CS 744: MESOS

Shivaram Venkataraman
Fall 2022
ADMINISTRIVIA

- Assignment 1: Due Sep 28\textsuperscript{th} at 11am!
- Assignment 2 out soon!
- Project details
  - Create project groups
  - Bid for projects/Propose your own
  - Work on Introduction
  - Final report / poster presentation
Scalable Storage Systems

Datacenter Architecture

Resource Management

Computational Engines

Machine Learning SQL Streaming Graph

Design

Mechanism

Policy

Spark, MapReduce
MapReduce

GFS

Spark
How do we share CPU between processes?

Time sharing

Context switching

Pre-empt switch to $p_2$
CLUSTER SCHEDULING

Space sharing
- Partition resources
  Many applications run at the same time

① Locality of frameworks to resources
② Minimize wasted work, but ensure frameworks get sufficient resources
③ Applications are complex and diverse
TARGET ENVIRONMENT

Multiple MapReduce versions

Mix of frameworks: MPI, Spark, MR

Avoid per-framework clusters. Why?

- Under utilization
- Workload time varying
- Data sharing, composing frameworks

~2006 - 2009 Hadoop releases

Share the clusters

<table>
<thead>
<tr>
<th>Spark</th>
<th>MPI</th>
<th>MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>40</td>
<td>35</td>
</tr>
</tbody>
</table>

100 machines
Two-level design

Framework-level schedulers

Centralized Mesos Master

Agent on every machine

Why do this?

Extensibility to handle diverse frameworks

Mesos Master

Why do this?

ZooKeeper quorum

Hadoop scheduler

MPI scheduler

Standby master

Standby master

Standby master

Hadoop executor

MPI executor

task

task

task

task

task

process that can spawn tasks

containers

map/reduce tasks
RESOURCE OFFERS

MPI resource → Are process start some offers - time r%%eriM ¥ +11.

Dual allocation?

MPI → All process start same time

resource offers

heart beats

free resources

Spark Driver

resource limits

Task Task

Executor

Framework 1
Job 1 | Job 2
FW Scheduler

2

Allocation module

3

Mesos master

Framework 2
Job 1 | Job 2
FW Scheduler

1

 Slave 1

Executor

Task Task

Slave 2

Executor

Task Task

<task1, s1, 2cpu, 1gb, ... >
<task2, s1, 1cpu, 2gb, ... >

<fw1, task1, 2cpu, 1gb, ... >
<fw1, task2, 1cpu, 2gb, ... >

<s1, 4cpu, 4gb, ... >
<s1, 4cpu, 4gb, ... >
CONSTRAINTS

Examples of constraints
- Data locality → soft constraint
- GPU machines → hard constraint

Constraints in Mesos:
- Applications can reject offers
- Optimization: Filters
  → only resources that pass this filter are offered.
DESIGN DETAILS

Allocation:
- Tasks are short, allocate when they finish
- Long tasks? Revocation beyond guaranteed allocation

Isolation
- Containers (Docker)

Example: If you allocate 1 CPU, 1GB to Spark to run a task, the framework can say that it's exceeding its guaranteed allocation.
Fault Tolerance can be "soft" by talking to framework schedulers and Mesos agents on machines.

Tasks running fail to launch it?
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
weighted offers based on job size → larger jobs get a fraction

→ Two-level sched. vs centralized sched.
CENTRALIZED VS DISTRIBUTED

Framework complexity → Means every framework needs to write their scheduler

Fragmentation, Starvation

Inter-dependent framework

→ if you a mix of small tasks and very large tasks

→ cannot enforce such constraints
COMPARISON: YARN

Per-job scheduler

AM asks for resource
RM replies
COMPARISON: BORG

Single centralized scheduler

Requests mem, cpu in cfg
Priority per user / service

Support for quotas / reservations

→ Kubernetes

Diagram showing the components of Borg and Kubernetes.
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/D1sqfzD3GqxQC4Y97
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)?

- Mesos master needs to make offers frequently
- Could lead to fragmentation/starvation
- Tasks might get slowed down waiting for offers
Cluster utilization is higher.

If framework is elastic get benefits.

(c) Spark

(d) Torque / MPI

Mesos is worse.
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