Good morning!

CS 744: MESOS

Shivaram Venkataraman

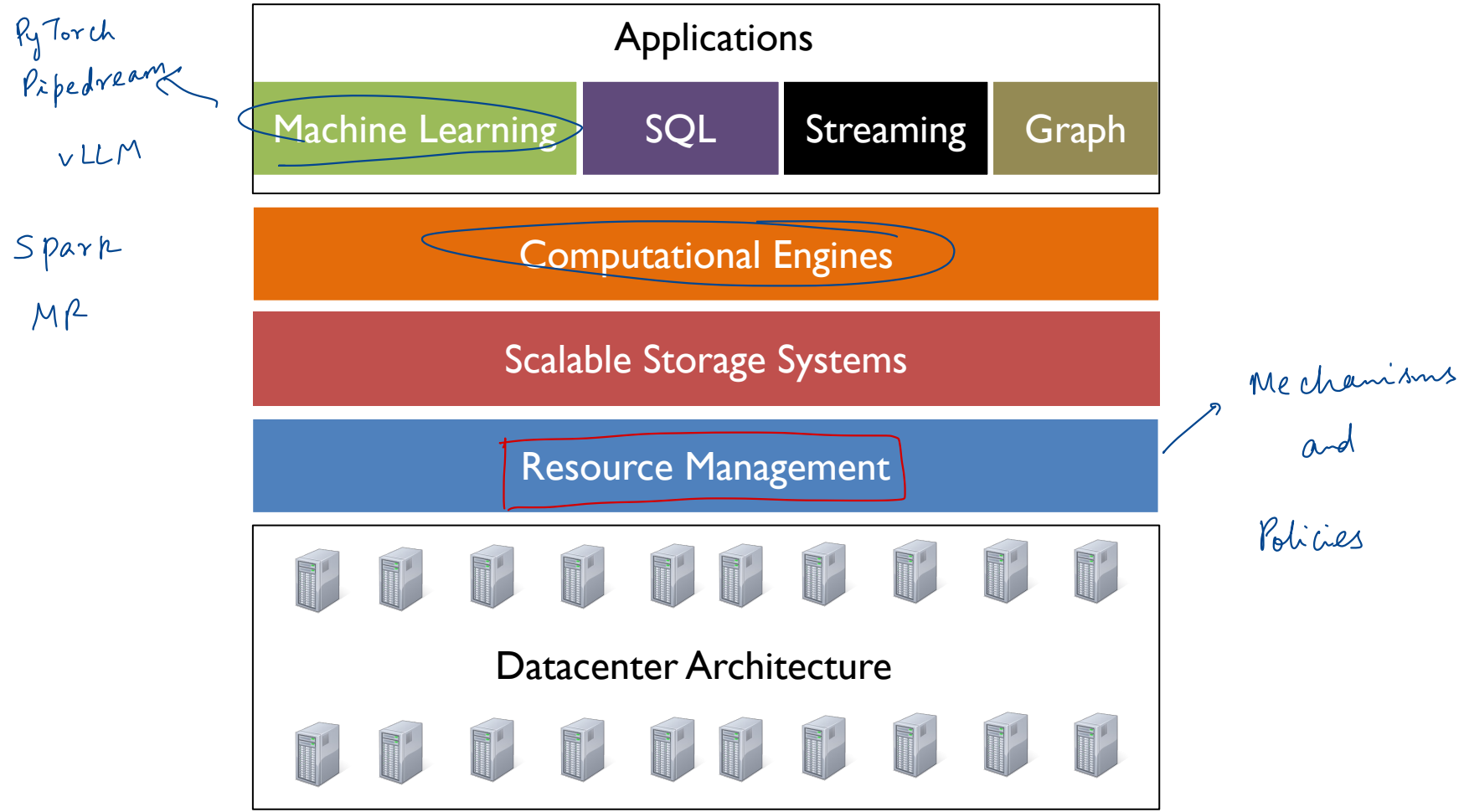
Spring 2024

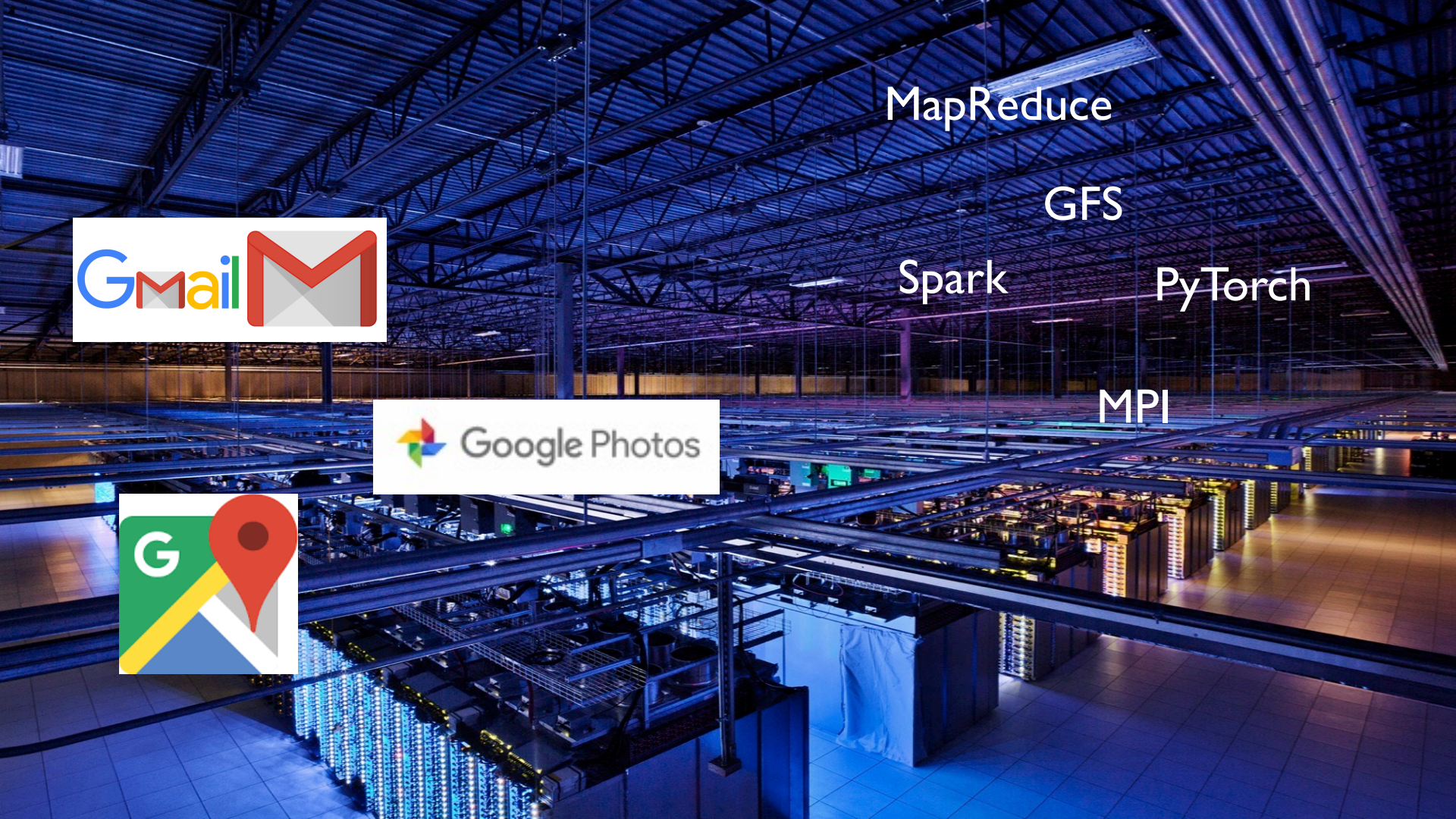
ADMINISTRIVIA

- Assignment 2: Due tomorrow!
- Project details
  - Create project groups
  - Bid for projects/Propose your own (Piazza, after class)
    - List of project ideas ~20
    - Come up with your own ideas!
    - Submit by Feb 27th Tue at 10pm

