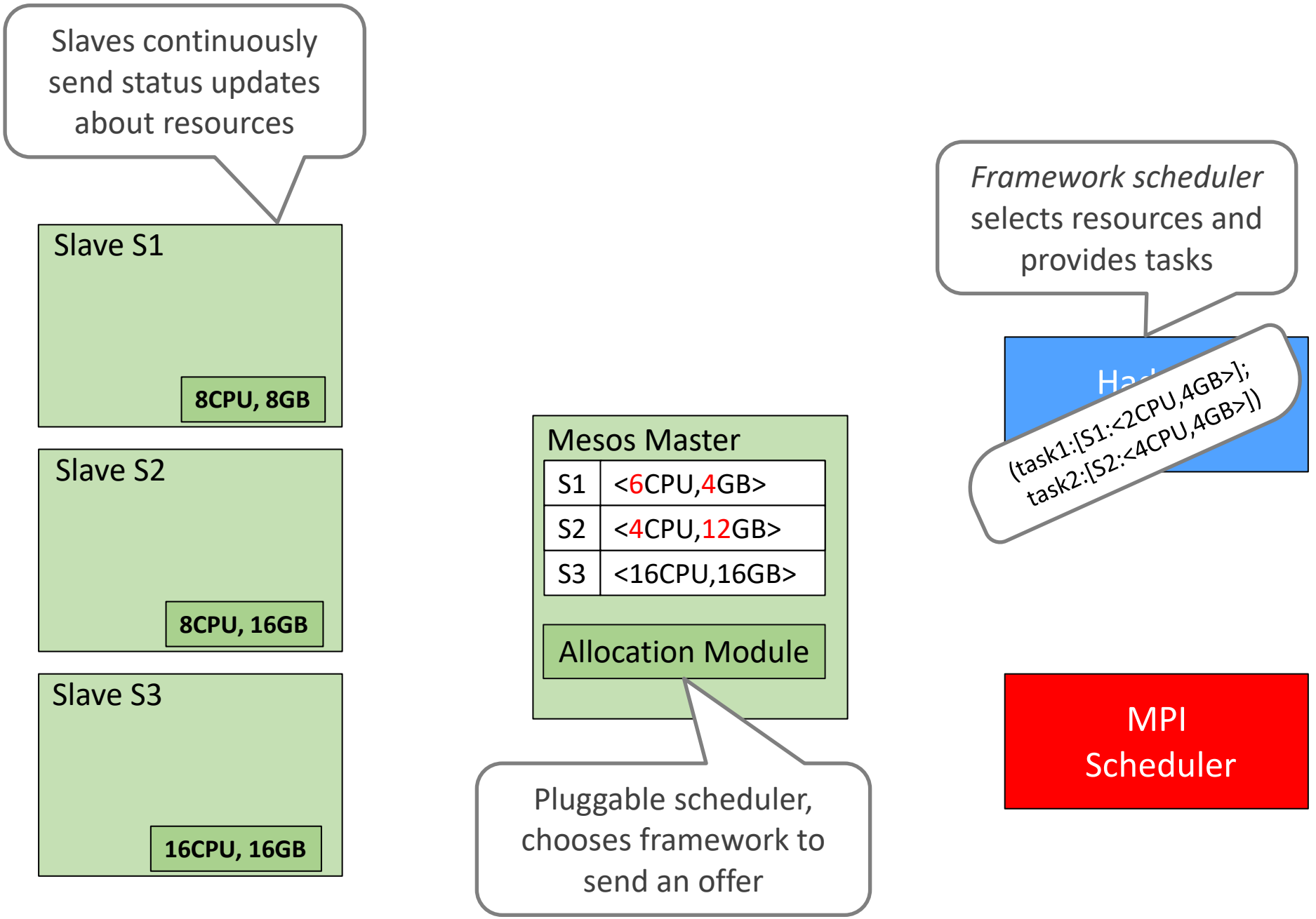
Mesos Example



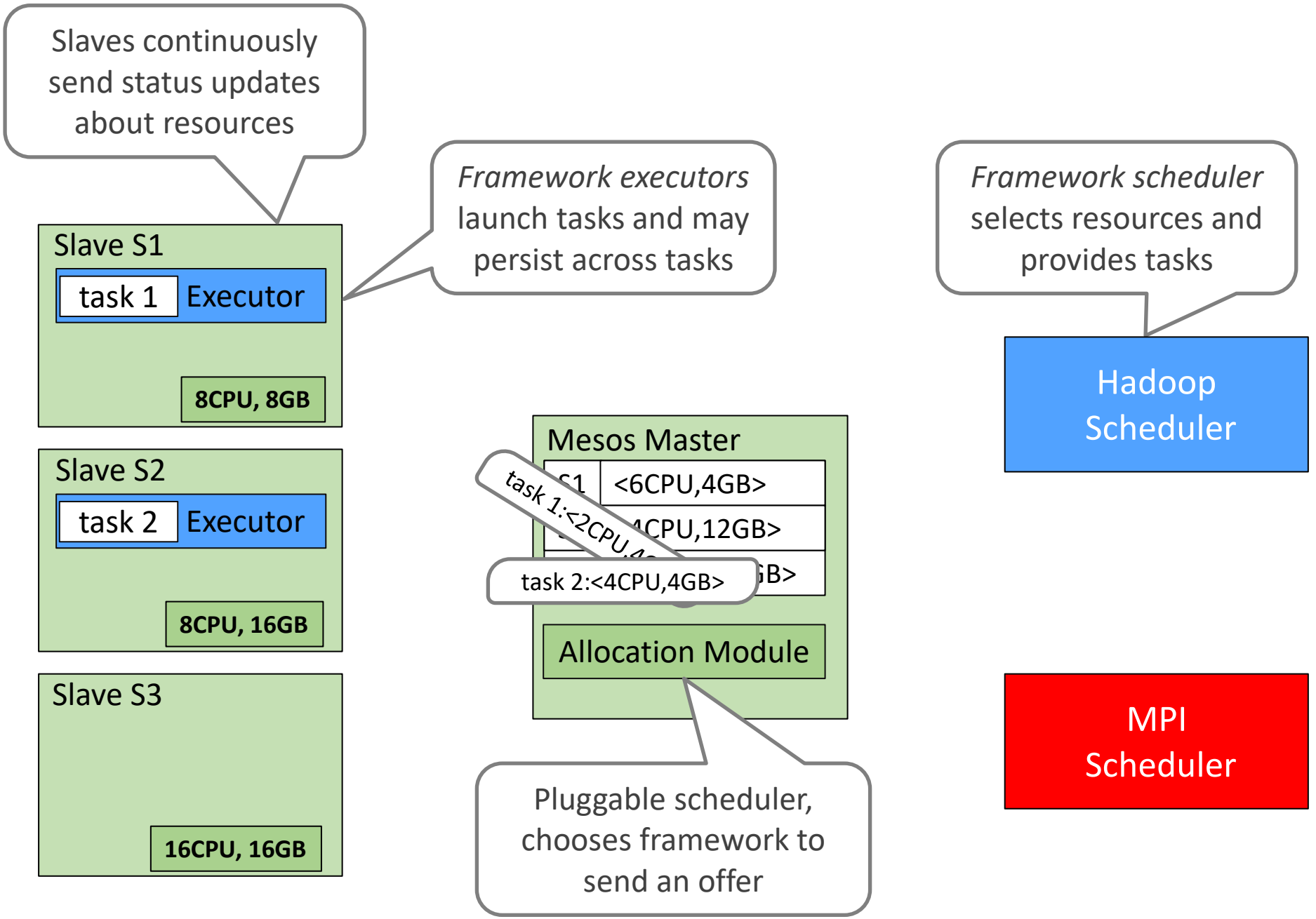
