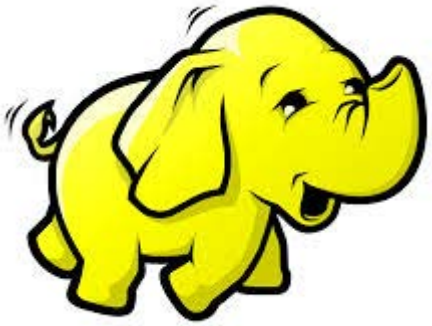
Hadoop/Pig Install

- Download and install
  - Virtualbox www.virtualbox.org
  - Virtualbox Extension Pack
- Download virtual machine
  - link in schedule (https://rmacchpcsymposium2015.sched.org/?iframe=no)
- Import virtual machine in Virtualbox
  - File → Import Appliance



Hadoop & Pig

RMACC
Rocky Mountain Advanced Computing Consortium

Dr. Karina Hauser
Senior Lecturer
Management & Entrepreneurship

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.
DilbertCartoonist@gmail.com
IT COMES FROM EVERYWHERE. IT KNOWS ALL.
ACCORDING TO THE BOOK OF WIKIPEDIA, ITS NAME IS "BIG DATA."
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.
$
www.dilbert.com
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

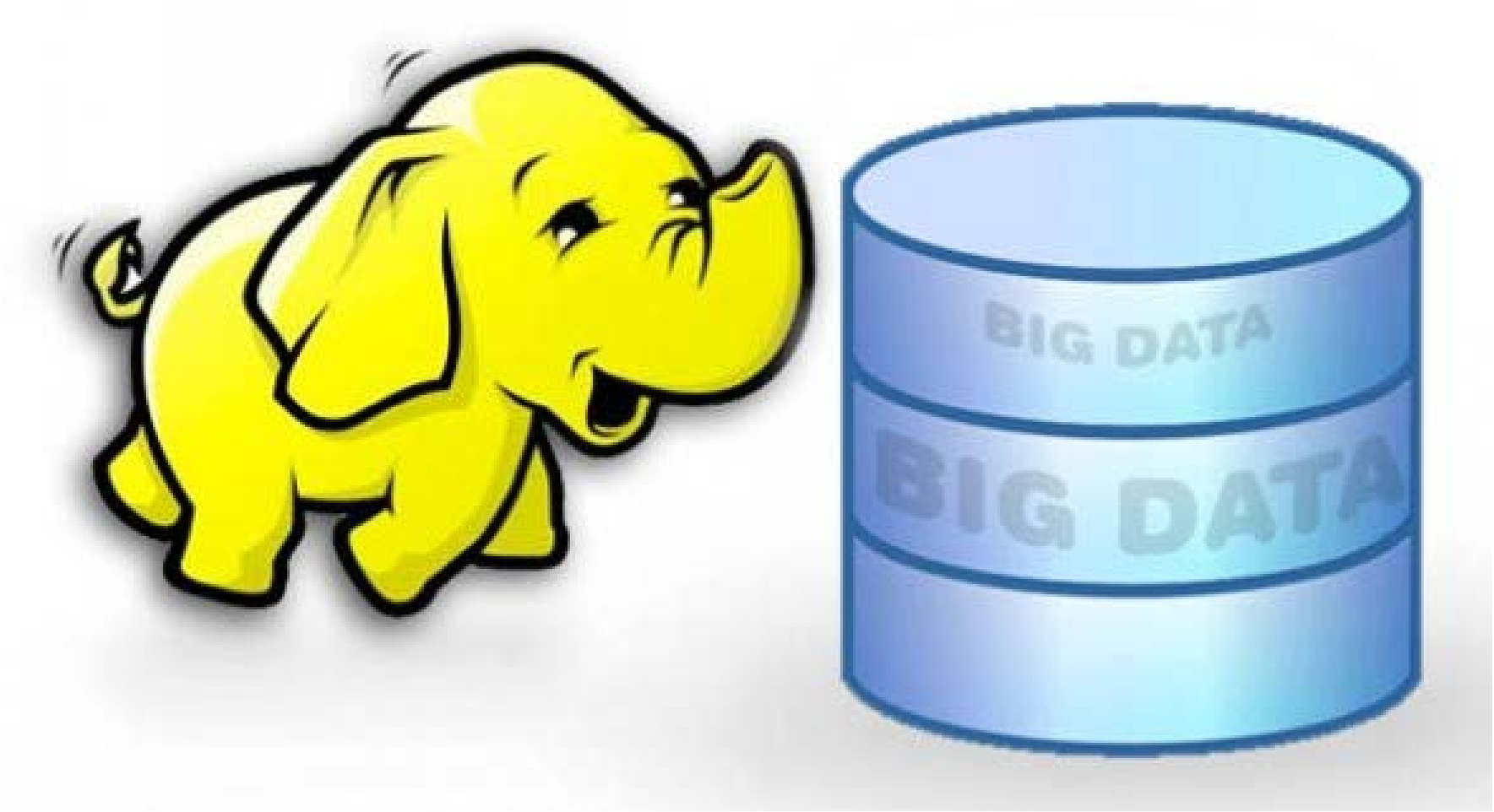
- The Economist Intelligence Unit study showed that nine out of 10 surveyed business leaders believe data is now the fourth factor of production, as fundamental to business as land, labor and capital.

The Economist, “Deciding Factor.”

