Hadoop & Pig

Dr. Karina Hauser
Senior Lecturer
Management & Entrepreneurship

Rocky Mountain Advanced Computing Consortium
Outline

• Introduction
• (Setup)
• Hadoop, HDFS and MapReduce
• Pig
Introduction

• What is Hadoop and where did it come from?
Big Data

Consultants say three quintillion bytes of data are created every day.

It comes from everywhere. It knows all.

According to the book of Wikipedia, its name is "Big Data."

Big data lives in the cloud. It knows what we do.

In the past, our company did many evil things.

But if we accept big data in our servers, we will be saved from bankruptcy.

Let us pay.

Is it too late to side with evil?

Shhhh! It hears you.
Big Data Sources

• Every day 2.5 quintillion bytes or 2.5 exabytes \((10^{18})\) are generated, that number is estimated to double every 40 month

• Astronomy
  – Sloan Digital Sky Survey (SDSS) began collecting astronomical data in 2000; 200 GB \((10^9)\) per night
  – Large Synoptic Survey Telescope (LSST) (~2020); estimated ~20 TB \((10^{12})\) per night

• Business
  – Twitter: 12 terabytes \((10^{12})\) of Tweets every day
  – Walmart: 2.5 petabytes \((10^{15})\) of data every hour from its customer transactions
Big Data - Big Business

- IDC predicts big data technology and services will grow worldwide from $3.2 billion in 2010 to $16.9 billion in 2015. This represents a compound annual growth rate of 40 percent — about seven times that of the overall information and communications technology market.
Big Data in the News

• The Economist Intelligence Unit study showed that nine out of 10 surveyed business leaders believe data is now the fourth factor of production, as fundamental to business as land, labor and capital.
What is Big Data?

• “When the data itself becomes part of the problem”

• Three (to five) dimensions:
  – Volume
  – Variety
  – Velocity
  – (Veracity)
  – (Value)
The Solution
Short History

- Created by Doug Cutting, named after his son’s toy elephant
- 2002 - Nutch, search engine, scalability problems
- 2004 - Google papers on GFS and MapReduce
- 2006 - Yahoo hires Doug to improve Hadoop
- 2008 - Hadoop becomes Apache Top Level Project
Hadoop Today

- Moving from “Internet” companies
  - Yahoo
  - Google
- to business and science applications
  - Customer relationship management
  - Bioinformatics
  - Astrophysics
Hadoop Definition

• “Framework that allows for the
  – distributed processing of
  – large data sets
  – across clusters of computers
  – using a simple programming model”

• Open-source software, maintained by
  “The Apache Software Foundation”

• http://hadoop.apache.org/
Apache Hadoop Projects

Apache Hadoop Subprojects
- Hadoop Common
- HDFS
- YARN (Vers. 2)
- MapReduce

Apache Hadoop Related Projects
- Ambri
- Avro
- Cassandra
- Chukwa
- HBase
- Hive
- Mahout
- Pig
- ZooKeeper

Apache Hadoop Projects
Hadoop Cluster

- **Commodity hardware**
- **Individual disk space on each node**
- **Hadoop framework handles:**
  - Data "backups" (through replication)
  - Hardware failure
  - Parallelization of code (through MapReduce paradigm)
HDFS

- Write-once, read-many
- Each file is stored as sequence of same-sized blocks (default size 64MB)
- Blocks are replicated across different nodes
- Highly reliable:
  - Redundant data storage
  - Heartbeat messages to detect connectivity problems
    → Automatic failover
WRITING DATA IN HDFS CLUSTER

REQUEST FROM USER
Let's start with writing some data...
Mr. Client, please write 200 MB data for me
It'll be my pleasure. But--

DIVIDE FILE INTO BLOCKS

BLOCK AND REPLICAION
--are you not forgetting something?
Ah yes, please: a) divide the data in 128MB blocks
b) copy each block in three places

A good client always knows these two things:

BLOCKSIZE: large file is divided in blocks (usually 64 or 128MB)
REPLICATION FACTOR: each block is stored in multiple locations (usually 3)

