Cluster Management Systems

Ion Stoica

CS 294

September 26, 2016

Motivation

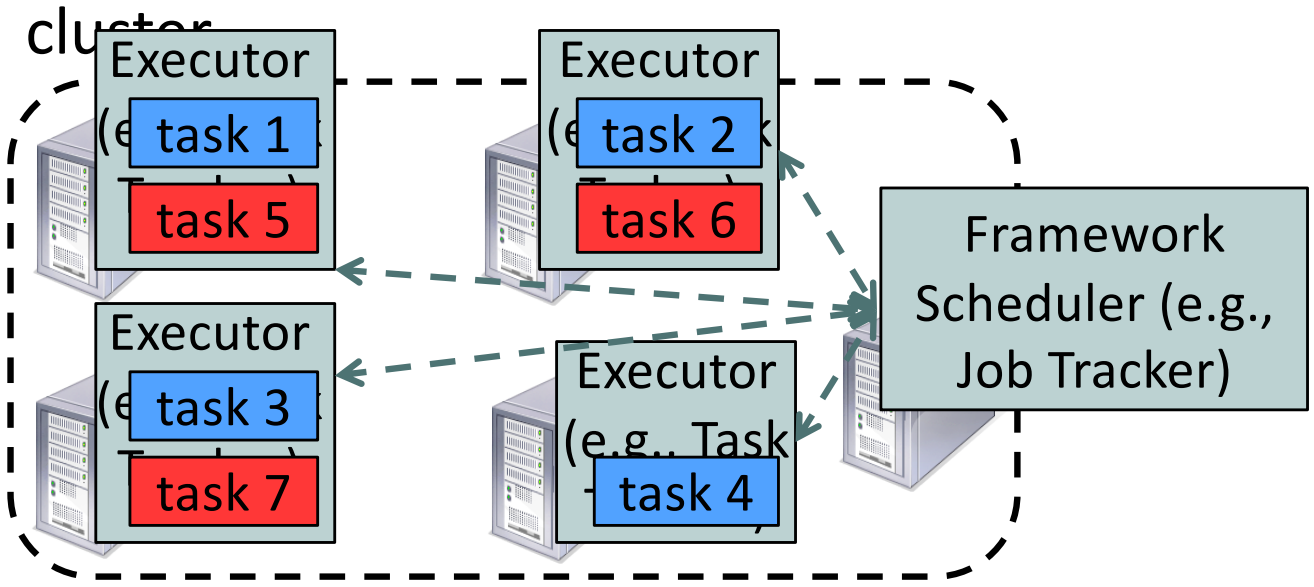
- Rapid innovation in cloud computing

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

2011 slide

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

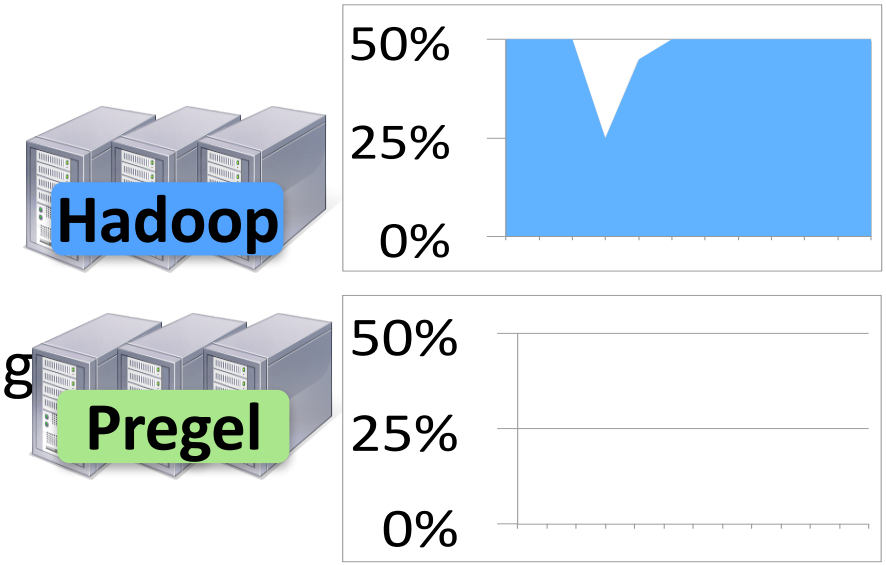


Job 1: tasks 1, 2, 3, 4