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
- 2013 - Hadoop 2.0

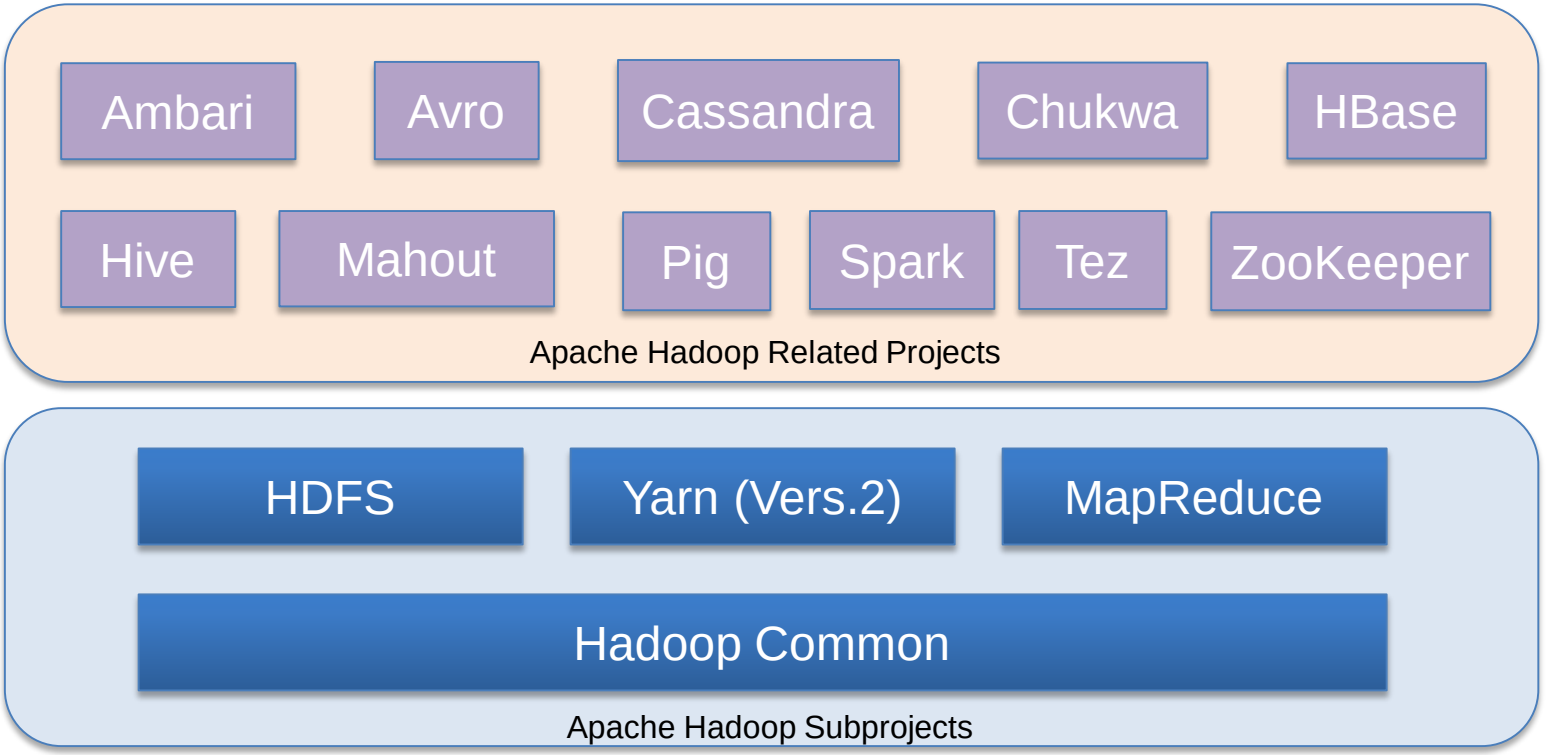
Hadoop Today

- 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



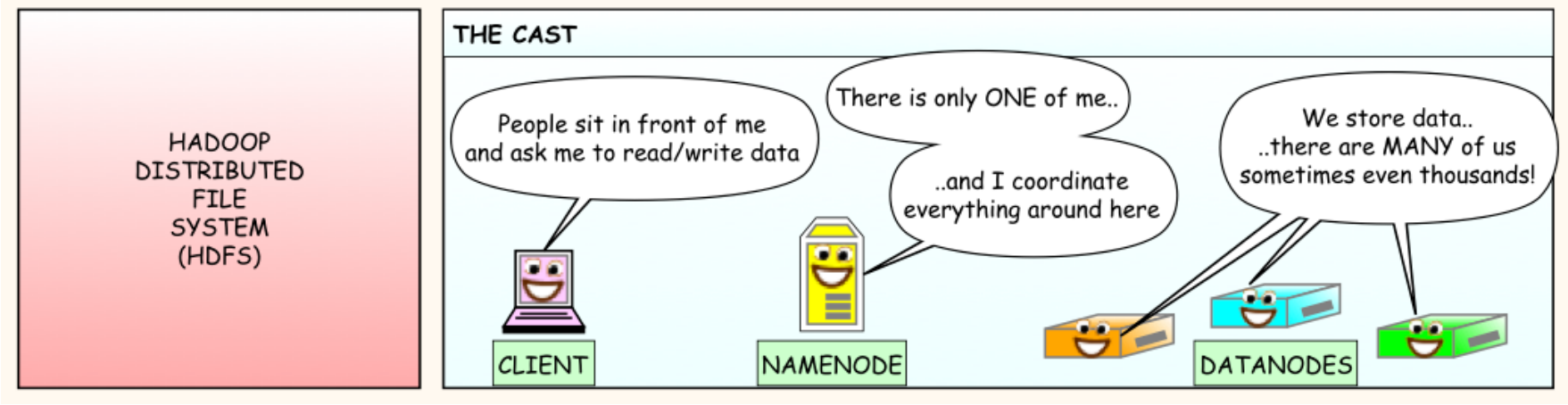
Hadoop Cluster

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

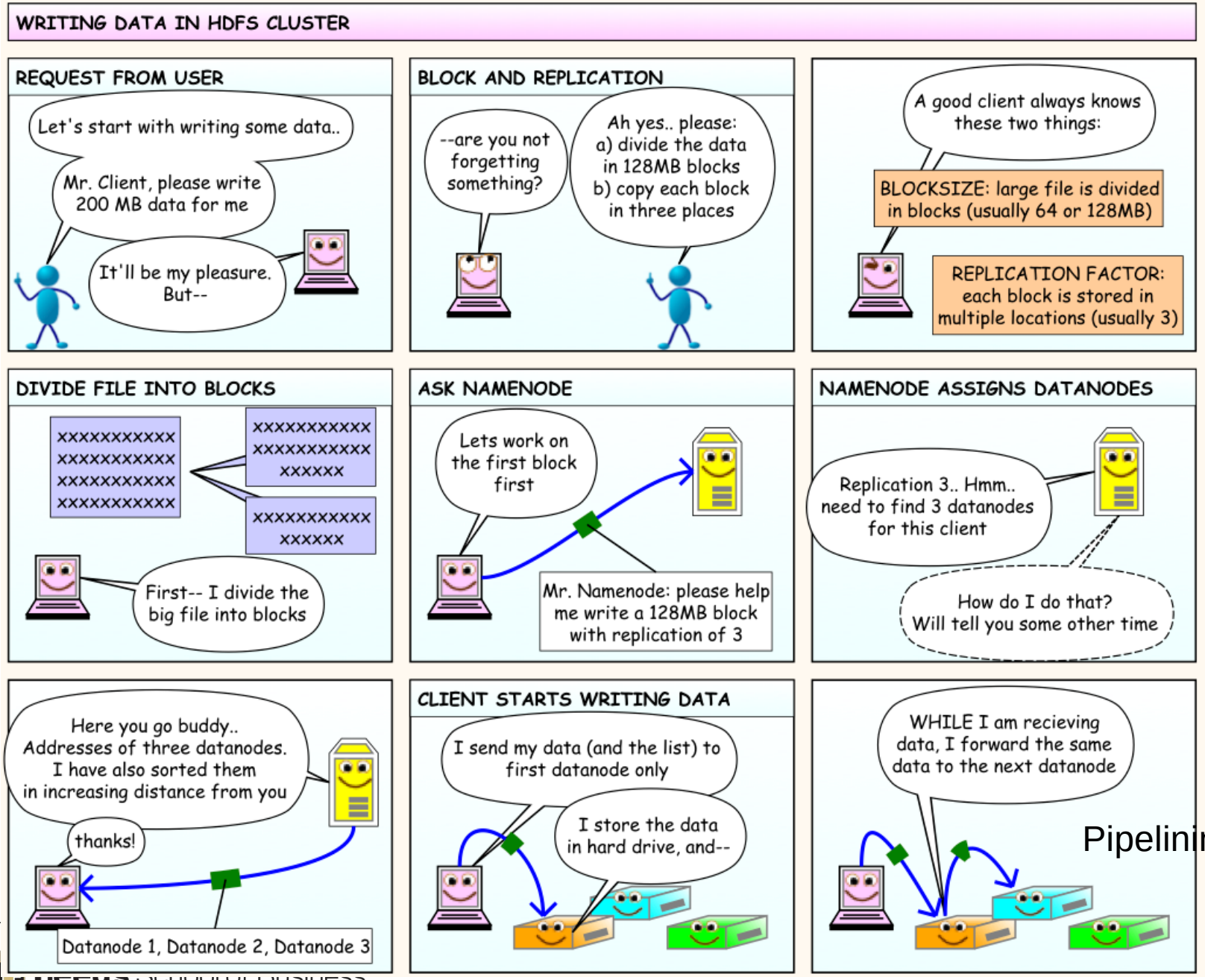
