

Hello!

CS 744: TWINE

Shivaram Venkataraman

Spring 2025

ADMINISTRIVIA

- Assignment grading → bottleneck Shivaram

- Project groups → Email from Tareq

- Next week plan

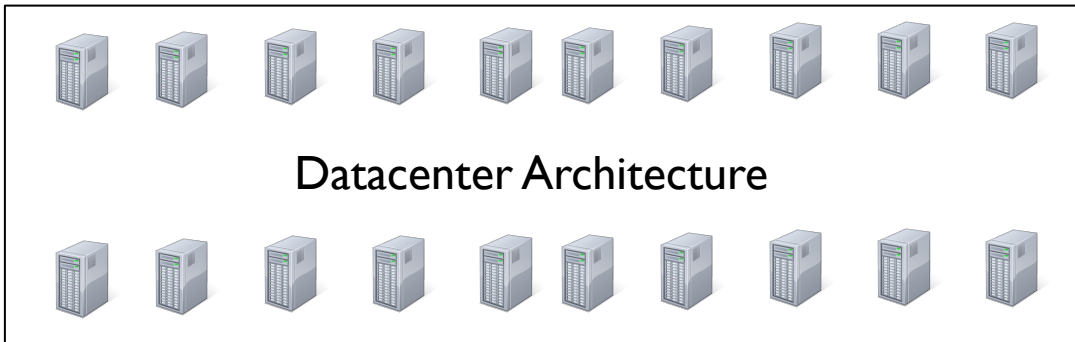
↳ Tue lecture : ML scheduler
Zoom (record it) 1pm: 2:15pm

