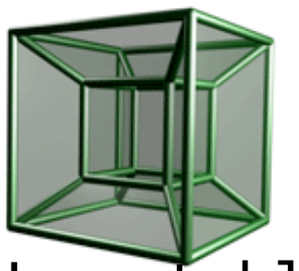
1. Metrics / goals for scheduling resources

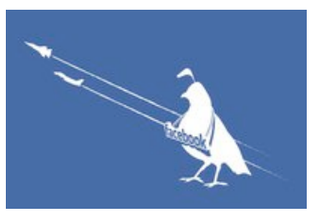
2. System architecture for big-data scheduling

Motivation

- Rapid innovation in cloud computing


Hypertable

Dryad

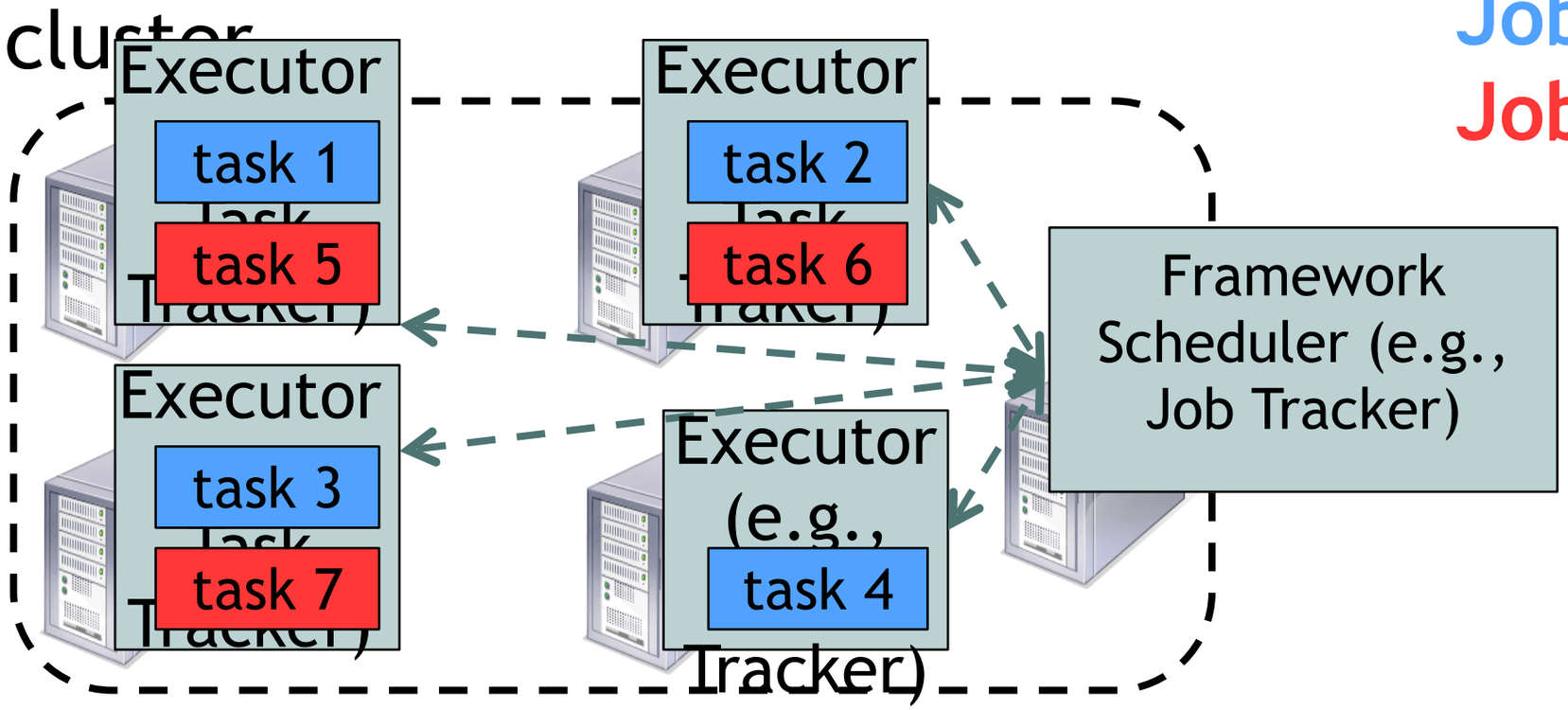

Cassandra


Pregel

- Today
  - No single framework optimal for all applications
  - Each framework runs on its dedicated cluster or cluster partition