Job 2: tasks 5, 6, 7

2011 slide

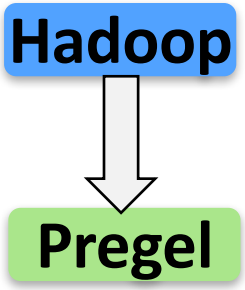
One Framework Per Cluster Challenges

- 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

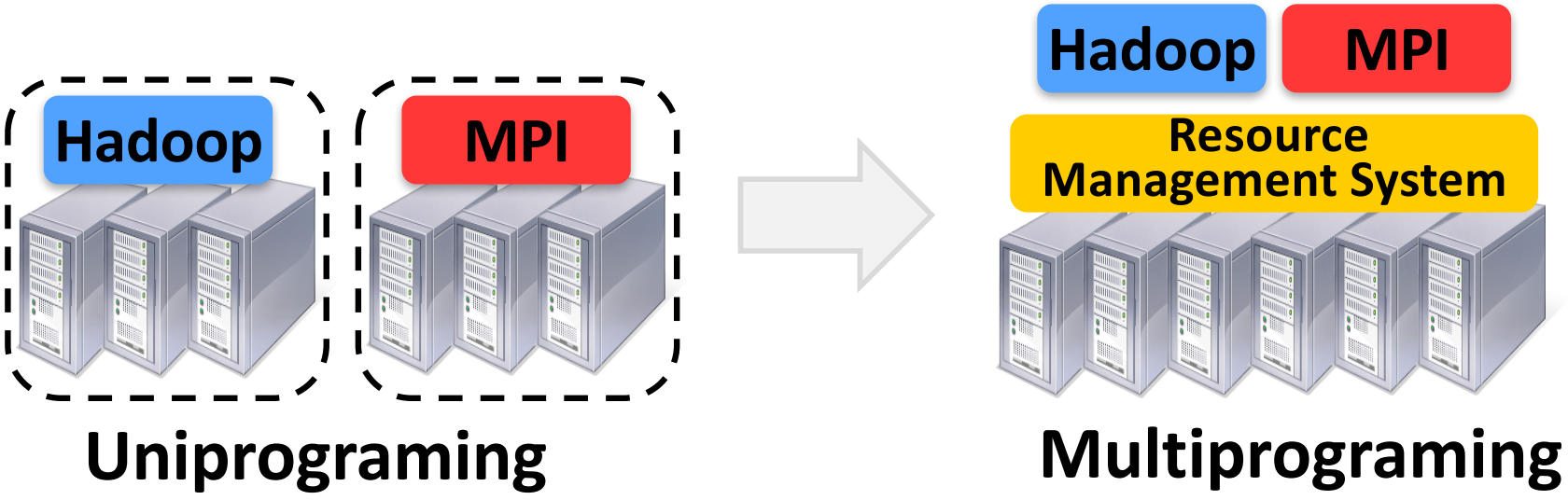
- Hard to cooperate
  - E.g., Not easy for Pregel to use graphs generated by Hadoop



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)



