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 DevHouse!
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.
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.
Example use cases

Distributed Grep.  
Distributed Sort.  
Link-graph traversal.  
Term-Vector per host.  
Web access log stats.  

Inverted index.  
Doc clustering.  
Machine learning.  
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
<?xml version="1.0" encoding="UTF-8"?>
<opml version="1.0">
<head>
<text/></text>
</head>
<body>
<outline htmlUrl="http://eventseer.net" title="EventSeer - A Digital Library of Call for Papers" useCustomFetchInterval="rss" type="rss" xmlUrl="http://eventseer.net/feeds/main/rss.xml" id="312053546" text="eventseer.net" />
<outline htmlUrl="http://www.theserverside.com" title="TheServerSide: Patterns" useCustomFetchInterval="rss" type="rss" xmlUrl="http://www.theserverside.com/rss/theserverside-120patterns-rss2.xml" text="Getting up-to-date news, discussions, patterns, resources, and media" />
<outline htmlUrl="http://chadwa.wordpress.com" title="Chad's Search Blog" useCustomFetchInterval="rss" type="rss" xmlUrl="http://chadwa.wordpress.com/feed/" id="545368194" text="Chad's Search Blog" descr="" />
<outline htmlUrl="http://emotion.inrialpes.fr/~dangauthier/blog" title="Yet Another Machine Learning blog" eMode="globalDefault" version="rss" type="rss" xmlUrl="http://emotion.inrialpes.fr/~dangauthier/blog_FEEDS.xml" />
<outline htmlUrl="http://yaroslavvb.blogspot.com/" title="Machine Learning, etc" useCustomFetchInterval="rss" type="rss" xmlUrl="http://yaroslavvb.blogspot.com/feeds/posts/default" id="805998569" text="" />
<outline htmlUrl="http://ptufts.blogspot.com/" title="Pinhead's Progress" useCustomFetchInterval="rss" type="rss" xmlUrl="http://ptufts.blogspot.com/feeds/posts/default" id="1019393988" text="Pinhead's P:"
<outline htmlUrl="http://absolutely-regular.blogspot.com/" title="Absolutely Regular" useCustomFetchInterval="rss" type="rss" xmlUrl="http://absolutely-regular.blogspot.com/feeds/posts/default" id="17850!" eMode="globalDefault" version="rss" type="rss" xmlUrl="http://atomai.blogspot.com/feeds/posts/default" text="Data Mining, Analytics and Artificial Intelligence -- the challenges and opportunities in data mining, artificial intelligence, analytics, intelligent agents, semiconductors, distributing business objects, Oracle, Intel, AMD, or Pentaho. Heuristic, Six Sigma, or CMM. Contractor or in-house. Hail.com" />
</body>
</opml>
isabel@h1349259:~$ more datafeeds.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
grep -o "$pattern" feeds | sort | REDUCE | uniq --count

MAP

SHUFFLE

sort

uniq --count

pattern="http://[0-9A-Za-z\-_\.]*"
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));
}
Hadoop ecosystem.
Higher level languages.
Cascading
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
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;
Srttd = order Smmd by clicks desc;
Top5 = limit Srttd 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'}
];

Example from JAQL documentation.
(Distributed) storage.
Libraries built on top.
Get involved!

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.