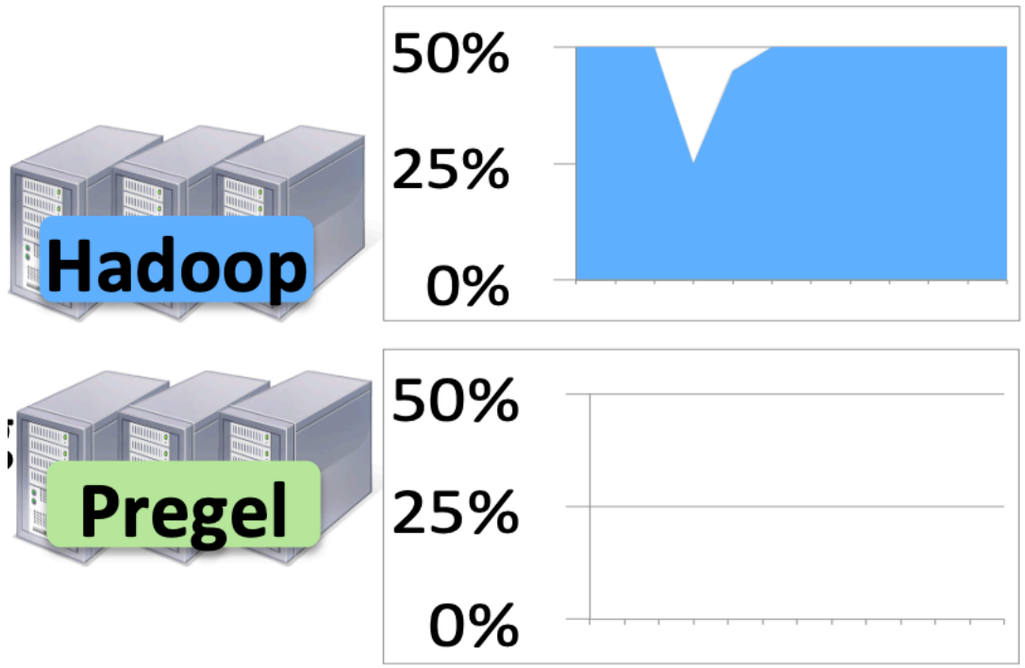
Computation Model: Frameworks

- A ***framework*** (e.g., Hadoop, MPI) manages one or more ***jobs*** in a computer cluster
- A ***job*** consists of one or more ***tasks***
- A ***task*** (e.g., map, reduce) is implemented by one or more processes running on a single machine



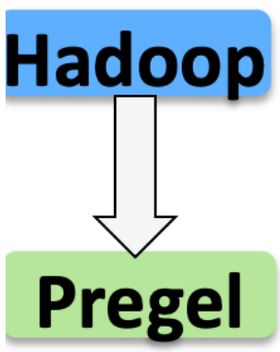
One Framework Per Cluster Challenges

- Inefficient resource usage
  - E.g., Hadoop cannot use available resources from Pregel's cluster
  - No opportunity for stat. multiplexing



- Hard to share data
  - Copy or access remotely, expensive

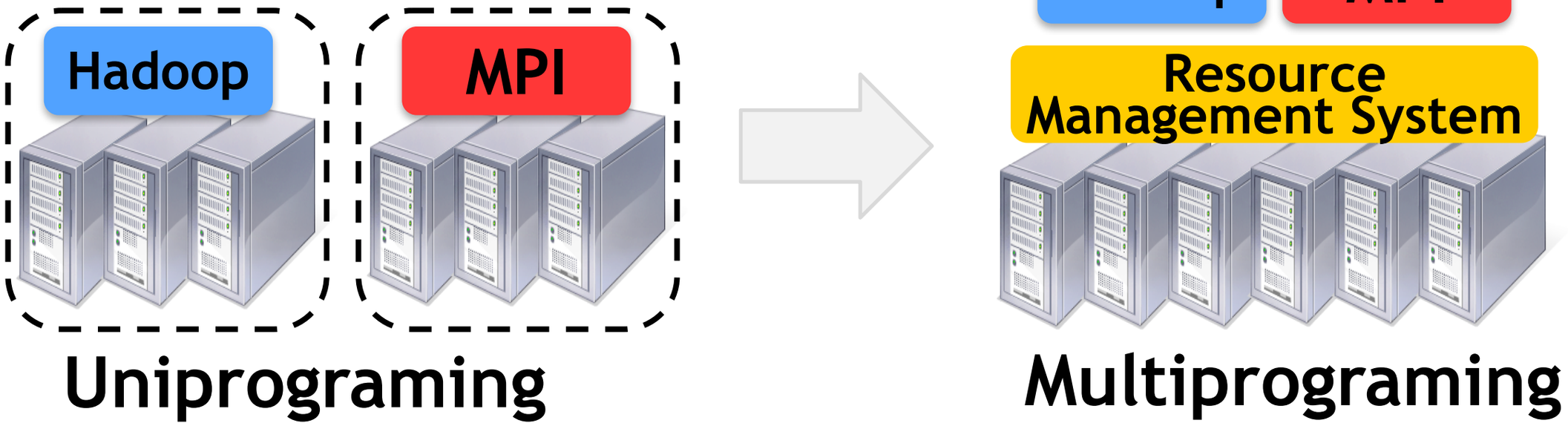
- Hard to cooperate
  - E.g., Not easy for Pregel to use graphs generated by Hadoop



Need to run multiple frameworks on same cluster

What do we want?

- Common resource sharing layer
  - Abstracts (“virtualizes”) resources to frameworks
  - Enable diverse frameworks to share cluster
  - Make it easier to develop and deploy new frameworks (e.g., Spark)



