Hadoop at Yahoo!

Owen O’ Malley
Yahoo!, Grid Team
owen@yahoo-inc.com
Who Am I?

• Yahoo! Architect on Hadoop Map/Reduce
  – Design, review, and implement features in Hadoop
  – Working on Hadoop full time since Feb 2006
  – Before Grid team, I worked on Yahooos’ WebMap
  – PhD from UC Irvine in Software Testing.

• VP of Apache for Hadoop
  – Chair of Hadoop Program Management Committee
  – Responsible for
    • Building the Hadoop community
    • Interfacing between the Hadoop PMC and the Apache Board
Problem

• How do you scale up applications?
  – 100’s of terabytes of data
  – Takes 11 days to read on 1 computer

• Need lots of cheap computers
  – Fixes speed problem (15 minutes on 1000 computers), but…
  – Reliability problems
    • In large clusters, computers fail every day
    • Cluster size is not fixed

• Need common infrastructure
  – Must be efficient and reliable
Solution

- Open Source Apache Project
- Hadoop is Doug Cutting’s son’s stuff elephant.
- Hadoop Core includes:
  - Distributed File System - distributes data
  - Map/Reduce - distributes application
- Written in Java
- Runs on
  - Linux, Mac OS/X, Windows, and Solaris
  - Commodity hardware
Hadoop Timeline

- 2004 – HDFS & map/reduce started in Nutch
- Dec 2005 – Nutch ported to map/reduce
- Jan 2006 – Doug Cutting joins Yahoo
- Feb 2006 – Factored out of Nutch.
- Apr 2006 – Sorts 1.9 TB on 188 nodes in 47 hours
- May 2006 – Yahoo sets up research cluster
- Jan 2008 – Hadoop is a top level Apache project
- Feb 2008 – Yahoo creating Webmap with Hadoop
- Apr 2008 – Wins Terabyte sort benchmark
- Aug 2008 – Ran 4000 node Hadoop cluster
Typically in 2 level architecture
- Nodes are commodity Linux PCs
- 40 nodes/rack
- Uplink from rack is 8 gigabit
- Rack-internal is 1 gigabit all-to-all
Distributed File System

- Single petabyte file system for entire cluster
  - Managed by a single namenode.
  - Files are written, read, renamed, deleted, but append-only.
  - Optimized for streaming reads of large files.
- Files are broken into large blocks.
  - Transparent to the client
  - Blocks are 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.
- Access from Java, C, or command line.
Block Placement

• Default is 3 replicas, but settable
• Blocks are placed (writes are pipelined):
  – On same node
  – On different rack
  – On the other rack
• Clients read from closest replica
• If the replication for a block drops below target, it is automatically re-replicated.
HDFS Dataflow

![Diagram of HDFS Dataflow with client nodes and data nodes connected through network nodes.]

Evergreen 2008
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
• Periodic validation by DataNode
Map/Reduce

- Map/Reduce is a programming model for efficient distributed computing
- It works like a Unix pipeline:
  - cat input | grep | sort | uniq -c | cat > output
  - Input | Map | Shuffle & Sort | Reduce | Output
- Efficiency from
  - Streaming through data, reducing seeks
  - Pipelining
- A good fit for a lot of applications
  - Log processing
  - Web index building
  - Data mining and machine learning
Map/Reduce Dataflow
Map/Reduce features

- Java, C++, and text-based APIs
  - In Java use Objects and C++ bytes
  - Text-based (streaming) great for scripting or legacy apps
  - Higher level interfaces: Pig, Hive, Jaql

- Automatic re-execution on failure
  - In a large cluster, some nodes are always slow or flaky
  - Framework re-executes failed tasks

- Locality optimizations
  - With large data, bandwidth to data is a problem
  - Map-Reduce queries HDFS for locations of input data
  - Map tasks are scheduled close to the inputs when possible
Word Count Example

- **Mapper**
  - Input: value: lines of text of input
  - Output: key: word, value: 1

- **Reducer**
  - Input: key: word, value: set of counts
  - Output: key: word, value: sum

- **Launching program**
  - Defines the job
  - submits job to cluster
Word Count Dataflow

Input:
- the quick brown fox
- the fox ate the mouse
- how now brown cow

Map:
- Map: quick, 1
- Map: ate, 1
- Map: cow, 1

Shuffle & Sort:
- the, 1
- brown, 1
- fox, 1
- how, 1
- now, 1
- brown, 1

Reduce:
- Reduce: brown, 2
- Reduce: fox, 2
- Reduce: how, 1
- Reduce: now, 1
- Reduce: the, 3

Output:
- Output: ate, 1
- Output: cow, 1
- Output: mouse, 1
- Output: quick, 1

Evergreen 2008
public static class MapClass extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value,
        OutputCollector<Text, IntWritable> output,
        Reporter reporter) throws IOException {
        String line = value.toString();
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            output.collect(word, one);
        }
    }
}
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
        int sum = 0;
        while (values.hasNext()) {
            sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
    }
}

import dumbo

def mapper(key, value):
    for word in value.split(): yield word, 1

def reducer(key, values):
    yield key, sum(values)

if __name__ == "__main__":
    dumbo.run(mapper, reducer)
Why Yahoo! is investing in Hadoop

