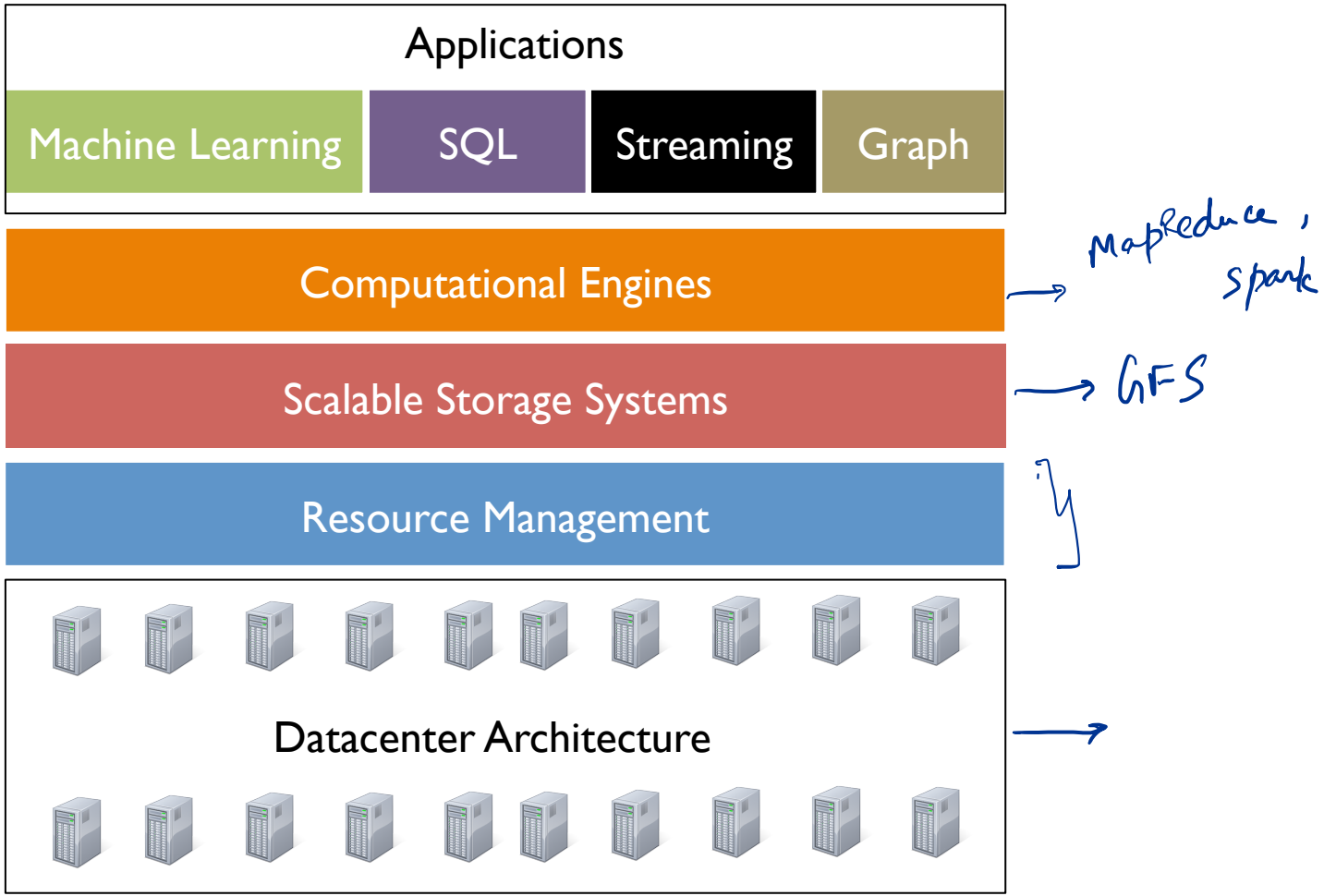
CS 744: MESOS

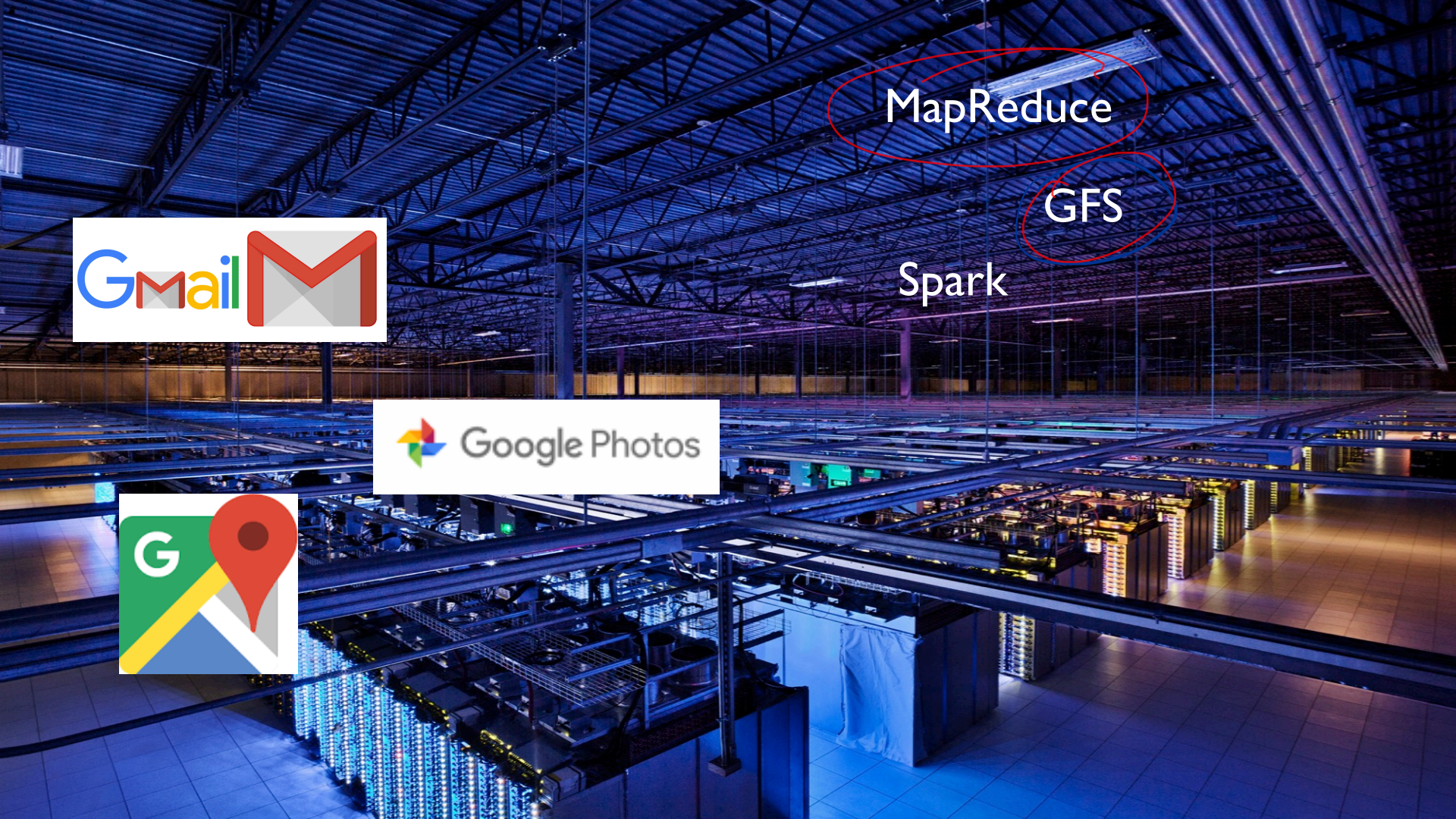
Shivaram Venkataraman

Fall 2019

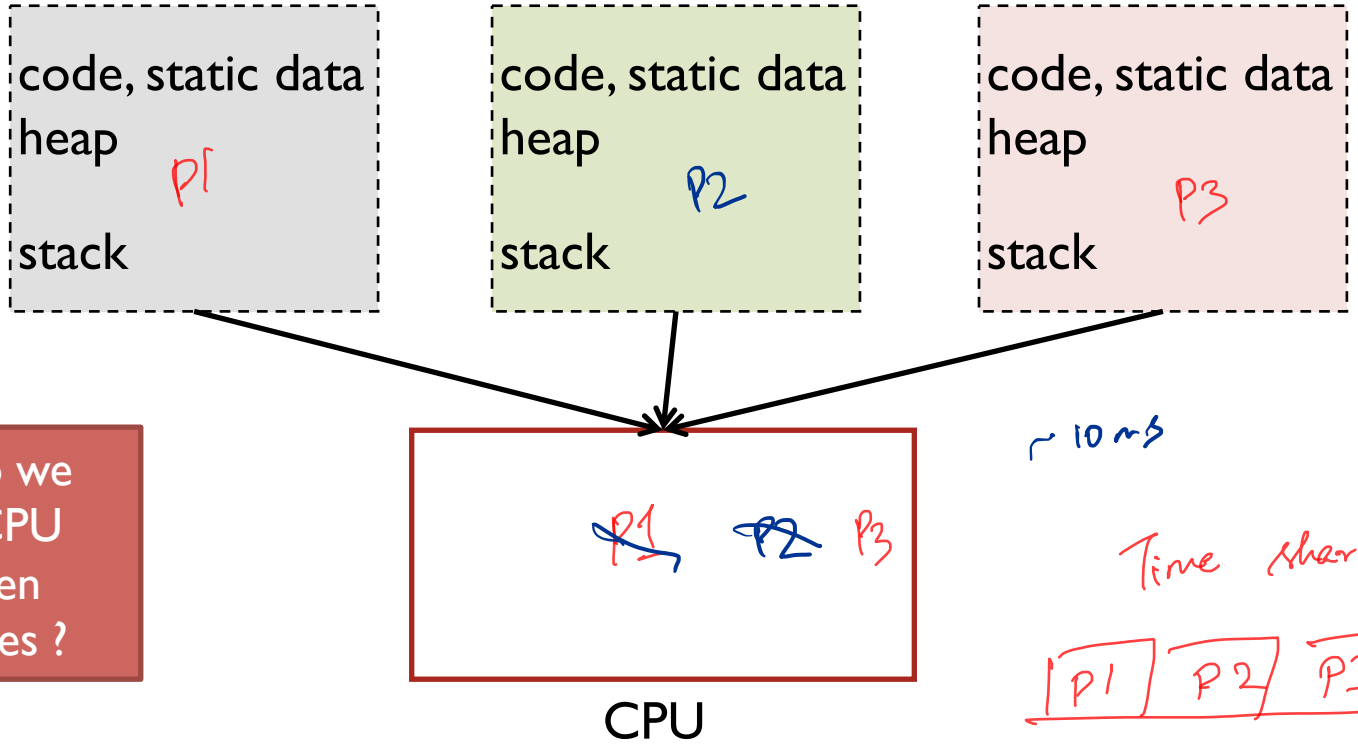
ADMINISTRIVIA

- Assignment 1: Due Tonight!
