From Data to Information
Apache Mahout

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 FrOSCon visitors!
Agenda

- Motivation.
- HowTo: A path from data to information.
- Introduction to Mahout.
News aggregation

September 10, 2008 by Alex Barth
http://www.flickr.com/photos/a-barth/2846621384

Today: Read news papers, Blogs, Twitter, RSS feed.  
Wish: Aggregate sources and track emerging topics.
Go to cinema

Today: IMDB, zitty, movie review pages, twitter, blogs, ask friends.

Wish: Reviews, sentiment detection, recommendations.
HowTo: From data to information.
From data to information.

- Start collecting and storing data.
- Analyse and understand data.
- Answer more complex questions.
Data storage options

- Structured, relational.
  - Customer data.
  - Bug database.
Data storage options

- Structured, relational:
  - Customer data.
  - Bug database.

- Continuous files:
  - Log data.
  - Document Stream.
Massive data as in:

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

Idea: Use multiple machines.
Challenges when scaling out.
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.
Requirements

- Built-in backup.
- Built-in failover.
- Easy to use.
- Parallel on rails.
Requirements

- Built-in backup.
- Built-in failover.
- Easy to use.
- Parallel on rails.
- Active development.
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.
• Active development.
Easy distributed programming.
Well known in industry and research.
Scales well beyond 1000 nodes.
## Petabyte sorting benchmark

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>500,000,000,000,000</td>
<td>1406</td>
</tr>
<tr>
<td>1,000,000,000,000,000</td>
<td>1460</td>
</tr>
<tr>
<td>100,000,000,000,000,000</td>
<td>3452</td>
</tr>
<tr>
<td>1,000,000,000,000,000,000</td>
<td>3658</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Replication</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>59 seconds</td>
</tr>
<tr>
<td>1</td>
<td>62 seconds</td>
</tr>
<tr>
<td>2</td>
<td>173 minutes</td>
</tr>
<tr>
<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.
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.
Data storage options

- Structured, relational:
  - Customer data.
  - Bug database.

- Semi-structured data:
  - Documents.
  - Independent rows.

- Continuous files:
  - Log data.
  - Document Stream.
Store in RDBMS?

- Possible.
- Becomes expensive pretty quickly.
Store in Hadoop DFS?

- Optimised for LARGE files.
- Throughput vs. latency.
Something in between?

- Transactions – can we do without?
- Joins – some applications don't need them.
About Dynomite

Dynomite is an eventually consistent d
Amazon’s Dynamo paper. Dynomite cu
plus some stuff not covered by the pa
From data to information.

- Start collecting and storing your data.
  - Analyse and understand your data.
  - Answer more complex questions.
Understanding your data

- Data profiling.

- Goals:
  - Identify usual behaviour.
  - Find exceptional cases.

- Exact questions depend on domain.
Example questions

• Structured data:
  – Shopping: Amount of money usually spent.
  – Average age of your customers.
  – Min/Max number of shopping sessions.

• Textual documents:
  – Average length of documents.
  – Distribution of document topics.
  – Distribution of authors.
Visualizations help
Understanding your data

• Structured data in RDBMS:
  – Functionality built-in (min, max etc.)

• Unstructured or Semistructured data in HDFS:
  – Write analysis code in Java. (Map/Reduce jobs).
  – Use higher level language.
Map/Reduce 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

REDUCE
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));
}
Higher level languages.
Cascading
Filtering/Aggregating in Hadoop

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.
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;
Srted = order Smmd by clicks desc;
Top5 = limit Srted 5;
store Top5 into 'top5sites';
From data to information.

