

Cloud Computing Tools

Presented By,
Sasikumar Venkatesh,
ME-CSE 2014-2016

Apache Hadoop 2.7



Outline

- ▶ Hadoop - Basics
- ▶ HDFS
 - ▶ Goals
 - ▶ Architecture
- ▶ MapReduce
 - ▶ Basics
 - ▶ Word Count Example
- ▶ Setting up of Hadoop
 - ▶ Single Node (Pseudo Distributed Mode)

Hadoop - Why ?

- ▶ Need to process huge datasets on large clusters of computers
- ▶ Very expensive to build reliability into each application
- ▶ Nodes fail every day
 - ▶ Failure is expected, rather than exceptional
 - ▶ The number of nodes in a cluster is not constant
- ▶ Need a common infrastructure
 - ▶ Efficient, reliable, easy to use
 - ▶ Open Source, Apache License

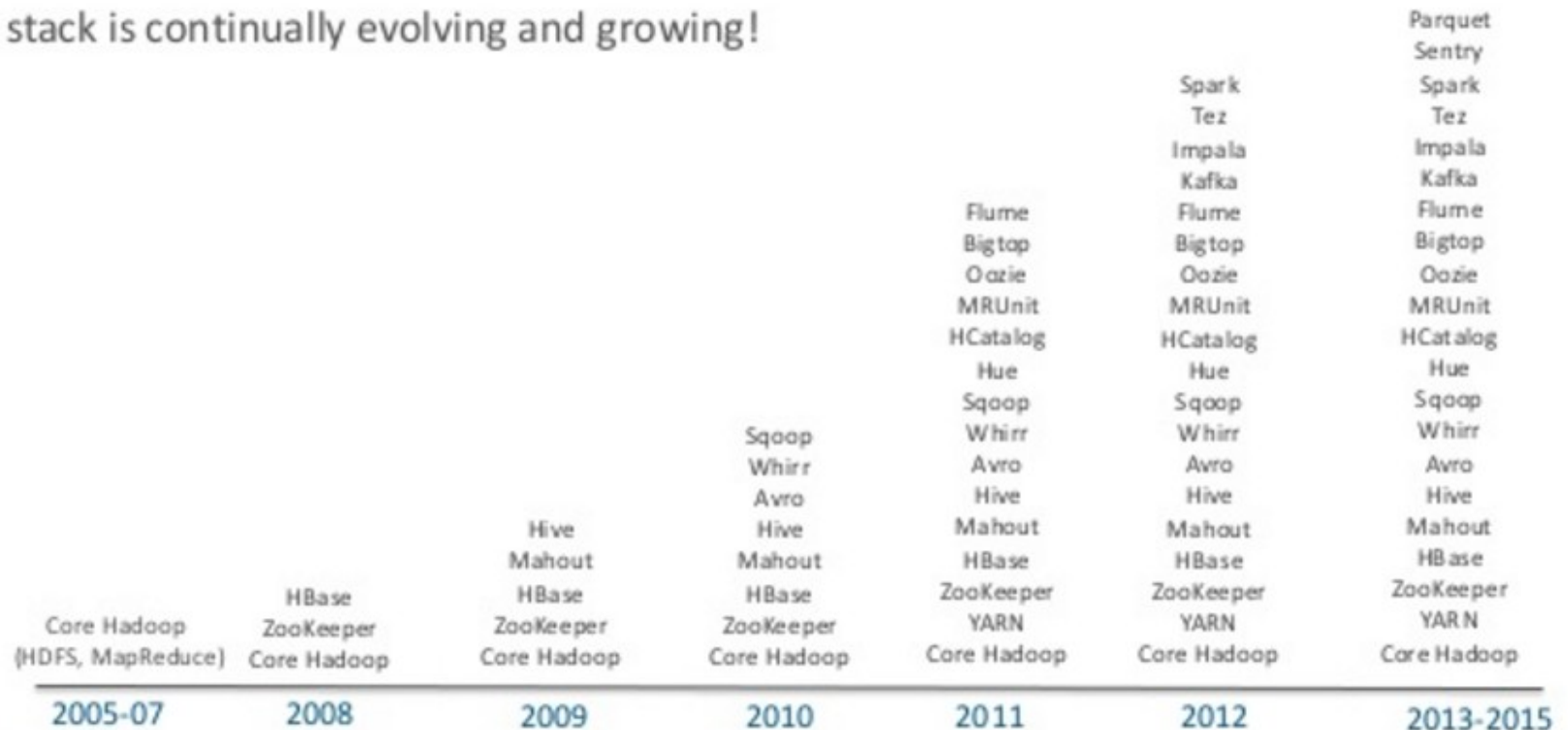
Who uses Hadoop?