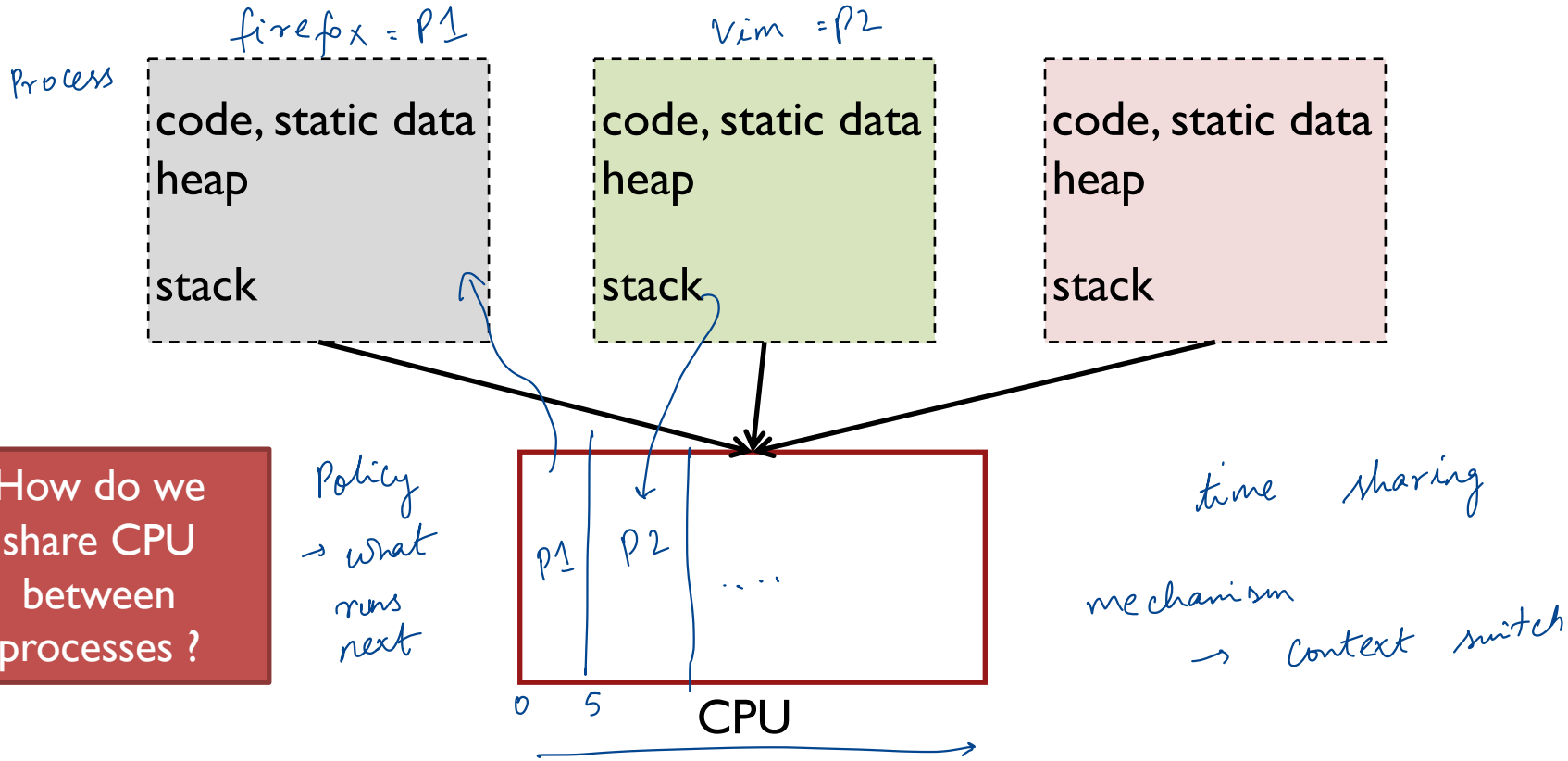
Form

→ Title


PyTorch
Pipedream
vLLM
Spark
MR
Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Mechanisms
and
Policies
Datacenter Architecture


