CS 744: MESOS

Shivaram Venkataraman
Fall 2021
- Assignment 1: Due tonight!
- Assignment 2 out soon
- Project details
  - Create project groups
  - Bid for projects/Propose your own
  - Work on Introduction 2 page
- Final report / poster presentation
Scalable Storage Systems

Datacenter Architecture

Resource Management

Computational Engines

Applications

Machine Learning

SQL

Streaming

Graph

Map Reduce

Spark

GFS

Allocate resources
How do we share CPU between processes?

Policy: Which process is chosen to run

Time sharing

CPU

P1 P2 P3 P1
0 10 20 30
CLUSTER SCHEDULING

- Time sharing
  - Context switch is more expensive

- Space sharing
  - Partition resources at the same time across jobs

- Policies
  - Aware of locality

Failures / Fault Tolerance

Spark

Web search

Clusters and servers diagram
TARGET ENVIRONMENT

Multiple MapReduce versions

Mix of frameworks: MPI, Spark, MR

Data sharing across frameworks

Avoid per-framework clusters

Static partitions
Not good for resource utilization
DESIGN

Two-level scheduling

Centralized master

Agent on every machine

Framework scheduler

Hadoop scheduler  →  Centralized master

MPI scheduler

ZooKeeper quorum

Simplicity across frameworks
RESOURCE OFFERS

MPI

Framework 1
- Job 1
- Job 2
- FW Scheduler

Allocation module

Framework 2
- Job 1
- Job 2
- FW Scheduler

Mesos master

Slave 1
- Executor
- Task

Slave 2
- Executor
- Task

Heartbeats (information)

Resource offer

Which framework should we offer?
CONSTRAINTS

Examples of constraints

Data locality → soft constraint → prefer to run task at location

GPU machines → hard constraint → cannot start task if constraint not satisfied

Constraints in Mesos:

Applications can reject offers

Optimization: Filters

→ reduces the number of rejected offers
DESIGN DETAILS

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

Isolation
Containers (Docker)

cgroups (linux support)

MPI: 8 machines will not revoke guaranteed resources

if using 12 machines can revoke 4 machines
FAULT TOLERANCE

- Soft state

- Forward this failure to Hadoop

- Hadoop scheduler
- MPI scheduler
- Mesos master
- Standby master
- Standby master
- ZooKeeper quorum

- Fails
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

You weigh resources proportional to priority / weight
CENTRALIZED VS DISTRIBUTED

Framework complexity → Every framework dev needs to implement a scheduler

Fragmentation, Starvation

Min offer 2 especially if a job has large resource needs 8GB, 8 CPU

Inter-dependent framework

≤ 2 frameworks cannot be colocated
COMPARISON: YARN

Per-job scheduler

AM asks for resource
RM replies

ResourceManager

Scheduler
AMService

client
Client -- RM

client

RM -- AM

MPI AM

Node Manager

MR

Node Manager

Node Manager

.. .

Node Manager

Umbilical

Container

Container

Container

Apache Hadoop

I support MR versions
COMPARISON: BORG

Single centralized scheduler

Requests mem, cpu in cfg
Priority per user / service

Support for quotas / reservations

bin packing
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/FSkKVbu94nLA4g3v9
What are some problems that could come up if we scale from 10 frameworks to 1000 frameworks in Mesos?

- Overhead of resource offers goes up linearly
  → Time to allocate can go up!

- Developer complexity to implement 1000 schedulers

- Failures
  → Time to recover from Master failure goes up!

- More possibility of fragmentation (minimum offer size)

Allocation module might be slower
Next class: Scheduling Policy

Further reading

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