Goals

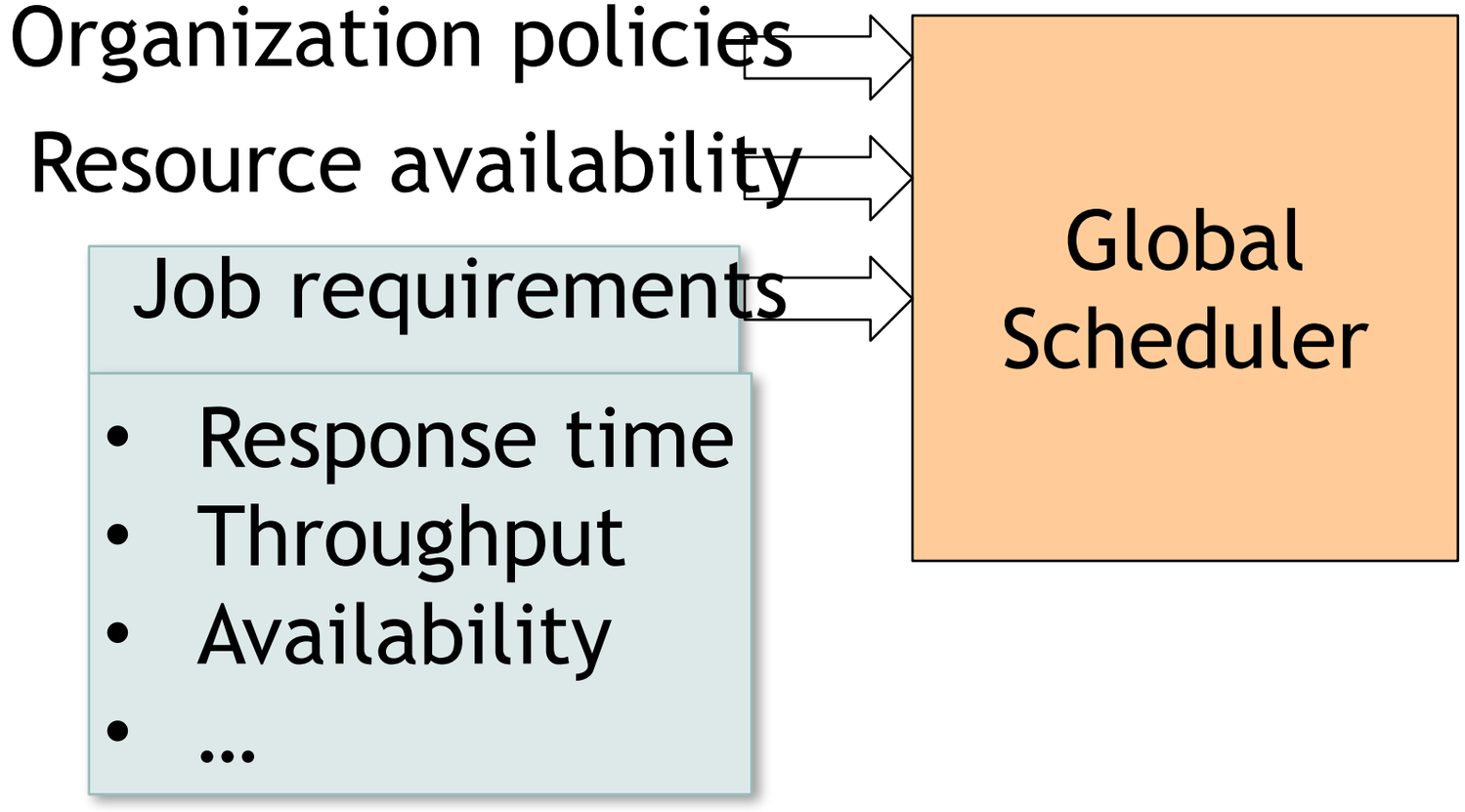
- **Efficient utilization** of resources
- **Support diverse frameworks** (existing & future)
- **Scalability** to 10,000's of nodes
- **Reliability** in face of node failures

Fine Grained Resource Sharing

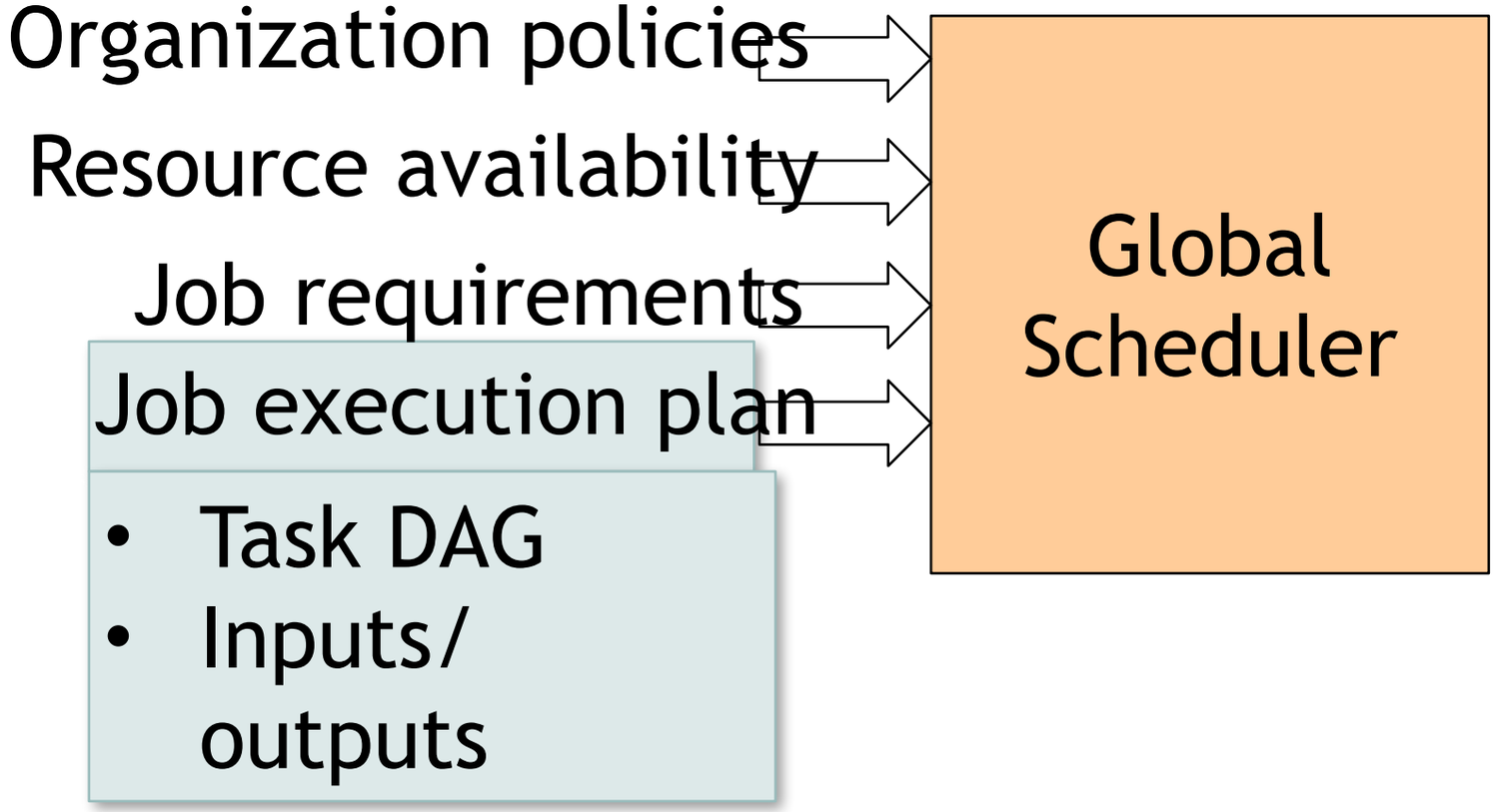
- Task granularity both in **time** & **space**
  - Multiplex node/time between tasks belonging to different jobs/frameworks
- Tasks typically short; median ~= 10 sec, minutes