Thu no lecture

Training

FT

Inference



Spark / MR

/

→ very large scale

→ ML Training jobs



MapReduce

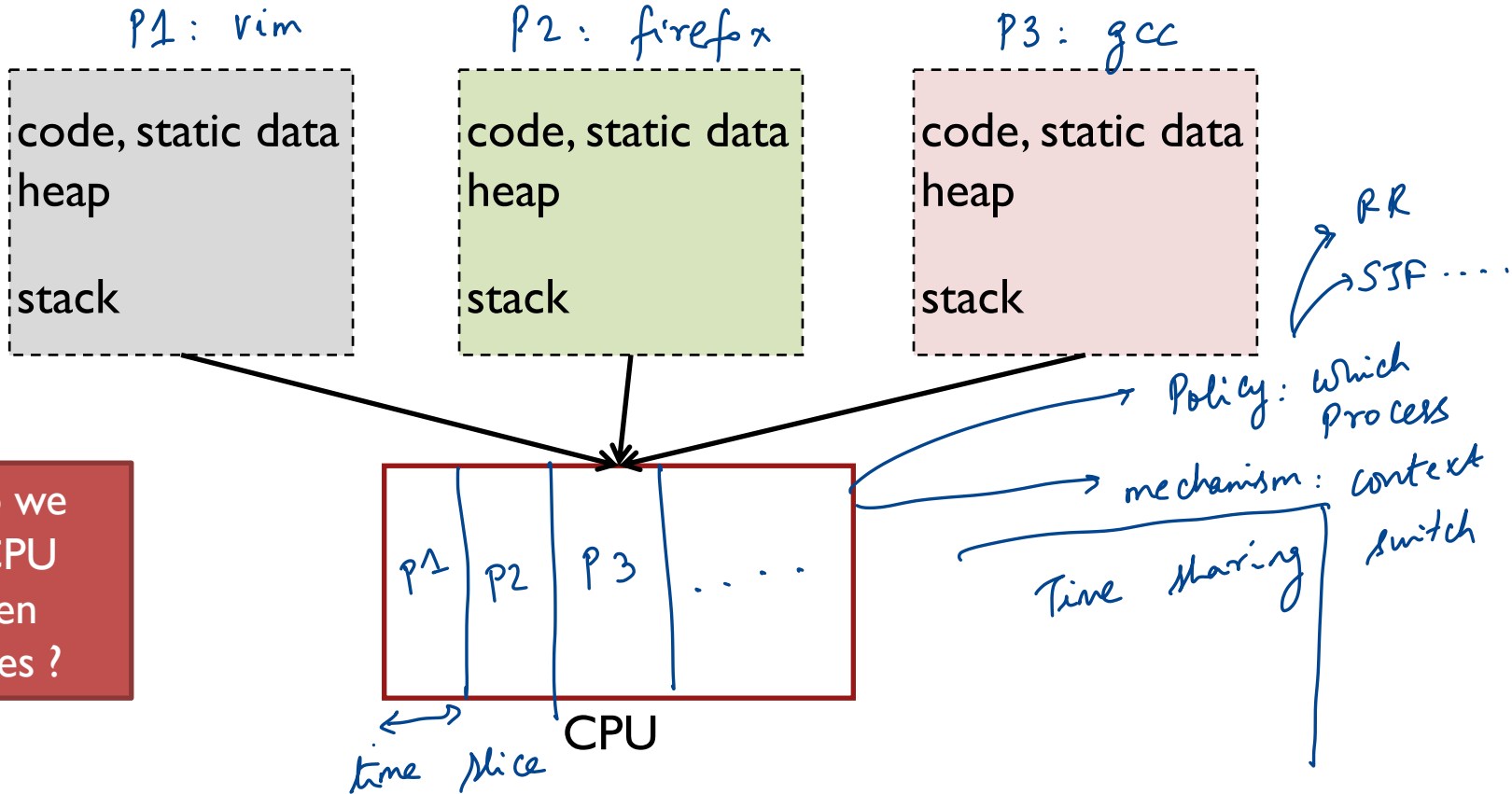
GFS

Spark

PyTorch

MPI

BACKGROUND: OS SCHEDULING



CLUSTER SCHEDULING

Processes

v.s.

Task



multiple task

PyTorch job → one of DDP workers is a task

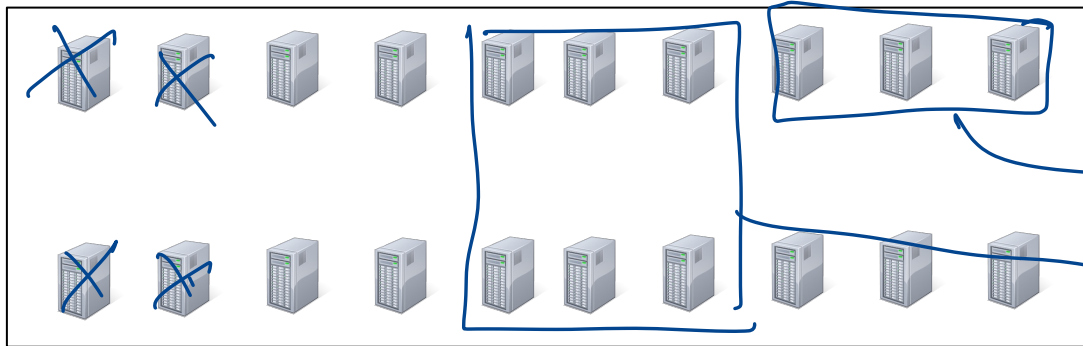
granularity

Where is scheduler

interface / bottlenecks

Failures / Auto scaling

Space sharing along with time sharing



Multiple machines!