- See project list on Piazza
- Assignment 2, Project groups


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture
MapReduce, Spark
GFS


MapReduce
GFS
Spark
Gmail
Google Photos

BACKGROUND: OS SCHEDULING



CLUSTER SCHEDULING

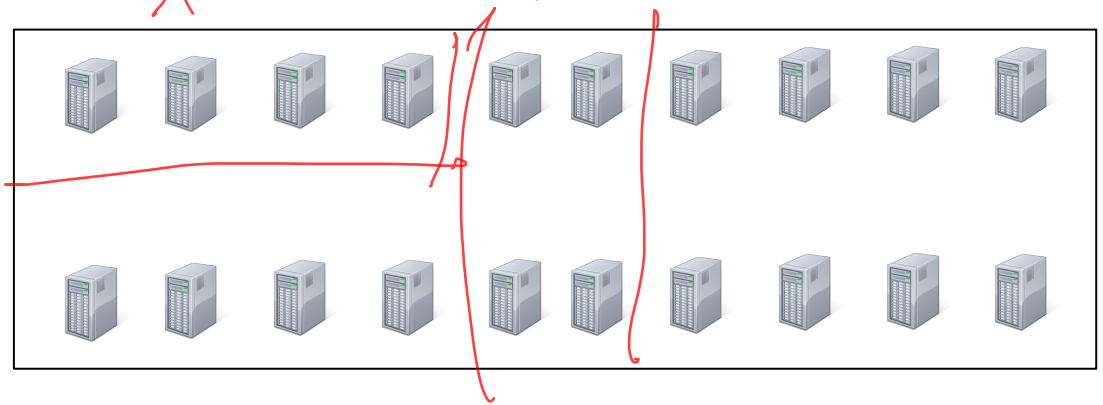
Mechanism + Policy

How What

space sharing

Resource management

Spark MapReduce



TARGET ENVIRONMENT

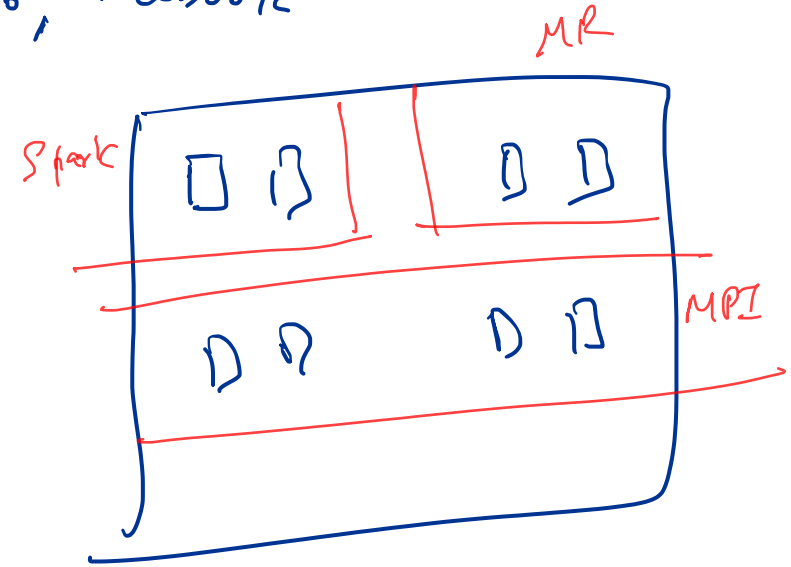
Multiple MapReduce versions → Yahoo, Facebook

