MapReduce, Hadoop and Amazon AWS

Yasser Ganjisaffar

http://www.ics.uci.edu/~yganjisa

February 2011

What is Hadoop?

- A software framework that supports data-intensive distributed applications.
- It enables applications to work with **thousands of nodes** and **petabytes of data**.
- Hadoop was inspired by Google's MapReduce and Google File System (GFS).
- Hadoop is a top-level Apache project being built and used by a global community of contributors, using the Java programming language.
- Yahoo! has been the largest contributor to the project, and uses Hadoop extensively across its businesses.

Who uses Hadoop?

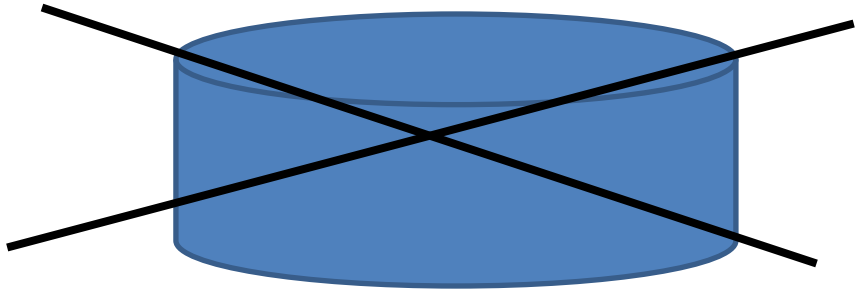
http://wiki.apache.org/hadoop/PoweredBy

Who uses Hadoop?

- Yahoo!
  - More than 100,000 CPUs in >36,000 computers.
- Facebook
  - Used in reporting/analytics and machine learning and also as storage engine for logs.
  - A 1100-machine cluster with 8800 cores and about 12 PB raw storage.
  - A 300-machine cluster with 2400 cores and about 3 PB raw storage.
  - Each (commodity) node has 8 cores and 12 TB of storage.