HDFS

- 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



From: http://nosql.mypopescu.com/post/15561851616/hadoop-distributed-file-system-hdfs-a-cartoon-is-worth

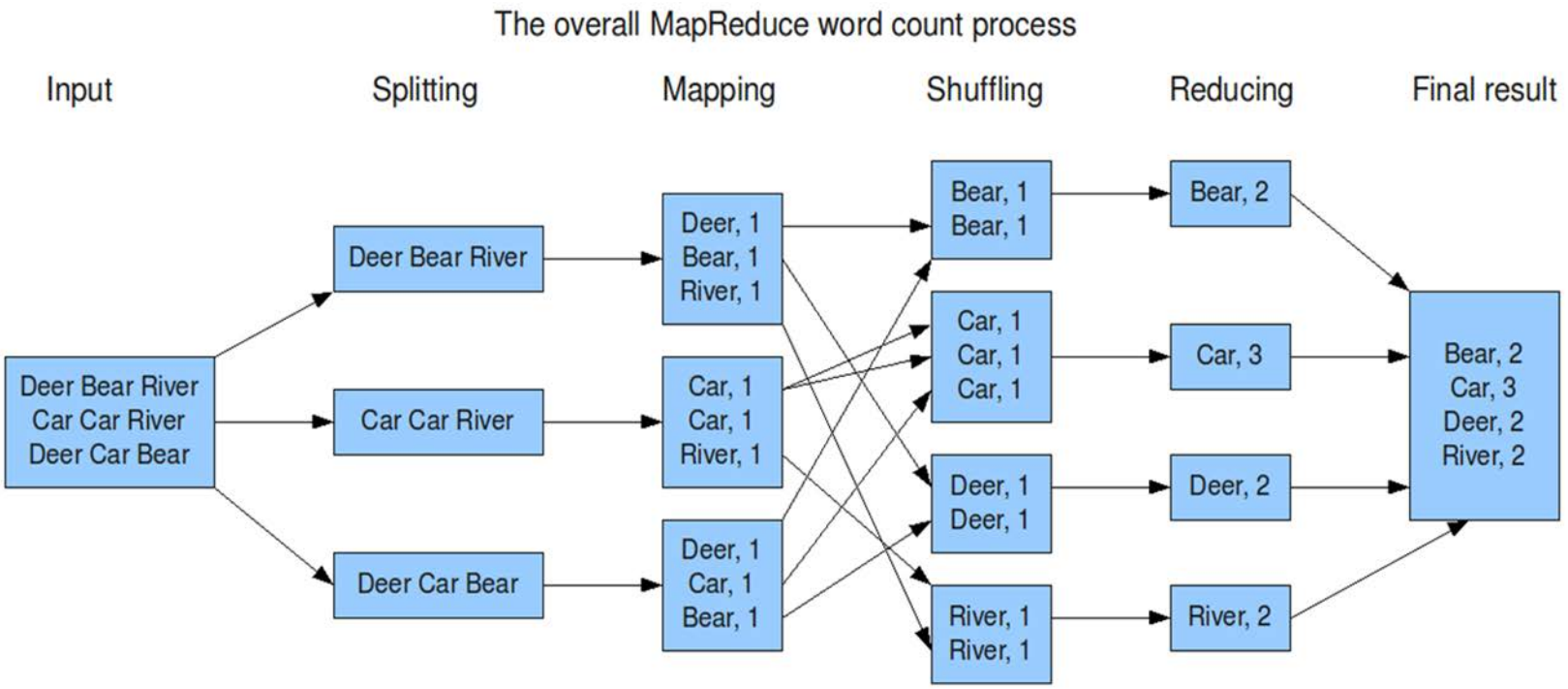


From: http://nosql.mypopescu.com/post/15561851616/hadoop-distributed-file-system-hdfs-a-cartoon-is-worth