ASK NAMENODE
Let's work on the first block first
Mr. Namenode: please help me write a 128MB block with replication of 3

NAMEODE ASSIGN DATANODES
Replication 3.. Hmm.. need to find 3 datanodes for this client
How do I do that? Will tell you some other time

DIVIDE FILE INTO BLOCKS
First-- I divide the big file into blocks

CLIENT STARTS WRITING DATA
I send my data (and the list) to first datanode only
I store the data in hard drive, and--

WHILE I am recieving data, I forward the same data to the next datanode

Here you go buddy. Addresses of three datanodes. I have also sorted them in increasing distance from you
thanks!

Datanode 1, Datanode 2, Datanode 3

Pipelining

MapReduce

- **Programming model designed for**
  - batch processing of large volumes of data
  - in parallel
  - by dividing the work into a set of independent tasks

- **Not limited to Hadoop**
MapReduce WordCount Example

The overall MapReduce word count process

Input
Deer Bear River
Car Car River
Deer Car Bear

Splitting
Deer Bear River
Car Car River
Deer Car Bear

Mapping
Deer, 1
Bear, 1
River, 1

Shuffling
Bear, 1
Bear, 1

Reducing
Bear, 2

Final result
Bear, 2
Car, 3
Deer, 2
River, 2
Problems suited for MapReduce

- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output
Hadoop 1.x Components
Hadoop 1.x Components

**Name Node**
- Stores metadata (filenames, replications factors …)
  - *fsimage*: latest checkpoint of namespace
  - *edits*: log of changes to namespace
- Checks data node availability (Heartbeat)
- If possible: Separate machine

**Data Node**
- Stores data
- Replicates blocks
- Computation
Hadoop 1.x Components

• **Secondary/Checkpoint Name Node**
  - Periodically creates checkpoints of namespace
    - *Downloads fsimage and edits, creates new namespace and updates Name Node*
  - Separate machine

• **Job Tracker**
  - Schedules and manages jobs

• **Task Tracker**
  - Executes MapReduce jobs on individual data node
Hadoop 2.0

HADOOP 1.0
- MapReduce (cluster resource management & data processing)
- HDFS (redundant, reliable storage)

HADOOP 2.0
- MapReduce (data processing)
- Others (data processing)
- YARN (cluster resource management)
- HDFS (redundant, reliable storage)

Applications Run Natively IN Hadoop
- BATCH (MapReduce)
- INTERACTIVE (Tez)
- ONLINE (Hoya)
- STREAMING (Storm)
- GRAPH (Giraph)
- IN-MEMORY (Spark)
- HPC MPI (OpenMPI)
- OTHER (Search) (Weave...)

YARN (Cluster Resource Management)
HDFS2 (Redundant, Reliable Storage)
Setup

- **Three options:**
  - Standalone (single Java process)
  - Pseudo-Distributed (separate Java processes)
  - Fully-Distributed

- **Prerequisites (on virtual machine):**
  - Ubuntu server 12.04
  - Ubuntu desktop (for monitoring)
  - Oracle (Sun) Java 1.7.0_25
  - SSH
Setup Files

• **hosts (ip address)**
• **.bashrc (Java dir, home dir)**
• **hadoop configuration files in /usr/local/hadoop/conf**
  – hadoop-env.sh (Java dir)
  – hdfs-site.xml (replication factor)
  – mapred-site.xml (host/port for jobtracker)
  – master/slave (ips for multi-node cluster)
Login

- **User:** rmacc
- **Password:** rmacc
- **Start terminal** (Ctrl+Alt+F1)
- **Login as hduser**
  - User: hduser
  - Password: hduser
- **Change directory to hadoop**
  - with `cd $HADOOP_PREFIX`
  - or `cd /usr/local/hadoop`
Starting Hadoop Daemons