↓ Generalize

Mix of frameworks: MPI, Spark, MR

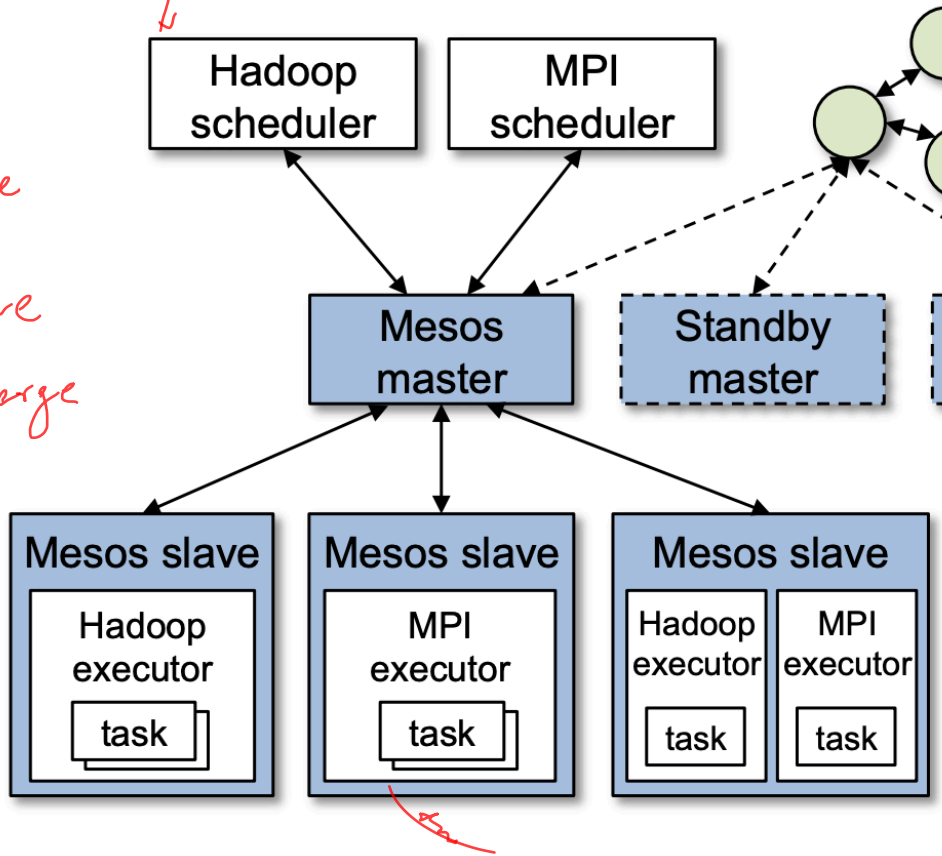
Data sharing across frameworks

Avoid per-framework clusters



1. Utilization
2. Data sharing

DESIGN



Per framework scheduler

Two-level scheduling

↳ Across frameworks

↳ Within frameworks

GFS / MapReduce

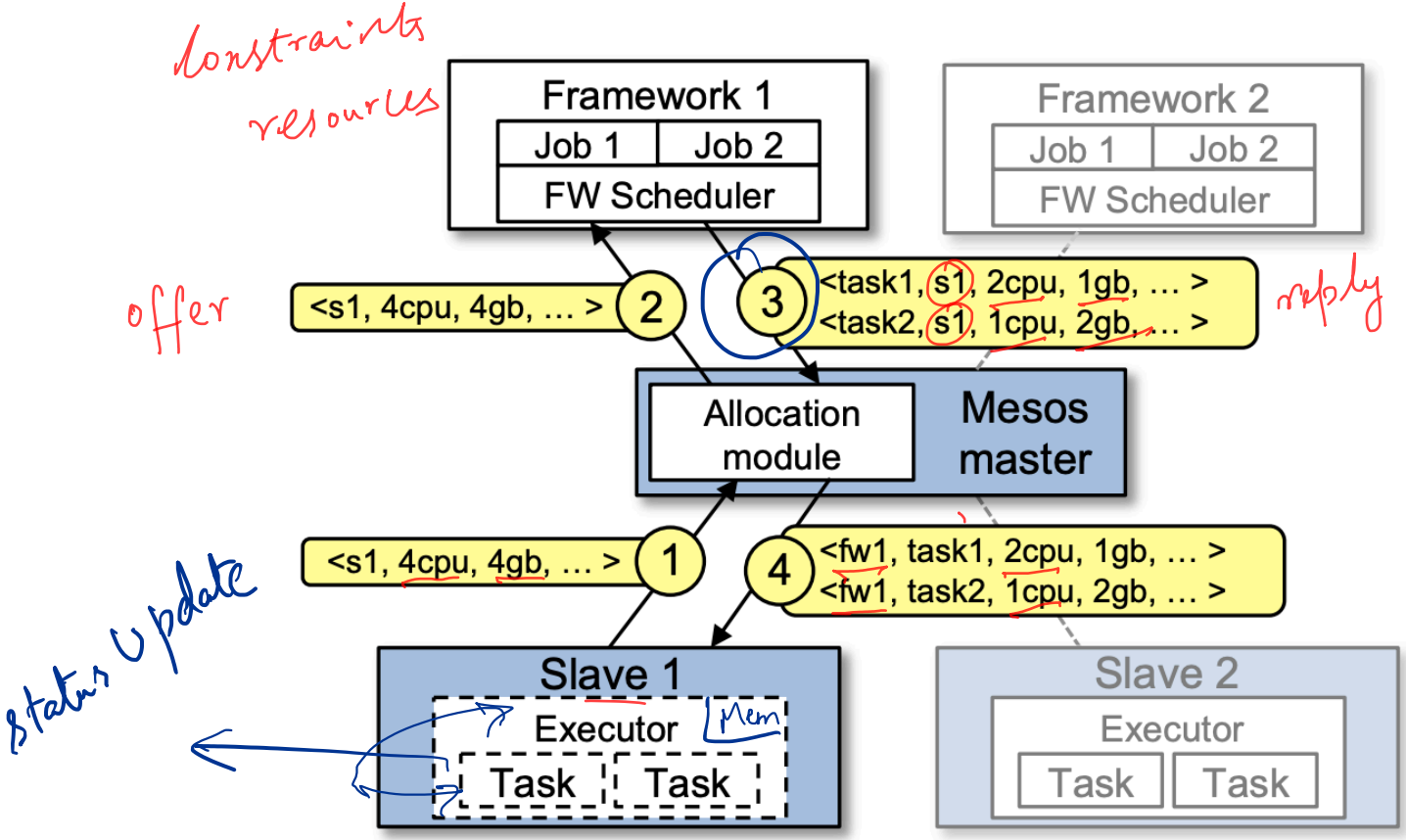
Master → slave

single large

Simplicity across frameworks

"Future proof"

RESOURCE OFFERS



Delay scheduling
→ wait for 5s