- Why fine grained?
  - Improve data locality
  - Easier to handle node failures

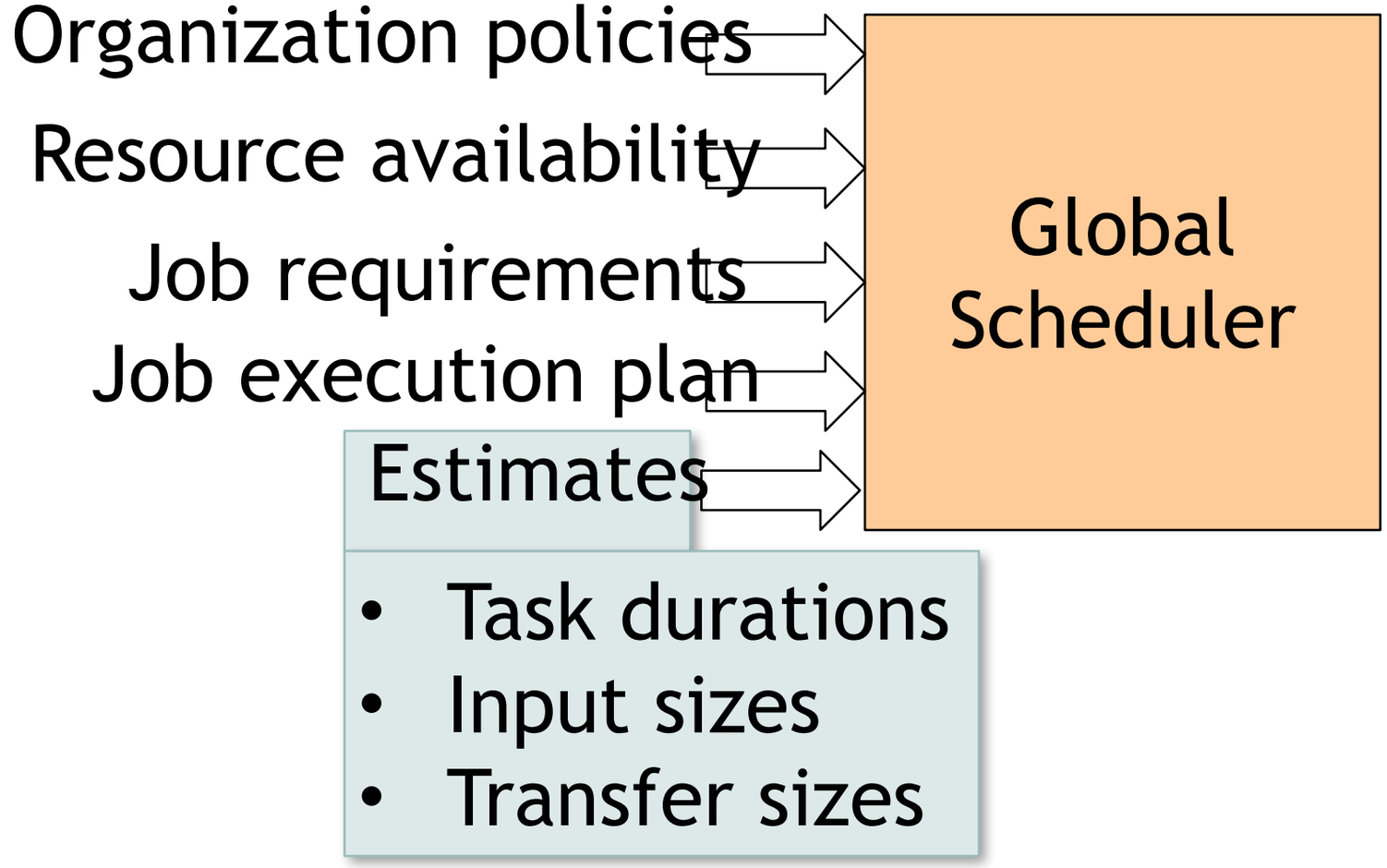
Approach: Global Scheduler



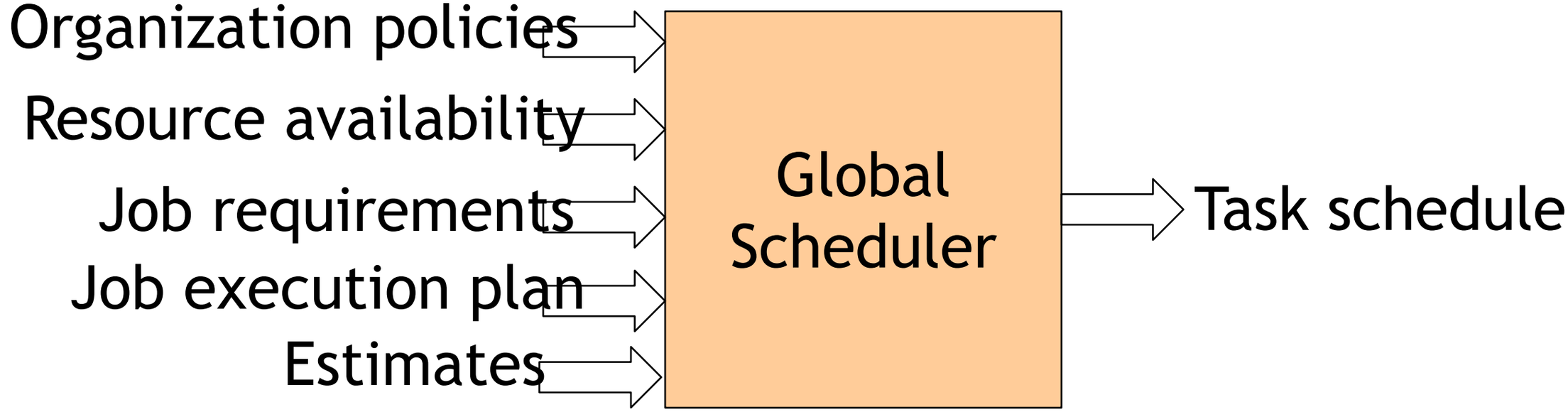
Approach: Global Scheduler



Approach: Global Scheduler



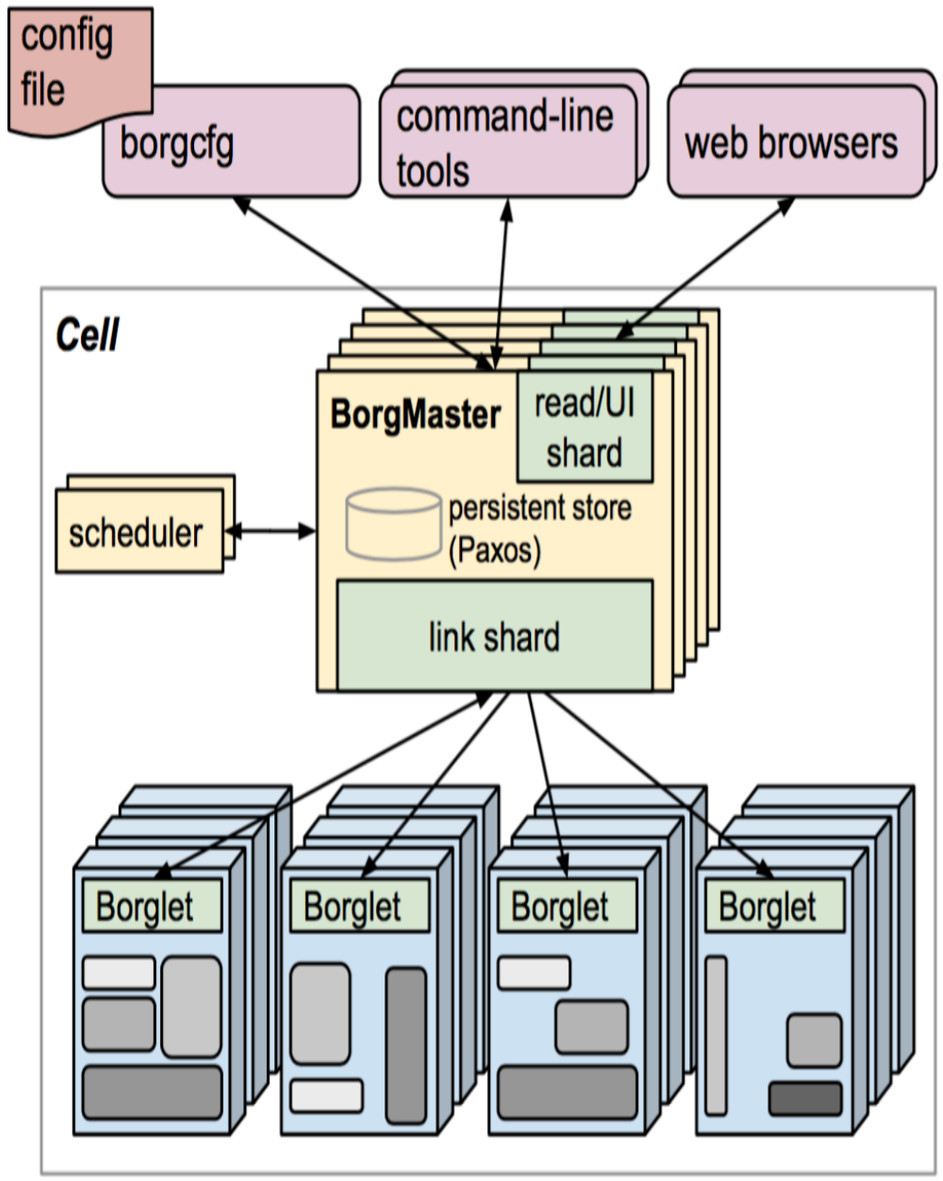
Approach: Global Scheduler



- Advantages: can achieve optimal schedule
- Disadvantages:
  - Complexity → hard to scale and ensure resilience
