What is Hadoop?

- Hadoop The Elephant
- Hadoop is an open source implementation of MapReduce
- Native API is written in Java
- Modular way of writing Mappers and Reducers
- Parallelization on distributed machines all done for you!
Practical Applications

- Searching
- Log Processing
- Data Warehouses
- Video and Image Analysis

Adobe
- Adobe uses Hadoop in various areas from social services to structured data storage and processing. They have about 30 nodes running Hadoop in clusters ranging from 5 to 14 nodes.

Careers
- Careers uses Hadoop to process company and job data and run machine learning algorithms for their recommendation engine.

EBay
- EBay has 532 node cluster (8*532 cores, 5.3 PB) Heavy usage of Java MapReduce, Apache Pig and Apache Hive for search optimization and research.

Facebook
- Facebook uses Hadoop to store copies of internal log and dimension data sources and use it as a source for reporting/analytics and machine learning.
Cluster

- Hadoop uses **Clusters**
  - Large number (100-100,000) of servers, i.e. nodes
  - Connected by a high speed network
  - Many racks, containing some number of nodes
Why Hadoop?

• Economical
  ‣ No License
  ‣ Open Source Framework

• Flexible
  ‣ Schema-Less, any type of data from any number of sources

• Smart
  ‣ Optimized Compression
  ‣ Query Expression
Why Hadoop?

- Scalable
  - New nodes can be added and removed without changing anything

- Reliable
  - When a node is lost, the task is retired and the continues on without missing a beat!!
    - Replication

- Helps Address Big Data 3 V’s
  - What are the three V’s?
Why Hadoop?

• Scalable
  ‣ New nodes can be added and removed without changing anything

• Reliable
  ‣ When a node is lost, the task is retired and the continues on without missing a beat!!
  ‣ Replication

• Helps Address Big Data 3 V's
  ‣ What are the three V's?
  ‣ Volume, Velocity, Variety
Hadoop Components

• Hadoop can be broken into two primary components
  ‣ Hadoop Distributed File System
  ‣ MapReduce

Hadoop

HDFS
Stores data in “Chunks” of 64 MB each

MapReduce
Processes the data in a massive parallel manner
Hadoop Distributed File System (HDFS)

- 64MBChunks
- Used to manage failure
- Improve performance

• Why might it improve performance?
Hadoop Distributed File System (HDFS)

- Replication Factor

- Replication factor of 3 here.
HDFS Components

Name Node

- Contains metadata - where chunks are stored, what nodes are online, who has what task, etc.
HDFS Components

Name Node

- The Heart Beat tells the name node that a datanode is alive.
HDFS Components

Name Node

- Datanodes update the name node on what chunks they contain.
HDFS Components

DataNode

- Contains chunks of data
HDFS Components

DataNode

- Processes tasks
HDFS Components

DataNode

- May get assigned more tasks/data as time goes on.
HDFS Components

**Rack**

- Contains data nodes
- Replication between racks or nodes
Parallel Data Processing in a Cluster

• Data Partitioning
  ‣ Partition (or repartition) data across nodes
  ‣ Compute output at each node
  ‣ Aggregate the results

• Map Reduce (Implementing the Data Partitioning)
  ‣ Programming model and framework that supports parallel data processing
  ‣ Proposed by Google researchers, natural model for many problems
  ‣ Simple data model
Map reduce Flow

- Data is declared as input
- Map splits into <key, value> pairs
- Sorted on <keys>
- Reduce performs operation on <key, value> pairs
Map Reduce Algorithm

• Input
  ‣ Set of records $X$
  ‣ User defined function $f$, called a mapper
  ‣ User defined function $r$, called a reducer

• Mapper:
  ‣ Apply $f$ to each record in $X$, which produces $<key, value>$ pairs
  ‣ Similar to Group By in SQL

• Reducer:
  ‣ Collect all values paired with each distinct key
  ‣ Apply $r$ to the key and values
  ‣ Similar to aggregate functions in SQL
Map Reduce

- Input is split into chunks
Map Reduce

- Mapped into <Key, Value> Pairs
Map Reduce

- Shuffled based on `<key>`
Map Reduce

- Each reducer is assigned a key(s).
Map Reduce

- Final output is combined from all reducers
Workflow of Hadoop Cluster

Why would the number of reducers matter?