- ▶ Amazon/A9
- ▶ Facebook
- ▶ Google
- ▶ New York Times
- ▶ Veoh
- ▶ Yahoo!
- ▶ Netflix
- ▶ many more

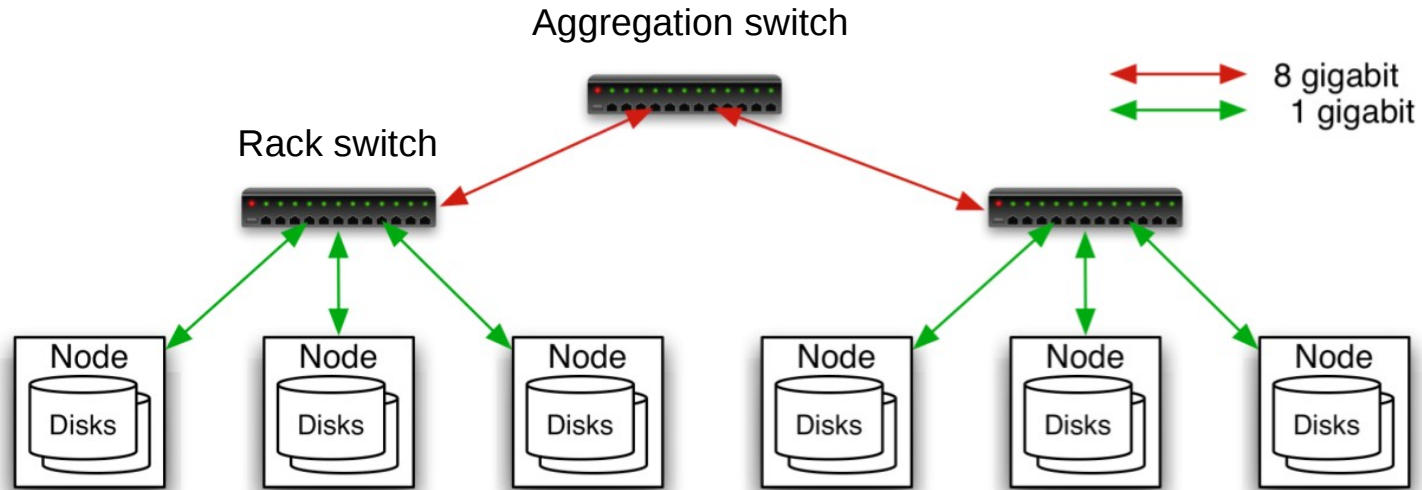
Evolution of Hadoop

Evolution of the Hadoop Platform

The stack is continually evolving and growing!

