Petabyte scale on commodity infrastructure

Owen O'Malley
Eric Baldeschwieler
Yahoo Inc!
{owen,eric14}@yahoo-inc.com

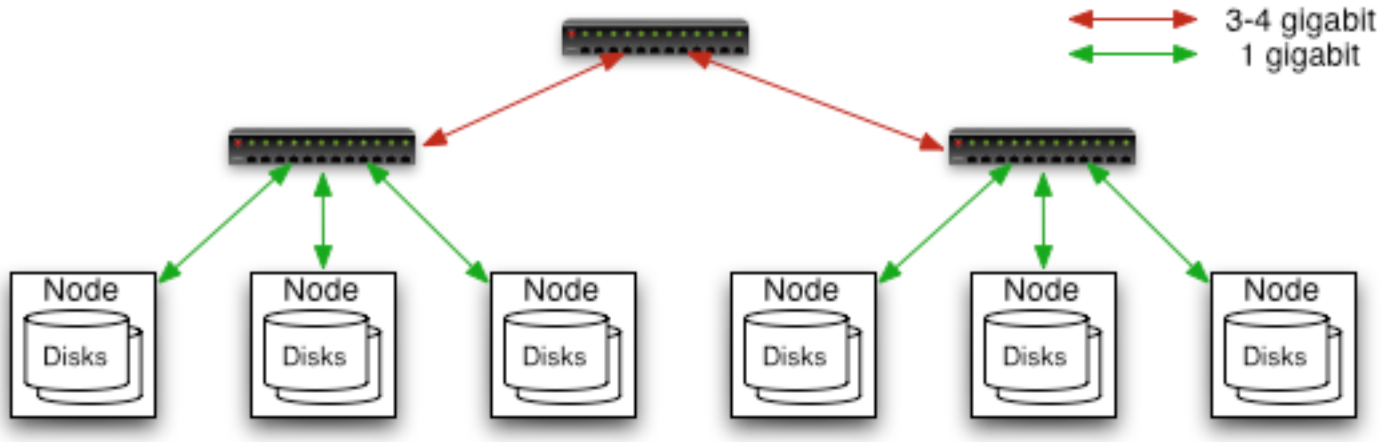
Hadoop: Why?

- **Need to process huge datasets on large clusters of computers**
- **In large clusters, nodes fail every day**
  - Failure is expected, rather than exceptional.
  - The number of nodes in a cluster is not constant.
- **Very expensive to build reliability into each application.**
- **Need common infrastructure**
  - Efficient, reliable, easy to use

Hadoop: How?

- **Commodity Hardware Cluster**
- **Distributed File System**
  - Modeled on GFS
- **Distributed Processing Framework**
  - Using Map/Reduce metaphor
- **Open Source, Java**
  - Apache Lucene subproject

Commodity Hardware Cluster



- **Typically in 2 level architecture**
  - Nodes are commodity PCs
  - 30-40 nodes/rack
  - Uplink from rack is 3-4 gigabit
  - Rack-internal is 1 gigabit

Distributed File System

- **Single namespace for entire cluster**
  - Managed by a single *namenode*.
  - Files are write-once.
  - Optimized for streaming reads of large files.
- **Files are broken in to large blocks.**
  - Typically 128 MB
  - Replicated to several *datanodes*, for reliability
- **Client talks to both namenode and datanodes**
  - Data is not sent through the namenode.
  - Throughput of file system scales nearly linearly with the number of nodes.

Block Placement

- **Default is 3 replicas, but settable**
- **Blocks are placed:**
  - On same node
  - On different rack
  - On same rack
  - Others placed randomly
- **Clients read from closest replica**
- **If the replication for a block drops below target, it is automatically rereplicated.**

Data Correctness

- **Data is checked with CRC32**
- **File Creation**
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- **File access**
  - Client retrieves the data and checksum from DataNode
  - If Validation fails, Client tries other replicas

Distributed Processing

- **User submits Map/Reduce *job* to *JobTracker***
- **System:**
  - Splits job into lots of *tasks*
  - Monitors tasks
  - Kills and restarts if they fail/hang/disappear
- **User can track progress of job via web ui**
- **Pluggable file systems for input/output**
  - Local file system for testing, debugging, etc…
  - KFS and S3 also have bindings…

Hadoop Map-Reduce

- **Implementation of the Map-Reduce programming model**
  - Framework for distributed processing of large data sets
    - Data handled as collections of key-value pairs
  - Pluggable user code runs in generic framework
- **Very common design pattern in data processing**
  - Demonstrated by a unix pipeline example:

```
cat *  | grep | sort    | unique -c | cat > file
input  | map  | shuffle | reduce    | output
```

  - Natural for:
    - Log processing
    - Web search indexing
    - Ad-hoc queries
  - Minimizes trips to disk and disk seeks
- **Several interfaces:**
  - Java, C++, text filter

Map/Reduce Optimizations

- **Overlap of maps, shuffle, and sort**
- **Mapper locality**
  - Schedule mappers close to the data.
- **Combiner**
  - Mappers may generate duplicate keys
  - Side-effect free reducer run on mapper node
  - Minimize data size before transfer
  - Reducer is still run
- **Speculative execution**
  - Some nodes may be slower
  - Run duplicate task on another node

Running on Amazon EC2/S3

- **Amazon sells cluster services**
  - EC2: $0.10/cpu hour
  - S3: $0.20/gigabyte month
- **Hadoop supports:**
  - EC2: cluster management scripts included
  - S3: file system implementation included
- **Tested on 400 node cluster**
- **Combination used by several startups**