Typical Components of a Hadoop Cluster

Client → JobTracker → NameNode → DataNode 1 → TaskTracker

Client → JobTracker → NameNode → DataNode 2 → TaskTracker

Client → JobTracker → NameNode → DataNode 3 → TaskTracker

Client → JobTracker → NameNode → DataNode 4 → TaskTracker

Client → JobTracker → NameNode → DataNode 5 → TaskTracker

Files: A, B, C
Mapper
Reducer

rohitmenon.com
Map Class

```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);

    @Override
    public void map(Object key, Text value, Context output) throws IOException,
                    InterruptedException {

        //Split on whitespace
        String[] words = value.toString().split(" ");
        //Get First Word
        output.write(new Text(words[0]), one);
    }
}
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);

    @Override
    public void map(Object key, Text value, Context output) throws IOException,
            InterruptedException {

        //Split on whitespace
        String[] words = value.toString().split(" ");
        //Get First Word
        output.write(new Text(words[0]), one);
    }
}
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);

    @Override
    public void map(Object key, Text value, Context output) throws IOException,
                    InterruptedException {
        //Split on whitespace
        String[] words = value.toString().split(" ");
        //Get First Word
        output.write(new Text(words[0]), one);
    }
}
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);

    @Override
    public void map(Object key, Text value, Context output) throws IOException, InterruptedException {

        //Split on whitespace
        String[] words = value.toString().split(" ");
        //Get First Word
        output.write(new Text(words[0]), one);
    }
}
Map Class

```java
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class WordCountMapper extends Mapper<Object, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);

    @Override
    public void map(Object key, Text value, Context output) throws IOException, InterruptedException {

        //Split on whitespace
        String[] words = value.toString().split(" ");
        //Get First Word
        output.write(new Text(words[0]), one);
    }
}
```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context output) throws IOException, InterruptedException {
        int voteCount = 0;
        for(IntWritable value : values){
            voteCount += value.get();
        }
        output.write(key, new IntWritable(voteCount));
    }
}
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context output) 
    throws IOException, InterruptedException {
        int voteCount = 0;
        for(IntWritable value : values) {
            voteCount += value.get();
        }
        output.write(key, new IntWritable(voteCount));
    }
}
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context output)
            throws IOException, InterruptedException {
        int voteCount = 0;
        for (IntWritable value: values) {
            voteCount += value.get();
        }
        output.write(key, new IntWritable(voteCount));
    }
}
Main Class

```java
public class WordCountMain17 extends Configured implements Tool{

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCountMain17(), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("usage: [input] [output]");
            System.exit(-1);
        }

        Job job = Job.getInstance(new Configuration());
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

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

        job.setJarByClass(WordCountMain17.class);