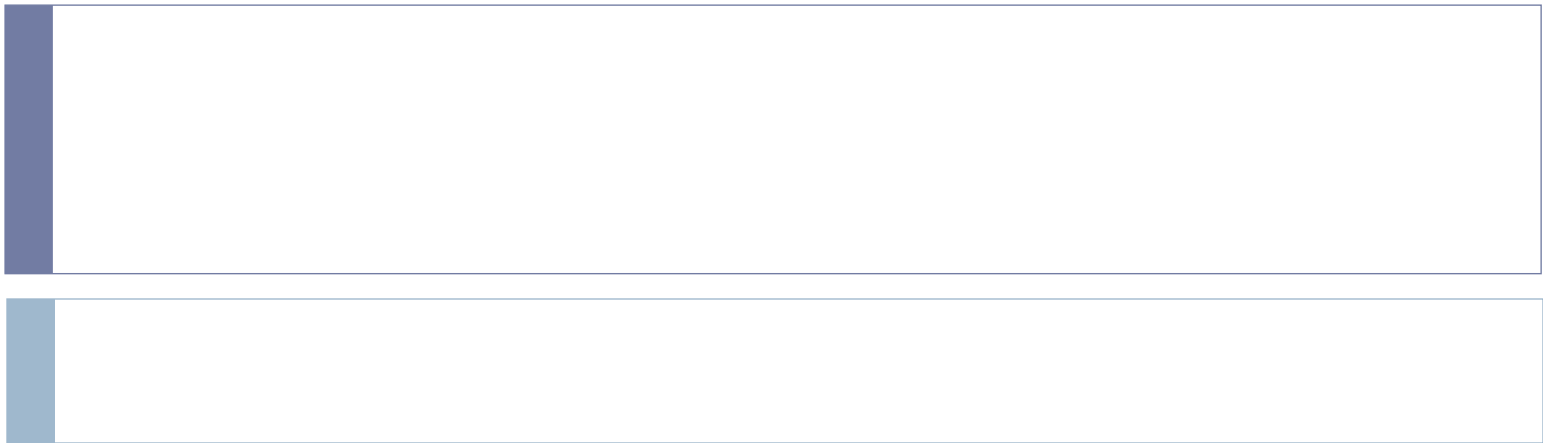


Commodity Hardware



- ▶ Typically in 2 level architecture
 - ▶ Nodes are commodity PCs
 - ▶ 30-40 nodes/rack
 - ▶ Uplink from rack is 3-4 gigabit
 - ▶ Rack-internal is 1 gigabit

Hadoop Distributed File System (HDFS)



