Apache Hadoop
Large scale data processing

Speaker: Isabel Drost
Isabel Drost

Nighttime:
- Came to nutch in 2004.
- Co-Founder Apache Mahout.
- Organizer of Berlin Hadoop Get Together.

Daytime:
- Software developer @ Berlin
Hello Information Retrieval course!
Agenda

- Motivation.
- A short tour of Map Reduce.
- Introduction to Hadoop.
- Hadoop ecosystem.
Massive data as in:

Cannot be stored on single machine.
Takes too long to process in serial.

Idea: Use multiple machines.
Challenges.
Single machines tend to fail:

- Hard disk.
- Power supply.

...
More machines – increased failure probability.
Requirements

- Built-in backup.
- Built-in failover.
Typical developer

- Has never dealt with large (petabytes) amount of data.
- Has no thorough understanding of parallel programming.
- Has no time to make software production ready.
Typical developer

- Has never dealt with large (petabytes) amount of data.
- Has no thorough understanding of parallel programming.
- Has no time to make software production ready.

Failure resistant: What if service X is unavailable?
Failover built in: Hardware failure does happen.
Documented logging: Understand message w/o code.
Monitoring: Which parameters indicate system's health?
Automated deployment: How long to bring up machines?
Backup: Where do backups go to, how to do restore?
Scaling: What if load or amount of data double, triple?
Many, many more.
Requirements

- Built-in backup.
- Built-in failover.
- Easy to use.
- Parallel on rails.
Developers world wide

Open source developers
Developers worldwide

- Developers interested in large scale applications
- Open source developers
Developers world wide

Developers interested in large scale applications

Java developers

Open source developers
Requirements

- Built-in backup.
- Built-in failover.
- Easy to use.
- Parallel on rails.
- Java based.
Go away or I will replace you with a very small shell script.
Requirements

- Built-in backup.
- Built-in failover.
- Easy to administrate.
- Single system.
- Easy to use.
- Parallel on rails.
- Java based.
We need a solution that:

- Is easy to use.
- Scales well beyond one node.
Java based implementation.

Easy distributed programming.

Well known in industry and research.

Scales well beyond 1000 nodes.
• 2008:
  - 70 hours runtime
  - 300 TB shuffling
  - 200 TB output

• In 2009
  - 73 hours
  - 490 TB shuffling
  - 280 TB output
  - 55%+ hardware
  - 2k CPUs (40% faster cpus)

• 2008
  - 2000 nodes
  - 6 PB raw disk
  - 16 TB RAM
  - 16k CPUs

• In 2009
  - 4000 nodes
  - 16 PB disk
  - 64 TB RAM
  - 32k CPUs (40% faster cpus)
Example use cases

- Distributed Grep.
- Distributed Sort.
- Link-graph traversal.
- Term-Vector per host.
- Web access log stats.
- Inverted index.
- Doc clustering.
- Machine translation.
Some history.
Feb '03 first Map Reduce library @ Google

Oct '03 GFS Paper

Dec '04 Map Reduce paper

Dec '05 Doug reports that nutch uses map reduce

Feb '06 Hadoop moves out of nutch

Apr '07 Y! running Hadoop on 1000 node cluster

Jan '08 Hadoop made an Apache Top Level Project
Hadoop assumptions
Assumptions:

Data to process does not fit on one node.
Each node is commodity hardware.
Failure happens.

Ideas:

Distribute filesystem.
Built in replication.
Automatic failover in case of failure.
Assumptions:
Moving data is expensive.
Moving computation is cheap.
Distributed computation is easy.

Ideas:
Move computation to data.
Write software that is easy to distribute.
Assumptions:

Systems run on spinning hard disks. Disk seek >> disk scan.