        job.submit();
        return 0;
    }
}
```
public class WordCountMain17 extends Configured implements Tool{
    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCountMain17(), args);
        System.exit(res);
    }
    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("usage: [input] [output]");
            System.exit(-1);
        }
        Job job = Job.getInstance(new Configuration());
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setJarByClass(WordCountMain17.class);
        job.submit();
        return 0;
    }
}
public class WordCountMain17 extends Configured implements Tool{

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCountMain17(), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("usage: [input] [output]);
            System.exit(-1);
        }

        Job job = Job.getInstance(new Configuration());
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

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

        job.setJarByClass(WordCountMain17.class);

        job.submit();
        return 0;
    }
}
public class WordCountMain17 extends Configured implements Tool{

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCountMain17(), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("usage: [input] [output]");
            System.exit(-1);
        }

        Job job = Job.getInstance(new Configuration());
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

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

        job.setJarByClass(WordCountMain17.class);
        job.submit();
        return 0;
    }
}
public class WordCountMain17 extends Configured implements Tool{

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCountMain17(), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("usage: [input] [output]");
            System.exit(-1);
        }

        Job job = Job.getInstance(new Configuration());
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

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

        job.setJarByClass(WordCountMain17.class);

        job.submit();
        return 0;
    }
}
public class WordCountMain17 extends Configured implements Tool{

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCountMain17(), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("usage: [input] [output]");
            System.exit(-1);
        }

        Job job = Job.getInstance(new Configuration());
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        TextInputFormat.setInputPaths(job, new Path(args[0]));
        TextOutputFormat.setOutputPath(job, new Path(args[1]));
        job.setJarByClass(WordCountMain17.class);
        job.submit();
        return 0;
    }
}
public class WordCountMain17 extends Configured implements Tool{

    public static void main(String[] args) throws Exception {
        int res = ToolRunner.run(new Configuration(), new WordCountMain17(), args);
        System.exit(res);
    }

    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 2) {
            System.out.println("usage: [input] [output]");
            System.exit(-1);
        }

        Job job = Job.getInstance(new Configuration());
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

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

        job.setJarByClass(WordCountMain17.class);
        job.submit();
        return 0;
    }
}
Creating the Jar

- Import Hadoop Jars
  - >Include External Jar
  - >/common and /mapreduce

- Package with external jars
Demo

• Create input directory
  ‣ hdfs dfs -mkdir /user/<yourusername>/<whateveryouwant>

• Move input files from encs server to hfs
  ‣ hdfs dfs -put <localinputfile> /user/<yourusername>/<whateveryoucreated>/

• Run
  ‣ hadoop jar example.jar <inputdirectory> <outputdirectory>
  ‣ Do NOT create output directory before hand, hadoop complains
Hadoop Streaming

- Comes with the Hadoop distribution
- Allows for Map Reduce jobs on any executable
  - Uses STDIN and STDOUT, executables need to read/output to these in order for this to work
C++ Specifics

- Need to compile on the server
- Executable permissions on executables
- Can actually test locally if you want
  
  cat input.txt | ./mapper | sort | reducer > output.txt
Command to Run

```
hadoop jar /opt/hadoop/hadoop-2.7.3/share/hadoop/tools/lib/hadoop-streaming-2.7.3.jar \
  -input /user/<yourusername>/<yourinputdirectory>/ \
  -output /user/<yourusername>/<outputdirectory>/ \
  -file ./mapper -mapper ./mapper \
  -file ./reducer -reducer ./reducer
```

- This is where the streaming file is
Command to Run

```
hadoop jar /opt/hadoop/hadoop-2.7.3/share/hadoop/tools/lib/hadoop-streaming-2.7.3.jar \
- input /user/<yourusername>/<yourinputdirectory>/ \
- output /user/<yourusername>/<outputdirectory>/ \
- file ./mapper -mapper ./mapper \
- file ./reducer -reducer ./reducer
```

- Define your input and output files here
  - They must be on HDFS still
  - Cannot already have the output directory
Command to Run

```
hadoop jar /opt/hadoop/hadoop-2.7.3/share/hadoop/tools/lib/hadoop-streaming-2.7.3.jar \
-input /user/<yourusername>/<yourinputdirectory>/ \
-output /user/<yourusername>/<outputdirectory>/ \
-file ./mapper -mapper ./mapper \
-file ./reducer -reducer ./reducer
```

- -file makes sure the executable is on every node
- Declare your mapper and reducer
#include "mapper.hpp"
#include <iostream>
#include <string>
#include <cstdlib>

using namespace std;

int main()
{
    string line;

    while(getline(cin,line)){
        // process input here
        // this input reads a single line from the input
        // so you will have to parse it out and then output it to stdout.
    }
}
Reducer

```cpp
#include "reducer.hpp"
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string line;
    getline(cin, line);
    // process input here
    // input is coming from the mapper, so it depends on how you output it
}
```
C++ Demo

• Away we go!
Map Reduce Components

- Nodes report status of jobs
Map Reduce Components

- Clients interact with job tracker to find status.
Map Reduce Components

- Can move tasks if a failure occurs
Multiple Mappers and Reducers