2011 slide

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

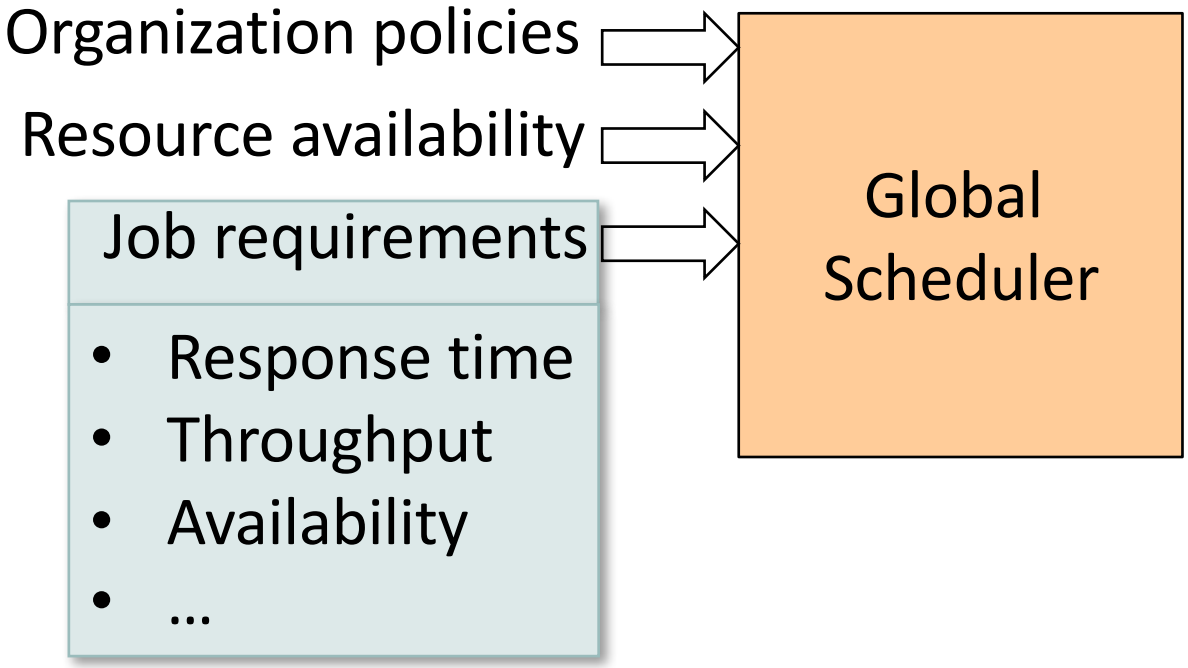
2011 slide

Goals

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

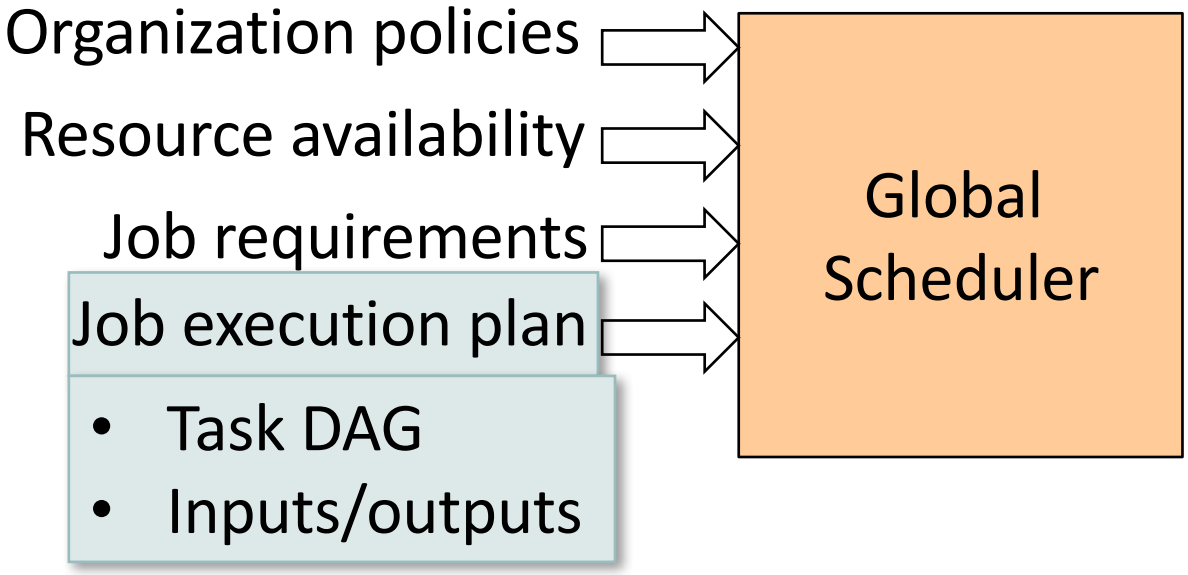
2011 slide

Approach: Global Scheduler



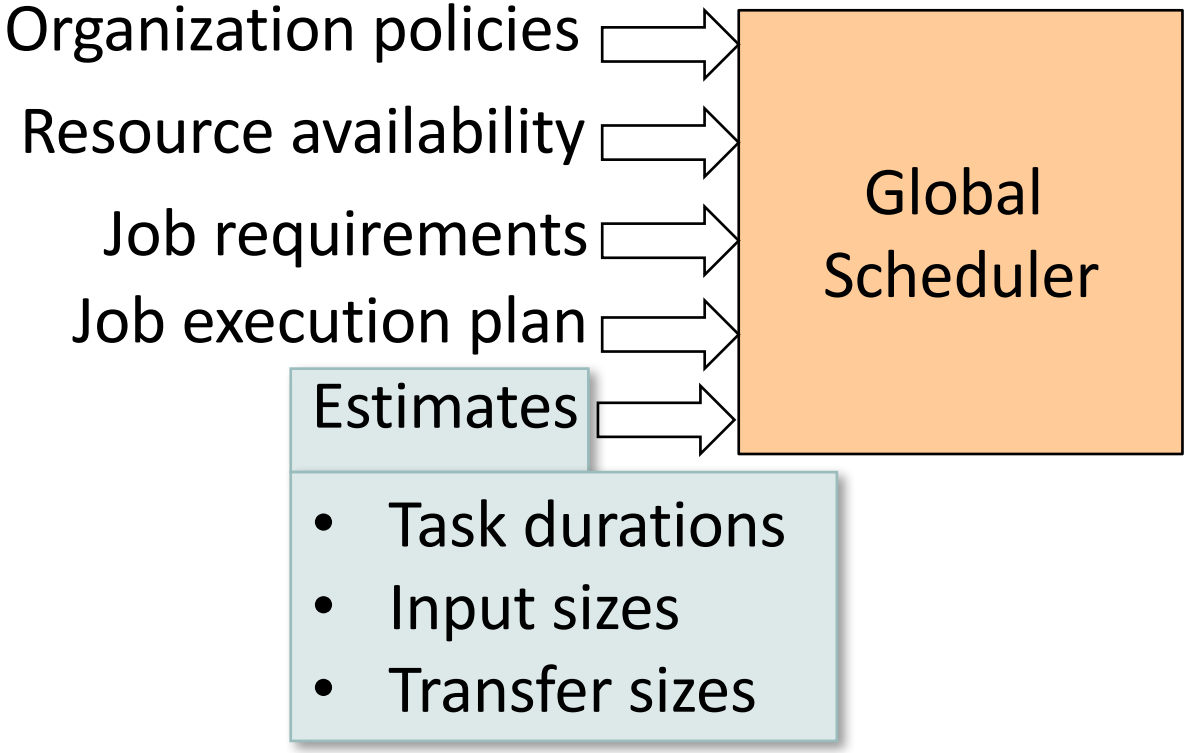
2011 slide

Approach: Global Scheduler



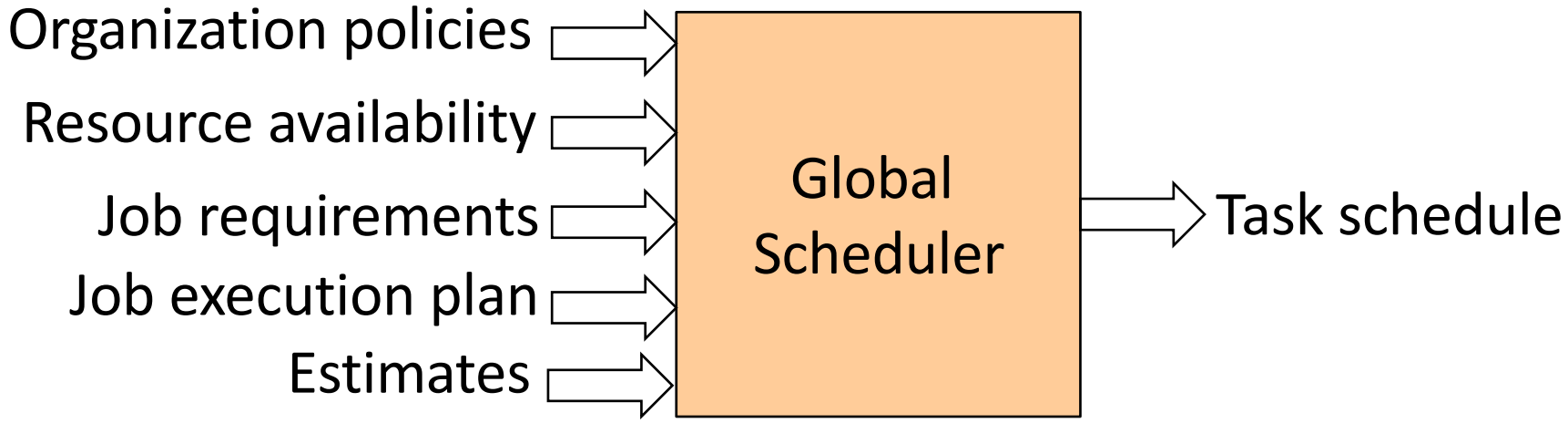
2011 slide

Approach: Global Scheduler



2011 slide

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

2011 slide

Motivations, Now and Then

- Recall main motivations in 2011:
  - Efficient resource usage
  - Enable rapid innovation
  - Main use case: big data frameworks

- What has changed since then?

What has Changed Since Then?

- Workloads: run long running applications (e.g., fron-end services) emerged as a common workload
  - No need for fine grain allocation

- Containers have evolved from an isolation mechanism to a packaging solution (e.g., Docker)
  - Container orchestration a major use case → manage entire app life cycle: develop, test, deploy

- Cloud has became prevalent
  - Easy to scale up using PAYG (statistic multiplexing less important)

How to classify systems?

- What is the granularity of resource sharing?
  - Tasks vs instance allocation
- How do they make resource management decisions?
  - Centralized vs. distributed allocation
  - Who maps resources to apps, and who maps resources to tasks?
  - What resource management polices they enforce?

This Lecture

- Mesos
- Yarn
- Omega (and Borg)
- Kubernetes