MapReduce

- Programing 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

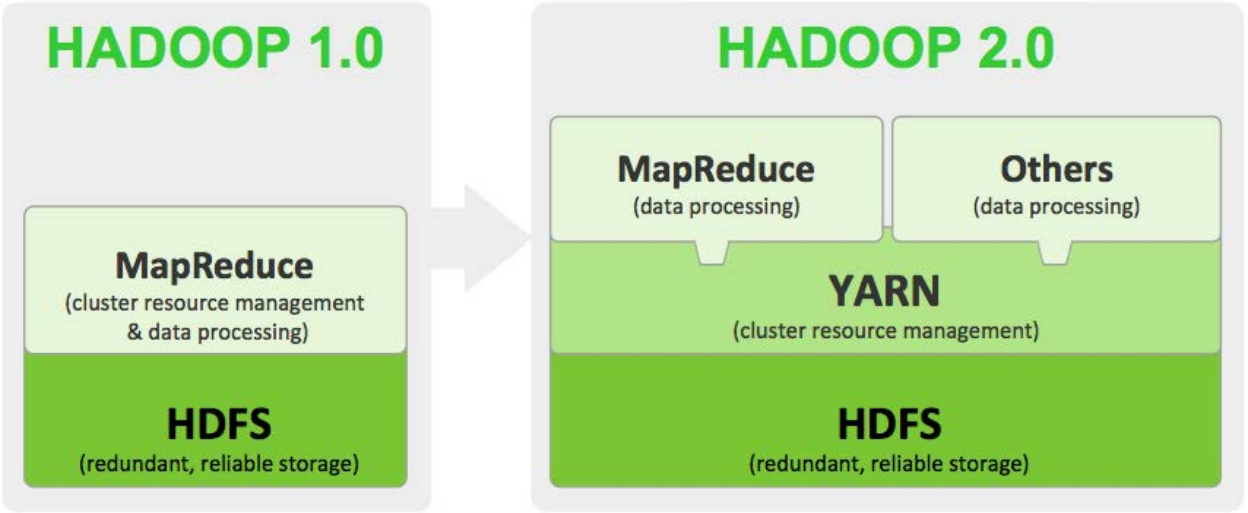
MapReduce WordCount Example

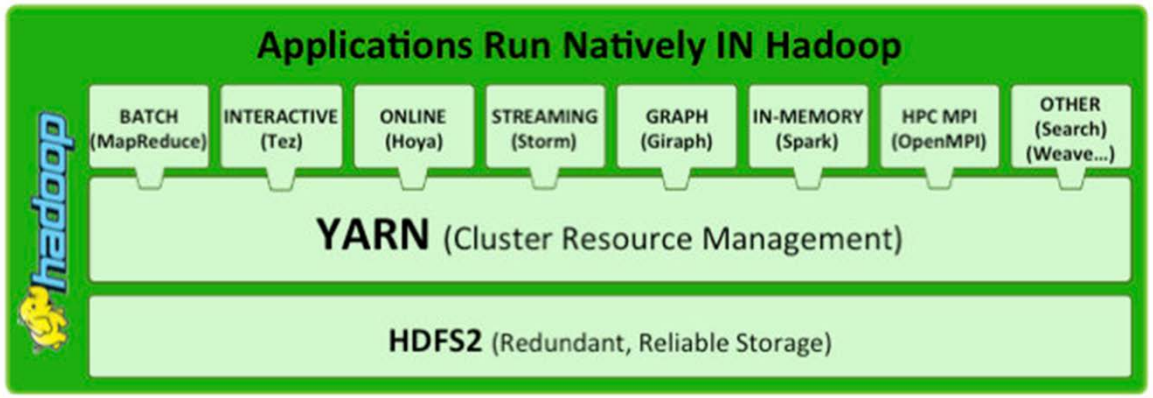


Problems suited for MapReduce

- 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 2.0

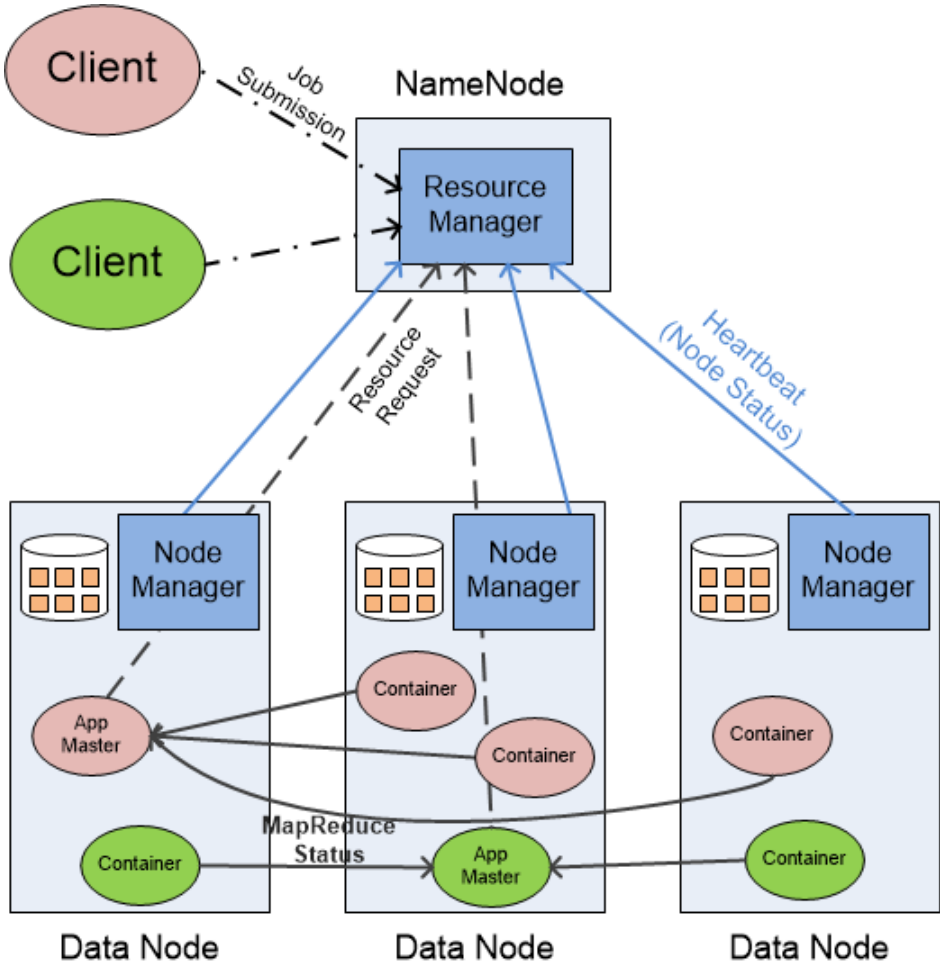

HADOOP 1.0
MapReduce
(cluster resource management & data processing)
HDFS
(redundant, reliable storage)
HADOOP 2.0
MapReduce
(data processing)
Others
(data processing)
YARN
(cluster resource management)
HDFS
(redundant, reliable storage)


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)

Yarn

- *Yet Another Resource Scheduler*
- "Operating System" for Hadoop
- Responsible for cluster management → improves resource utilization
- Allocates resources to different applications
- Supports multiple processing modes; not just MapReduce
- Security

Hadoop 2.0 Components



Hadoop Components

- Name Node
  - Stores metadata (filenames, replications factors …)
  - Checks data node availability (Heartbeat)
  - Hadoop 1: One Hadoop 2: Mulitple
- Data Node
  - Stores data
  - Replicates blocks
  - Computation

Hadoop Components

- Resource Manager (RM)
  - Scheduler that allocates available cluster resources amongst the competing applications
- Node Manager (NM)
  - Takes direction from the RM. Manages resources on single node.
- Application Master
  - An **instance** of a framework-specific library, that runs a specific YARN job and is responsible for negotiating resources from the RM. Working with the NM to execute and monitor containers (allocated resources on node).

What makes Hadoop different?

Traditional Clusters vs. Hadoop

- Traditional
  - Computationally intensive