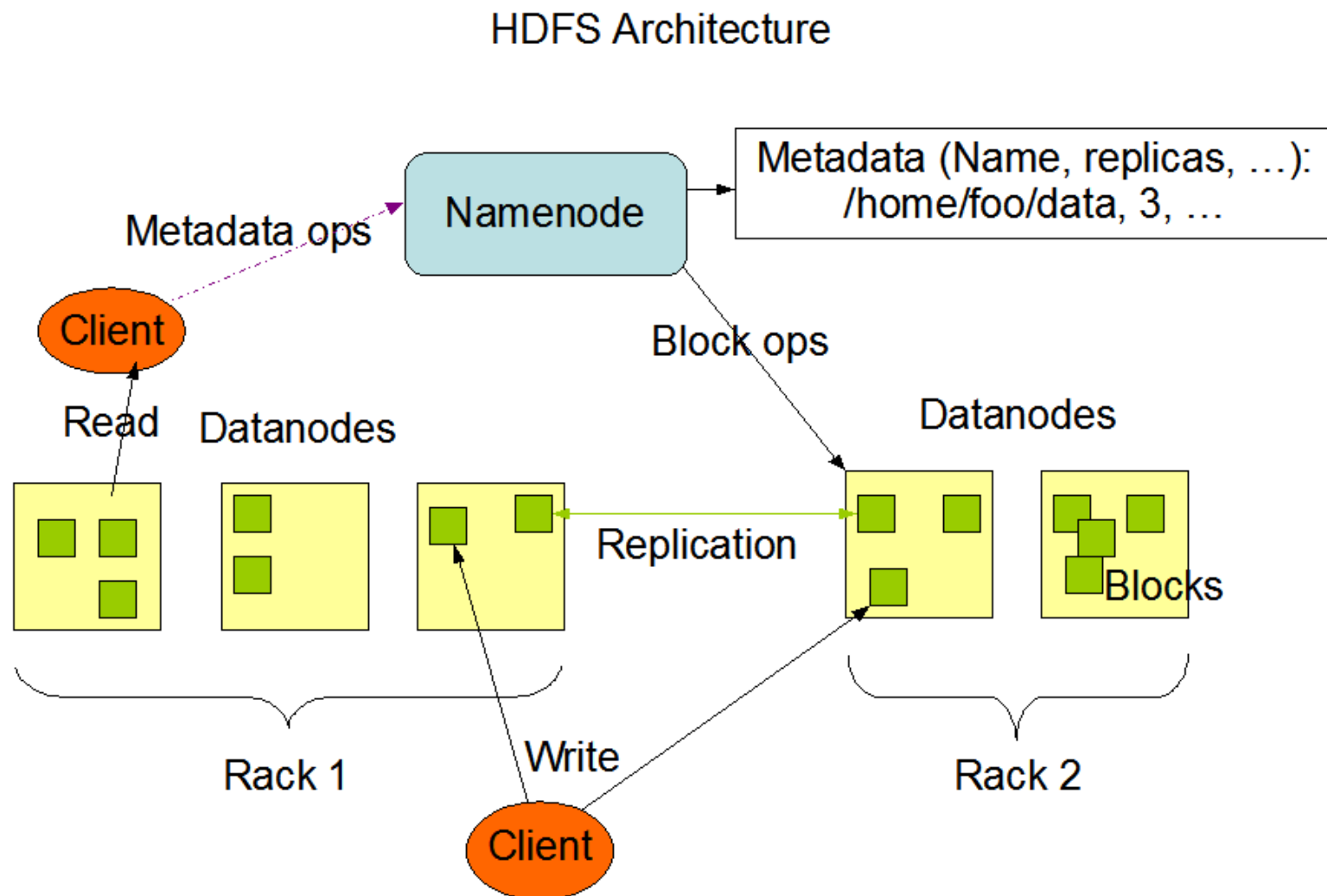
Goals of HDFS

- ▶ Very Large Distributed File System
 - ▶ 10K nodes, 100 million files, 10PB
- ▶ Assumes Commodity Hardware
 - ▶ Files are replicated to handle hardware failure
 - ▶ Detect failures and recover from them
- ▶ Optimized for Batch Processing
 - ▶ Data locations exposed so that computations can move to where data resides
 - ▶ Provides very high aggregate bandwidth

Distributed File System

- ▶ Single Namespace for entire cluster
- ▶ Data Coherency
 - ▶ Write-once-read-many access model
 - ▶ Client can only append to existing files
- ▶ Files are broken up into blocks
 - ▶ Typically 64MB block size
 - ▶ Each block replicated on multiple DataNodes
- ▶ Intelligent Client
 - ▶ Client can find location of blocks
 - ▶ Client accesses data directly from DataNode

HDFS Architecture



Functions of a NameNode

- ▶ **Manages File System Namespace**
 - ▶ Maps a file name to a set of blocks
 - ▶ Maps a block to the DataNodes where it resides
- ▶ **Cluster Configuration Management**
- ▶ **Replication Engine for Blocks**

NameNode Metadata

- ▶ **Metadata in Memory**
 - ▶ The entire metadata is in main memory
 - ▶ No demand paging of metadata
- ▶ **Types of metadata**
 - ▶ List of files
 - ▶ List of Blocks for each file
 - ▶ List of DataNodes for each block
 - ▶ File attributes, e.g. creation time, replication factor
- ▶ **A Transaction Log**
 - ▶ Records file creations, file deletions etc

DataNode

- ▶ **A Block Server**
 - ▶ Stores data in the local file system (e.g. ext3)
 - ▶ Stores metadata of a block (e.g. CRC)
 - ▶ Serves data and metadata to Clients
- ▶ **Block Report**
 - ▶ Periodically sends a report of all existing blocks to the NameNode
- ▶ **Facilitates Pipelining of Data**
 - ▶ Forwards data to other specified DataNodes

Heartbeats

- ▶ DataNodes send heartbeat to the NameNode
 - ▶ Once every 3 seconds
- ▶ NameNode uses heartbeats to detect DataNode failure

Replication Engine

- ▶ **NameNode detects DataNode failures**
 - ▶ Chooses new DataNodes for new replicas
 - ▶ Balances disk usage
 - ▶ Balances communication traffic to DataNodes

Data Correctness

- ▶ Use Checksums to validate data
 - ▶ Use CRC32
- ▶ File Creation
 - ▶ Client computes checksum per 512 bytes
 - ▶ DataNode stores the checksum
- ▶ File access
 - ▶ Client retrieves the data and checksum from DataNode
 - ▶ If Validation fails, Client tries other replicas

NameNode Failure

- ▶ A single point of failure
- ▶ Transaction Log stored in multiple directories
 - ▶ A directory on the local file system
 - ▶ A directory on a remote file system (NFS/CIFS)
- ▶ Need to develop a real HA solution