CONSTRAINTS

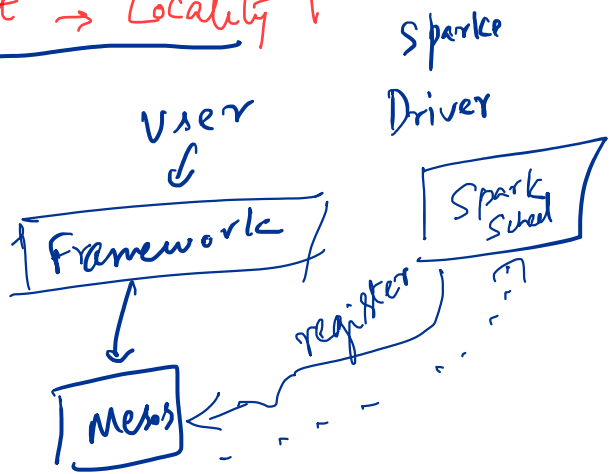
Examples of constraints

↳ specific hardware (GPUs)
↳ Large allocation
↳ Data placement → Locality

long tasks

Constraints in Mesos:

↳ Reject offers
↳ Filters : Boolean clauses



DESIGN DETAILS

Allocation:

Guaranteed allocation, revocation

Crude mechanism for time slicing ?

min limit of how many tasks

long tasks

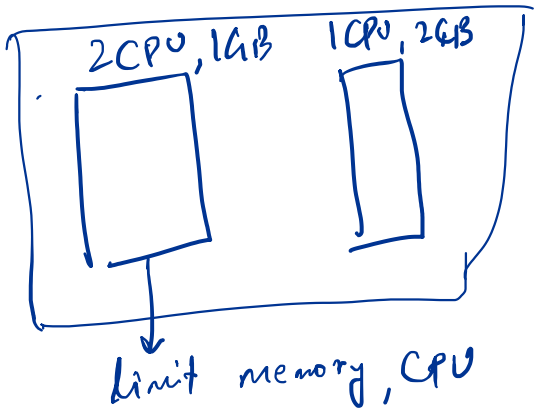
How much progress

Isolation

Containers (Docker)