• **All:**
  
  $ bin/start-all.sh

• Log output in *logs* directory
Check Daemons

- **jps**
  - 1367 Jps
  - 8695 DataNode
  - 8609 NameNode
  - 6318 SecondaryNameNode
  - 2600 JobTracker
  - 2830 TaskTracker
Web Interfaces

- [http://localhost:50030](http://localhost:50030) → Cluster status and jobs
- [http://localhost:50070](http://localhost:50070) → HDFS
HDFS

- Hadoop Distributed File System
Ubuntu (Linux) | Hadoop
---|---
**local file system** | **(HDFS)**
```
/usr/local/hadoop/...
```
```
/user/hduser/...
```
**Files** | **KDD**
**Scripts** | **Movies**

**Create Directory**
```
mkdir
```
```
bhadoop dfs -mkdir
```

**Show content of directory**
```
ls
```
```
bhadoop dfs -ls
```
```
sh ls
```
```
fs -ls
```
```
fs -mkdir
```
```
bhadoop dfs -mkdir
```

---

Leeds School of Business
UNIVERSITY OF COLORADO BOULDER
HDFS Shell

- `bin/hadoop dfsadmin -help` → all admin commands
- `bin/hadoop dfs -help` → all commands
- Most commands similar to unix
  - `dfs -copyFromLocal`
  - `dfs -ls`
Importing Data

- **KDD Example:** Legcare sales data from KDD Cup 2000
  "We wish to thank Blue Martini Software for contributing the KDD Cup 2000 data"
- Cleaned for easier/faster use
- Copy file KDDCupCleaned.txt to hdfs
Copying File to HDFS

- **Create new directory:**
  
  $ bin/hadoop dfs -mkdir rmacc

- **Copy files:**
  
  $ bin/hadoop dfs -copyFromLocal ../rmacc/Beowulf.txt rmacc/
  $ bin/hadoop dfs -copyFromLocal ../rmacc/KDD* rmacc/

- **Check:**
  
  $ bin/hadoop dfs -lsr rmacc

- **Localhost:**
  
  50070 →Browse the filesystem
MapReduce WordCount Example

The overall MapReduce word count process

Input
- Deer
- Bear
- River
- Car
- Bear

Splitting
- Deer Bear River
- Car Car River
- Deer Car Bear

Mapping
- Deer, 1
- Bear, 1
- River, 1
- Car, 1
- Car, 1
- Car, 1
- Deer, 1
- Deer, 1
- Bear, 1
- River, 1
- River, 1

Shuffling
- Bear, 1
- Bear, 1
- Car, 1
- Car, 1
- Car, 1
- Deer, 1
- Deer, 1
- River, 1
- River, 1

Reducing
- Bear, 2
- Bear, 2
- Car, 3
- Deer, 2
- River, 2

Final result
- Bear, 2
- Car, 3
- Deer, 2
- River, 2

From: http://www.rabidgremlin.com/data20/#(3)
Java Code for WordCount Example

```java
package org.myorg;

import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class WordCount {

    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text word = 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()) {
                word.set(tokenizer.nextToken());
                output.collect(word, one);
            }
        }
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
```
Java Code for WordCount Example

```java
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));
}

public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");

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

    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    conf.set_inputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
}
```
There Must be an Easier Way
Pig (Latin)
Pig

- High-level data processing language (Pig Latin)
- Resides on user machine, not cluster
- Pig Latin compiled into efficient MapReduce jobs
Test Pig Installation

- (Hadoop has to be running)
  - $ cd $PIG_HOME
  - $ bin/pig -help
How to Run Pig

• **Grunt interactive shell**
  • Two modes:
    • *Local, standalone* (pig –x local)
    • *Hadoop, distributed* (pig –x mapreduce)

• **Scripts** (.pig)
• **Embedded in Java or Python**
• **PigPen, Eclipse plugin**
Pig Statements in Grunt

- **LOAD** → **Transform data** → **DUMP or STORE**
- **Example:**
  - `grunt> A = load 'student' using PigStorage()
    AS (name:chararray, age:int, gpa:float);
  - `grunt> B = foreach A generate name;
  - `grunt> dump B