Mesos Example



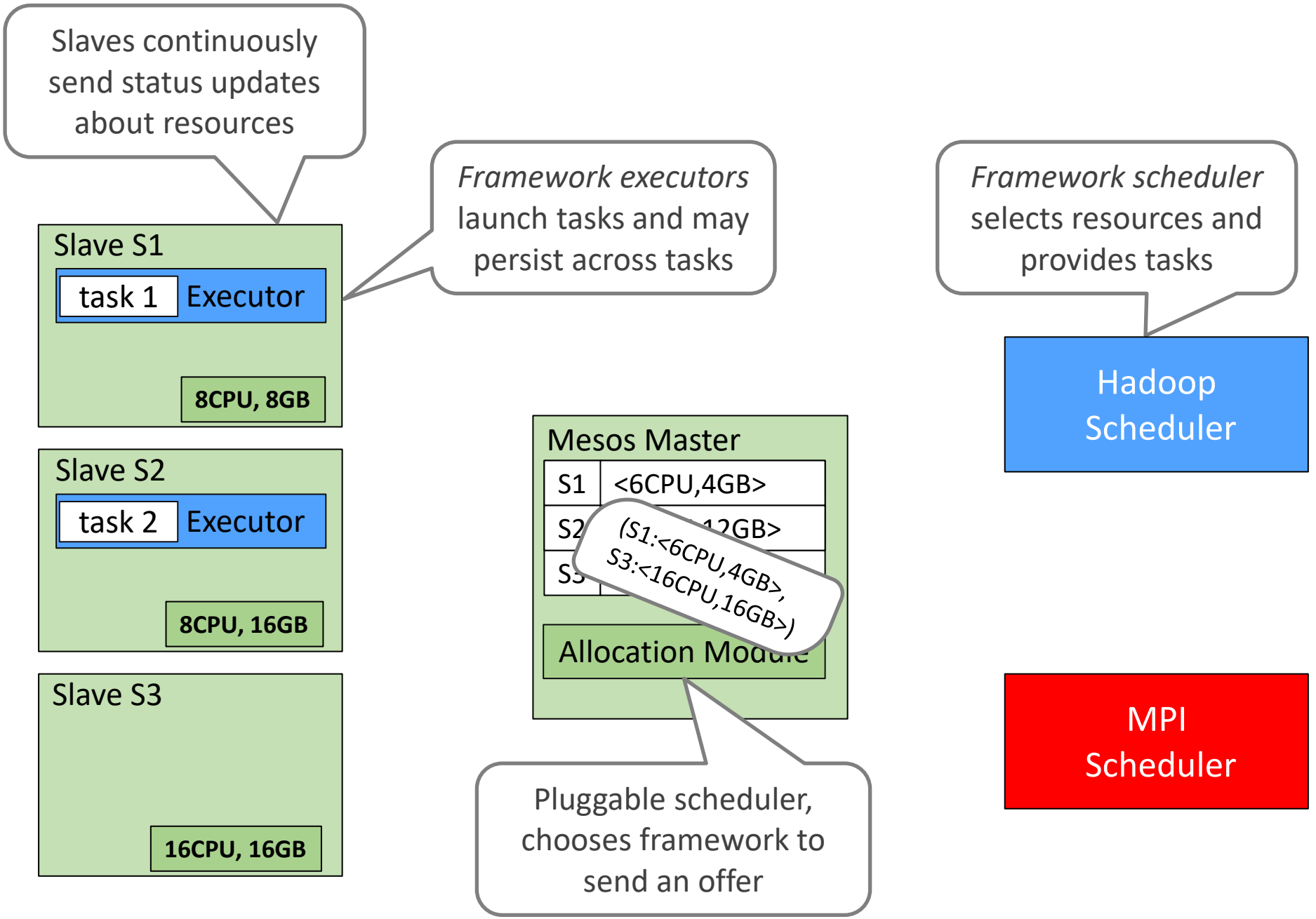
Mesos Example



