Hadoop In 45 Minutes or Less
Large-Scale Data Processing for Everyone

Tom Wheeler
Who Am I?

- Principal Software Engineer at Object Computing, Inc.
- I worked on large-scale data processing in a previous job – *If only I’d had Hadoop back then...*
What I’m Going to Cover

▶ I’ll explain what Hadoop is
▶ I’ll tell you what problems it can (and can’t) solve
▶ I’ll describe how it works
▶ I’ll show examples so you can see it in action
What is Hadoop?

It’s a framework for large-scale data processing:

- Inspired by Google’s architecture: Map Reduce and GFS
- A top-level Apache project – Hadoop is open source
- Written in Java, plus a few shell scripts
How Did Hadoop Originate?

- Google publishes GFS paper
- Nutch rewritten to use MapReduce
- Hadoop factored out of Nutch
- Becomes a top-level Apache project
- Wins TeraSort benchmark (209 secs)

Timeline:
- 2004: Google publishes GFS paper
- 2005: Nutch rewritten to use MapReduce
- 2006: Hadoop factored out of Nutch
- 2007: Becomes a top-level Apache project
- 2008: Wins TeraSort benchmark (209 secs)
- 2009:
Why Should I Care About Hadoop?

- Fault-tolerant hardware is expensive
- Hadoop is designed to run on cheap commodity hardware
- It automatically handles data replication and node failure
- It does the hard work – you can focus on processing data
Who’s Using Hadoop?

facebook

YAHOO!

The New York Times

Adobe

amazon.com

hulu

Baidu 百度

AOL

Microsoft

Source: Hadoop Wiki, September 2009

Trademarks belong to respective owners and do not necessarily represent an endorsement of Hadoop or this presentation
What Features Does Hadoop Offer?

- API + implementation for working with Map Reduce
- More importantly, it provides infrastructure

**Hadoop Infrastructure**

- Job configuration and efficient scheduling
- Browser-based monitoring of important cluster stats
- Handling failures in both computation and data nodes
- A distributed FS optimized for *HUGE* amounts of data
When is Hadoop a Good Choice?

- When you must process lots of unstructured data
- When your processing can easily be made parallel
- When running batch jobs is acceptable
- When you have access to lots of cheap hardware
When is Hadoop Not A Good Choice?

- For intense calculations with little or no data
- When your processing cannot be easily made parallel
- When your data is not self-contained
- When you need interactive results
- If you own stock in Cray!
**Hadoop Examples/Anti-Examples**

<table>
<thead>
<tr>
<th>Hadoop would be a good choice for...</th>
</tr>
</thead>
<tbody>
<tr>
<td>➤ Indexing log files</td>
</tr>
<tr>
<td>➤ Sorting vast amounts of data</td>
</tr>
<tr>
<td>➤ Image analysis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hadoop would be a poor choice for...</th>
</tr>
</thead>
<tbody>
<tr>
<td>➤ Figuring Pi to 1,000,000 digits</td>
</tr>
<tr>
<td>➤ Calculating Fibonacci sequences</td>
</tr>
<tr>
<td>➤ A general RDBMS replacement</td>
</tr>
</tbody>
</table>
HDFS is perhaps Hadoop’s most interesting feature.

- HDFS = Hadoop Distributed Filesystem (userspace)
- Inspired by Google File System
- High aggregate throughput for streaming large files
- Replication and locality
Data copied into HDFS is split into blocks

Typical block size: UNIX = 4KB vs. HDFS = 128MB
How HDFS Works: Replication

- Each data blocks is replicated to multiple machines
- Allows for node failure without data loss
The Hadoop Cast of Characters: Name Node

- There is only one (active) name node per cluster
- It manages the filesystem namespace and metadata
- SPOF: the one place to spend $$$ for good hardware
There are typically lots of data nodes
It manages data blocks + serves them to clients
Data is replicated – failure is no big deal
The Hadoop Cast of Characters: Job Tracker

- There is exactly one job tracker per cluster
- Receives job requests submitted by client
- Schedules and monitors MR jobs on task trackers
There are typically lots of task trackers
Responsible for executing MR operations
Reads blocks from data nodes
Hadoop supports three modes of operation:

- Standalone
- Pseudo-distributed
- Fully-distributed
Installing Hadoop

The installation process, for distributed modes:

- Requirements: Linux, Java 1.6, sshd, rsync
- Configure SSH for password-free authentication
- Unpack Hadoop distribution
- Edit a few configuration files
- Format the DFS on the name node
- Start all the daemon processes
The basic steps for running a Hadoop job are:

- Compile your job into a JAR file
- Copy input data into HDFS
- Execute `bin/hadoop jar` with relevant args
- Monitor tasks via Web interface (optional)
- Examine output when job is complete
A Simple Hadoop Job

I’ll demonstrate Hadoop with a simple Map Reduce example

- Input is historical data for 30 stocks, 1987-2009
- Records are CSV: symbol, low price, high price, etc.
- Goal is to find largest intra-day price fluctuation
- Output is one record per stock, showing max delta
Our Hadoop Job Illustrated

- Hadoop is all about data processing
- Seeing the data will help to explain the job
Our Hadoop Job Illustrated: Mapper Input

CSV

Symbol, Date, Open, High, Low, ...
AET, 2009-09-21, 30.49, 31.09, ...
AET, 2009-09-18, 31.01, 31.44, ...
...
MRK, 1988-11-25, 55.00, 55.25, ...

Input to map function
Our Hadoop Job Illustrated: Mapper Output

Input to map function

CSV
Symbol,Date,Open,High,Low,...
AET,2009-09-21,30.49,31.09,...
AET,2009-09-18,31.01,31.44,...
...
MRK,1988-11-25,55.00,55.25,...

Output of map function

key=AET, value=0.61
key=AET, value=0.33
...
key=MRK, value=0.25
Our Hadoop Job Illustrated: Reducer Output

Output of map function

```
key=\text{AET}, \text{value}=0.61
key=\text{AET}, \text{value}=0.33
...  
key=\text{MRK}, \text{value}=0.25
```

Output of reduce function

```
key=\text{AET}, \text{value}=13.75
key=\text{AXP}, \text{value}=15.12
...  
key=\text{MRK}, \text{value}=29.0
```
Now that you understand the data... let’s see the code!
public class StockAnalyzerMapper extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, FloatWritable> {

    @Override
    public void map(LongWritable key, Text value,
                     OutputCollector<Text, FloatWritable> output,
                     Reporter reporter)
        throws IOException {

        String record = value.toString();

        if (record.startsWith("Symbol")) {
            // ignore header row
            return;
        }
    }
String[] fields = record.split("","");
String symbol = fields[0];
String high = fields[3];
String low = fields[4];

float lowValue = Float.parseFloat(low);
float highValue = Float.parseFloat(high);
float delta = highValue - lowValue;

output.collect(new Text(symbol),
          new FloatWritable(delta));
public class StockAnalyzerReducer extends MapReduceBase implements Reducer<Text, FloatWritable, Text, FloatWritable> {

    @Override
    public void reduce(Text key, Iterator<FloatWritable> values,
                        OutputCollector<Text, FloatWritable> output,
                        Reporter reporter)
                        throws IOException {

        float maxValue = Float.MIN_VALUE;
        while (values.hasNext()) {
            maxValue = Math.max(maxValue, values.next().get());
        }

        output.collect(key, new FloatWritable(maxValue));
    }
}
public class StockAnalyzerConfig {

    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: StockAnalyzerConfig <
                              input> <output> ");
            System.exit(1);
        }

        JobConf conf = new JobConf(StockAnalyzerConfig.class);
        conf.setJobName("Stock analyzer");

        Path inputPath = new Path(args[0]);
        FileInputFormat.addInputPath(conf, inputPath);

        Path outputPath = new Path(args[1]);
        FileOutputFormat.setOutputPath(conf, outputPath);
    }
}
conf.setMapperClass(StockAnalyzerMapper.class);
conf.setReducerClass(StockAnalyzerReducer.class);

conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(FloatWritable.class);

JobClient.runJob(conf);
Demonstration

Let’s run the example so we can see Hadoop in action!
Conclusion

- Tonight I’ve explained the basics of Hadoop
- It’s a highly-scalable framework for data processing
- It’s inspired by Google
- And essential to Yahoo, Facebook and many others
- Hopefully you can find a use for it too!