lightweight → fast to start

4 CPU
4 GB



FAULT TOLERANCE

Soft state

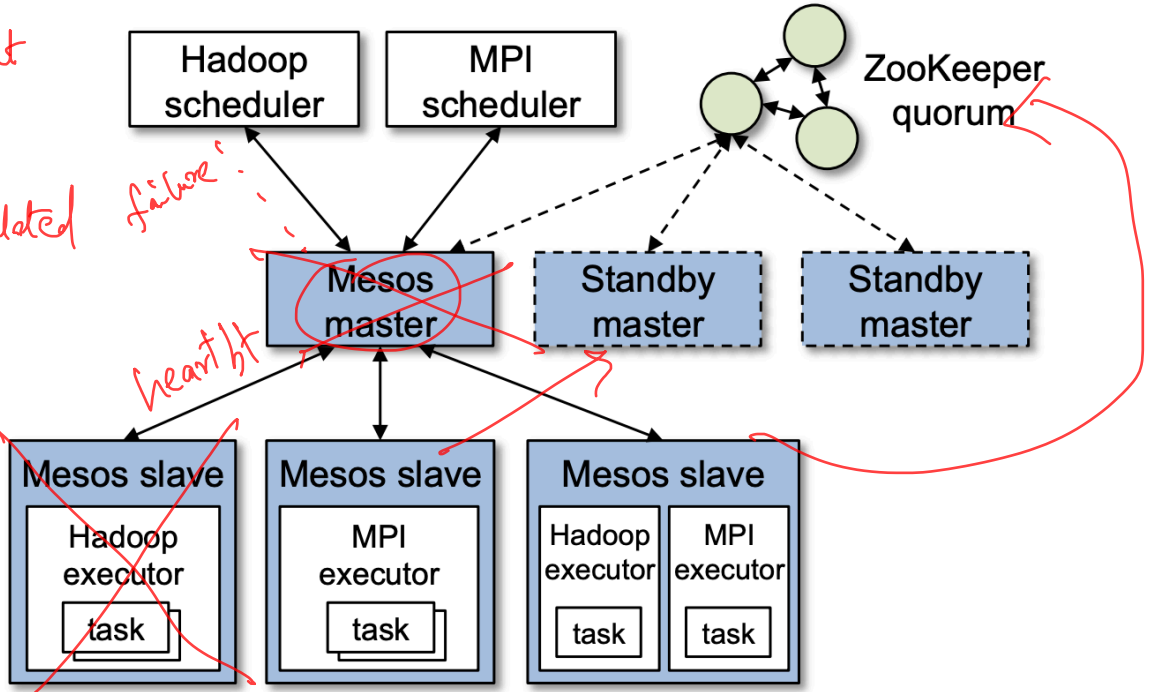
↳ No checkpoint

↳ All DS can be repopulated

↳ Slaves Mesos & Framework schedulers

GFS Master

↳ checkpoint



↳ Tasks can keep running even if master is down

PLACEMENT PREFERENCES

What is the problem?

↳ list of frameworks, who gets the offer

↳

How do we do allocations?