  — “A “ is called a “relation” or “outer bag”
## Pig Example: KDD Data

<table>
<thead>
<tr>
<th>Key</th>
<th>Date</th>
<th>Time</th>
<th>Unit</th>
<th>Order LineID</th>
<th>Qty</th>
<th>Order Status</th>
<th>Tax</th>
<th>Amount</th>
<th>Weekday</th>
<th>Hour</th>
<th>City</th>
<th>State</th>
<th>Customer ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2/27/2000</td>
<td>08:06:35</td>
<td>15</td>
<td>1</td>
<td>15</td>
<td>Shipped</td>
<td>1.3</td>
<td>16.3</td>
<td>Sunday</td>
<td>8</td>
<td>Westport</td>
<td>CT</td>
<td>62</td>
</tr>
<tr>
<td>2</td>
<td>3/30/2000</td>
<td>10:00:18</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>Shipped</td>
<td>0</td>
<td>9</td>
<td>Thursday</td>
<td>10</td>
<td>Westport</td>
<td>CT</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>1/28/2000</td>
<td>14:43:34</td>
<td>12</td>
<td>1</td>
<td>12</td>
<td>Shipped</td>
<td>1.02</td>
<td>13.02</td>
<td>Friday</td>
<td>14</td>
<td>San Francisco</td>
<td>CA</td>
<td>96</td>
</tr>
<tr>
<td>4</td>
<td>1/29/2000</td>
<td>10:22:37</td>
<td>12</td>
<td>1</td>
<td>12</td>
<td>Shipped</td>
<td>0.87</td>
<td>12.87</td>
<td>Saturday</td>
<td>10</td>
<td>Novato</td>
<td>CA</td>
<td>132</td>
</tr>
<tr>
<td>5</td>
<td>2/1/2000</td>
<td>08:44:48</td>
<td>6.5</td>
<td>1</td>
<td>6.5</td>
<td>Shipped</td>
<td>0.55</td>
<td>7.05</td>
<td>Tuesday</td>
<td>8</td>
<td>Cupertino</td>
<td>CA</td>
<td>168</td>
</tr>
<tr>
<td>7</td>
<td>2/29/2000</td>
<td>10:31:42</td>
<td>14</td>
<td>1</td>
<td>14</td>
<td>Shipped</td>
<td>1.16</td>
<td>15.16</td>
<td>Tuesday</td>
<td>10</td>
<td>San Ramon</td>
<td>CA</td>
<td>184</td>
</tr>
<tr>
<td>8</td>
<td>2/29/2000</td>
<td>10:31:42</td>
<td>6.5</td>
<td>2</td>
<td>6.5</td>
<td>Shipped</td>
<td>1.07</td>
<td>14.07</td>
<td>Tuesday</td>
<td>10</td>
<td>San Ramon</td>
<td>CA</td>
<td>184</td>
</tr>
<tr>
<td>9</td>
<td>3/8/2000</td>
<td>16:48:47</td>
<td>11</td>
<td>3</td>
<td>11</td>
<td>Shipped</td>
<td>2.72</td>
<td>35.72</td>
<td>Wednesday</td>
<td>16</td>
<td>San Ramon</td>
<td>CA</td>
<td>184</td>
</tr>
<tr>
<td>10</td>
<td>1/30/2000</td>
<td>14:13:57</td>
<td>10</td>
<td>1</td>
<td>10</td>
<td>Shipped</td>
<td>0</td>
<td>10</td>
<td>Sunday</td>
<td>14</td>
<td>Scarsdale</td>
<td>NY</td>
<td>224</td>
</tr>
<tr>
<td>12</td>
<td>2/26/2000</td>
<td>03:42:17</td>
<td>12.7</td>
<td>1</td>
<td>12.7</td>
<td>Shipped</td>
<td>0</td>
<td>12.7</td>
<td>Saturday</td>
<td>3</td>
<td>Novato</td>
<td>CA</td>
<td>236</td>
</tr>
<tr>
<td>13</td>
<td>3/30/2000</td>
<td>11:51:44</td>
<td>6.5</td>
<td>1</td>
<td>4.88</td>
<td>Shipped</td>
<td>0</td>
<td>4.88</td>
<td>Thursday</td>
<td>11</td>
<td>Novato</td>
<td>CA</td>
<td>236</td>
</tr>
<tr>
<td>14</td>
<td>3/30/2000</td>
<td>11:51:44</td>
<td>6.5</td>
<td>1</td>
<td>4.88</td>
<td>Shipped</td>
<td>0</td>
<td>4.88</td>
<td>Thursday</td>
<td>11</td>
<td>Novato</td>
<td>CA</td>
<td>236</td>
</tr>
<tr>
<td>15</td>
<td>3/30/2000</td>
<td>11:51:44</td>
<td>7</td>
<td>1</td>
<td>7</td>
<td>Shipped</td>
<td>0</td>
<td>7</td>
<td>Thursday</td>
<td>11</td>
<td>Novato</td>
<td>CA</td>
<td>236</td>
</tr>
<tr>
<td>16</td>
<td>3/30/2000</td>
<td>11:51:44</td>
<td>12</td>
<td>1</td>
<td>12</td>
<td>Shipped</td>
<td>0</td>
<td>12</td>
<td>Thursday</td>
<td>11</td>
<td>Novato</td>
<td>CA</td>
<td>236</td>
</tr>
</tbody>
</table>
Pig LOAD Function