  - Moves data to code
  - Shared file storage

- Hadoop
  - Data intensive
  - Moves code to data
  - Individual disks

MPI vs. MapReduce

- MPI
  - Requires the **user** to implement
    - ***data splitting,***
    - ***data management,***
    - ***parallelization,***
    - ***synchronization and***
    - ***fault-tolerance in their applications***
- Hadoop/MapReduce
  - **Framework** handles
    - ***data splitting***
    - ***etc.***

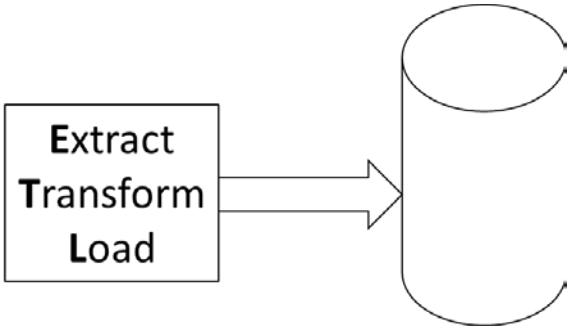
Databases vs. HDFS

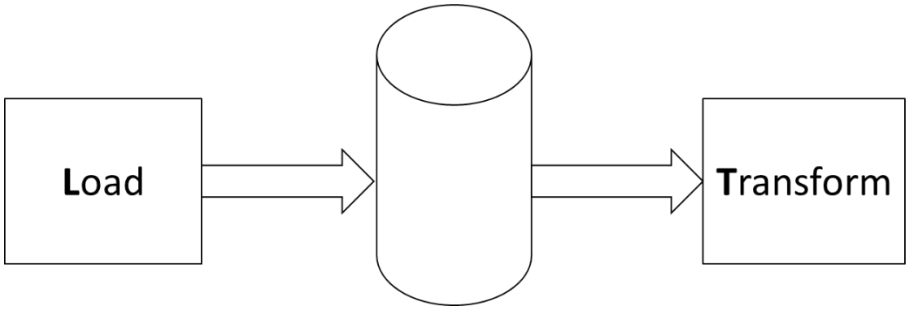
Databases

- Schema created before storage
- Data cleaned before storage

HDFS

- Data loaded without schema
- Schema created during data processing





Setup

Setup

- Three options:
  - Standalone (single Java process)
  - **Pseudo-Distributed (separate Java processes)**
  - Fully-Distributed
- Prerequisites (on virtual machine):
  - Ubuntu server 14.04 with SSH
  - Ubuntu desktop (for monitoring)
  - Oracle (Sun) Java 1.7.0_80-b15
    - ***http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in-ubuntu-via.html***

Setup Files

- hosts (ip address)
- .bashrc (Java dir, home dir)
- hadoop configuration files in /usr/local/hadoop/conf
  - hadoop-env.sh (Java dir)
  - core-site.xml (default file system)
  - hdfs-site.xml (replication factor)
  - mapred-site.xml (mapreduce framework)
  - yarn-site.xml (nodemanager aux-services)
  - master/slave (ips for multi-node cluster)

Login

- User: rmacc2015
- Password: rmacc2015
- 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

Format Namenode

- All existing data will be deleted !!!
- Delete tmp directory

  $ rm -R /tmp/hadoop-hduser/dfs/data

- Format Namenode

  $ bin/hdfs namenode -format

Starting Hadoop Daemons

- All:

  $ sbin/start-dfs.sh

  $ sbin/start-yarn.sh

- Log output in *logs* directory

Check Daemons

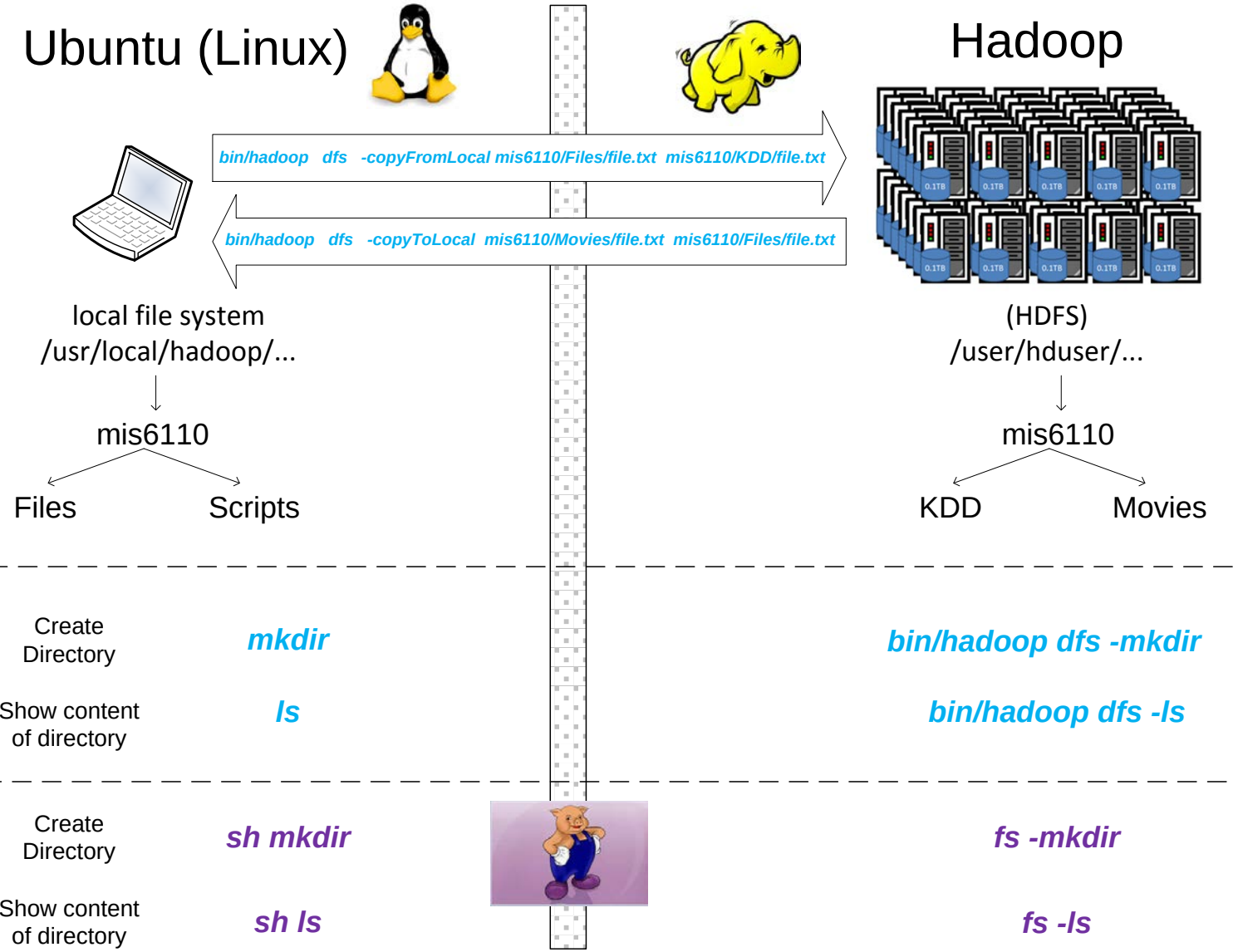
- jps →
  - 1367 NameNode
  - 8695 DataNode
  - 2600 SecondaryNameNode
  - 4786 ResourceManager
  - 5019 NodeManager
  - 4558 Jps

Web Interfaces

- http://localhost:8088 → Cluster status and jobs
- http://localhost:50070 → HDFS

HDFS

- Hadoop Distributed File System


Ubuntu (Linux)
Hadoop
bin/hadoop dfs -copyFromLocal mis6110/Files/file.txt mis6110/KDD/file.txt
bin/hadoop dfs -copyToLocal mis6110/Movies/file.txt mis6110/Files/file.txt
local file system
/usr/local/hadoop/...
(HDFS)
/user/hduser/...
mis6110
Files
Scripts
mis6110
KDD
Movies
Create Directory
mkdir
bin/hadoop dfs -mkdir
Show content of directory
ls
bin/hadoop dfs -ls
Create Directory
sh mkdir
fs -mkdir
Show content of directory
sh ls
fs -ls

HDFS Shell

- bin/hdfs dfsadmin –help → all admin commands
- bin/hdfs dfs –help → all commands
- Most commands similar to unix
  - dfs –copyFromLocal
  - dfs -ls
- Shell commands: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

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 and KDDrmacc.txt to hdfs

Copying File to HDFS

- Create new directory:

  $ bin/hdfs dfs -mkdir /rmacc

- Copy files:

  $ bin/hdfs dfs -copyFromLocal ../rmacc/beowulf.txt /rmacc/

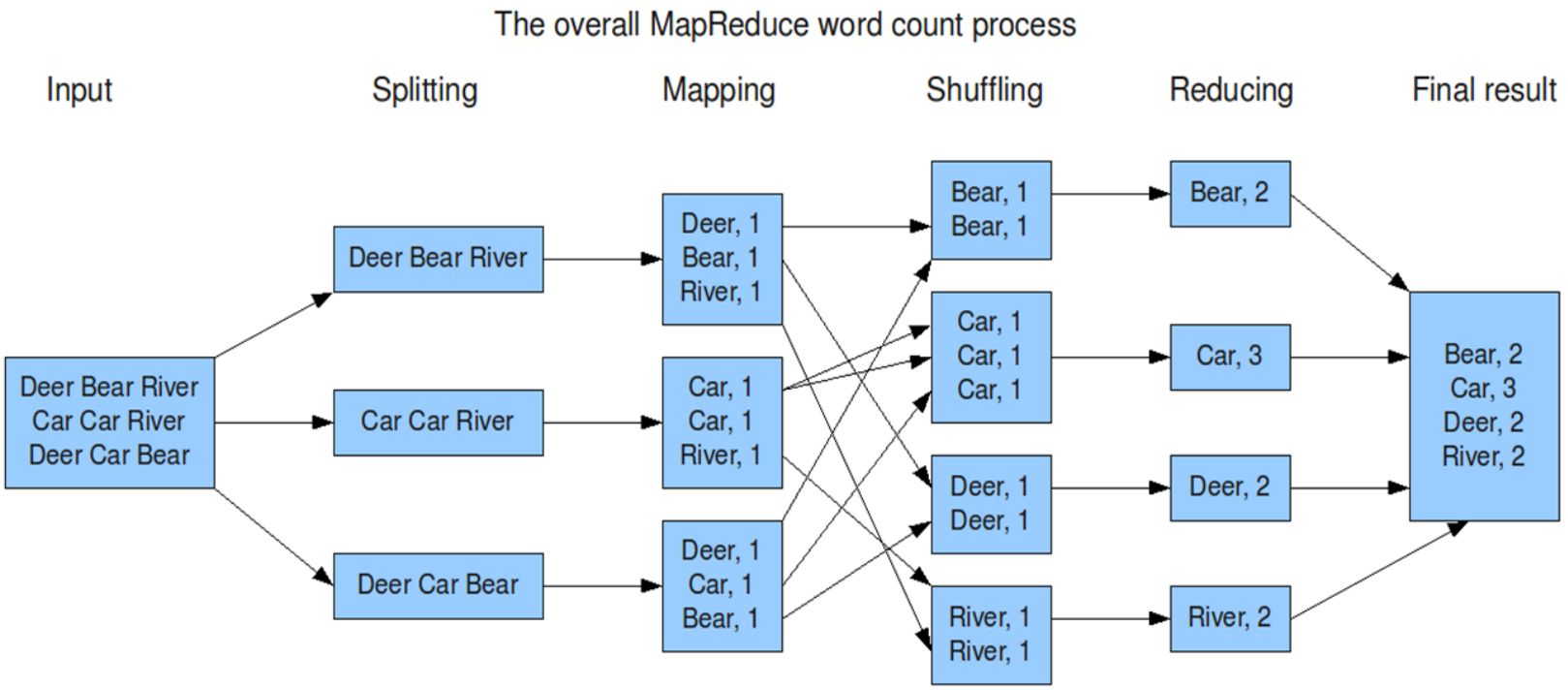
  $ bin/hdfs dfs -copyFromLocal ../rmacc/KDD* /rmacc/

- Check:

  bin/hdfs dfs -ls -R /rmacc

  - ***or localhost:50070 → Utilities → Browse the filesystem***

MapReduce WordCount Example



From: http://www.rabidgremlin.com/data20/#(3)

Java Code for WordCount Example

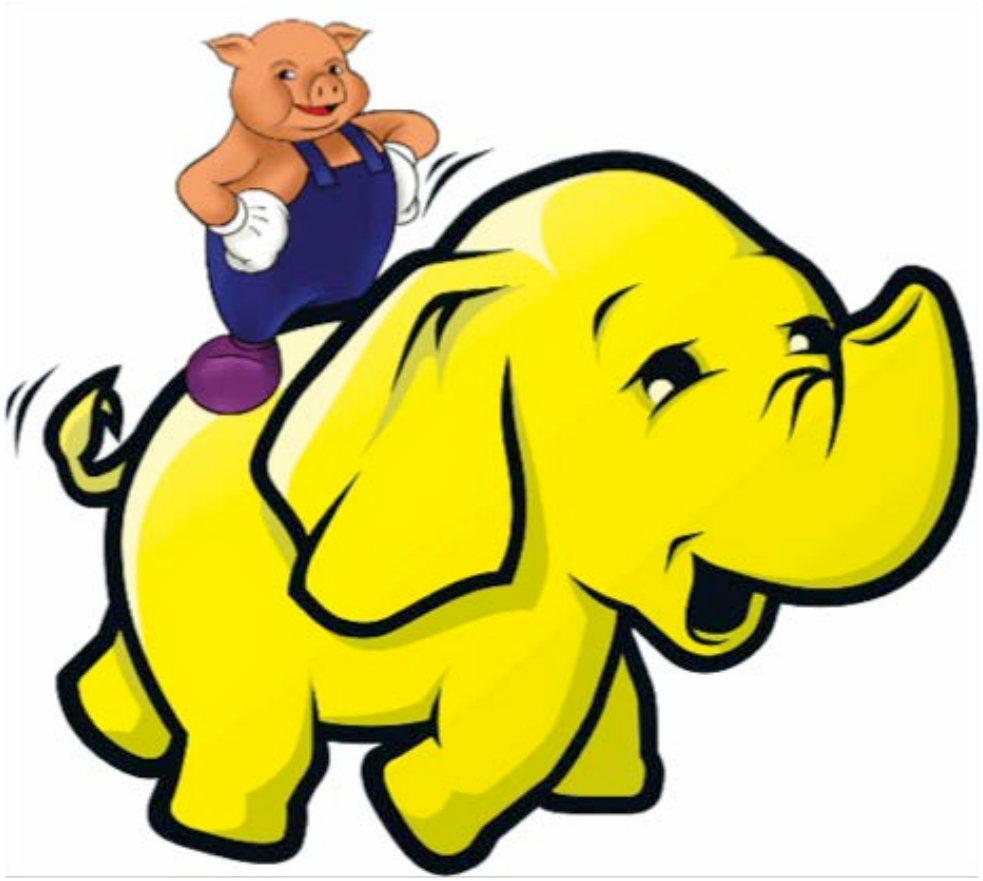
```
1.  package org.myorg;
2.
3.  import java.io.IOException;
4.  import java.util.*;
5.
6.  import org.apache.hadoop.fs.Path;
7.  import org.apache.hadoop.conf.*;
8.  import org.apache.hadoop.io.*;
9.  import org.apache.hadoop.mapred.*;
10. import org.apache.hadoop.util.*;
11.
12. public class WordCount {
13.
14.     public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
15.       private final static IntWritable one = new IntWritable(1);
16.       private Text word = new Text();
17.
18.       public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
19.         String line = value.toString();
20.         StringTokenizer tokenizer = new StringTokenizer(line);
21.         while (tokenizer.hasMoreTokens()) {
22.           word.set(tokenizer.nextToken());
23.           output.collect(word, one);
24.         }25.               }
26.     }
27.
28.     public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
```