  - Hard to anticipate future frameworks' requirements
  - Need to refactor existing frameworks

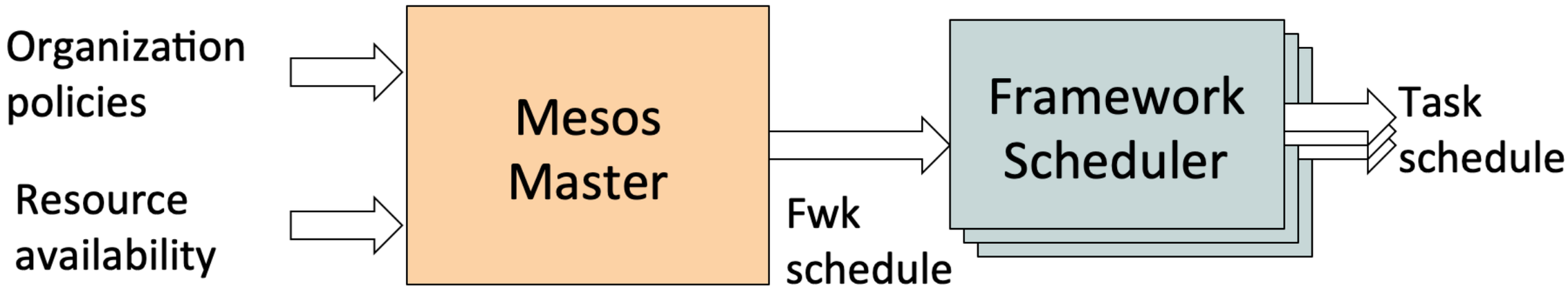
Borg Architecture

- Centralized Borgmaster + Localized Borglet (manage/monitor tasks)
- Goal: Find machines for a given job
- Used across all Google services
  - – Services: Gmail, web search, GFS
  - – Analytics: MapReduce, streaming
  - Framework controller sends master allocation request to Borg for full job



