Apache Hadoop
Large scale data processing

Speaker: Isabel Drost
Isabel Drost

Nighttime:
Co-Founder Apache Mahout. Organizer of Berlin Hadoop Get Together.

Daytime:
Software developer
Hello FOSDEM visitors!
Hello FOSDEM visitors!

How many know Hadoop?
Hello FOSDEM visitors!

How many Hadoop users?
How many nodes?

Hello FOSDEM visitors!
Hello FOSDEM visitors!

Zookeeper?
Hello FOSDEM visitors!

Hive?
Hello FOSDEM visitors!
Hello FOSDEM visitors!

Pig?
Hello FOSDEM visitors!

Lucene?
Solr?

Hello FOSDEM visitors!
Hello FOSDEM visitors!

Mahout?
Agenda

- Collecting and storing data.
- Analysing data.
- Tour of Hadoop.
- Hadoop ecosystem.
Collecting and storing data.
Data storage options

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

September 10, 2007 by sanden
http://www.flickr.com/photos/daphid/1354523220/
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.
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
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.
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:

Distributed computation is easy.
Moving computation is cheap.
Moving data is expensive.

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.
HDFS building blocks
(Graphics: Thanks to Thilo.)
NameNode
- Stores file meta data.
- In memory.
- Block-node mapping.

DataNode
- Stores file contents.
- On disk.
- Block-Id to disk.
Anatomy of a file write

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Anatomy of a file write

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Anatomy of a file write

HDFS client

Client node

Create file

Close file

Name Node

Write packet

Ack packet

Data Node

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Anatomy of a file write

HDFS client

Client node

Name Node

Create file

Close file

Write packet

Ack packet
HDFS Replication Strategy

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Anatomy of a file read

HDFS client

Client node
Anatomy of a file read

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Anatomy of a file read

HDFS client

Client node

Open file

Close file

Name Node

Read blocks

Data Node

Read blocks

Data Node

Read blocks

Data Node

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Analyse and understand your data.
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

MAP

| SHUFFLE
| sort
| uniq --count

REDUCE
Input

Map
Map
Map
Map
Map

Intermediate Output

k1:v1, k2:v1, k1:v2
k2:v1, k1:v2

Shuffle
Groups by key

Intermediate Output

k2:v1, k1:v3
Input

Map
Map
Map
Map
Map
Map

Intermediate Output

k1:v1, k2:v1, k1:v2
k2:v1, k1:v2
k2:v1, k1:v3

Shuffle
Groups by key

Intermediate Output

k1:v1, k1:v2, k1:v3
k2:v1, k2:v1, k2:v1

Reduce
Reduce

Output

k1:v1, k1:v2, k1:v3
k2:v1, k2:v1, k2:v1
private IntWritable one = new IntWritable(1);
private Text hostname = new Text();

public void map(K key, V value, Context context) {
    String line = value.toString();
    StringTokenizer tokenizer = new StringTokenizer(line);
    while (tokenizer.hasMoreTokens()) {
        hostname.set(getHostname(tokenizer.nextToken()));
        context.write(hostname, one);
    }
}

public void reduce(K2 key, Iterable<V2> values, OutputCollector<K3, V2> output) {
    int sum = 0;
    while (values.hasNext()) {
        sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
}
Anatomy of a map/reduce job
Anatomy of a map/reduce job

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Anatomy of a map/reduce job

Client app

Client node

Submit Job

Job Tracker

TaskTracker

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Anatomy of a map/reduce job
Anatomy of a map/reduce job

Slide inspired by: “Hadoop – The definitive guide”, Tom White, O'Reilly
Requirements to get started
Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Amazon EC2 reduces the time required to obtain
Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon EC2’s simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Amazon EC2 reduces the time required to obtain

Amazon Elastic MapReduce

Amazon Elastic MapReduce is a web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data. It utilizes a hosted Hadoop framework running on the web-scale infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3).

Using Amazon Elastic MapReduce, you can instantly provision as much or as little capacity as you like to perform data-intensive tasks for applications such as web indexing, data mining, log file analysis, machine learning, financial
(Thanks to Thilo for helping set up the cluster, Thanks to packet and masq for two of the three machines.)
Requirements to get started
Up next.
Up next.

- In 0.21:
  - append/sync in HDFS
  - more advanced task schedulers

- In 0.22:
  - security
  - avro-based rpc for cross-version rpc compatibility
  - symbolic links
  - federated NameNodes
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.
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;
Smmdd = foreach Grpd generate group,
    COUNT(Jnd) as clicks;
Srttd = order Smmdd by clicks desc;
Top5 = limit Srttd 5;
store Top5 into 'top5sites';
(Distributed) storage.
Hypertable

HBase

Cassandra

A highly scalable, eventually consistent, distributed, structured key-value store.
Libraries built on top.
Jumpstart your project with proven code.
Discuss ideas and problems online.

November 16, 2005 [phil h]
http://www.flickr.com/photos/hi-phi/64055296
Interest in solving hard problems.
Being part of lively community.
Engineering best practices.

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

Image by: Patrick McEvoy
Mar., 10th 2010: Hadoop* Get Together in Berlin

- Bob Schulze (eCircle/ Munich): Database and Table Design Tips with HBase
- Dragan Milosevic (zanox/ Berlin): Product Search and Reporting powered by Hadoop
- Chris Male (JTeam/ Amsterdam): Spatial Search

http://upcoming.yahoo.com/event/5280014/

* UIMA, Hbase, Lucene, Solr, katta, Mahout, CouchDB, pig, Hive, Cassandra, Cascading, JAQL, ... talks welcome as well.
June 7/8th: Berlin Buzzwords 2010

Store, Search, Scale

Solr
HBase
Lucene
Sphinx
Business Intelligence
Cloud Computing
MongoDB
Hadoop
Distributed computing
NoSQL
Scalability

This is to announce the Berlin Buzzwords 2010 scalability conference. Berlin Buzzwords 2010 is scheduled for the start of June. Topics of interest include NoSQL databases, Hadoop, Lucene and others. Our goal is to bring developers and users together in central Europe for a conference featuring talks on scaling data analysis. The team organizing this event is deeply rooted in the Hadoop, Lucene, and CouchDB communities. Interested in helping? See the requests for helping hands. Also note that we are just getting off the ground. Please be patient as we get the various infrastructure pieces in place.
Interest in solving hard problems.
Being part of lively community.
Engineering best practices.

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

Image by: Patrick McEvoy
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.