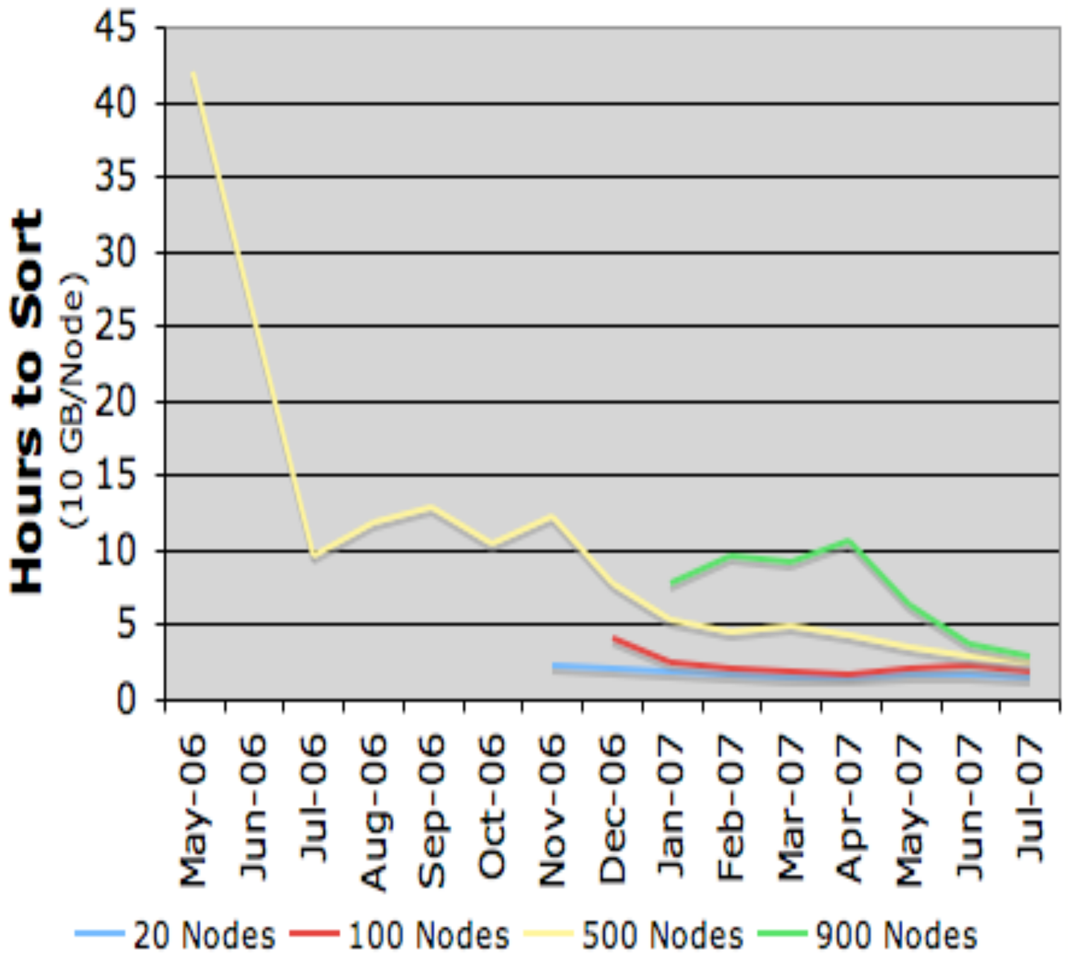
Hadoop On Demand

- **Traditionally Hadoop runs with dedicated servers**
- **Hadoop On Demand works with a batch system to allocate and provision nodes dynamically**
  - Bindings for Condor and Torque/Maui
- **Allows more dynamic allocation of resources**

Yahoo's Hadoop Clusters

- We have ~10,000 machines running Hadoop
- Our largest cluster is currently 2000 nodes
- 1 petabyte of user data (compressed, unreplicated)
- We run roughly 10,000 research jobs / week

Scaling Hadoop



- **Sort benchmark**
  - Sorting random data
  - Scaled to 10GB/node
- **We've improved both scalability and performance over time**
- **Making improvements in frameworks helps a lot.**

Coming Attractions

- **Block rebalancing**
- **Clean up of HDFS client write protocol**
  - Heading toward file append support
- **Rack-awareness for Map/Reduce**
- **Redesign of RPC timeout handling**
- **Support for versioning in Jute/Record IO**
- **Support for users, groups, and permissions**
- **Improved utilization**
  - Your feedback solicited

Upcoming Tools

- **Pig**
  - A scripting language/interpreter that makes it easy to define complex jobs that require multiple map/reduce jobs.
  - Currently in Apache Incubator.
- **Zookeeper**
  - Highly available directory service
  - Support master election and configuration
  - Filesystem interface
  - Consensus building between servers
  - Posting to SourceForge soon
- **HBase**
  - BigTable-inspired distributed object store, sorted by primary key
  - Storage in HDFS files
  - Hadoop community project

Collaboration

- **Hadoop is an Open Source project!**
- **Please contribute your xtrace hooks**
- **Feedback on performance bottleneck welcome**
- **Developer tooling for easing debugging and performance diagnosing are very welcome.**
  - IBM has contributed an Eclipse plugin.
- **Interested your thoughts in management and virtualization**
- **Block placement in HDFS for reliability**

Thank You

- **Questions?**
- **For more information:**
  - Blog on http://developer.yahoo.net/
  - Hadoop website: http://lucene.apache.org/hadoop/