CS 677: Big Data

Cluster Orchestration

Lecture 11

Today’s Schedule

- Modern cluster orchestration
- Scaling Big Data Infrastructure
- Log Management

Today's Schedule

- **Modern cluster orchestration**
- Scaling Big Data Infrastructure
- Log Management

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

Stepping Back: 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 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

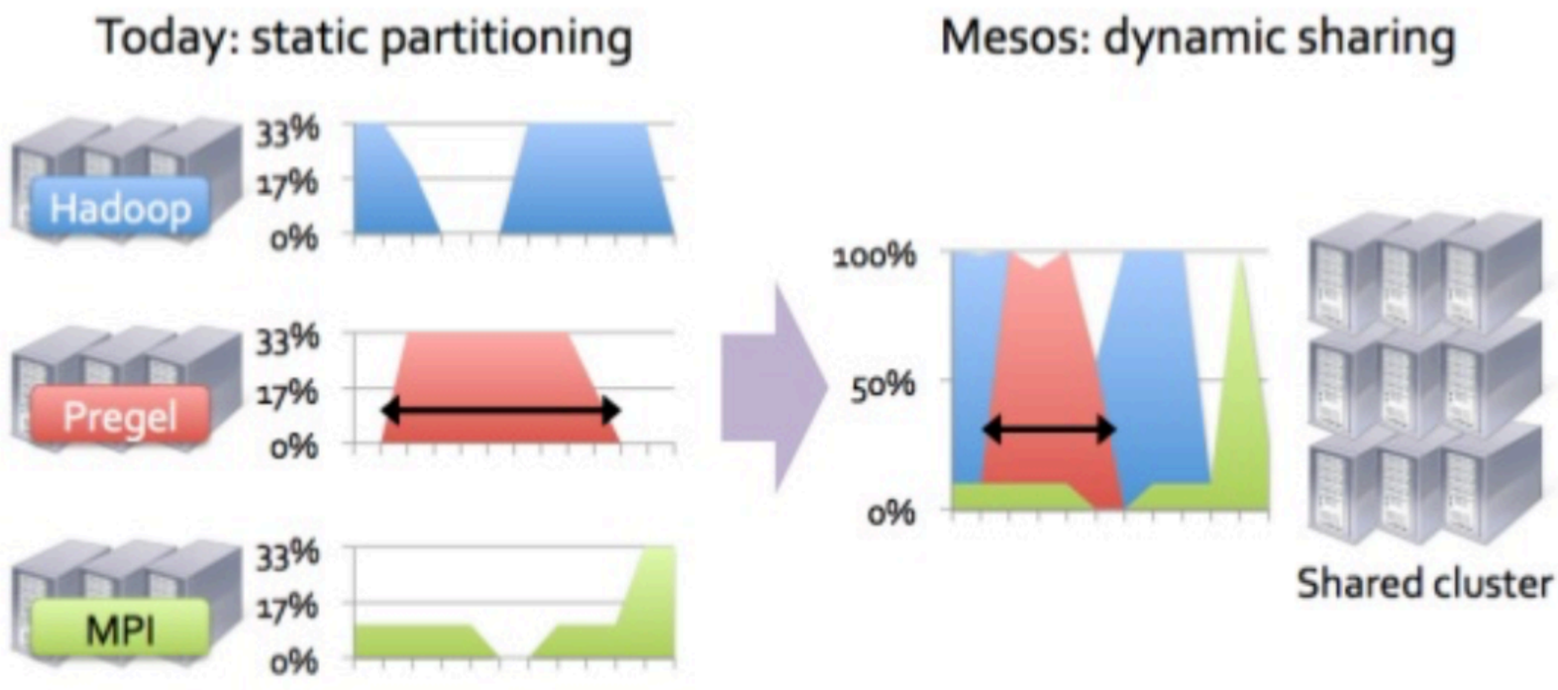
Secret Internal Google Project

- 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
  - And of course, design a new cluster management system based on their expertise in this area of research
  - Mesos

Mesos / Borg



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

Containers

- How do you make all these "clusters" work together seamlessly?
- VMs
  - Great and all, but take a long time to spin up, shut down, 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.)
  - Run on the host kernel

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
  - 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

Namespace Isolation Demo

- `unshare --user --pid --map-root-user \`
  `--mount-proc --fork bash`

```
unshare --user --pid --map-root-user \
      --mount-proc --fork bash
```

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

Today's Schedule

- Modern cluster orchestration
- **Scaling Big Data Infrastructure**
- Log Management

Scaling Big Data Mining Infrastructure: The Twitter Experience

- Insights from two vantage points:
  - Jimmy Lin – Prof at University of Maryland (extended sabbatical from 2010 – 2012 at Twitter)
  - Dmitriy Ryaboy – Engineering manager of the analytics infrastructure team, Twitter
- Future work, research directions, gaps in the literature
- Recommendations for industry use of big data applications

