CS 744: MESOS

Shivaram Venkataraman
Fall 2020

Good morning!
- Assignment 1: How did it go?  → Fill out the poll!
- Assignment 2 out tonight  → ML distributed
- Project details
  - Create project groups  → ~3 students
  - Bid for projects/Propose your own  → next week
- Work on Introduction  → 1-2 page
- Check-in
- Final report and poster session
COURSE FORMAT

Paper reviews

“Compare, contrast and evaluate research papers”

Discussion
Scalable Storage Systems

Computational Engines

Resource Management

Datacenter Architecture

Applications

Machine Learning | SQL | Streaming | Graph

→ MR, A part

⇒ GFS

Assignment
How do we share CPU between processes?
CLUSTER SCHEDULING

Scale → large number of machines

Fairness → similarity between OS & cluster

Multi-core or NUMA-aware scheduling

Space sharing

Fault tolerance

Preferences (placement, constraint)

MR

Spark

Time
TARGET ENVIRONMENT

Multiple MapReduce versions

Mix of frameworks: MPI, Spark, MR

Data sharing across frameworks

Avoid per-framework clusters

Utilization ▼ Not all resources are used

Sharing ▼

Different kinds of applications on same cluster

→ Spark → PageRank
→ MR → Grep
→ Indexing...

100 machine

30 mc Mark

40 MR

30 MPI

Data
Two-level scheduling

- Odd numbers: 3 or 5 is common

- Mesos
  - Scheduling across framework
  - Per-framework scheduler

- Single master
  - Single namenode
  - Chukserver

- Single master
  - Simple at a cluster wide scheduler
  - Add new frameworks in the future

- Scalability, Flexibility

Design diagram:

- Hadoop scheduler
- MPI scheduler
- Mesos master
- Standby master
- Standby master
- Mesos slave
  - Hadoop executor
  - MPI executor
  - Task
  - Task
  - Task

- ZooKeeper quorum
- Spark driver

Note: Graphical representation of the design of a distributed computing system focusing on scheduling and scalability.
RESOURCE OFFERS

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
Examples of constraints:

- Data locality → soft
- GPU machines → hard

Constraints in Mesos:

- Frameworks can reject offers
- "Filters" → Boolean functions
DESIGN DETAILS

Allocation:
Guaranteed allocation, revocation

→ Short-lived tasks!

Isolation
Containers (Docker)

Other frameworks express interest
FAULT TOLERANCE

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

Mesos slave
- Hadoop executor
- MPI executor
- task

Heartbeat

Mesos master failure doesn't affect jobs

Soft state

Talking to Mesos slaves
PLACEMENT PREFERENCES

What is the problem?

If you have more frameworks with prep than machines available in the cluster

How do we do allocations?

weighted lottery scheme

make offers proportional in size to the overall resources that a framework needs
Centralized vs Decentralized

Centralized

- Scalability
  - ~100s of frameworks
  - each ~100s of apps
- Optimal solution

Decentralized

- Handle new frameworks
- Complexity for framework developer
CENTRALIZED VS DECENTRALIZED

Framework complexity ✓

Fragmentation, Starvation → If resource offers are too small

Inter-dependent framework
COMPARISON: YARN

Per-job scheduler

RM -- NodeManager

"Mesos matter"

AM asks for resource
RM replies

per-job scheduler
COMPARISON: BORG

Single centralized scheduler

Requests mem, cpu in cfg
Priority per user / service

Support for quotas / reservations

Better 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/urHSeukfyipCKjue6
What are some problems that could come up if we scale from 10 frameworks to 1000 frameworks in Mesos?

→ Fragmentation / starvation odds go up

→ Master bottleneck?
  
  → it takes time to wait for frameworks to reply

→ Pre-emption? Yes.

→ Failure recovery takes longer? Why?
  
  → Failure recovery takes longer? Why?
  
  → Mesos master has soft state ~ unclear?
Poncée: Ponge

Rigid framework: Terror

Static Partitioning, Mesos

Share of Cluster

Time (s)

(a) Facebook Hadoop Mix

(b) Large Hadoop Mix

(c) Spark

(d) Torque / MPI

Elastic 2x framework can use

1.5x faster completion

Similar completion time

Elastic framework

Able to maintain MPIs tenure
List any one difference between an OS scheduler and Mesos

Motivation part of the lecture

Spark on oversubscribed clusters

- long running jobs
  - Executor → cache RDDs
    - → "shuffle files" on local disk
    - → gets preempted → cache is blown away
    - → shuffle files long lived beyond "guaranteed share"

Data locality

- Input (HDFS)
- Memory (Executor)
- Shuffle (Executor)
- Output

Coarse Grained Executor Backend
Resource offers

How does it "perform" better?

- "ramp-up"
- Optimal policy

(i) Time to schedule
(ii) Time to completion

Comparisons with YARN, Borg
Next class: Scheduling Policy

Assignment 2 will be released

Further reading

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