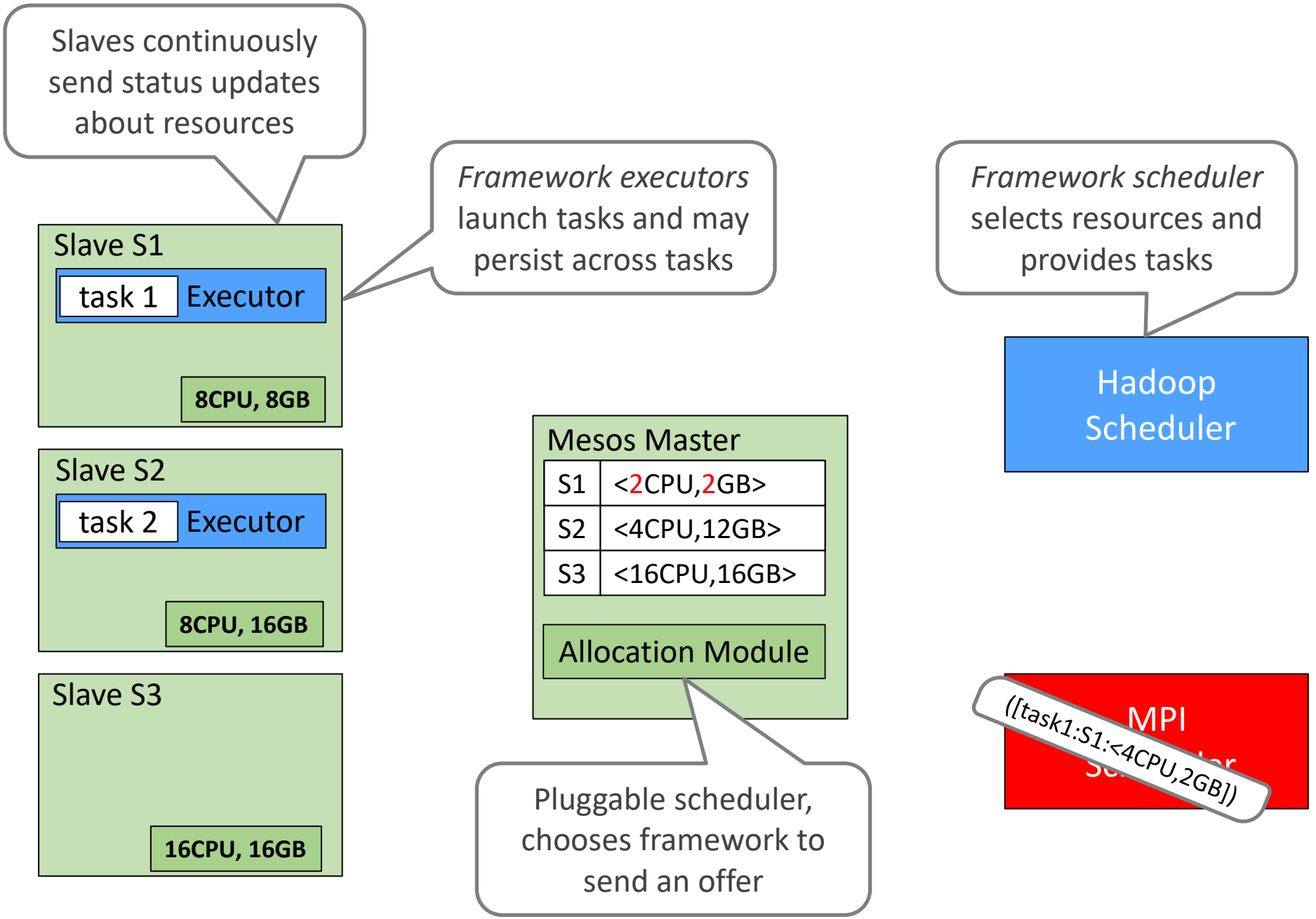
Mesos Example



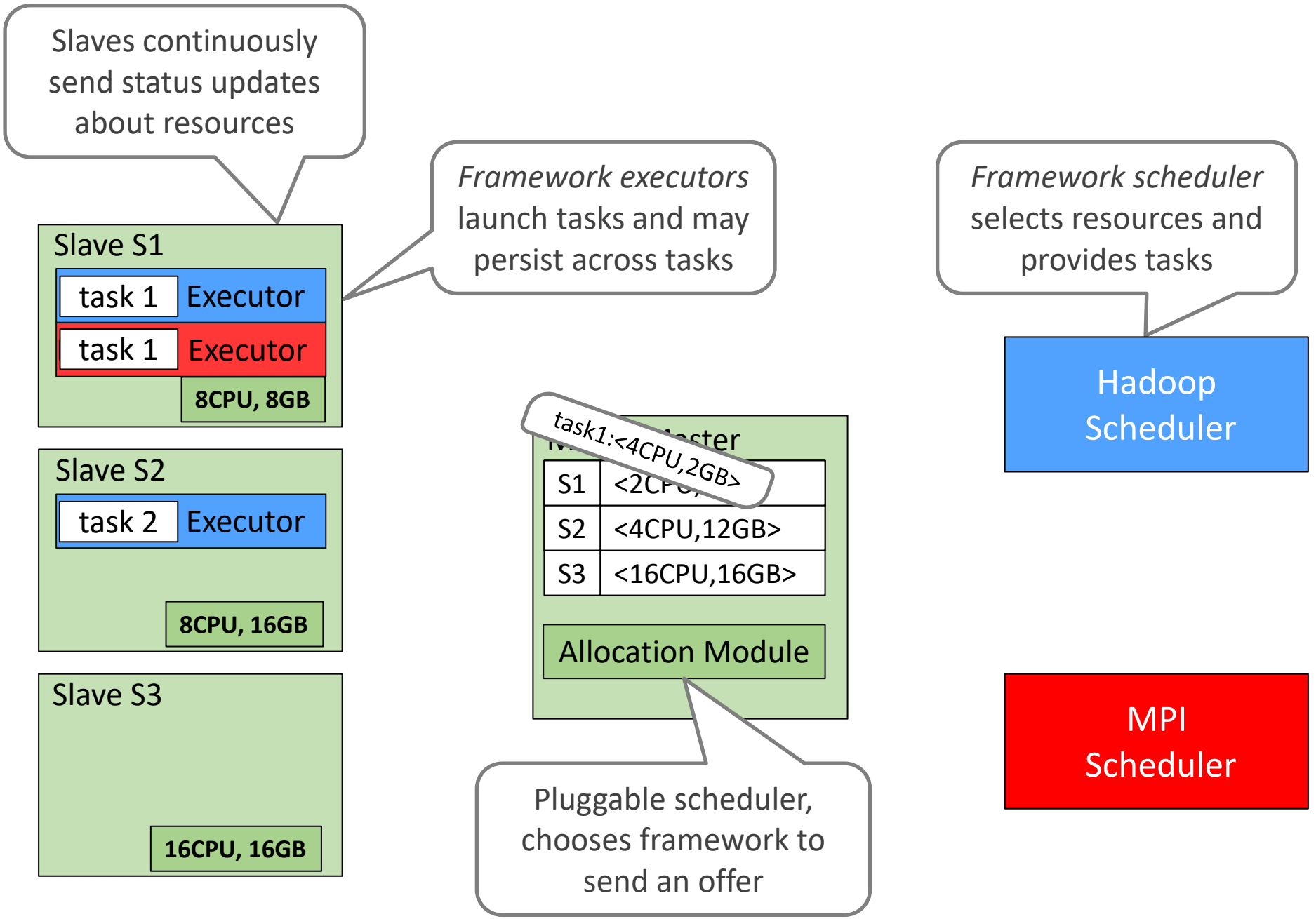
Mesos Example