Spark

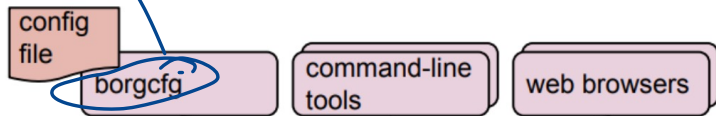
PyTorch

BORG, MESOS ETC

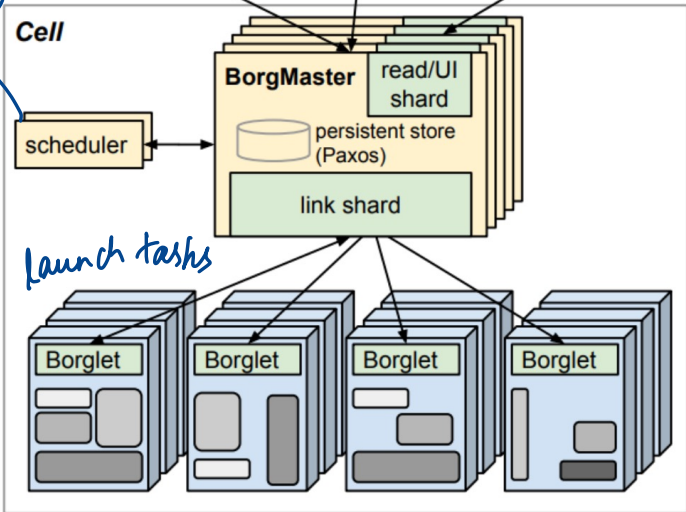
Google →

Apache Mesos
hints or pref / constraints

users submit jobs

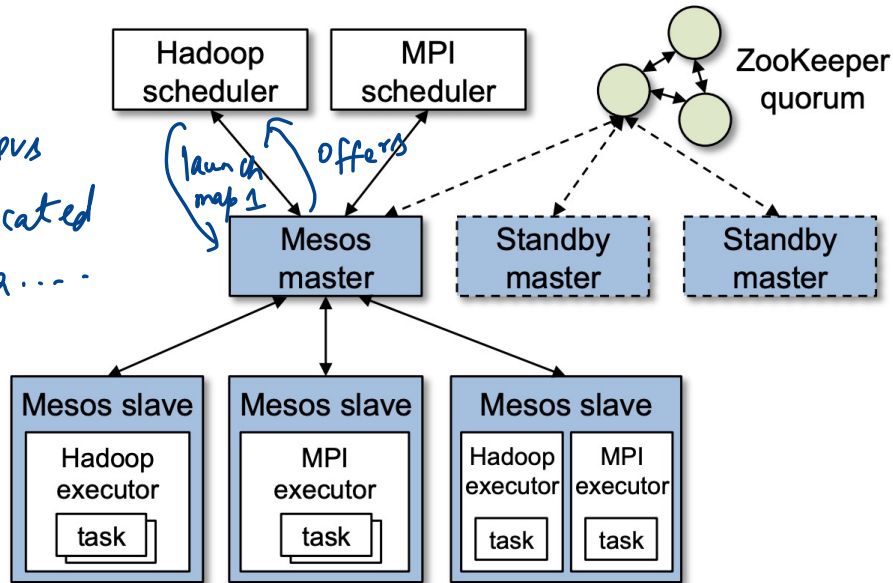


selecting MCS to jobs



cell ~10,000 machines

4 CPUs
co-located data...

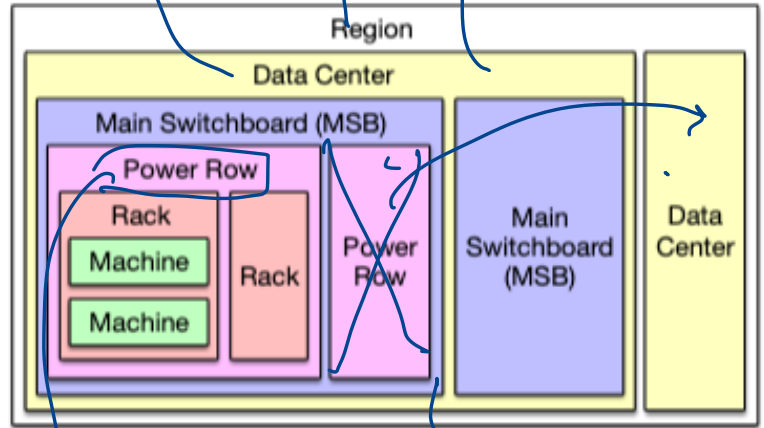


TWINE

~1M machines in the region

sell

~10,000 - 100,000 machines



10s of racks

power failure

Goals:

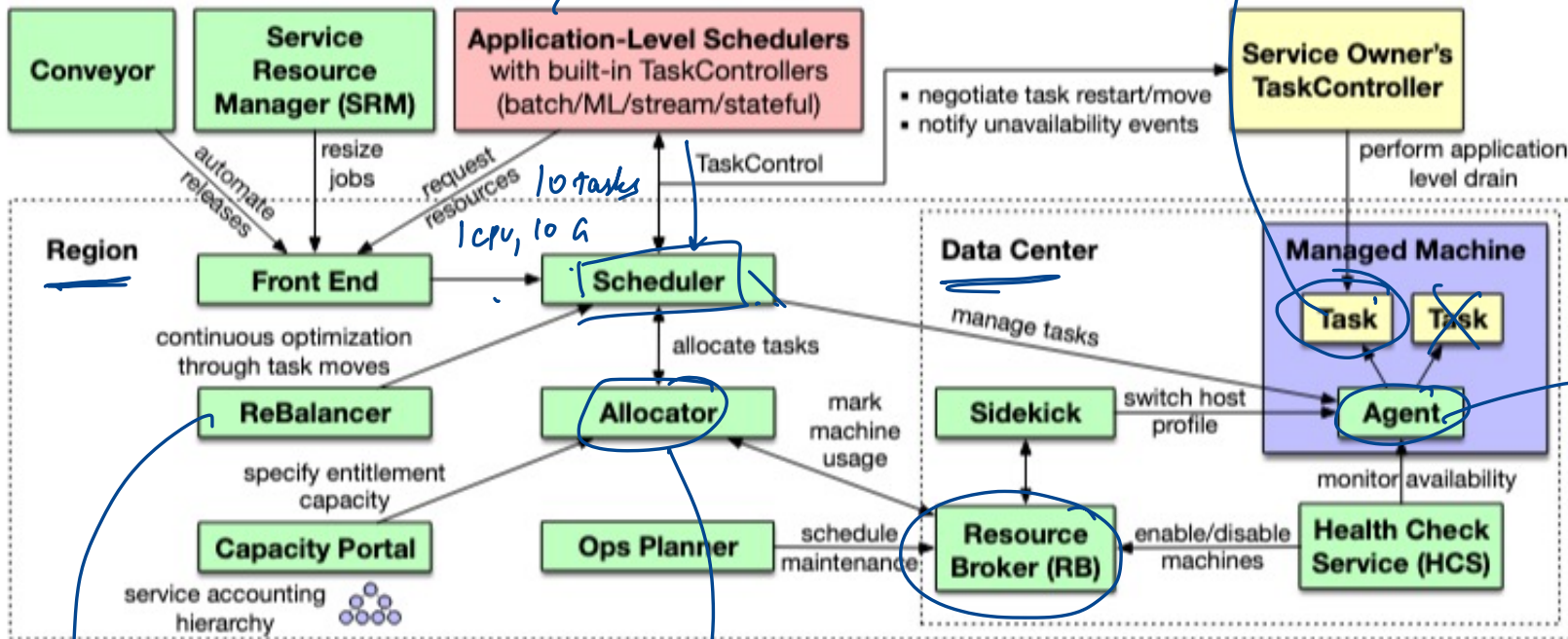
fragmentation / stranded resources

- Share resources across clusters
- Improve scheduler → application lifecycle
- Allow hardware customization

TWINE: ARCHITECTURE

map / PyTorch etc.