Java Code for WordCount Example

```
29.     public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
30.       int sum = 0;
31.       while (values.hasNext()) {
32.         sum += values.next().get();
33.       }
34.       output.collect(key, new IntWritable(sum));
35.     }
36.   }
37.
38.   public static void main(String[] args) throws Exception {
39.     JobConf conf = new JobConf(WordCount.class);
40.     conf.setJobName("wordcount");
41.
42.     conf.setOutputKeyClass(Text.class);
43.     conf.setOutputValueClass(IntWritable.class);
44.
45.     conf.setMapperClass(Map.class);
46.     conf.setCombinerClass(Reduce.class);
47.     conf.setReducerClass(Reduce.class);
48.
49.     conf.setInputFormat(TextInputFormat.class);
50.     conf.setOutputFormat(TextOutputFormat.class);
51.
52.     FileInputFormat.setInputPaths(conf, new Path(args[0]));
53.     FileOutputFormat.setOutputPath(conf, new Path(args[1]));
54.
55.     JobClient.runJob(conf);
57.   }
58. }
```

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

Starting Job History Daemon

- Pig relies on job history daemon
  (Error: ... 0.0.0.0:10020 ... retrying to connect to server)

  $ sbin/mr-jobhistory-daemon.sh start historyserver

Check Daemons

- jps →
  - 1367 NameNode
  - 8695 DataNode
  - 2600 SecondaryNameNode
  - 4786 ResourceManager
  - 5019 NodeManager
  - 7409 JobHistoryServer
  - 4558 Jps

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 or just pig)***
- 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 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

> bin/pig

> a = LOAD '/rmacc/beowulf.txt';

> All Statements end with semicolon !!!

Pig Debugging Statements

Debug Operator	Description
DUMP	Display results
DESCRIBE	Display schema of relation
EXPLAIN	Display execution plan
ILLUSTRATE	Display step-by-step execution
Full list	http://pig.apache.org/docs/r0.15.0/test.html#diagnostic-ops

- Describe and illustrate only work if schema is provided

Pig Example: KDD Data

Key	Date	Time	Unit Price	Order LineID	Qty	Order Status	Tax	Amount	Weekday	Hour	City	State	Customer ID
1	2/27/2000	08\:06\:35	15	1	15	Shipped	1.3	16.3	Sunday	8	Westport	CT	62
2	3/30/2000	10\:00\:18	9	1	9	Shipped	0	9	Thursday	10	Westport	CT	62
3	1/28/2000	14\:43\:34	12	1	12	Shipped	1.02	13.02	Friday	14	San Francisco	CA	96
4	1/29/2000	10\:22\:37	12	1	12	Shipped	0.87	12.87	Saturday	10	Novato	CA	132
5	2/1/2000	08\:44\:48	6.5	1	6.5	Shipped	0.55	7.05	Tuesday	8	Cupertino	CA	168
6	2/29/2000	10\:31\:42	15	1	15	Shipped	1.24	16.24	Tuesday	10	San Ramon	CA	184
7	2/29/2000	10\:31\:42	14	1	14	Shipped	1.16	15.16	Tuesday	10	San Ramon	CA	184
8	2/29/2000	10\:31\:42	6.5	2	6.5	Shipped	1.07	14.07	Tuesday	10	San Ramon	CA	184
9	3/8/2000	16\:48\:47	11	3	11	Shipped	2.72	35.72	Wednesday	16	San Ramon	CA	184
10	1/30/2000	14\:13\:57	10	1	10	Shipped	0	10	Sunday	14	Scarsdale	NY	224
11	1/30/2000	14\:13\:57	13.5	1	13.5	Shipped	0	13.5	Sunday	14	Scarsdale	NY	224
12	2/26/2000	03\:42\:17	12.7	1	12.7	Shipped	0	12.7	Saturday	3	Novato	CA	236
13	3/30/2000	11\:51\:44	6.5	1	4.88	Shipped	0	4.88	Thursday	11	Novato	CA	236
14	3/30/2000	11\:51\:44	6.5	1	4.88	Shipped	0	4.88	Thursday	11	Novato	CA	236
15	3/30/2000	11\:51\:44	7	1	7	Shipped	0	7	Thursday	11	Novato	CA	236
16	3/30/2000	11\:51\:44	12	1	12	Shipped	0	12	Thursday	11	Novato	CA	236

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

> a = LOAD '/rmacc/KDDCupCleaned.txt' AS (key:int,date,time,up,ol,qty,os,tax,amount:float, wd,hour,city,state,ci);

> b = FOREACH a GENERATE key, date, time, amount;

> STORE b INTO '/rmacc/KDDCupShort.txt' USING PigStorage('*') ;

(always NEW directory)

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)
  - bag contains tuples (with data)

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);
> STORE sumday INTO '/rmacc/sumday';
```

Evaluation Functions

(Case Sensitive)

Function	Description
AVG	Calculates average
CONCAT	Concatenates two expressions of identical type
COUNT	Counts the number of elements in a bag
COUNT_STAR	Like count by includes NULL values in count
DIFF	Compares two fields in a tuple
IsEmpty	Checks if a bag or map is empty
MAX	Calculates maximum
MIN	Calculates minimum
SIZE	Computes the number of elements (characters)
SUM	Calculates sum
TOKENIZE	Splits a string and outputs a bag of words
List with examples	http://pig.apache.org/docs/r0.15.0/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;
> DUMP  bigpur;
```

Relational Operators

Operators	Description
LOAD	Loads data from the file system
GROUP	Groups the data in one or more relations
FOREACH	Generates data transformations based on columns of data
FILTER	Selects tuples from a relation based on some condition
Full list with examples	http://pig.apache.org/docs/r0.15.0/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 beo;
> 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***
  - Online Searches
  - Books for overview, not technical details
- “Evolving project” → constantly changing, documentation can’t keep up with development

Questions ?