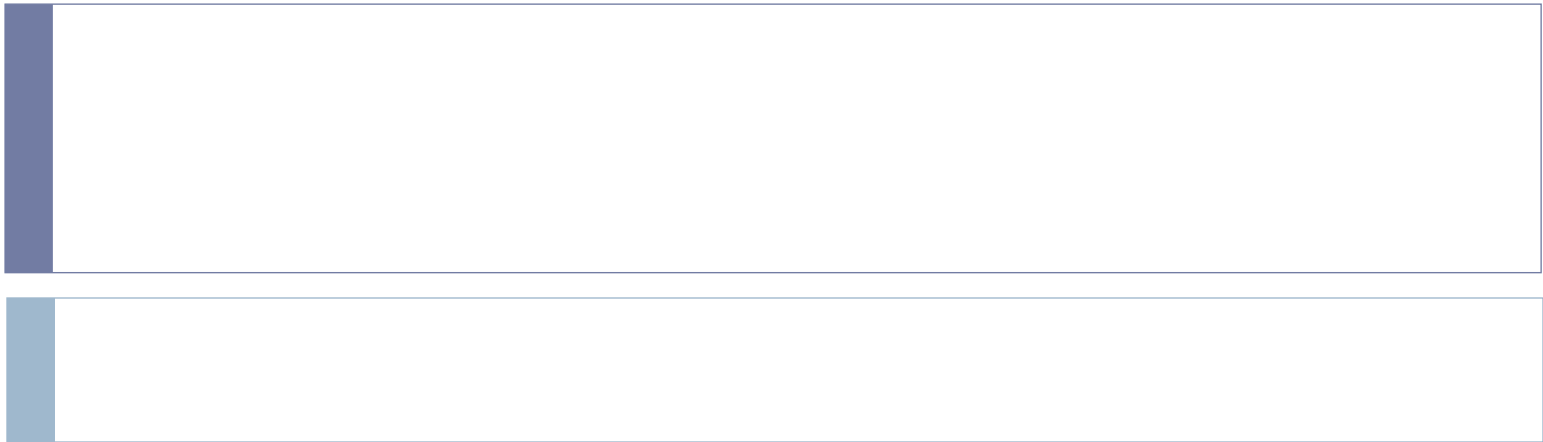
Secondary NameNode

- ▶ Copies FSImage and Transaction Log from Namenode to a temporary directory
- ▶ Merges FSImage and Transaction Log into a new FSImage in temporary directory
- ▶ Uploads new FSImage to the NameNode
 - ▶ Transaction Log on NameNode is purged

User Interface

- ▶ **Commads for HDFS User:**
 - ▶ `hadoop dfs -mkdir /foodir`
 - ▶ `hadoop dfs -cat /foodir/myfile.txt`
 - ▶ `hadoop dfs -rm /foodir/myfile.txt`
- ▶ **Commands for HDFS Administrator**
 - ▶ `hadoop dfsadmin -report`
 - ▶ `hadoop dfsadmin -decommision datanodename`
- ▶ **Web Interface**
 - ▶ `http://localhost:50070/dfshealth.jsp`