Mesos Example

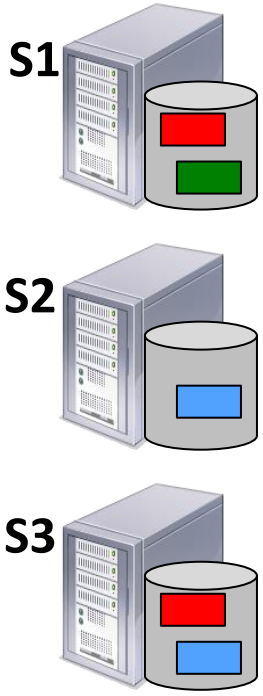


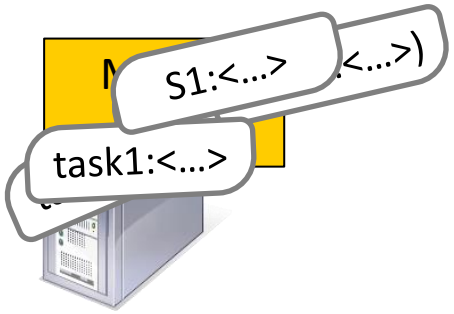
Mesos Example

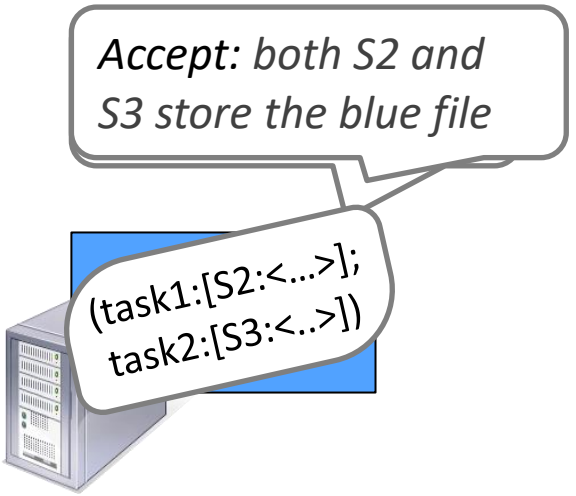


Why Do Resource Offers Work?