Growth of Analytics at Twitter

- 2010
  - 100 employees
  - 4 people devoted to analytics
  - 30-node Hadoop cluster
- 2012
  - 1000+ employees
  - Thousands of Hadoop nodes, many data centers
  - 100 TB of raw data ingested per day
  - ~10,000+ MapReduce jobs per day

Key Insights (1/2)

- Exploratory/predictive analytics is under-represented in the literature
  - Data scientists have a difficult time figuring out ***what*** data points exist, how they're structured, and their relationships
- Big data analytics is no longer a competitive advantage, it's a requirement
  - Everybody is doing it!
    - EVERYBODY
- Often, analysis of analytics data is important

Key Insights (2/2)

- Ensembles used heavily for production machine learning operations
  - Better results with large training sets
- Best practices when handling logging at scale
- Service architecture
  - Amazon-style microservices with well-defined interfaces

Ensembles

- Rather than dumping all your data into one gigantic model, build multiple models
  - Statistical, machine learning, etc.
- Use predictions/classifications from these models as results from a "group of experts"
- This is good for performance (parallelism) and also means that you can have models specialize for a particular part of the dataset

Today's Schedule

- Modern cluster orchestration
- Scaling Big Data Infrastructure
- **Log Management**

Data Source for Analytics: Logs

- From a high level, this entire paper is about logs
- Each application manages its own logging system
  - Printing to stdout/stderr
  - Logging with several different Java frameworks
  - System logs
- Every action taken at Twitter is logged
- Logs become the analytics dataset

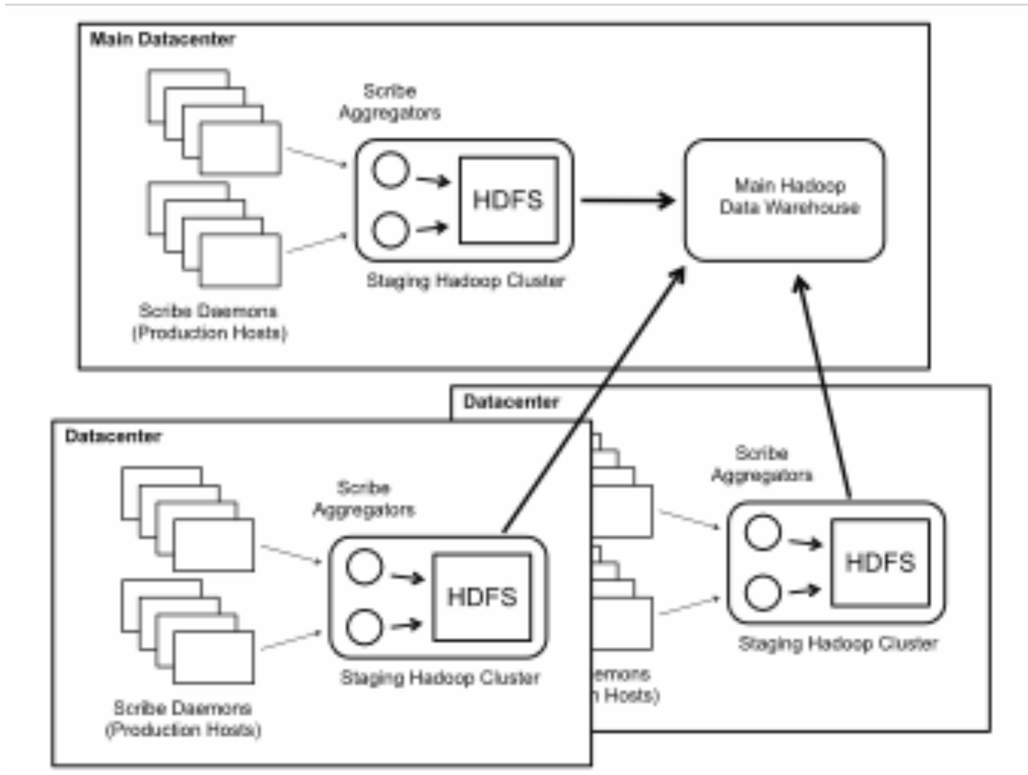
Logging Issues: Software

- Sharded MySQL databases were used to handle log messages
- Relational databases are **not** a good fit for logs
  - Scaling is tough
  - Transaction support is not necessary
  - Data points do not need to be mutable
  - Schemas are expensive to change
- Better solution: online analytics processing (OLAP) systems

Solution: Scribe

- Developed by Facebook
- Designed specifically for managing large amounts of log data
- Scribe aggregators collect logs and push them out to Hadoop *staging clusters*
- Eventually, logs all find their way to the *main data warehouse*
- Alternatives: Apache Flume, Kafka