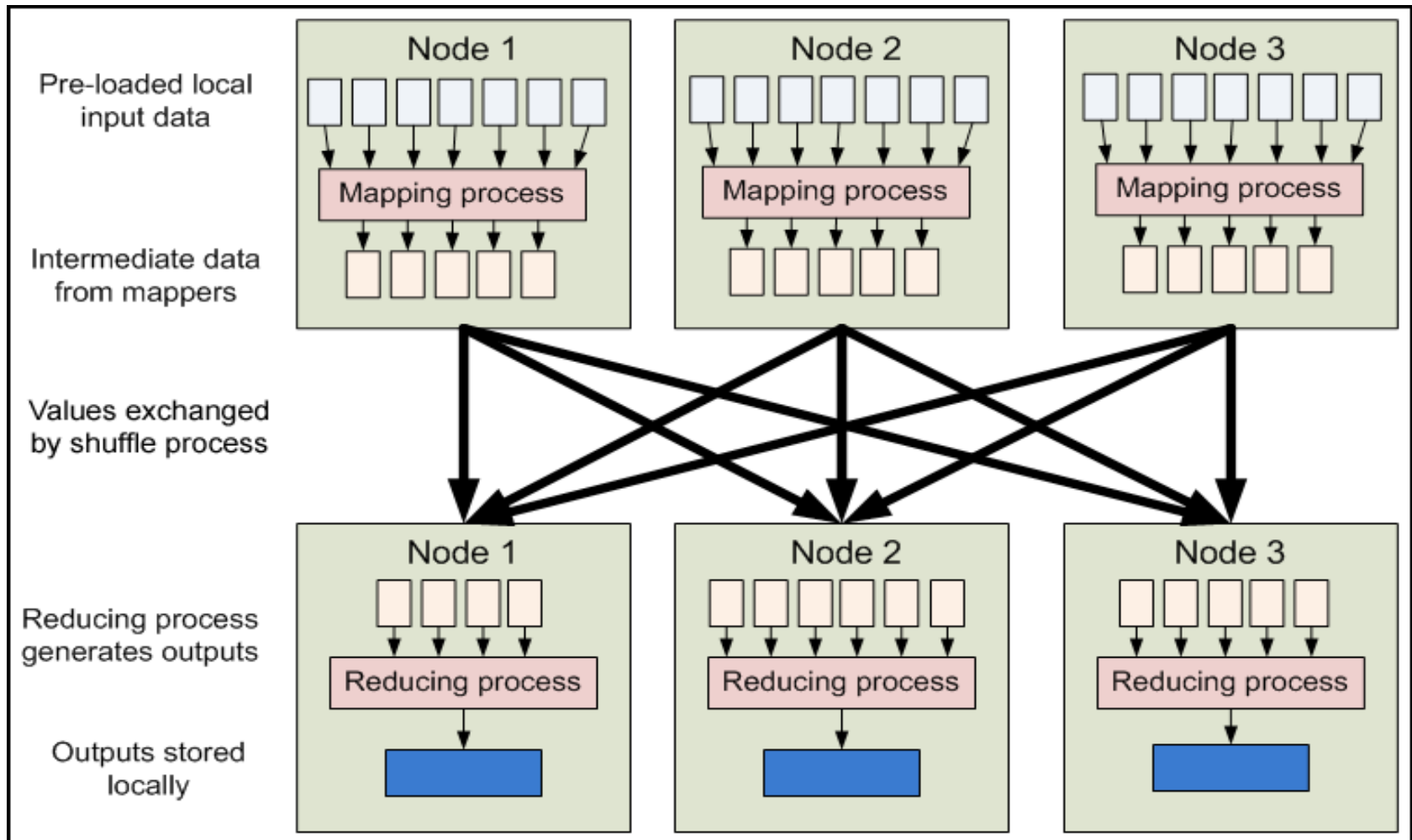
MapReduce



MapReduce - What?

- ▶ MapReduce is a programming model for efficient distributed computing
- ▶ It works like a Unix pipeline
 - ▶ `cat input | grep | sort | uniq -c | cat > output`
 - ▶ **Input** | **Map** | Shuffle & Sort | **Reduce** | **Output**
- ▶ Efficiency from
 - ▶ Streaming through data, reducing seeks
 - ▶ Pipelining
- ▶ A good fit for a lot of applications
 - ▶ Log processing
 - ▶ Web index building

MapReduce - Dataflow



MapReduce - Features

- ▶ Fine grained Map and Reduce tasks
 - ▶ Improved load balancing
 - ▶ Faster recovery from failed tasks
- ▶ Automatic re-execution on failure
 - ▶ In a large cluster, some nodes are always slow or flaky
 - ▶ Framework re-executes failed tasks
- ▶ Locality optimizations
 - ▶ With large data, bandwidth to data is a problem
 - ▶ Map-Reduce + HDFS is a very effective solution
 - ▶ Map-Reduce queries HDFS for locations of input data
 - ▶ Map tasks are scheduled close to the inputs when possible