- A framework can just wait for an offer that matches its constraints!
  - It can reject offers it does not like
- Example: Hadoop's job input is blue file







Optimization: Filters

- Frameworks can short-circuit rejection by providing a predicate on resources to be offered
  - E.g., offer me “nodes from list L” or “nodes with > 8 GB RAM”
- Ability to reject still ensures correctness when needs cannot be expressed using filters

Mesos API

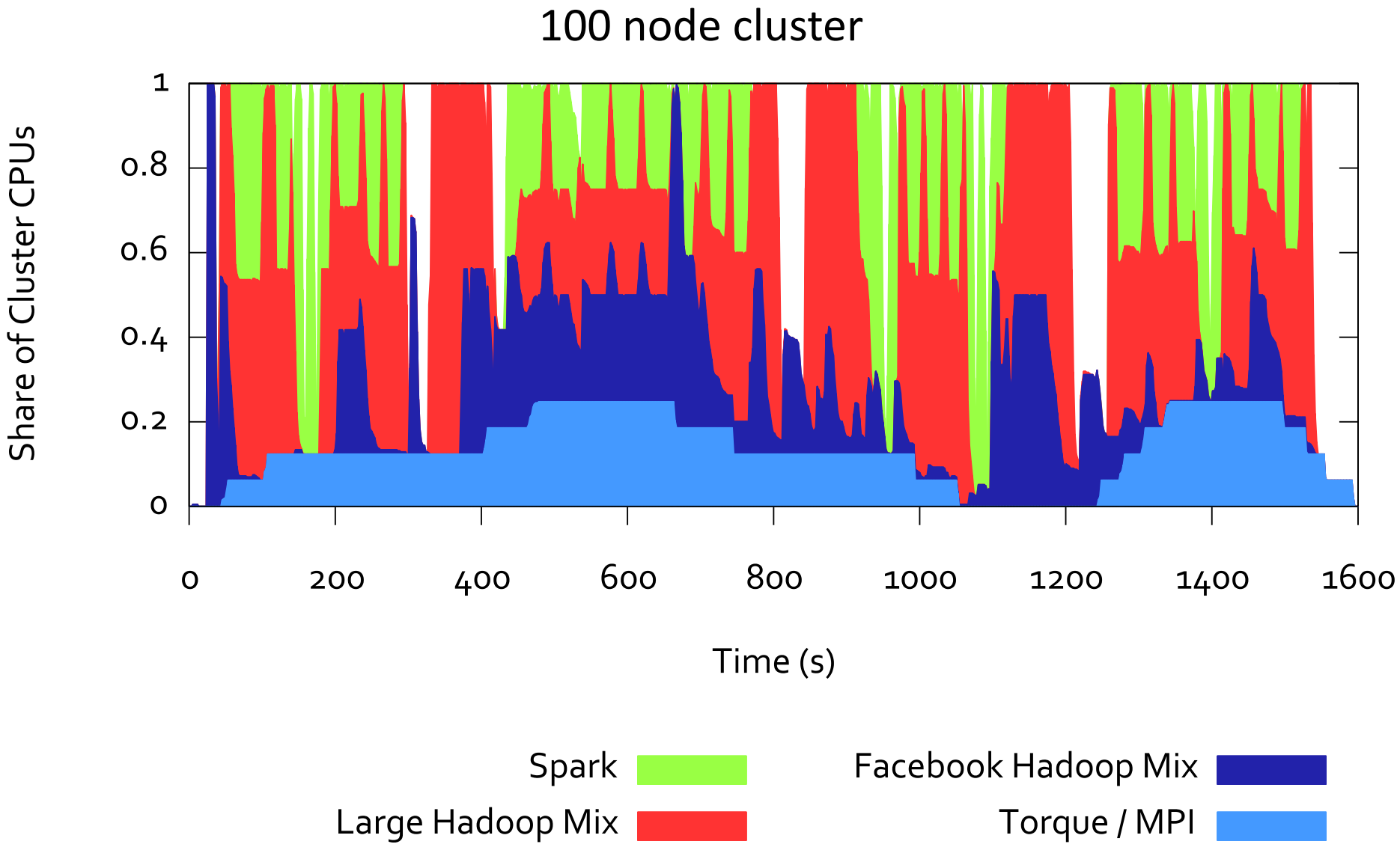
Scheduler Callbacks	Scheduler Actions
resourceOffer(offerId, offers) offerRescinded(offerId) statusUpdate(taskId, status) slaveLost(slaveId)	replyToOffer(offerId, tasks) setNeedsOffers(bool) setFilters(filters) getGuaranteedShare() killTask(taskId)