Mesos

Distributed Scheduler



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

Resource Offers

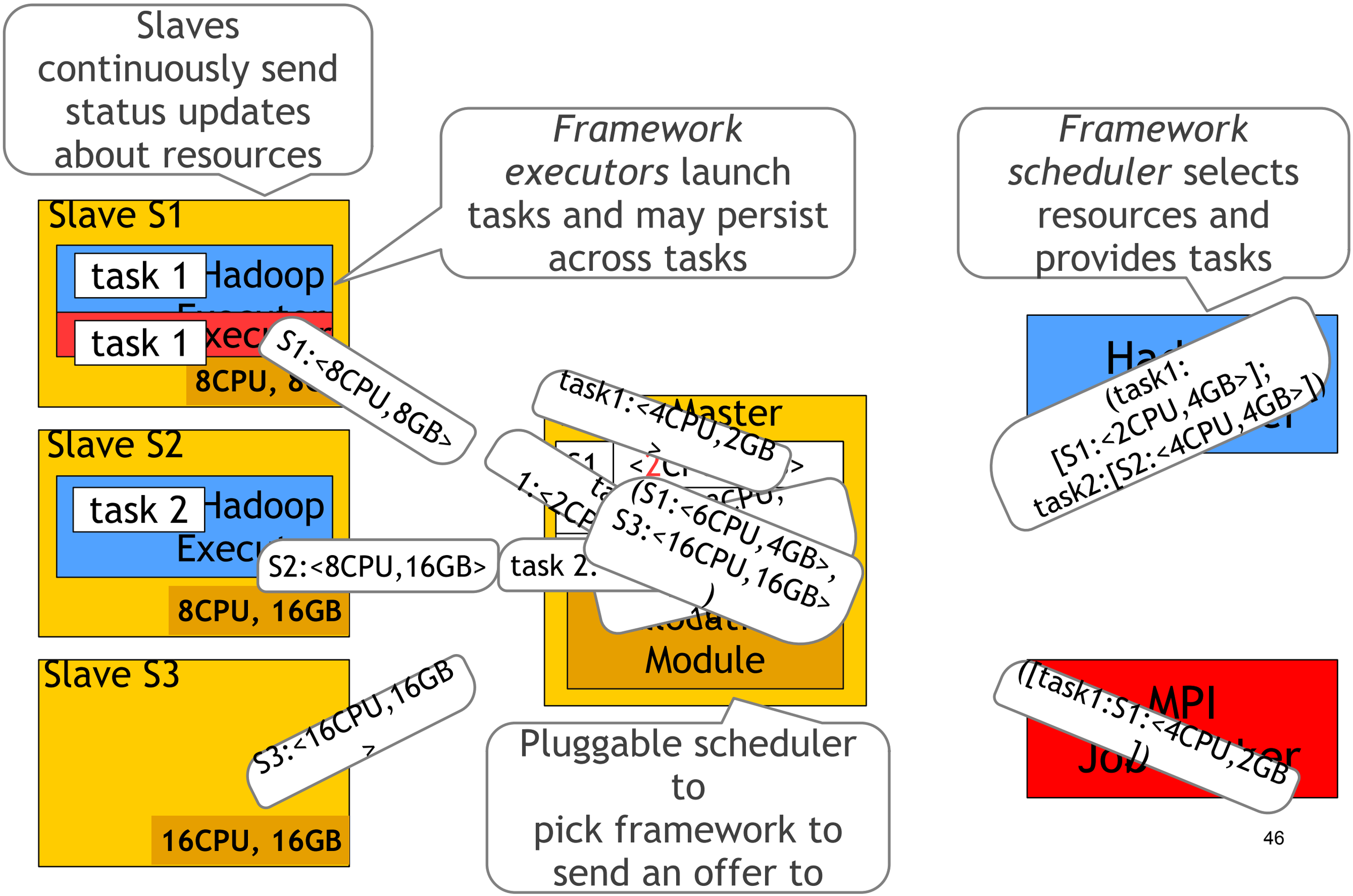
- Unit of allocation: ***resource offer***
  - Vector of available resources on a node
  - E.g., node1: <1CPU, 1GB>, node2: <4CPU, 16GB>