MapReduce
GFS
Spark
PyTorch
MPI
Gmail
Google Photos

BACKGROUND: OS SCHEDULING



CLUSTER SCHEDULING

Not just time sharing

→ no job uses the entire cluster

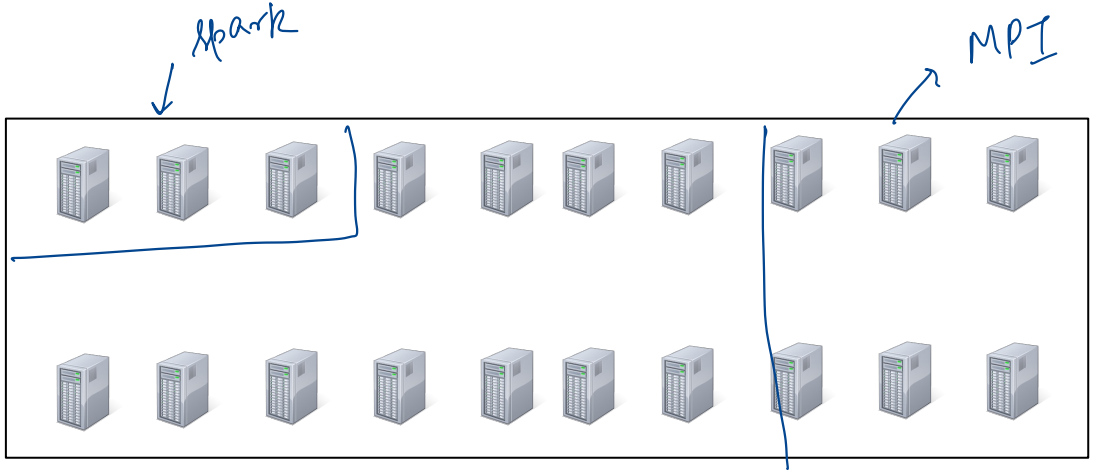
≡ Space sharing

① locality becomes important

② Hardware heterogeneity

→ GPUs

→ faster SSDs



TARGET ENVIRONMENT

Multiple MapReduce versions → 2006 ~ 2009 Hadoop releases

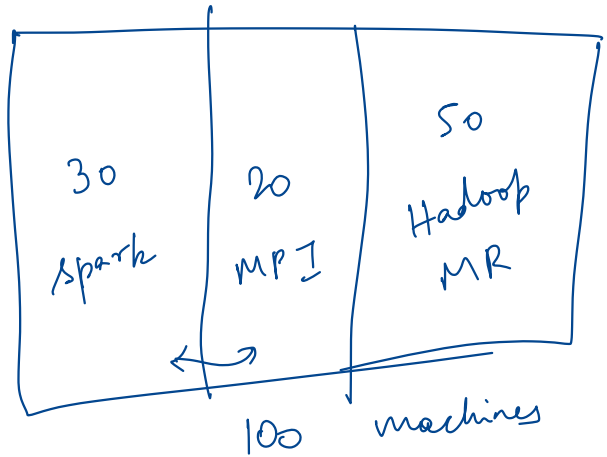
Mix of frameworks: MPI, Spark, MR → similar execution footprint as PyTorch

cluster utilization = HIGH!

Avoid per-framework clusters. Why?