Ideas:

Improve support for large files. File system API makes scanning easy.
Hadoop by example
isabel@h1349259:~$ more data/feeds.opml | grep -o "http://[0-9A-Za-z\-_\.]*" | sort | uniq --count | sort | tail
  3 http://agbs.kyb.tuebingen.mpg.de
  3 http://irgupf.com
  3 http://jeffsutherland.com
  4 http://ml.typepad.com
  4 http://weblogs.java.net
  4 http://www.gridvm.org
  4 http://yaroslavvb.blogspot.com
  5 http://feeds.feedburner.com
  6 http://blogsearch.google.com
10 http://arxiv.org
pattern="http://[0-9A-Za-z\-_\.]*"
grep -o "$pattern" feeds.opml | sort | uniq --count
pattern="http://[0-9A-Za-z\-\_\.]*"

grep -o "$pattern" feeds.opml

M A P | sort | SHUFFLE | uniq --count
Local to data.
Local to data.
Outputs a lot less data.
Output can cheaply move.
Local to data.
Outputs a lot less data.
Output can cheaply move.
Local to data.
Outputs a lot less data.
Output can cheaply move.

Shuffle sorts input by key.
Reduces output significantly.
private IntWritable one = new IntWritable(1);
private Text hostname = new Text();

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
        hostname.set(getHostname(tokenizer.nextToken()));
        output.collect(hostname, one);
    }
}

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));
}
Input

Map

Map

Map

Map

Map

Input

k1:v1, k2:v1, k1:v2

k2:v1, k1:v2

k2:v1, k1:v3

Intermediate Output

Shuffle

Groups by key

Intermediate Output

k1:v1, k1:v2, k1:v3

k2:v1, k2:v1, k2:v1

Reduce

Reduce

Output
# Petabyte sorting benchmark

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Nodes</th>
<th>Maps</th>
<th>Reduces</th>
<th>Replication</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>500,000,000,000,000</td>
<td>1406</td>
<td>8000</td>
<td>2600</td>
<td>1</td>
<td>59 seconds</td>
</tr>
<tr>
<td>1,000,000,000,000,000</td>
<td>1460</td>
<td>8000</td>
<td>2700</td>
<td>1</td>
<td>62 seconds</td>
</tr>
<tr>
<td>100,000,000,000,000,000</td>
<td>3452</td>
<td>190,000</td>
<td>10,000</td>
<td>2</td>
<td>173 minutes</td>
</tr>
<tr>
<td>1,000,000,000,000,000,000</td>
<td>3658</td>
<td>80,000</td>
<td>20,000</td>
<td>2</td>
<td>975 minutes</td>
</tr>
</tbody>
</table>

Per node: 2 quad core Xeons @ 2.5ghz, 4 SATA disks, 8G RAM (upgraded to 16GB before petabyte sort), 1 gigabit ethernet.

Per Rack: 40 nodes, 8 gigabit ethernet uplinks.
500 GB Task Timeline

Running Tasks

Seconds

reduce
merge
shuffle
maps
Waste = Failed or killed, speculative execution.
What was left out