- “Pigs eat anything”
- LOAD 'data' [USING function] [AS schema];
- USING
  - PigStorage → structured text file (default)
  - TextLoader → unstructured UTF-8 data
  - Other and User Defined Functions
- AS
  - (Field1[:type], Field2[:type], ... FieldX[type])
  - Bytearray default type
Pig Example: Loading Data

> pig -x mapreduce
> a = LOAD 'rmacc/KDDCupCleaned.txt';

> All Statements end with semicolon !!!
Pig Debugging Statements

<table>
<thead>
<tr>
<th>Debug Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUMP</td>
<td>Display results</td>
</tr>
<tr>
<td>DESCRIBE</td>
<td>Display schema of relation</td>
</tr>
<tr>
<td>EXPLAIN</td>
<td>Display execution plan</td>
</tr>
<tr>
<td>ILLUSTRATE</td>
<td>Display step-by-step execution</td>
</tr>
</tbody>
</table>

*Full list* [http://pig.apache.org/docs/r0.11.1/test.html#diagnostic-ops](http://pig.apache.org/docs/r0.11.1/test.html#diagnostic-ops)

- **Describe and illustrate only work if schema is provided**
Pig Example: Loading Data

> a = LOAD ‘rmacc/KDDrmacc‘ , AS (key,date,time, qty:float);

• **Columns can be accessed by**
  
  • $0$ for second column (first contains key)
  • or name (key, date, time....)
Pig Data Types

• **Simple:**
  - int, long, float, double, chararray, bytearray, boolean

• **Complex:**
  - tuple - a set of fields (10, 5, alpha)
  - bag - a collection of tuples { (10,5, alpha) (8,2,beta) }
  - map - a set of key value pairs [key#value]
Pig Example: Reduce # of Fields

\[
\begin{align*}
> & \quad a = \text{LOAD} \ '\text{rmacc/KDDCupCleaned.txt}' \ AS \\
& \quad \text{(key:int,date,time,up,ol,qty,os,tax,amount:float,wd,hour,city,state,ci);}
> & \quad b = \text{FOREACH} \ a \ \text{GENERATE} \ \text{key, date, time, amount;}
> & \quad \text{STORE} \ b \ \text{INTO} \ '\text{rmacc/KDDCupShort'.txt USING PigStorage('‚,’);}
\end{align*}
\]
Pig Example: Group per Date

> a = LOAD 'rmacc/KDDrmacc.txt' AS
  (key,date,time,qty:float);