→ If workload mix varies then we waste resources

→ Data sharing is non-trivial



DESIGN

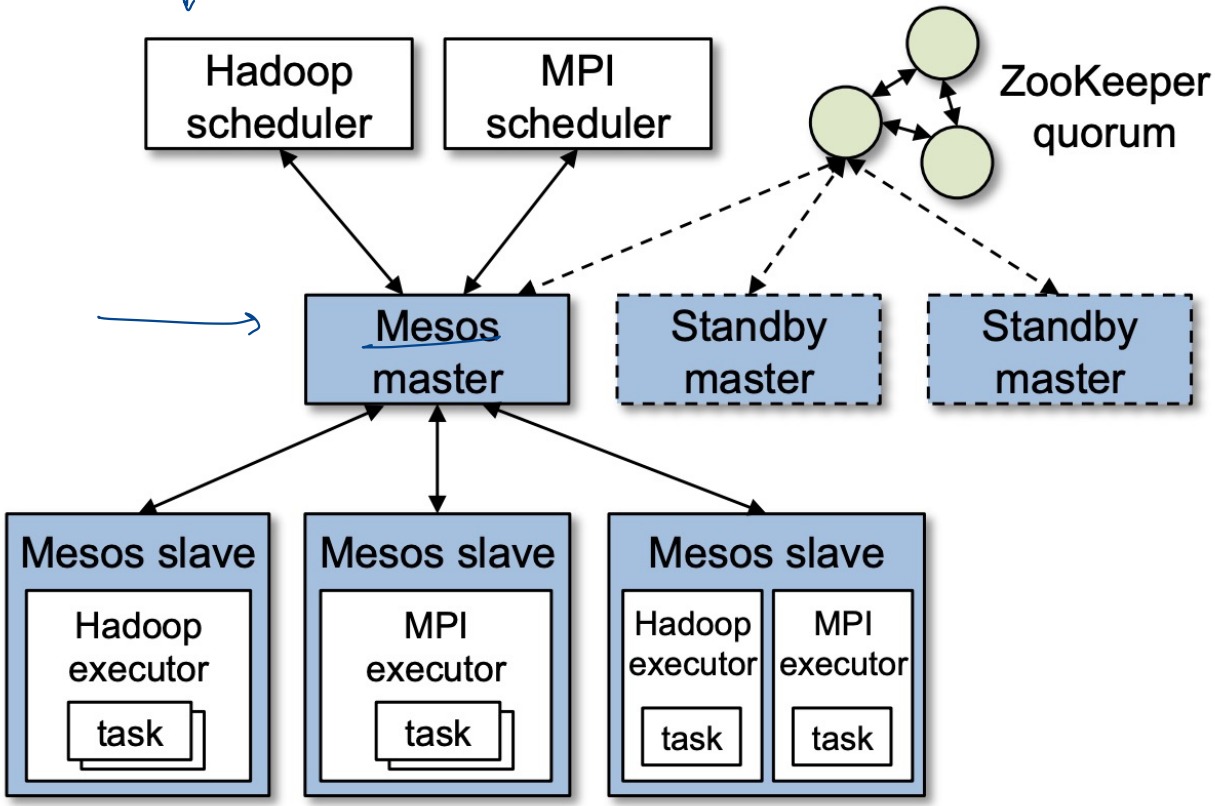
Per-Framework Scheduler

"Two-level" scheduling

framework

↳ Across frameworks

→ within

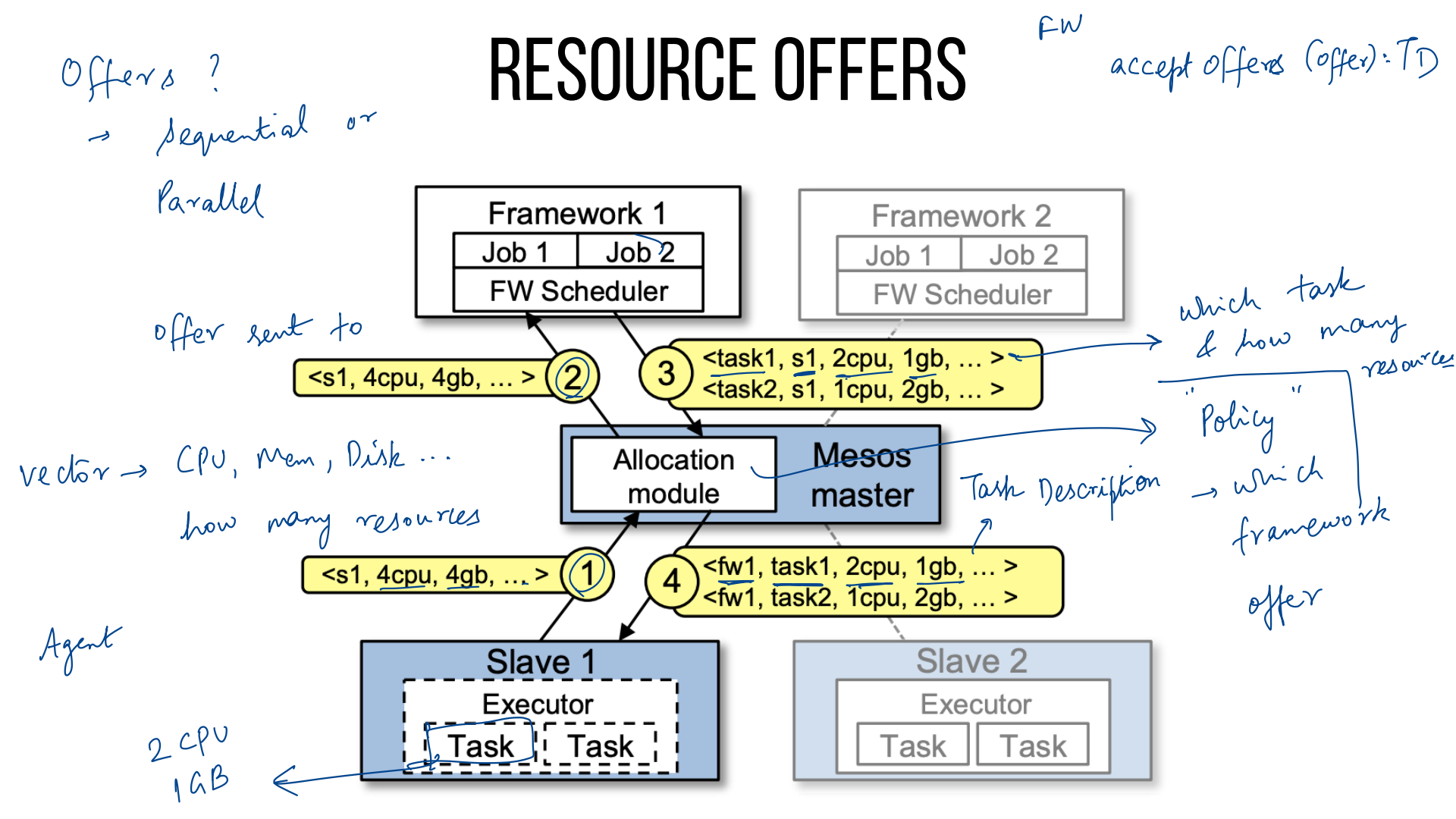


Centralized coordinator

similar to GFS design

Worker / Agent runs on every machine

RESOURCE OFFERS



Offers ?
→ Sequential or Parallel

FW
accept offers (offer): TD

offer sent to

which task & how many resources

"Policy"

vector → CPU, Mem, Disk …

how many resources

Task Description → which framework

offer

Agent

2 CPU
1 GB

CONSTRAINTS

Examples of constraints

- Data locality → soft constraint
- GPU machines → hard constraint

Constraints in Mesos:

- Applications can reject offers
- Optimization: Filters

DESIGN DETAILS

When do you do allocation?