Word Count Example

- ▶ Mapper

- ▶ Input: value: lines of text of input
- ▶ Output: key: word, value: 1

- ▶ Reducer

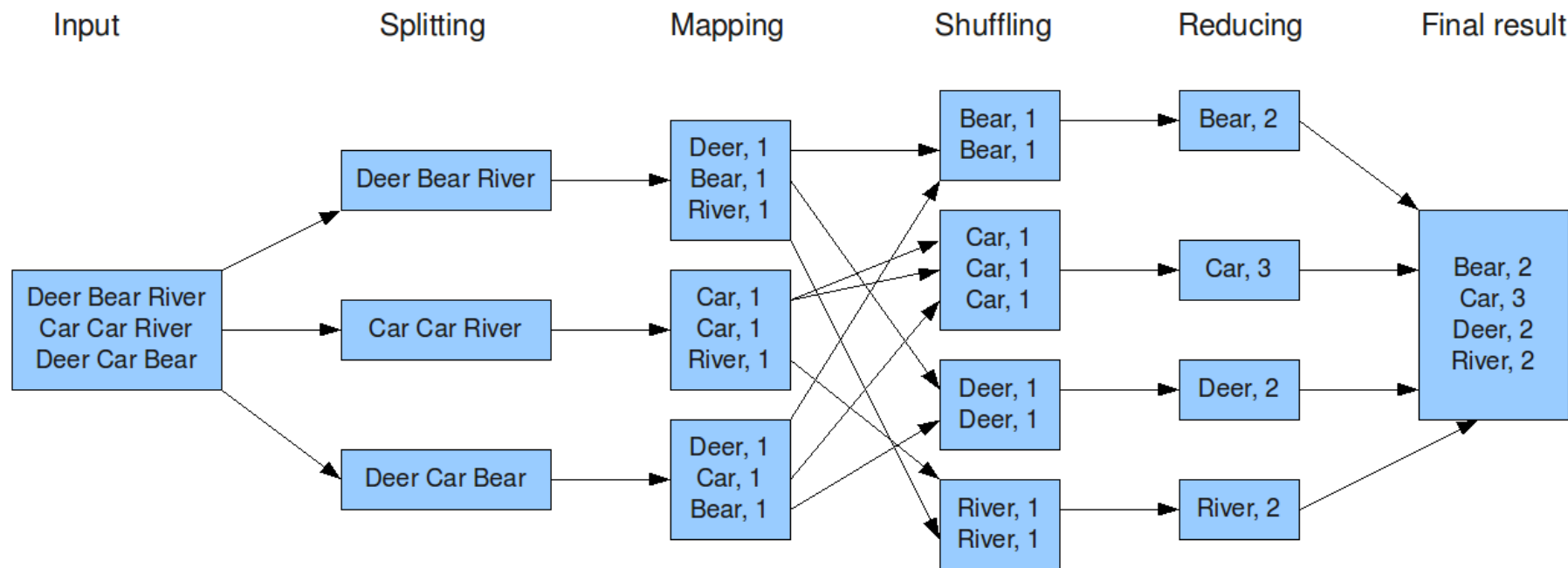
- ▶ Input: key: word, value: set of counts
- ▶ Output: key: word, value: sum

- ▶ Launching program

- ▶ Defines this job
- ▶ Submits job to cluster

Word Count Dataflow

The overall MapReduce word count process



Word Count Mapper

```
public static class Map extends MapReduceBase implements
    Mapper<LongWritable,Text,Text,IntWritable> {
    private static final IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public static void map(LongWritable key, Text value,
        OutputCollector<Text,IntWritable> output, Reporter reporter) throws
        IOException {
        String line = value.toString();
        StringTokenizer = new StringTokenizer(line);
        while(tokenizer.hasNext()) {
            word.set(tokenizer.nextToken());
            output.collect(word,one);
        }
    }
}
```

Word Count Reducer

```
public static class Reduce extends MapReduceBase implements
    Reducer<Text,IntWritable,Text,IntWritable> {
    public static void map(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));
    }
}
```

Word Count Example

- ▶ Jobs are controlled by configuring *JobConfs*
- ▶ JobConfs are maps from attribute names to string values
- ▶ The framework defines attributes to control how the job is executed
 - ▶ `conf.set("mapred.job.name", "MyApp");`
- ▶ Applications can add arbitrary values to the JobConf
 - ▶ `conf.set("my.string", "foo");`
 - ▶ `conf.set("my.integer", 12);`
- ▶ JobConf is available to all tasks

Hadoop Installation

Single Node (Pseudo Distributed Mode)

Steps to Follow

- ▶ Prerequisite

- ▶ Ubuntu 14.04 / 15.04 LTS 64-Bit Machines

- ▶ Login as root/sudo user

- Linux > sudo apt-get update

- ▶ Download the JDK1.8 tar.gz file from the following URL

- <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

- ▶ Untar the file using the following command,

- Linux > tar xzf jdk-8u45-linux-x64.tar.gz

- ▶ Move the java to /usr/local/java

- Linux > mv jdk1.8.0_45 /usr/local/java

-Contd,

- ▶ Set the Path and ClassPath Variables in ~/.bashrc

```
Linux > gedit ~/.bashrc
```

- ▶ Add the following line in the editor, save the file and exit.

```
export JAVA_HOME=/usr/local/java  
export PATH=$JAVA_HOME/bin:${PATH}  
export CLASSPATH=$JAVA_HOME/lib
```