Spark Scheduler



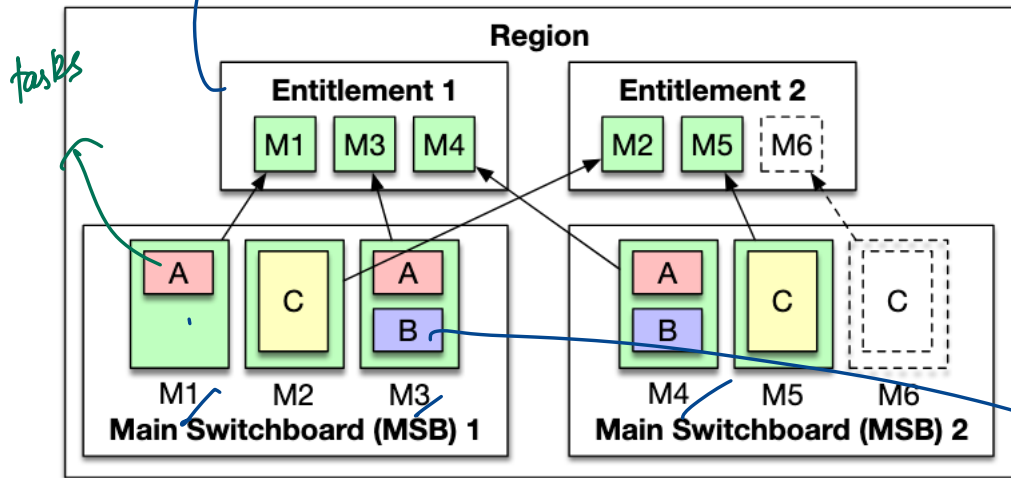
Twine Components

which machines to use?

launch task

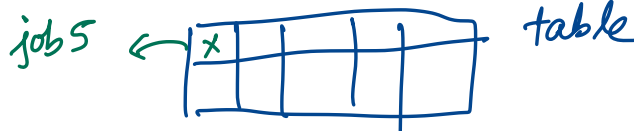
ALLOCATOR

entitlements
↳ quota or capacity for a user / team



fast filtering
↳ only machines GPU

In-memory index of all machines



Optimistic concurrency control to resolve conflicts [Omega]

→ Job inside entitlement spread across MSB / DCs

↳ shared between jobs that belong to same entitlement

TASK CONTROL, APP LEVEL SCHEDULERS

Application knows how to handle lifecycle (Mesos)

Containers managed by
application-level schedulers

Twine
between scheduler

*f App-level
scheduler*



```
service TaskController {
    TaskControlResponse process(TaskControlRequest request);
}
struct TaskControlRequest {
    string jobHandle;
    list<> request; // Pending task operations to be approved.
    list<> completed; // Completed task operations.
    list<> advanceNotices; // Upcoming planned maintenance.
    list<> allUnhealthyTasks; // Tasks unhealthy due to any reason.
    int sequenceNumber; // Increase after each call.
}
struct TaskControlResponse {
    list<> ack; // Approved task operations.
}
```

SCALING TO 1M MACHINES

Sharding vs. federation

Shard frontend, scheduler,
allocator, resource broker...

metadata

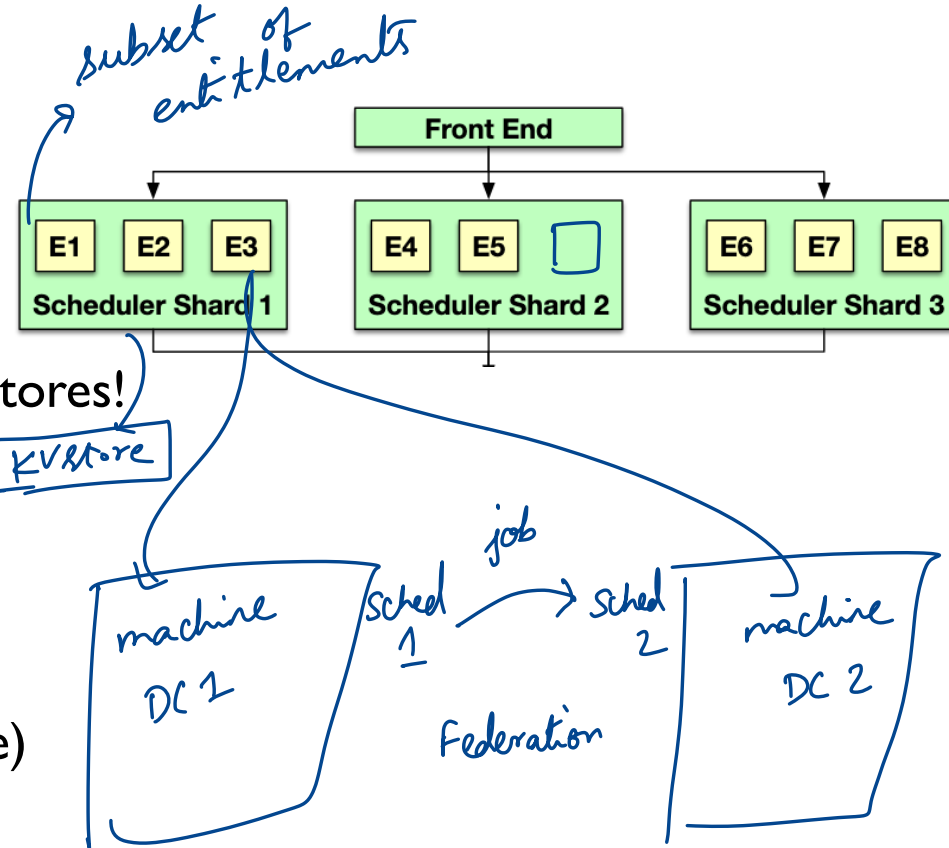
← Use separate persistent external stores!

