MapReduce and Apache Hadoop

Vahid Ghadakchi
Some Slides Adapted from Nitin Subramanian
Motivation

• Want to process lots of data
  - Google’s index as of 2016 is 100 PB
• Want to distribute through thousands of machines
• Want to make this easy
MapReduce

- Automatic parallelization and distribution
- Fault-tolerant
- Provides monitoring tools
- Clean abstraction / API
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.
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!
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?
  - Volume, Velocity, Variety
Hadoop Components

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

- Stores data in “Chunks” of 64 MB each
- Processes the data in a massive parallel manner
Hadoop Distributed File System (HDFS)

- 64MB Chunks
  - Used to manage failure
  - Improve performance

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

- Replication Factor

- Replication factor of 3 here.
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 $\langle\text{key},\text{value}\rangle$ 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?
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);
    }
}
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 extendsMapper<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));
    }
}
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);

        FileInputStream.setInputPaths(job, new Path(args[0]));
        FileOutputStream.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;
    }
}
Demo

- ssh hadoop-master.engr.oregonstate.edu

- 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>
  - View Output
  - Make sure there is no output dir before running
Demo

- View Status
  - [http://hadoop-master.engr.oregonstate.edu:8088](http://hadoop-master.engr.oregonstate.edu:8088)

- View Output
  - `hdfs dfs -cat <path to your folder>/output/part-r-00000`
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-input-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-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-input-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
class WordCountMapper : public HadoopPipes::Mapper {
public:
    // constructor: does nothing
    WordCountMapper( HadoopPipes::TaskContext& context ) {
    }

    // map function: receives a line, outputs (word,"1")
    // to reducer.
    void map( HadoopPipes::MapContext& context ) {
        //--- get line of text ---
        string line = context.getInputValue();

        //--- split it into words ---
        vector< string > words =
            HadoopUtils::splitString( line, " ");

        //--- emit each word tuple (word, "1") ---
        for ( unsigned int i=0; i < words.size(); i++ ) {
            context.emit( words[i], HadoopUtils::toString( 1 ) );
        }
    }
};
// reduce function
void reduce( HadoopPipes::ReduceContext& context ) {
    int count = 0;

    //--- get all tuples with the same key, and count their numbers ---
    while ( context.nextValue() ) {
        count += HadoopUtils::toInt( context.getInputValue() );
    }

    //--- emit (word, count) ---
    context.emit(context.getInputKey(), HadoopUtils::toString( count ));
}

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
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](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>
```
Using Python to Stream Mapper

```python
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s	%s' % (word, 1)
```
Using Python to Stream Mapper

```python
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```
Using Python to Stream Mapper

```python
#!/usr/bin/env python
import sys
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```
Using Python to Stream Mapper

```python
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        print '%s\t%s' % (word, 1)
```
Using Python to Stream Mapper

```python
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()
    words = line.split()

for word in words:
    print '%s\t%s' % (word, 1)
```
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()

    word, count = line.split(' \t')

    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)

        current_word = word
        current_count = count

# Since we are only printing the words once we encounter a new one
# we need to be sure to output the last word
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
Using Python to Stream Reducer

```python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    # remove extra newlines and spaces before and after line
    line = line.strip()

    # split on tab
    word, count = line.split(' \	')

    # the count comes in as a str, so need to convert to an integer
    count = int(count)

    # still on the same word, so increase the count
    if current_word == word:
        current_count += count
    else:
        # we got a new word, so output the one we are storing
        if current_word:
            print '%s: %s' % (current_word, current_count)

        # update what the word and count are
        current_count = count
        current_word = word

    # since we are only printing the words once we encounter a new one
    # we need to be sure to output the last word
    if current_word == word:
        print '%s: %s' % (current_word, current_count)
```

Using Python to Stream Reducer

```python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    # remove extra newlines and spaces before and after line
    line = line.strip()

    # split on tab
    word, count = line.split(' \
')

    # the count comes in as a str, so need to convert to an integer
    count = int(count)

    # Still on the same word, so increase the count
    if current_word == word:
        current_count += count
    else:
        # We got a new word, so output the one we are storing
        if current_word:
            print '%s\t%s' % (current_word, current_count)

        # update what the word and count are
        current_count = count
        current_word = word

    # Since we are only printing the words once we encounter a new one
    # we need to be sure to output the last word
    if current_word == word:
        print '%s\t%s' % (current_word, current_count)
```
Using Python to Stream Reducer

```python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    # remove extra newlines and spaces before and after line
    line = line.strip()

    # Split on tab
    word, count = line.split('	')

    # the count comes in as a str, so need to convert to an integer
    count = int(count)

    # Still on the same word, so increase the count
    if current_word == word:
        current_count += count
    else:
        # We got a new word, so output the one we are storing
        if current_word:
            print '%s\t%s' % (current_word, current_count)

        # update what the word and count are
        current_count = count
        current_word = word

# Since we are only printing the words once we encounter a new one
# we need to be sure to output the last word
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```
Using Python to Stream Reduce

```python
from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

for line in sys.stdin:
    line = line.strip()
    word, count = line.split('	')
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word

if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

All Finished!

Questions?