Scribe Setup



"We consider the problem of log transport mostly solved"

Logging Issues: Schema

```
^(\\w+\\s+\\d+\\s+\\d+:\\d+:\\d+)\\s+
([^@]+?)@(\\S+)\\s+(\\S+):\\s+(\\S+)\\s+(\\S+)
\\s+((?:\\S+?,\\s+)*(?:\\S+?))\\s+(\\S+)\\s+(\\S+)
\\s+\\[([^\\]]+)\\]\\s+\"(\\w+)\\s+([^\"\\\\]*
(?:\\\\.[^\"\\\\]*)*)\\s+(\\S+)\"\\s+(\\S+)\\s+
(\\S+)\\s+\"([^\"\\\\]*(?:\\\\.[^\"\\\\]*)*)
\"\\s+\"([^\"\\\\]*(?:\\\\.[^\"\\\\]*)*)\"\\s*
(\\d*-[\\d-]*)?\\s*(\\d+)?\\s*(\\d*\\.[\\d\\.]*)?
(\\s+[-\\w]+)?.*$
```

Log-parsing regex, 2010

MySQL Log Schema

```
create table `my_audit_log` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `created_at` datetime,
  `user_id` int(11),
  `action` varchar(256),
  ...
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Table 1: Using MySQL for logging is a bad idea.

Issues with Plain Text

- No types
- No formatting requirements
  - Only inter-organization conventions that aren't followed
- What is the delimiter?!?!
  - Newlines? What about multi-line stack traces?
  - Spaces? Tabs?
    - Both were found in Twitter usernames

What about JSON? (1/2)

- Key-value storage, primitive support for types
- JSON logs “generally start out as an adequate solution, but then gradually spiral into a persistent nightmare”
- No naming conventions to identify keys
  - CamelCase, smallCamelCase, snake_case, and the mythical dunder__snake!
  - You really need to inspect each JSON document from each source

JSON Log “Schema”

```
{
  "token": 945842,
  "feature_enabled": "super_special",
  "userid": 229922,
  "page": "null",
  "info": { "email": "my@place.com" }
}
```

What about JSON? (2/2)

- There is no standard way of representing null
  - null, NULL, nil
- Arbitrary nesting is supported
  - How to know when the nesting stops?
  - Turtle { Turtle { Turtle { Turtle { Turtle { Turtle { Turtle {
- Some Solutions:
  - Reading, remembering code of each component
  - Generating histograms of the keys used

Log Schema Solution: Apache Thrift

- Supports data types
- Optional fields are clearly marked
- Fields can be deprecated
- Structure is well-defined
- Binary serialization improves performance
  - Decouples logical and physical representations
- Alternatives: Google's Protobufs, Apache Avro
- Still no solution for consistent naming…

Sample Thrift "Schema"

```
struct MessageInfo {
  1: optional string name
  2: optional string email  // NOTE: unverified.
}

enum Feature {
  super_special,
  less_special
}

struct LogMessage {
  1: required i64 token
  2: required string user_id
  3: optional list<Feature> enabled_features
  4: optional i64 page = 0
  5: optional MessageInfo info
}
```

Additional Schema Enhancements

- Hive's HCatalog provides global schema and naming conventions for all HDFS-based applications
  - So far, only used by some teams
- Provenance data: determining the chain of custody for outputs
  - Hooks into HCatalog load/store operations
  - Builds a graph of dependencies for datasets

Scaling out Machine Learning

- At Twitter, they found:
  - The more data, the better
  - Simple features are often strikingly effective
- Most user-friendly ML toolkits (Weka, Mallet) are designed for single-node setups
  - Leads to computational inefficiencies
- Sampling can hurt rather than help
  - Reduces dataset sizes (see point #1)

Distributed ML Tools

- Bottou, Vowpal Wabbit show promise, but do not offer integrated solutions
- Impedance mismatch between Hadoop/MapReduce and common ML tasks
- Issues with dataset formats: transforming the data to fit into the framework takes longer than the machine learning!
- MLBase, RDDs seem to address some of these points

Twitter's Open Questions

- Big Data Visualization
  - Aggregating and displaying terabytes of information on a user's laptop still hasn't been solved adequately
  - Solutions still under development
- Real-time interaction with large datasets
  - Data mining is an iterative, exploratory task
  - Current workflow: write Pig script, submit job, wait 5 minutes, discover error, correct error, wait another 5 minutes

Some Thoughts

- If logging (write-heavy operation) is the most important analytics component at Twitter, why didn't they consider Cassandra?
- Sampling **can** be very useful, if done carefully
- An interesting choice to not develop many of the tools in-house
  - Many of the "hip" tech companies have "not invented here" (NIH) syndrome… so maybe this is a good thing
  - Analytics may be the difference between a profitable Twitter and a bankrupt one