KV store

Move machines vs. move jobs

Separation of concerns

App level scheduler (outside Twine)



FAULT TOLERANCE

Risks? Failures, network partitions

Shard, replicate components

Scheduler failure doesn't affect running tasks

Network redundancy

Twine manages itself?



*bootstrap the system / reuse logic
for Twine!*

SUMMARY

- Twine: Scheduler across a region (multiple datacenters)
- Shard all components to ensure scalability
- App-level schedulers, rebalancer for expensive decisions

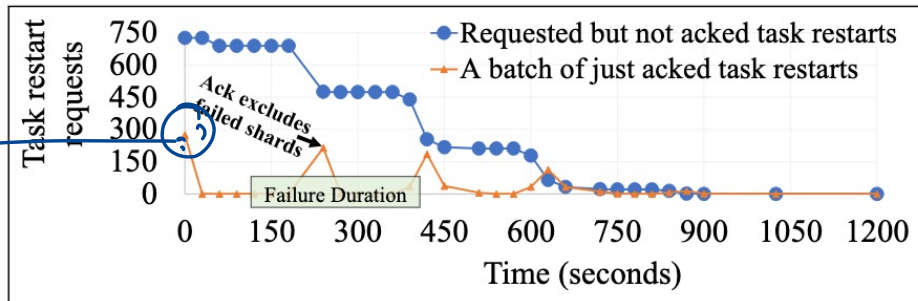


DISCUSSION

<https://forms.gle/TzaxuHu3kvVM673M9>

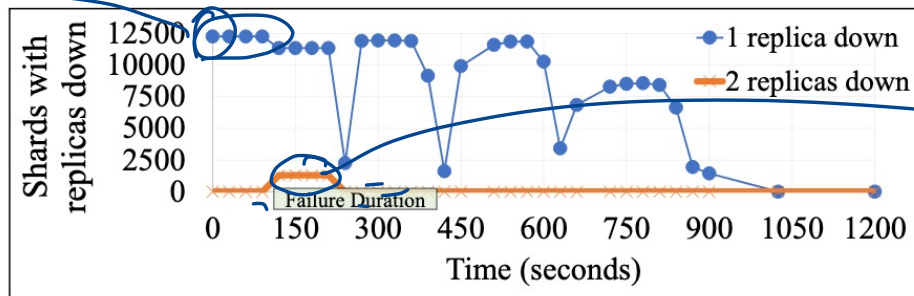
Update their
binary version

ack for
275 tasks



(a) Acknowledged and pending task-restart requests.

lots of
shards with
1 replica
down?



50 tasks

failures

(b) Shards with some replicas down.

What are some features you might implement if you were writing an application level scheduler for ML training jobs?

Hardware Customization GPU

→ Data used for training → locality

→ 3D parallelism → placement

TP → same machine

NEXT STEPS

Next class: ML Scheduling with Blox