- Combiners compact map output.
- Language choice: Java vs. Dumbo vs. PIG …
- Size of input files does matter.
- Facilities for chaining jobs.
- Logging facilities.
- Monitoring.
- Job tuning (number of mappers and reducers)
- …
Hadoop ecosystem.
Higher level languages.
Suppose you have user data in one file, website data in another, and you need to find the top 5 most visited pages by users aged 18 - 25.
Example from PIG presentation at Apache Con EU 2009
Users = load 'users' as (name, age);
Fltrd = filter Users by
    age >= 18 and age <= 25;
Pages = load 'pages' as (user, url);
Jnd = join Fltrd by name, Pages by user;
Grpd = group Jnd by url;
Smmd = foreach Grpd generate group,
    COUNT(Jnd) as clicks;
Srtld = order Smmd by clicks desc;
Top5 = limit Srtld 5;
store Top5 into 'top5sites';
Example from JAQL documentation.

```json
[
  {publisher: 'Scholastic',
   author: 'J. K. Rowling',
   title: 'Deathly Hallows',
   year: 2007},

  {publisher: 'Scholastic',
   author: 'J. K. Rowling',
   title: 'Chamber of Secrets',
   year: 1999,
   reviews: [
     {rating: 10, user: 'joe', review: 'The best ...'},
     {rating: 6, user: 'mary', review: 'Average ...'}]},

  {publisher: 'Scholastic',
   author: 'J. K. Rowling',
   title: 'Sorcerers Stone',
   year: 1998},

  {publisher: 'Scholastic',
   author: 'R. L. Stine',
   title: 'Monster Blood IV',
   year: 1997,
   reviews: [
     {rating: 8, user: 'rob', review: 'High on my list...'},
     {rating: 2, user: 'mike', review: 'Not worth the paper ...'},
     discussion: [
       {user: 'ben', text: 'This is too harsh...'},
       {user: 'jill', text: 'I agree ...'}]],

  {publisher: 'Grosset',
   author: 'Carolyn Keene',
   title: 'The Secret of Kane',
   year: 1930}
]
// Query 2. Find the authors and titles of books that have received a review.
for( $b in hdfsRead('books') )
  if( exists($b.reviews) )
    [{ $b.author, $b.title }];

// result...
[
  {author: 'J. K. Rowling', title: 'Chamber of Secrets'},
  {author: 'R. L. Stine', title: 'Monster Blood IV'}
];
(Distributed) storage.
About Dynomite

Dynomite is an eventually consistent distributed database that's based on Amazon's Dynamo paper. Dynomite currently provides a drop-in replacement for Dynamo along with some additional features not covered by the paper.
Libraries built on top.
Alternative approaches.
Get involved!
Do you love:
Do you love:

Solving hard problems?
Do you love:

Solving hard problems?
Communicating your solution?
Do you love:

Solving hard problems?
Communicating your solution?
Working with excellent teams?
Do you love:

Solving hard problems?
Communicating your solution?
Working with excellent teams?
Skills to learn:

- Technical
- Soft Skills
Source control system.
Continuous integration.
Test-first development.
Issue-tracker.
Create readable patches.

Communicate and discuss solutions.

Review others code.

Work in large, distributed teams.
How?

• First time users:
  – Documentation in wiki.

• Found a bug:
  – Go to JIRA, file a bug.
  – Describe the bug.
  – Create a test to show.
  – Provide a patch.

• Experimenting:
  – Write examples.

• Evaluating:
  – Test performance.
  – Provide comparison.

• Participate on-list.
  – Answer questions.
  – Discuss your use-case.
Recipe to Apache

- Download the release and use it.
- Subscribe to the mailing-list.
- Questions:
  - Documentation: Wiki.
  - Discussions: Mailing list.
  - Current status: JIRA.
  - History: JIRA for patches, mailing-list for votes.
- Checkout the code and built it.

Bug reports, patches, features. Documentation, code, examples.

Contact Ross Gardler for more information on Apache at universities worldwide.
No, you are not the only one... Many a sleepless night spent on it... :-)

I usually try to refer to it as Lucene Java, but old habits die hard and often times I just call it Lucene. I think the name has a good brand at this point and is very strongly associated w/ the Java library. I seem to recall when they were forming the TLP, that the original proposal was search.a.o, but then changed b/c the ASF didn't like generic names (or at least that is how I recall it.) And, of course, with Hadoop and the potential for Tika/Lius, it isn't just search anymore. I have often thought about an Apache "Text" project, that could eventually hold a whole family of text based tools like Lucene, Tika, Hadoop, Solr, etc. plus things like part of speech taggers, clustering/classification algorithms, UIMA, etc. all under one roof. But that is just my two cents and I don't know if it fits with what other people have in mind. There are a lot of OSS tools out there for these things, but none bring together a whole suite under a brand like Apache.

-Grant
Why go for Apache?
Jumpstart your project with proven code.
Discuss ideas and problems online.

November 16, 2005 [phil h]
http://www.flickr.com/photos/hi-phi/64055296
Become part of the community.