OS scheduling → Fixed time quanta Pre-empt processes

Starvation for waiting jobs

Allocation:

Tasks are short, allocate when they finish

Long tasks? Revocation beyond guaranteed allocation

Kill tasks

Each framework

Policy is only invoked when tasks finish

Isolation

Containers (Docker)

↳ CPU, mem

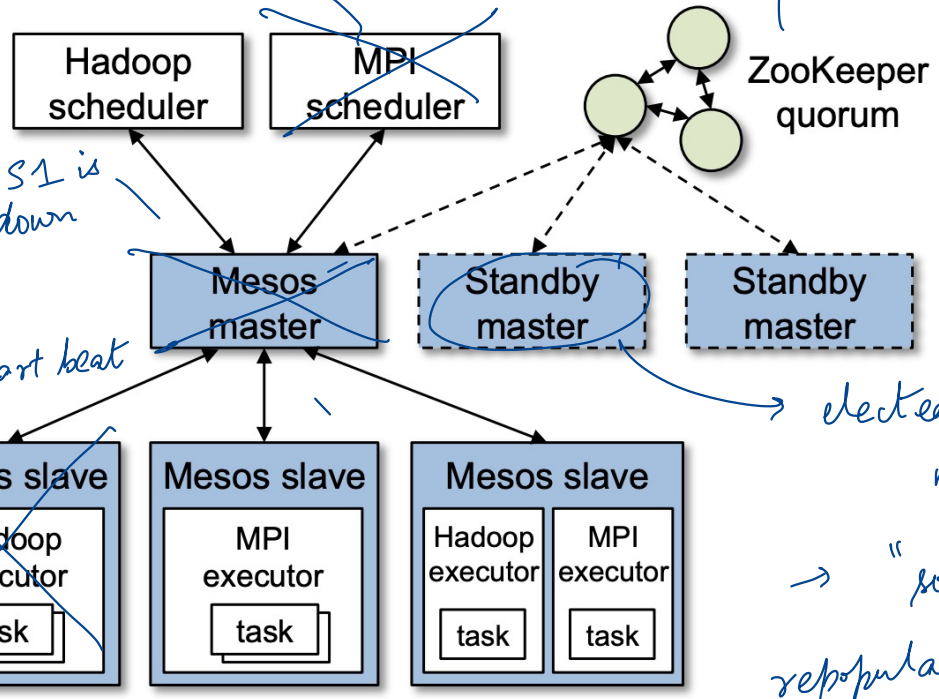
Spark / Hadoop

FAULT TOLERANCE

FW scheduler might crash

user relaunch them & register with mesos

Consistent kv store



Workers machines Crashes

↳ fwd this to per-framework Scheduler

→ "soft state" repopulated from other machines

HANDLING PLACEMENT PREFERENCES

What is the problem?

- More frameworks have preferred nodes than available
- Who gets the offers?

How do we do allocations?

- Lottery scheduling – offers weighted by num allocations

Policy

CENTRALIZED VS DISTRIBUTED

Framework complexity

↳ easier to support new frameworks as core scheduler doesn't need to change.