↳ weighted lottery

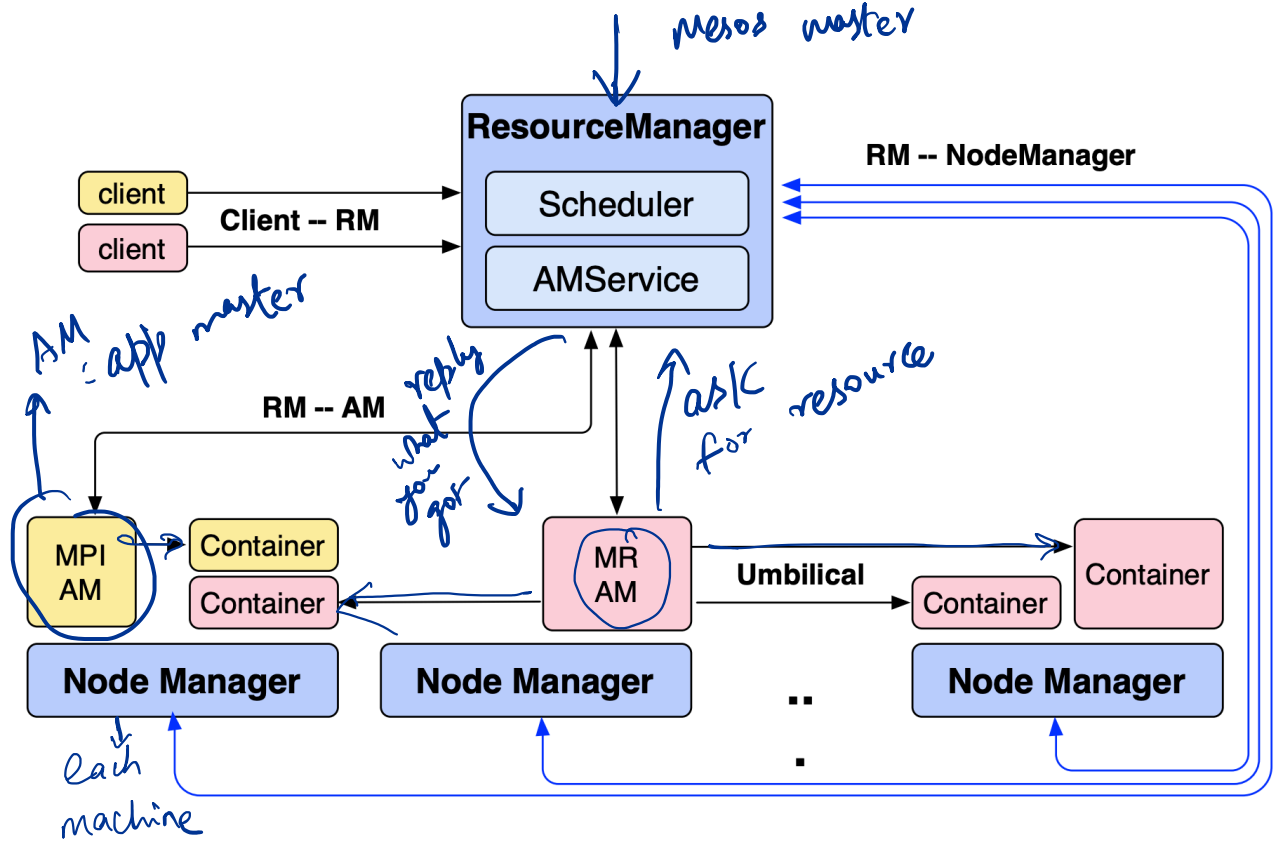
<fraction of allocation>

scheduler

weighted random number

[0, 0.5] [0.5, 0.8] [0.8, 1.0]

COMPARISON: YARN → Apache Hadoop



Per-job scheduler

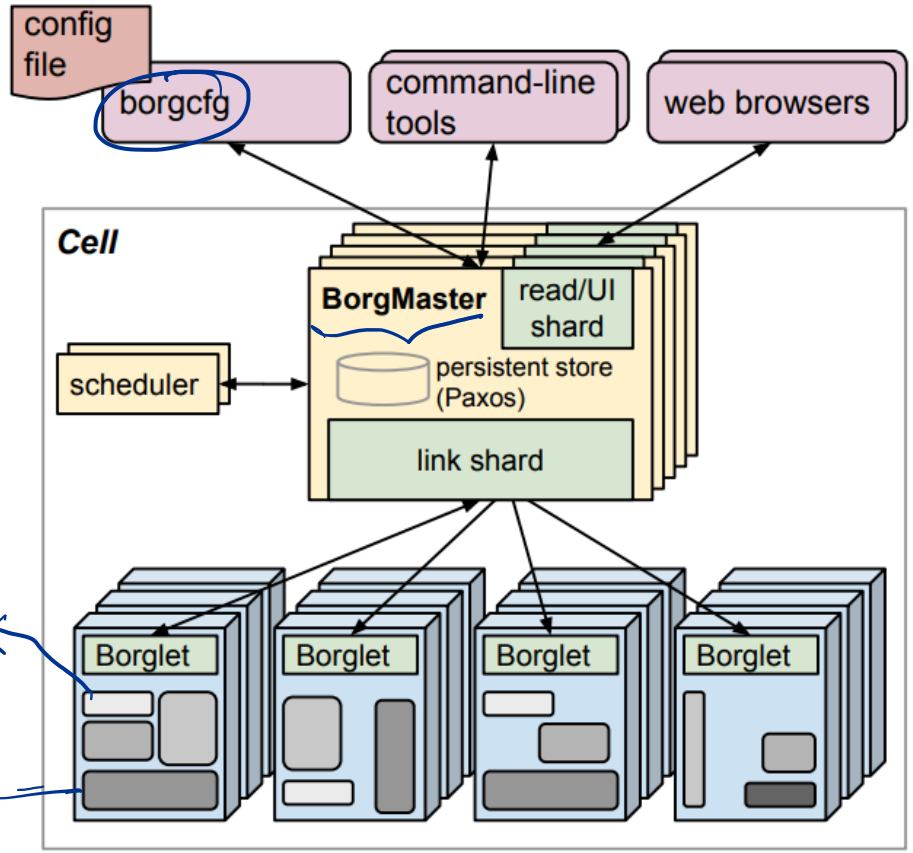
AM asks for resource

COMPARISON: BORG

Single centralized scheduler

Requests mem, cpu in cfg
Priority per user / service

Support for quotas / reservations



CENTRALIZED VS DECENTRALIZED

Decentralized

Simple Mechanism
Future frameworks

Scalability

"Short tasks"

Centralized

Global optimum

better packing
avoid fragmentation

Avoid Starvation

CENTRALIZED VS DECENTRALIZED

Framework complexity → Sparic sched
MPI sched

XML file
cpu = 1
mem = 5g

Fragmentation, Starvation → lottery sched
min offer size
8GB, 8CPU

Low-priority tasks for utilization?

DISCUSSION

https://forms.gle/oYYdvTAcczamnxvT7

What are some problems that could come up if we scale from 10 frameworks to 1000 frameworks in Mesos?

→ framework → Mesos master

→ Starvation?

