CS 677: Big Data

Computation Orchestration

Lecture 15

Orchestration

- How do these large organizations manage their huge fleets of clusters?
  - MR, Spark, Hive, … the list goes on
- Clearly hand-configuring each cluster does not scale
  - But it used to be pretty normal
  - More common: configuration management tool
- Better: declare cluster state, let a scheduler enforce whatever policies you set

Distributed Schedulers

- Basic approach for orchestrating computations over a distributed system
- Given a set of resources, allocate incoming tasks
  - Based on priority, workload size, past usage, etc.
  - Queue up tasks if there aren't enough resources available
- Deal with fault tolerance via speculative execution
- The hard part: accounting for heterogeneity
  - (hard, but doable)

The Real Issue

- Nobody just runs Hadoop, Spark, Kafka, Flink, etc.
- It is extremely wasteful to allocate hardware for a specific purpose
  - E.g., “these machines are a Hadoop cluster”
- Many of these applications depend on or support one another
  - E.g., HDFS

Cluster Management: Zookeeper

- Many open source big data platforms use(d) Apache Zookeeper to manage their individual clusters
- ZK is great at electing leaders, coming to a consensus, etc…
  - …but it's not really designed to manage an entire datacenter
- Minimal KV storage, configuration management functionality

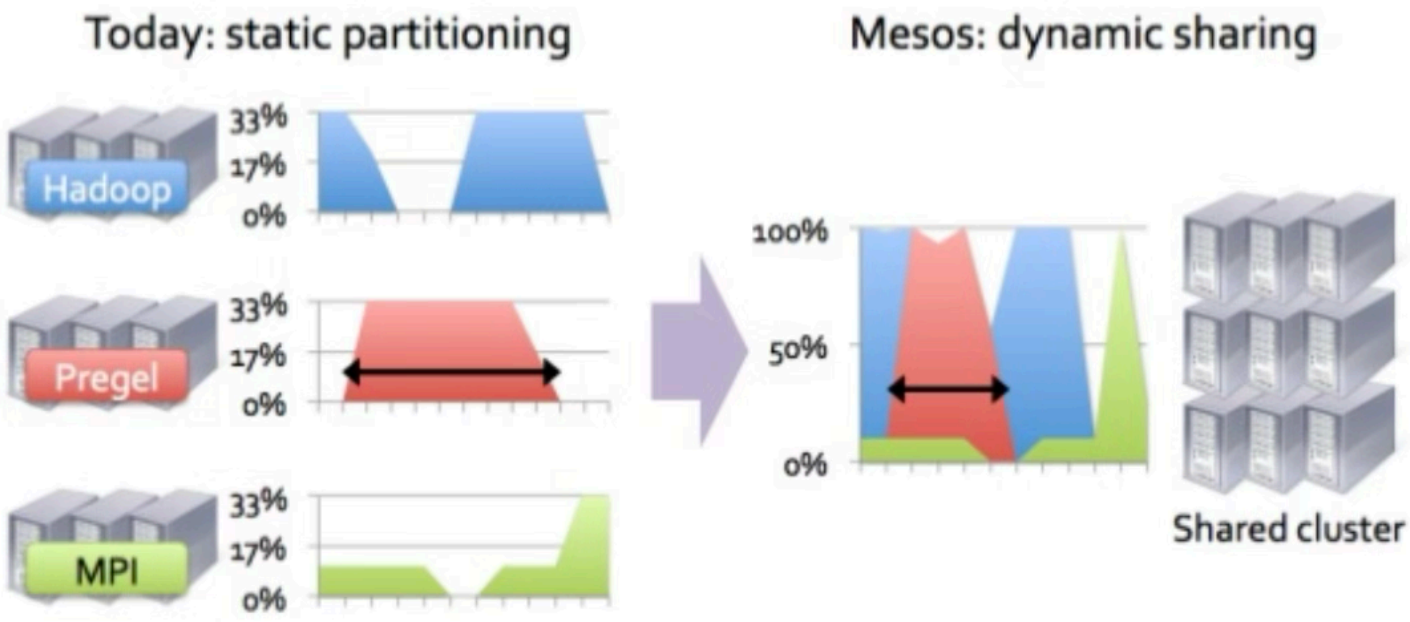
Secret Internal Google Project

- For a very long time, Google operated a cluster management platform to handle these issues
  - Unlike many of their projects, they didn't publish many details about how the system worked
- Generally… If they haven't published it yet, they're still using it 😄
- "Borg"

Borg

- According to Google, Borg saved them from building an entire extra datacenter!
  - Based on the huge increase in resource utilization
  - More utilization = more efficiency
    - Same logic as Amazon behind creating AWS
- Proprietary, but folks at Berkeley AmpLab were able to piece together details based on conversations with Google engineers
  - Designed a new cluster management system (Mesos)

Mesos / Borg


Today: static partitioning
Hadoop
Pregel
MPI
33%
17%
0%
Mesos: dynamic sharing
100%
50%
0%
Shared cluster

Containers

- How do you make all these ”clusters” work together seamlessly?
- VMs? They’re great, but take a long time to start, stop and are resource intensive
- Containers
  - Processes in a container believe that they are running on their own machine
  - Isolated from all other processes on the host
  - Can dynamically change resource allocations (CPU, Memory, etc.)

cgroups, namespaces

- On Linux, control groups and namespaces allow rapid changes to how resources are allocated
- Want to limit a container's disk write speed, CPU usage, memory, etc.? You can do it on the fly
- Much of this infrastructure (basically, the stuff that makes containers work) was built by Google engineers

Control Groups Features

- Resource Limits (*e.g. Putting an upper bound on memory*)
- Prioritization (*Give certain groups higher CPU usage, disk I/O, or network I/O throughput*)
- Accounting (*monitoring various resource usage metrics*)
- Control (*Freezing, checkpointing, restarting*)

Namespace Isolation

- Beyond the control offered by cgroups, namespaces provide container isolation
- Users, process IDs, hostname, timezone are distinct from the host OS
  - Even though they're running the same kernel
- Can have separate mount points (both physical and virtual devices)
- Perhaps most importantly: network isolation so the container appears to be an individual network host

Kubernetes

- Many “borg” features eventually found their way to their open-source Kubernetes project
- Run most cluster software in lightweight containers that can be moved around, have resource limitations, etc.
- This practice has spread to most large orgs managing huge datasets / processing needs