- ▶ Linux> source ~/.bashrc or . ~/.bashrc
- ▶ Verify JAVA is Installed or not using this Command

```
Linux> java -version
```

-
- ▶ ³² make sure that you can see the JAVA version

-Contd, **Installing Secure Shell**

- ▶ Linux> sudo apt-get install ssh
- ▶ Linux> sudo apt-get install rsync
- ▶ **Best Practice,**
 - ▶ Create a new user hadoop or hduser for running hadoop
 - ▶ Create and Setup SSH Certificates (Setup passphraseless ssh)
 - ▶ To enable password-less login, generate a new SSH key with an empty passphrase:
 - ▶ Use Hadoop User(hduser/hadoop):
- ▶ Linux>ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
- ▶ Linux>cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
- ▶ Verify SSH,
 - ▶ Linux > ssh localhost
 - ▶ Make sure that you are able to connect localhost without the password.

Now, You are Ready to Install Hadoop 😊

▶ Fetch Hadoop (Stable Version)

- ▶ Linux > `wget http://apache.tradefair.com/pub/hadoop/common/current/hadoop-2.6.0.tar.gz`
- ▶ Extract File,
- ▶ Linux> `tar xzf hadoop-2.6.0.tar.gz`
- ▶ Move the hadoop to local dir,
- ▶ Linux> `mv hadoop-2.6.0 /usr/local/hadoop`

▶ Add the JAVA_HOME to hadoop-env.sh file

- ▶ Linux> `gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh`
- ▶ Locate `java_home` and set
- ▶ `JAVA_HOME=/usr/local/java`
- ▶ Now, save and Exit the file.

-Contd,

► Add the following lines to ~/.bashrc

```
► Linux> gedit ~/.bashrc
#HADOOP VARIABLES START
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
#HADOOP VARIABLES END
```

Now, save the file and Exit.

```
Linux> . ~/.bashrc
```

-Contd,

- ▶ Add the following `<property>` tag to `core-site.xml`

```
Linux> gedit /usr/local/hadoop/etc/hadoop/core-site.xml
```

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
</property>
```

Now, save the file and Exit.

- ▶ Add the following `<property>` tags to `yarn-site.xml`

```
Linux> gedit /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

```
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

Now, save the file and Exit.

-Contd,

- ▶ Add the following <property> tag to core-site.xml

- ▶ Copy the MapRed-site.xml.template file to MapRed-site.xml
- ▶ Linux>cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
- ▶ Linux> gedit /usr/local/hadoop/etc/hadoop/mapred-site.xml

```
<property>  
    <name>mapreduce.framework.name</name>  
    <value>yarn</value>
```

```
</property> Now, save the file and Exit.
```

- ▶ Add the following <property> tags to hdfs-site.xml

- ▶ Create Namenode and Datanode directories
- ▶ Linux> mkdir -p /usr/local/hadoop_store/hdfs
- ▶ Linux> gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```
<property>  
    <name>dfs.replication</name>  
    <value>1</value>  
</property>  
<property>  
    <name>dfs.namenode.name.dir</name>  
    <value>file:/usr/local/hadoop_store/hdfs/namenode</value>  
</property>  
<property>  
    <name>dfs.datanode.data.dir</name>  
    <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
```

```
</property> Now, save the file and Exit.
```

Cheers, two more steps to go.

- ▶ Replace `sasz:` with your hadoop users(`hduser/hadoop`) to be the owner of the folder

```
Linux> sudo chown sasz:sasz -R /usr/local/hadoop
```

```
Linux> sudo chown sasz:sasz -R /usr/local/hadoop_store
```

also give the folder the full permission

```
Linux> sudo chmod -R 777 /usr/local/hadoop
```

```
Linux> sudo chmod -R 777 /usr/local/hadoop_store
```

- ▶ Format your HDFS, make sure you have logged in as `hadoop/hduser` user.

```
Linux> hdfs namenode -format
```

- ▶ Start/Stop the Hadoop Cluster.

```
Linux> start-all.sh or stop-all.sh
```

Congratulations!!, You have successfully done installation of Hadoop

- ▶ Access the User Interfaces
- ▶ ResourceManager @- <http://localhost:8088/>
- ▶ NameNode @- <http://localhost:50070/>

Thank You!