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

Failure Recovery

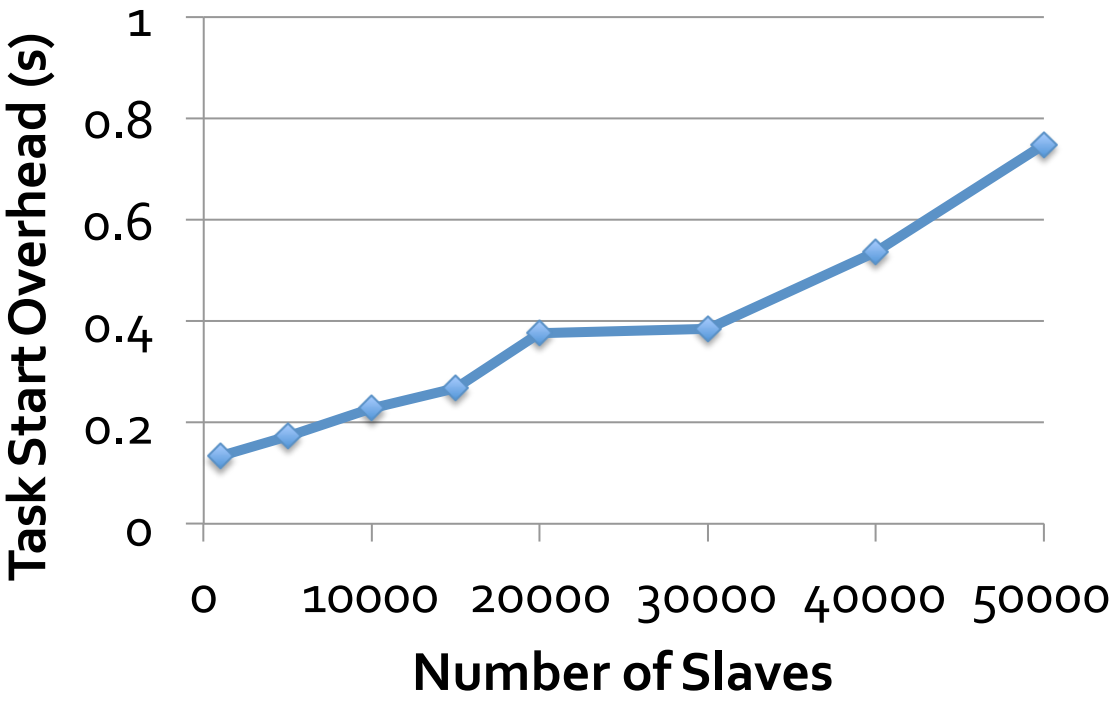
- Mesos master only keeps soft state
  - List of currently running frameworks, slave nodes, available resources and tasks
- Master uses zookeeper for leader election
- After master failure, new master rebuilds state when frameworks and slaves re-register with new master
- Fault detection and recovery in ~10 sec

Evaluation


100 node cluster
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

Scalability

- Mesos only performs inter-framework scheduling (e.g., fair sharing), easier than intra-framework scheduling
- Result: Scaled to 50,000 emulated slaves, 200 frameworks, 100K tasks (30s task length)



Conclusions

- Mesos shares cluster among different frameworks efficiently with two key design ideas
  - Fine-grained sharing at the level of tasks
  - Use hierarchical scheduling
    - Resource manager offers resources to frameworks
    - Frameworks control their own task scheduling
- Enables co-existence of current frameworks and development of newer ones
- Hundreds of deployments in productions
  - E.g., Twitter, GE, Apple
  - Managing 10K node datacenters!

Discussion

Q1

- Mesos offers resources to frameworks, and frameworks accept or reject these offers. For what types of frameworks will this approach work well?

Q2

- Will the Resource Offers approach work well for tasks with small versus large resource requirements? (think about a restaurant that needs to serve small versus large groups)

Q3

- How would you handle the problem in the previous slide?

Q4

- How can Mesos handle a framework that delays responding to a Resource Offer?

Q5

- How can Mesos handle a framework that never releases resources?