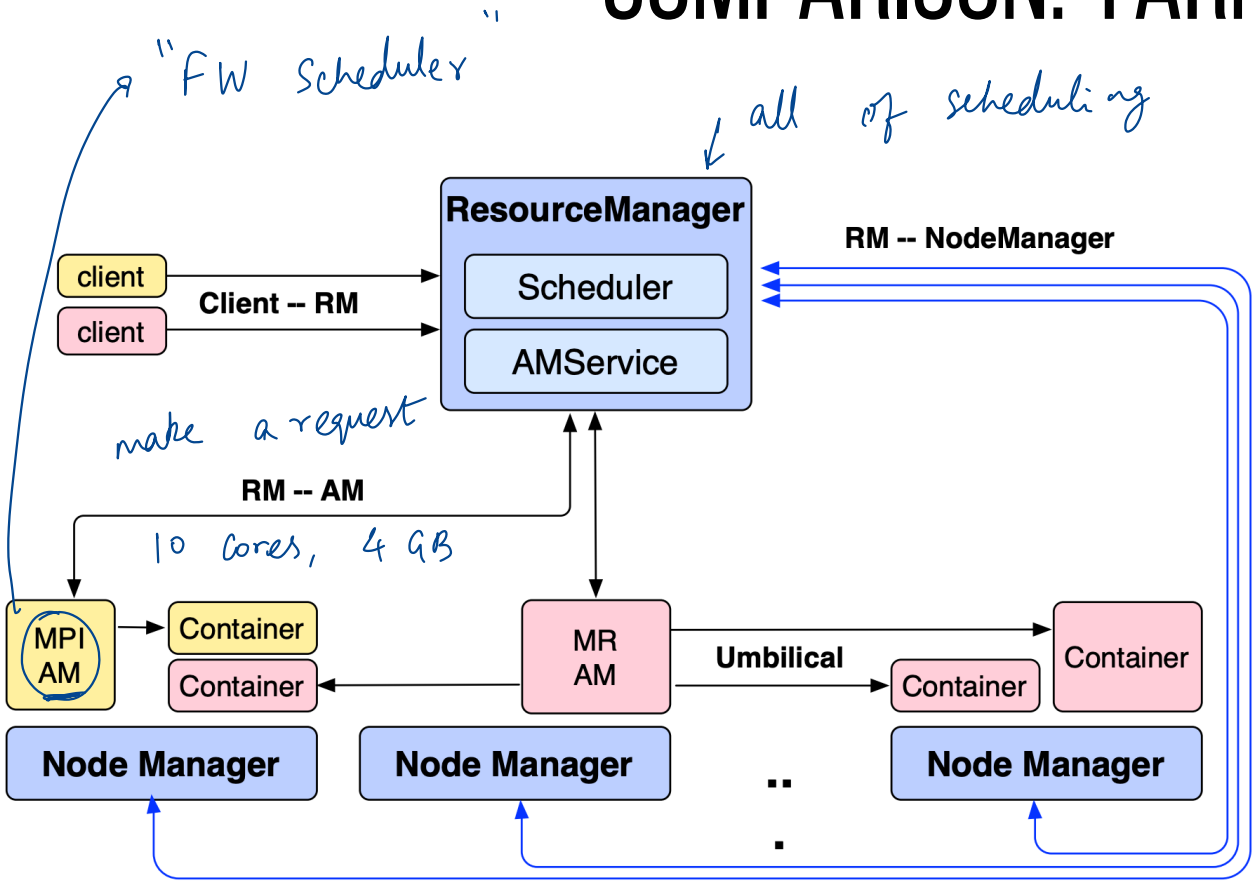
Fragmentation, Starvation

↳ mix of small and large jobs sharing the cluster

Inter-dependent framework

↳ MPI job after Spark job runs

COMPARISON: YARN → Hadoop



Per-job scheduler

AM asks for resource
RM replies

COMPARISON: BORG (KUBERNETES!?)

Single centralized scheduler

gets all requests

comes up with assignment

Requests mem, cpu in cfg

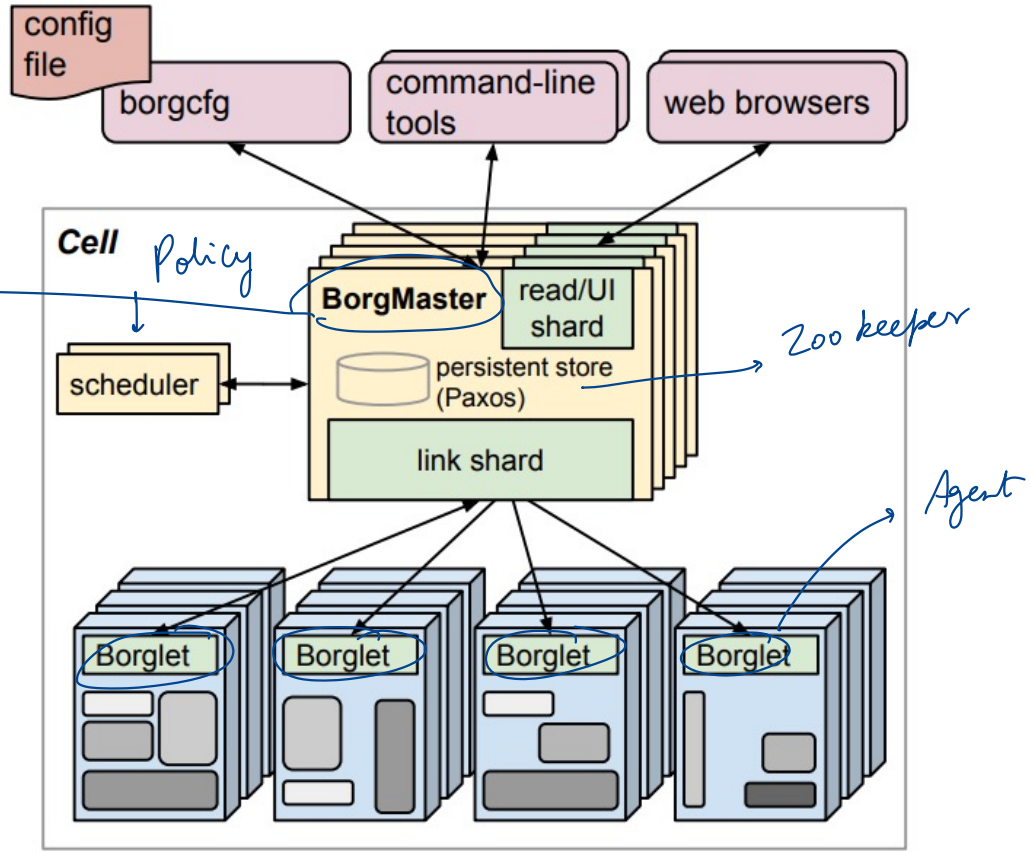
Priority per user / service

→ websearch = 100

→ indexing = 10

Support for quotas / reservations

→ Pre-emption, packing etc.