→ Fault recovery → longer?

→ Allocation module slower?

→ Executor life cycle?

1 slow sched among 1000 sched

↳ "Pessimistic sched" exclusive

List any one difference between an OS scheduler and Mesos

OS

Process Context, premption

Time slicing

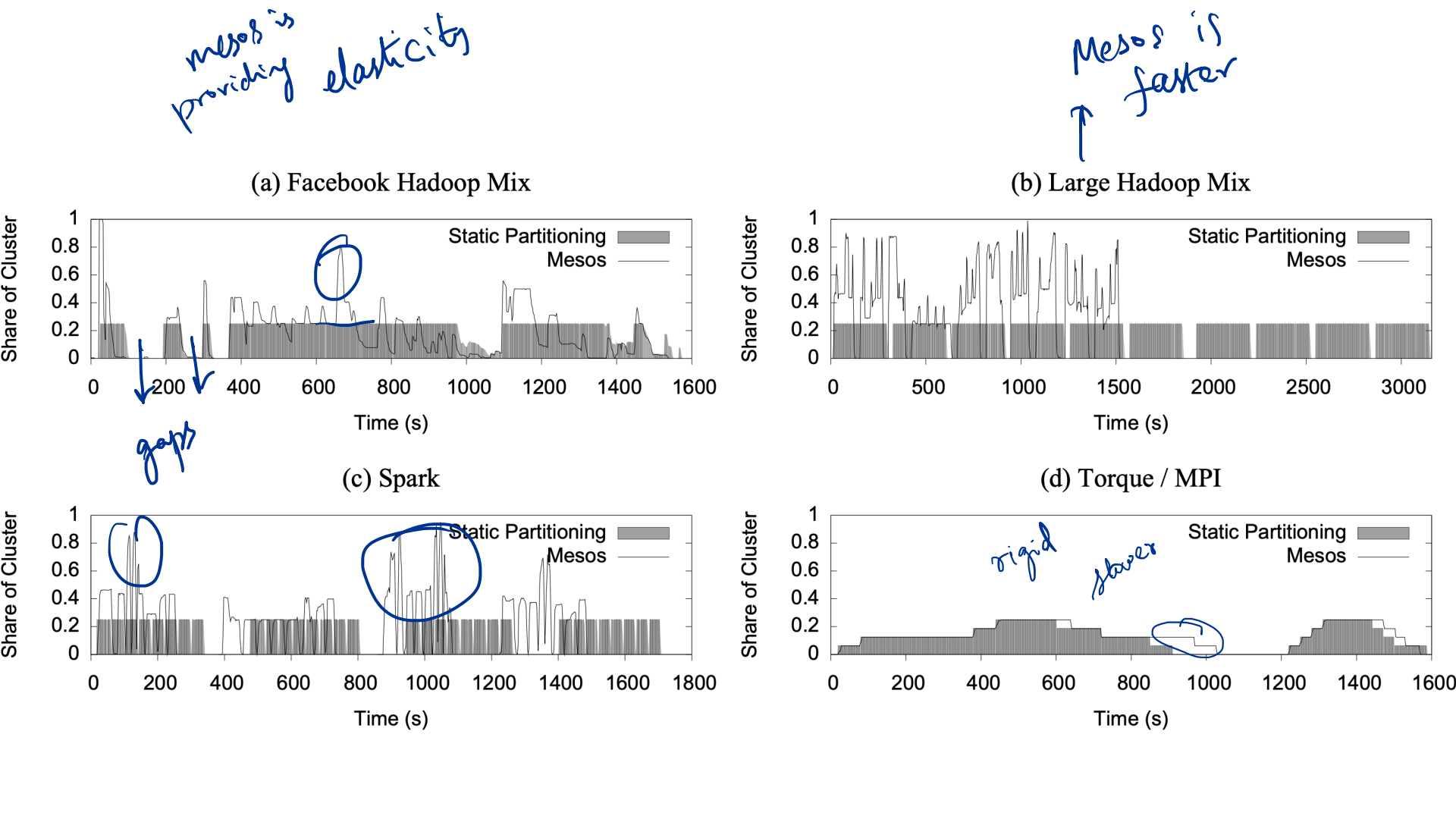
Request → new thread
→ allocate

Mesos

Dependencies across frameworks?

Space sharing +
↳ killing tasks

Offers


mesos is providing elasticity
Mesos is faster
(a) Facebook Hadoop Mix
(b) Large Hadoop Mix
(c) Spark
(d) Torque / MPI
Share of Cluster
Time (s)
Static Partitioning
Mesos
gaps
rigid
slower

NEXT STEPS

Next class: Scheduling Policy

Further reading

- https://www.umbrant.com/2015/05/27/mesos-omega-borg-a-survey/
- https://queue.acm.org/detail.cfm?id=3173558

Assignment 1 due tonight!

Assignment 2 out Thu