```java
//conf.setMapperClass(Map.class);
conf.setReducerClass(Reduce.class);
conf.setInputFormat(TextInputFormat.class);
conf.setOutputFormat(TextOutputFormat.class);
MultipleInputs.addInputPath(conf, new Path(args[0]), TextInputFormat.class, Map.class);
MultipleInputs.addInputPath(conf, new Path(args[1]), TextInputFormat.class, Map.class);
FileOutputFormat.setOutputPath(conf, new Path(intermediateFile));
conf.set("mapred.job.priority", JobPriority.VERY_HIGH.toString());
JobClient.runJob(conf);
getAvg(intermediateFile, args[2]);
Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path path = new Path(intermediateFile);
boolean isDeleted = hdfs.delete(path, true);```
Multiple Mappers and Reducers

```java
//conf.setMapperClass(Map.class);
cnf.setReducerClass(Reduce.class);

cnf.setInputFormat(TextInputFormat.class);
cnf.setOutputFormat(TextOutputFormat.class);

MultipleInputs.addInputPath(conf, new Path(args[0]), TextInputFormat.class, MapAge.class);
MultipleInputs.addInputPath(conf, new Path(args[1]), TextInputFormat.class, Map.class);

FileOutputFormat.setOutputPath(conf, new Path(intermediateFile));

cnf.set("mapred.job.priority", JobPriority.VERY_HIGH.toString());

JobClient.runJob(conf);

getAvg(intermediateFile, args[2]);

Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path path = new Path(intermediateFile);
boolean isDeleted = hdfs.delete(path, true);```
Multiple Mappers and Reducers

```java
//conf.setMapperClass(Map.class);
conf.setReducerClass(Reduce.class);

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

MultipleInputs.addInputPath(conf, new Path(args[0]), TextInputFormat.class, Map.class);
MultipleInputs.addInputPath(conf, new Path(args[1]), TextInputFormat.class, Map.class);

FileOutputFormat.setOutputPath(conf, new Path(intermediateFile));

conf.set("mapred.job.priority", JobPriority.VERY_HIGH.toString());

JobClient.runJob(conf);

getAvg(intermediateFile, args[2]);

Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path path = new Path(intermediateFile);
boolean isDeleted = hdfs.delete(path, true);```
Multiple Mappers and Reducers
Multiple Mappers and Reducers

```java
//conf.setMapperClass(Map.class);
conf.setReducerClass(Reduce.class);

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

MultipleInputs.addInputPath(conf, new Path(args[0]), TextInputFormat.class, MapAge.class);
MultipleInputs.addInputPath(conf, new Path(args[1]), TextInputFormat.class, Map.class);

FileOutputFormat.setOutputPath(conf, new Path(intermediateFile));

conf.set("mapred.job.priority", JobPriority.VERY_HIGH.toStrin());

JobClient.runJob(conf);

getAvg(intermediateFile, args[2]);

Configuration config = new Configuration();
FileSystem hdfs = FileSystem.get(config);
Path path = new Path(intermediateFile);
boolean isDeleted = hdfs.delete(path, true);
```
Multiple Mappers and Reducers

```java
conf.setMapperClass(MapAvg.class);
conf.setReducerClass(ReduceAvg.class);

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

FileInputFormat.setInputPaths(conf, new Path(inFile));
FileOutputFormat.setOutputPath(conf, new Path(outFile));

conf.set("mapred.reduce.tasks", "1");
conf.set("mapred.job.priority", JobPriority.VERY_HIGH.toString());

JobClient.runJob(conf);
```
Multiple Mappers and Reducers

```java
conf.setMapperClass(MapAvg.class);
conf.setReducerClass(ReduceAvg.class);

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

FileInputFormat.setInputPaths(conf, new Path(inFile));
FileOutputFormat.setOutputPath(conf, new Path(outFile));

conf.set("mapred.reduce.tasks", "1");
conf.set("mapred.job.priority", JobPriority.VERY_HIGH.toString());

JobClient.runJob(conf);
```
Configuration Files

• Hadoop has many configuration files
  ▸ All the values have defaults, but some you will want to change

• Found in etc/hadoop/
  ▸ Depends on how you installed Hadoop

• The files on the server are shared, so you don’t have any write permissions
  ▸ Still important to know how to change though
core-site.xml


```xml
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/usr/local/Cellar/hadoop/hdfs/tmp</value>
    <description>base for name, data, name secondary, etc</description>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```
mapred-site.xml


```xml
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>localhost:9010</value>
  </property>
</configuration>
```
HDFS-site.xml

- https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml

```xml
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```
All Finished!

Questions?