- Master sends resource offers to frameworks

- Frameworks select which offers to accept and which tasks to run

Push task scheduling to frameworks

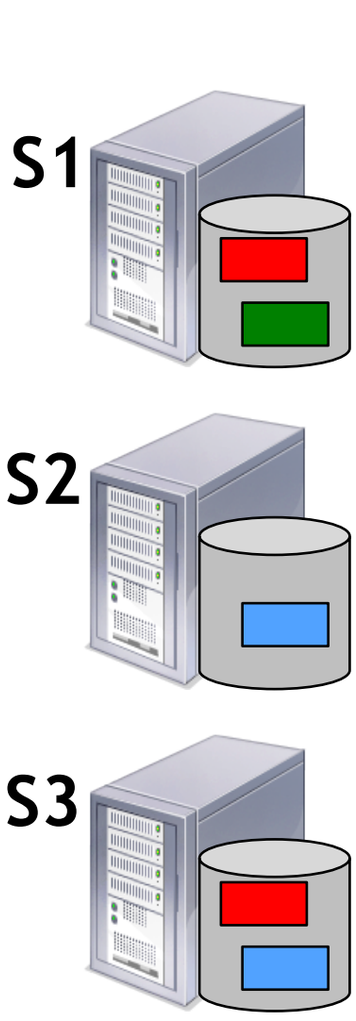
Mesos Architecture: Example

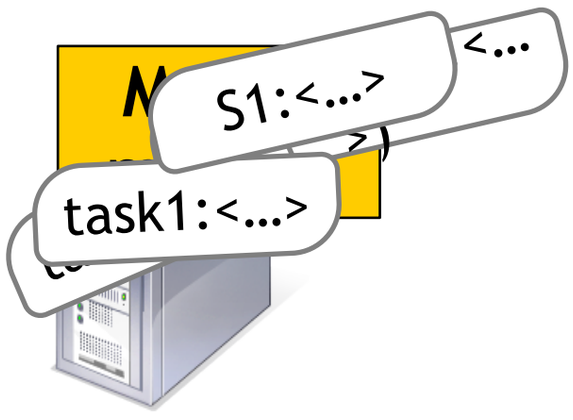


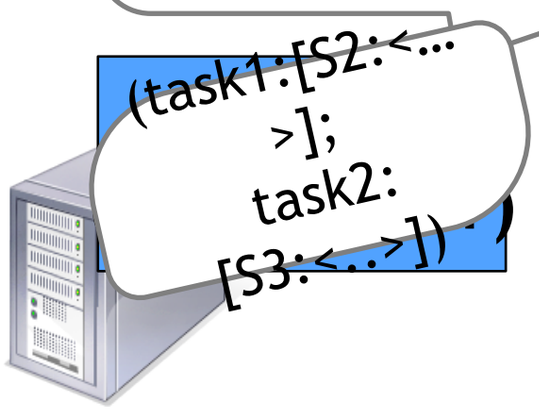
Why does it Work?

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

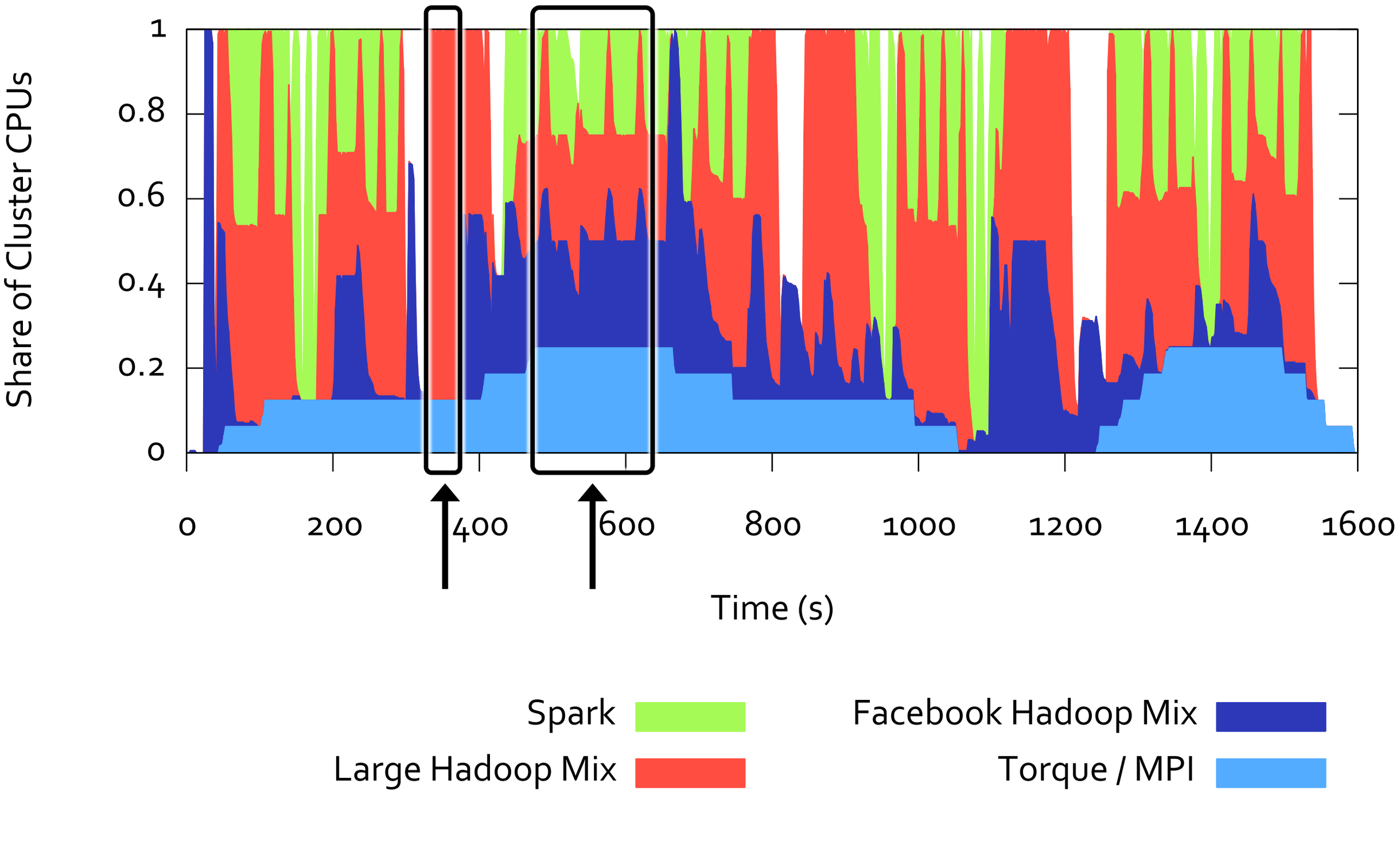
Accept: *both S2 and S3 store the blue file*