SUMMARY

- Mesos: Scheduler to share cluster between Spark, MR, etc.
- Two-level scheduling with app-specific schedulers
- Provides scalable, decentralized scheduling
- Pluggable Policy ? Next class!

DISCUSSION

https://forms.gle/hWQeyW6om3XhnqDS8

What are some problems that might arise if you wanted to use Mesos with frameworks that had very low latency tasks (e.g., for interactive analytics)

→ Overhead of offers

→ Utilization high

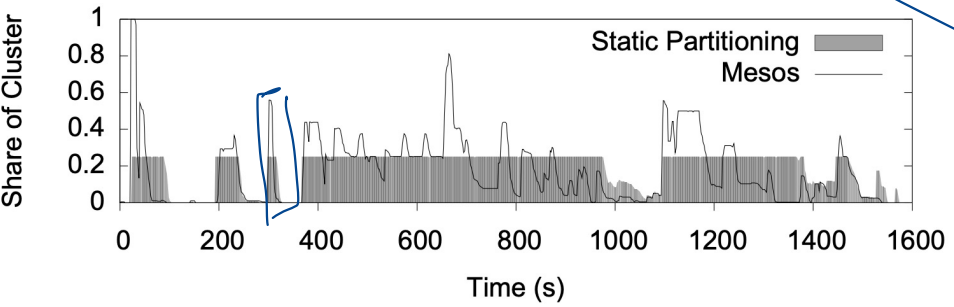
vs.

Responsiveness

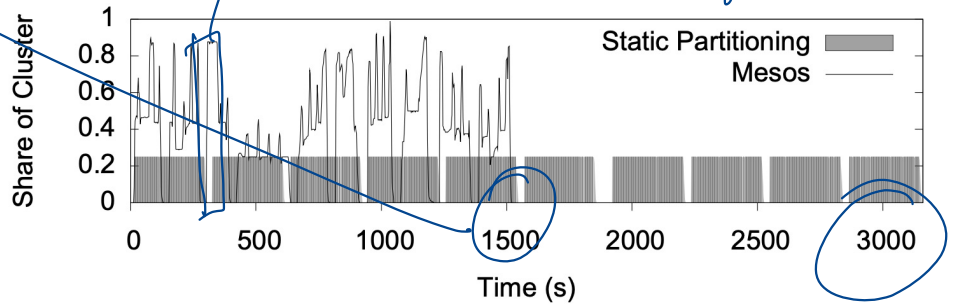
Mesos is almost ~2× faster → elastic workloads good

Mesos is high utilization: adjusting burst of jobs arriving

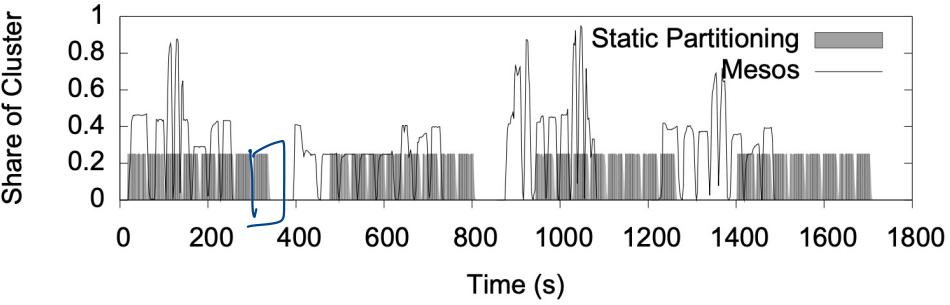
(a) Facebook Hadoop Mix



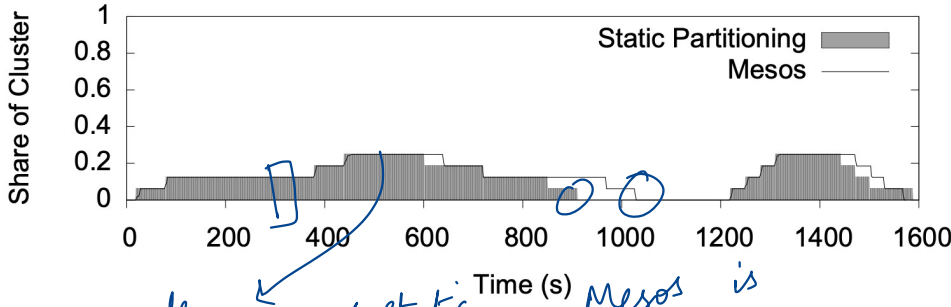
(b) Large Hadoop Mix



(c) Spark



(d) Torque / MPI



made more progress static Partitioning Mesos is 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