- Start collecting and storing your data.
- Analyse and understand your data.
  - Answer more complex questions.
More complex questions

- Which products are commonly bought together.
- What groups of search results were returned.
- Predict probability of user clicking an ad.
- Identify emerging topics in news stories.
- Find source code commonly changed together.
- Identify malicious access patterns to servers.
Machine learning – what's that?
Archimedes taking a Warm Bath
Archimedes model of nature

\[ \frac{\text{Density of Object}}{\text{Density of Fluid}} = . \]

\[ \frac{\text{Weight}}{\text{Weight} - \text{Apparent immersed weight}} \]
An SVM's model of nature

- Margin
- Separating hyperplane
- Class +1: $w^*x + b > 1$
- Class -1: $w^*x + b < -1$
- $\xi$
Scaling machine learning.
Contributions need not be Java based:

PIG, JAQL, Cascading, ...?
• Industry ready.
  – Friendly license.
  – Scalable.

• Easy to use.
  – Well documented.
  – Well maintained by healthy and active community.

• Easy to extend and contribute to.
  – Automated tests.
  – Easy to build and deploy.
What does Mahout have to offer.
Discover groups of items

- Group items by similarity.

- Examples:
  - Group news articles by topic.
  - Find developers with similar interests.
  - Discovery of groups of related search results.
Discover groups of similar items

- Canopy.

- k-Means.

- Fuzzy k-Means.

- Dirichlet based.

- Others upcoming.
Identify dominant topics

• Given a dataset of texts, identify main topics.

  Algorithms: Parallel LDA

• Examples:
  - Dominant topics in set of mails.
  - Identify news message categories.
Assign items to defined categories.

- Given pre-defined categories, assign items to it.

Examples:
- Spam mail classification.
- Discovery of images depicting humans.
Assign items to defined categories.

- Naïve Bayes.
- Complementary naïve bayes.
- Winnow/Perceptron.
- Others upcoming.
Recommendation mining.

- Recommend items to users.

- Examples:
  - Find movies I might want to watch.
  - Find books related to the book I am buying.
  - Find people I might want to meet.
  - Recommend locations to items.
Recommendation mining.

- Integrated Taste.
- Mature Java library.
- Java-based, web service / HTTP bindings.
- Batch mode based on EC2 and Hadoop.
Frequent pattern mining

- Given groups of items, find commonly co-occurring items.

- Examples:
  - In shopping carts find items bought together.
  - In query logs find queries issued in one session.
Release: 0.1

Big Thanks to those who made this possible!

Mahout is running on Amazon EMR.

October 22, 2008 by e_calamar
http://www.flickr.com/photos/e_calamar/2964991182/
Why go for Apache Mahout?
Jumpstart your project with proven code.
Discuss ideas and problems online.
Become part of the community.

Bug reports, patches, features. Documentation, code, examples.
Sept., 29th 2009: Hadoop* Get Together in Berlin

- Thilo Götz: “JAQL”
- Thorsten Schütt: “Solving puzzles with Map/Reduce”
- Uwe Schindler: “Lucene 2.9 with focus on range search.”
- nugg.ad GmbH: “Using Hadoop for ad recommendation.”

newthinking store
Tucholskystr. 48


* UIMA, Hbase, Lucene, Solr, katta, Mahout, CouchDB, pig, Hive, Cassandra, Cascading, JAQL, ... talks welcome as well.
Interest in solving hard problems.
Being part of lively community.
Engineering best practices.

Bug reports, patches, features.
Documentation, code, examples.
<table>
<thead>
<tr>
<th>From</th>
<th>Grant Ingersoll <a href="mailto:gsing...@apache.org">gsing...@apache.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>Re: Lucene Branding: the TLP, and &quot;Lucene Java&quot;</td>
</tr>
<tr>
<td>Date</td>
<td>Wed, 11 Apr 2007 01:13:36 GMT</td>
</tr>
</tbody>
</table>

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
Going parallel: k-Means
Until stable.
Data intensive.
Output: Cluster assignment.
Pre-Compute centers.

Done in Map.
Data intensive.
Output: Cluster assignment.
Pre-Compute centers.

Done in Map.  

Done in Reduce.