Dynamic Resource Sharing



Performance

- Ramp-up time low under most scenarios
- Barely any performance differences between global and distributed schedulers in Facebook workload
- Optimizations
  - Master doesn’t send an offer already rejected by a framework (negative caching)
  - Allow frameworks to specify white and black lists of nodes

Mesos vs Static Partitioning

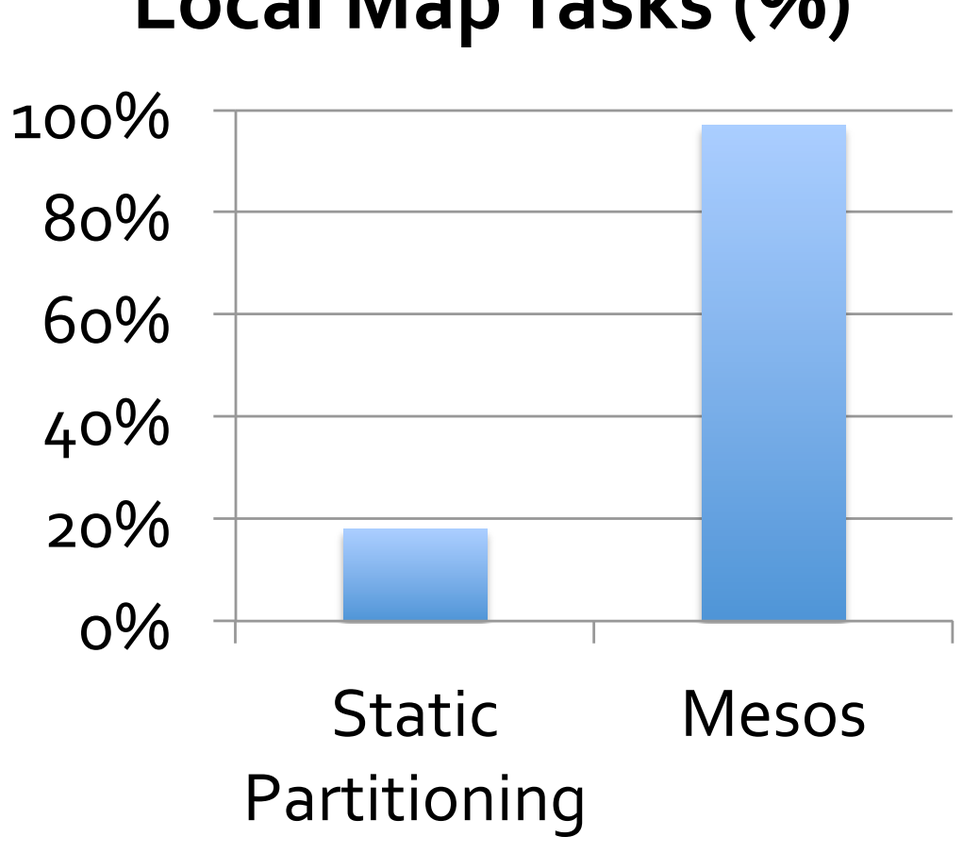
Compared performance with statically partitioned cluster where each framework gets 25% of nodes

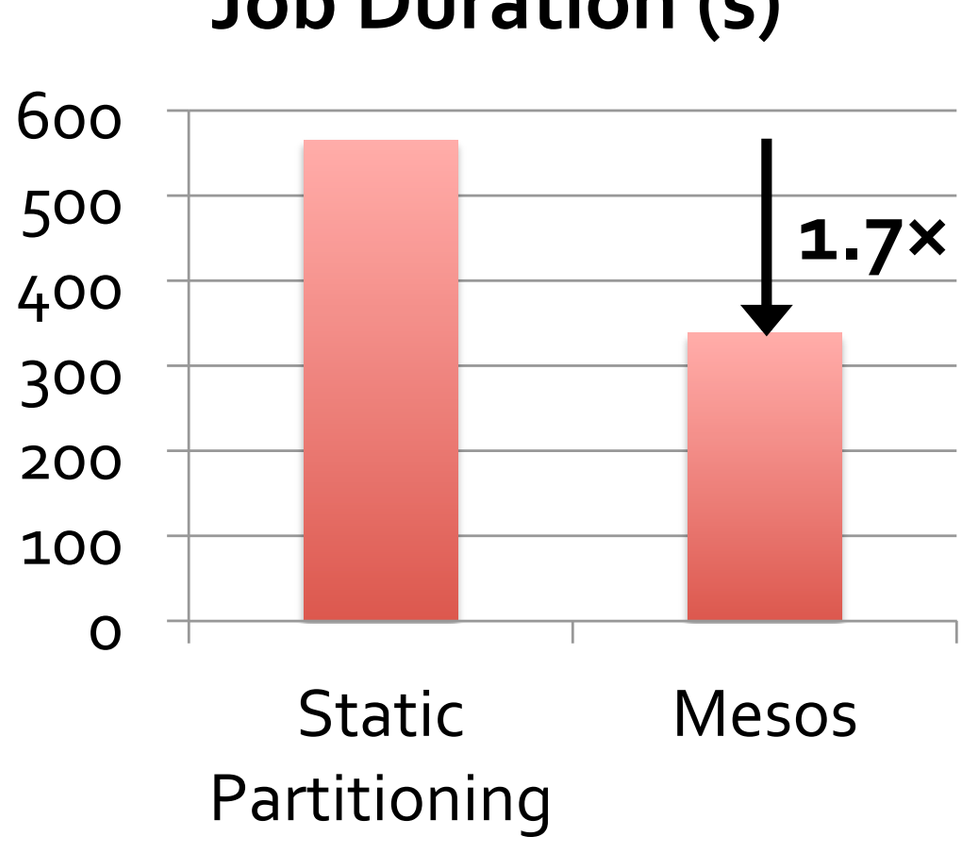
Framework	Speedup on Mesos
Facebook Hadoop Mix	1.14×
Large Hadoop Mix	2.10×
Spark	1.26×
Torque / MPI	0.96×

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)





Next: Mid-term Review