• We started with building better applications
  – Scale up web scale batch applications (search, ads, …)
  – Factor out common code from existing systems, so new applications will be easier to write
  – Manage the many clusters we have more easily

• The mission now includes research support
  – Build a huge data warehouse with many Yahoo! data sets
  – Couple it with a huge compute cluster and programming models to make using the data easy
  – Provide this as a service to our researchers
  – We are seeing great results!
    • Experiments can be run much more quickly in this environment
Running the Production WebMap

- Search needs a graph of the “known” web
  - Invert edges, compute link text, whole graph heuristics
- Periodic batch job using Map/Reduce
  - Uses a chain of ~100 map/reduce jobs
- Scale
  - 100 billion nodes and 1 trillion edges
  - Largest shuffle is 450 TB
  - Final output is 300 TB compressed
  - Runs on 10,000 cores
- Written mostly using Hadoop’s C++ interface
WebMap in Context

Web

Crawler -> Webmap

Indexer

Batch

Real-time

Users -> Front End

Search Engine

Evergreen 2008
Inverting the Webmap

• One job inverts all of the edges in the Webmap
  – From a cache of web pages
  – Finds the text in the links that point to each page.

• Mapper:
  – Input: key: URL; value: page HTML
  – Output: key: target URL; value: source URL, Text from link

• Reducer:
  – Output: new column in URL table with set of linking pages and link text
Finding Duplicate Web Pages

- Find “close” textual matches in the cache of web pages
- Want to compare each pair of web pages
  - But it would take way too long

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>T</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

- Instead use a hash based on page’s visible text
  - Ignore headers and footers
  - Shingles
Finding Duplicates

• Job 1 finds the “groups”
  – Mapper:
    • Input: key: URL; value: page HTML
    • Output: key: page text hash, goodness; value: URL
  – Reducer:
    • Output: key: page text hash; value: URL (best first)

• Job 2 sorts results back into URL order
  – Mapper:
    • Output: key: URL; value: best URL
  – Reducer:
    • Tags duplicate URLs with their best representative
Research Clusters

• The grid team runs research clusters as a service to Yahoo researchers
  – Analytics as a Service
• Mostly data mining/machine learning jobs
• Most research jobs are *not* Java:
  – 42% Streaming
    • Uses Unix text processing to define map and reduce
  – 28% Pig
    • Higher level dataflow scripting language
  – 28% Java
  – 2% C++
Hadoop clusters

- We have ~20,000 machines running Hadoop
- Our largest clusters are currently 2000 nodes
- Several petabytes of user data (compressed, unreplicated)
- We run hundreds of thousands of jobs every month
• Needed offline conversion of public domain articles from 1851-1922.
• Used Hadoop to convert scanned images to PDF
• Ran 100 Amazon EC2 instances for around 24 hours
• 4 TB of input
• 1.5 TB of output

Published 1892, copyright New York Times
Evergreen 2008

Terabyte Sort Benchmark

- Started by Jim Gray at Microsoft in 1998
- Sorting 10 billion 100 byte records
- Hadoop won general category in 209 seconds (prev was 297 )
  - 910 nodes
  - 2 quad-core Xeons @ 2.0Ghz / node
  - 4 SATA disks / node
  - 8 GB ram / node
  - 1 gb ethernet / node and 8 gb ethernet uplink / rack
  - 40 nodes / rack
- Only hard parts were:
  - Getting a total order
  - Converting the data generator to map/reduce
- http://developer.yahoo.net/blogs/hadoop/2008/07
Hadoop Community

• Apache is focused on project communities
  – Users
  – Contributors
    • write patches
  – Committers
    • can commit patches too
  – Project Management Committee
    • vote on new committers and releases too

• Apache is a meritocracy

• Use, contribution, and diversity is growing
  – But we need and want more!
Size of Developer Community

Developer Messages per Month

- Zookeeper
- Hbase
- Core
Size of User Community

User Message by Month

- Zookeeper
- Hbase
- Core

Evergreen 2008
Who Uses Hadoop?

- Amazon/A9
- Facebook
- Google
- IBM
- Joost
- Last.fm
- New York Times
- PowerSet (now Microsoft)
- Quantcast
- Veoh
- Yahoo!
- More at http://wiki.apache.org/hadoop/PoweredBy
What’s Next?

• 0.19
  – File Appends
  – Total Order Sampler and Partitioner
  – Pluggable scheduler

• 0.20
  – New map/reduce API
  – Better scheduling for sharing between groups
  – Splitting Core into sub-projects (HDFS, Map/Reduce, Hive)

• And beyond
  – HDFS and Map/Reduce security
  – High Availability via Zookeeper
  – Get ready for Hadoop 1.0

Evergreen 2008
• For more information:
  – Website: http://hadoop.apache.org/core
  – Mailing lists:
    • core-dev@hadoop.apache
    • core-user@hadoop.apache
  – IRC: #hadoop on irc.freenode.org