> groupday = GROUP a BY date;

> illustrate groupday →

> group is new key for each bag (day)

> tuples within data within each bag
Pig Example: Sum per Date

> a = LOAD 'rmacc/KDDrmacc.txt' AS (key,date,time,qty:float);
> groupday = GROUP a BY date;
> sumday = FOREACH groupday GENERATE group, SUM(a.qty); sr
> STORE sumday INTO 'rmacc/sumday';
# Evaluation Functions
(Case Sensitive)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVG</td>
<td>Calculates average</td>
</tr>
<tr>
<td>CONCAT</td>
<td>Concatenates two expressions of identical type</td>
</tr>
<tr>
<td>COUNT</td>
<td>Counts the number of elements in a bag</td>
</tr>
<tr>
<td>COUNT_STAR</td>
<td>Like count by includes NULL values in count</td>
</tr>
<tr>
<td>DIFF</td>
<td>Compares two fields in a tuple</td>
</tr>
<tr>
<td>IsEmpty</td>
<td>Checks if a bag or map is empty</td>
</tr>
<tr>
<td>MAX</td>
<td>Calculates maximum</td>
</tr>
<tr>
<td>MIN</td>
<td>Calculates minimum</td>
</tr>
<tr>
<td>SIZE</td>
<td>Computes the number of elements (characters)</td>
</tr>
<tr>
<td>SUM</td>
<td>Calculates sum</td>
</tr>
<tr>
<td>TOKENIZE</td>
<td>Splits a string and outputs a bag of words</td>
</tr>
</tbody>
</table>

*List with examples*  
[http://pig.apache.org/docs/r0.11.1/func.html#eval-functions](http://pig.apache.org/docs/r0.11.1/func.html#eval-functions)
Other Functions

- Math Functions
- String Functions
- Datetime Functions
- Tuple, Bag, Map Functions
- User Defined Functions
Pig Example: Filter Purchases> $100

> a = LOAD ‘rmacc/sumday‘ AS (date,sum:float);
> bigpur = FILTER a BY sum>1000;
> STORE bigpur INTO ‘rmacc/bigpur’;
## Relational Operators

<table>
<thead>
<tr>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD</td>
<td>Loads data from the file system</td>
</tr>
<tr>
<td>GROUP</td>
<td>Groups the data in one or more relations</td>
</tr>
<tr>
<td>FOREACH</td>
<td>Generates data transformations based on columns of data</td>
</tr>
<tr>
<td>FILTER</td>
<td>Selects tuples from a relation based on some condition</td>
</tr>
</tbody>
</table>

*Full list with examples*  
[http://pig.apache.org/docs/r0.11.1/basic.html#Relational+Operators](http://pig.apache.org/docs/r0.11.1/basic.html#Relational+Operators)
Other Operators

- Arithmetic Operators
- Boolean Operators
- Cast Operators
- Comparison Operators
- Type Construction Operators
- Dereference Operators
- Disambiguate Operator
- Flatten Operator
- Null Operators
- Sign Operators
Pig WordCount Code

> b = LOAD 'rmacc/Beowulf.txt' AS be;
> beowords = FOREACH b GENERATE flatten(TOKENIZE(beo)) as bw;
> wg = GROUP beowords BY bw;
> wc = FOREACH wg GENERATE group, COUNT(beowords) as bc;
> sumord =ORDER wc BY bc;
> STORE sumord INTO 'rmacc/beowulfwc'
Hadoop Summary

• Accessible – runs on commodity hardware
• Robust – handles hardware failures
• Scalable – by adding more nodes
• Simple – allows users to quickly write efficient parallel code
• Pig - easy to learn
Conclusions

• **Individual components “easy” to setup → integration more complicated**

• **Resources**
  - Apache Hadoop (download and docu)
    - [http://hadoop.apache.org](http://hadoop.apache.org)
  - Online Searches
  - Books for overview, not technical details

• **“Evolving project” → constantly changing, documentation can’t keep up with development**
Questions ?