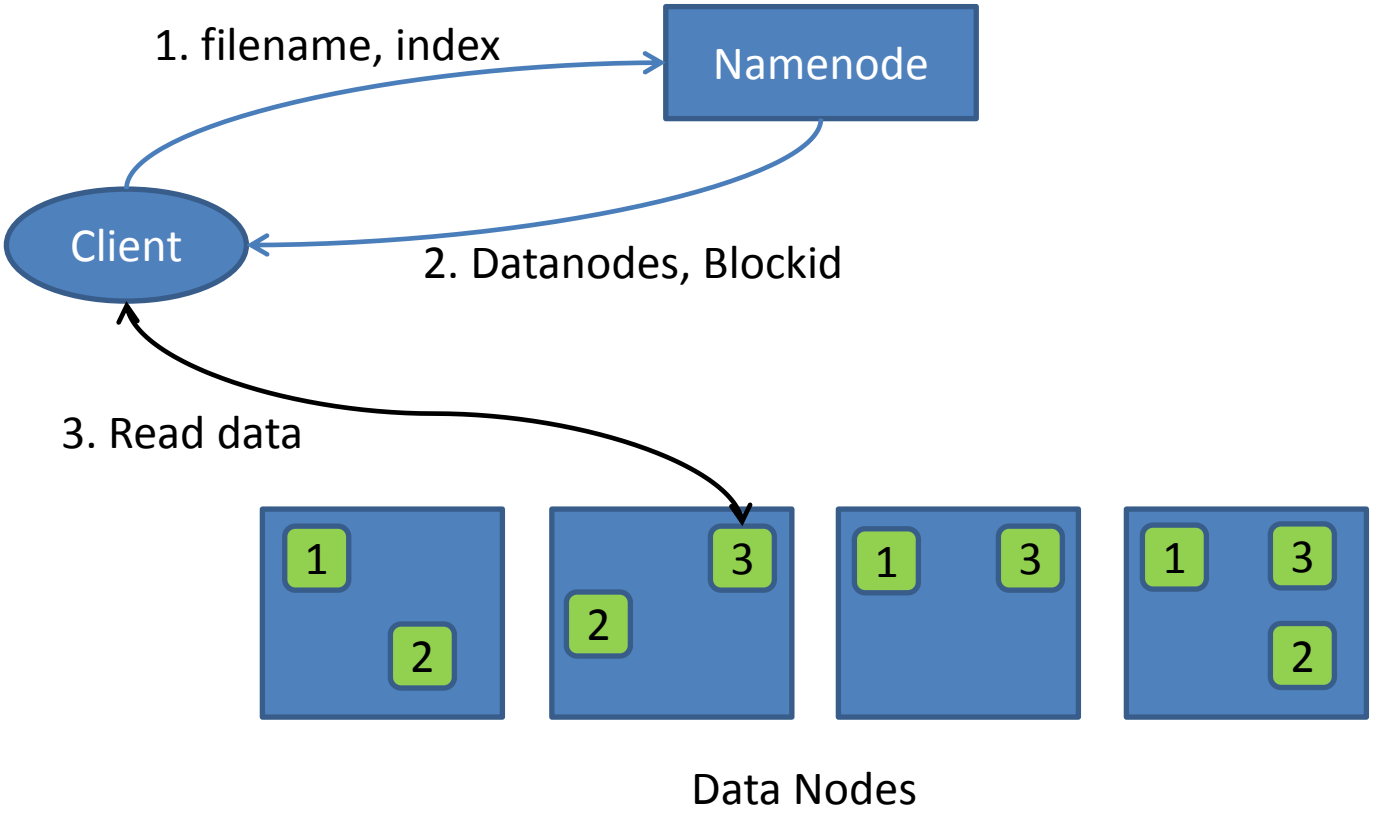
Very Large Storage Requirements

- Facebook has Hadoop clusters with 15 PB of raw storage (15,000,000 GB).
- No single storage can handle this amount of data.



- We need a large set of nodes each storing part of the data.

HDFS: Hadoop Distributed File System



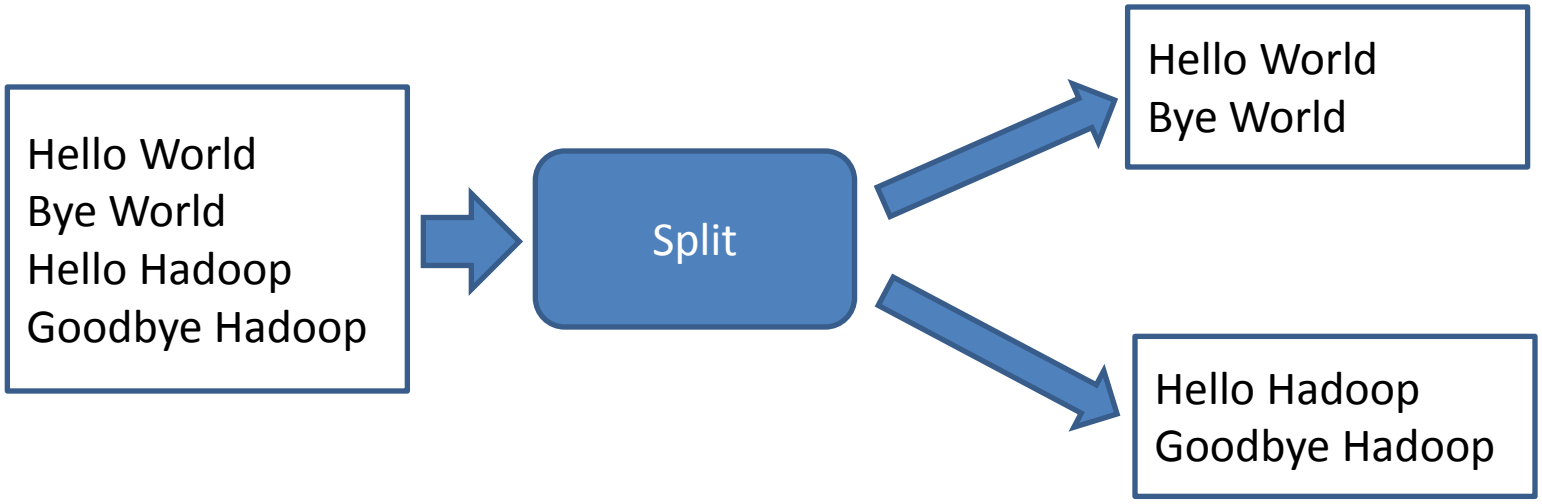
Terabyte Sort Benchmark

- http://sortbenchmark.org/
- Task: Sorting 100TB of data and writing results on disk (10^12 records each 100 bytes).

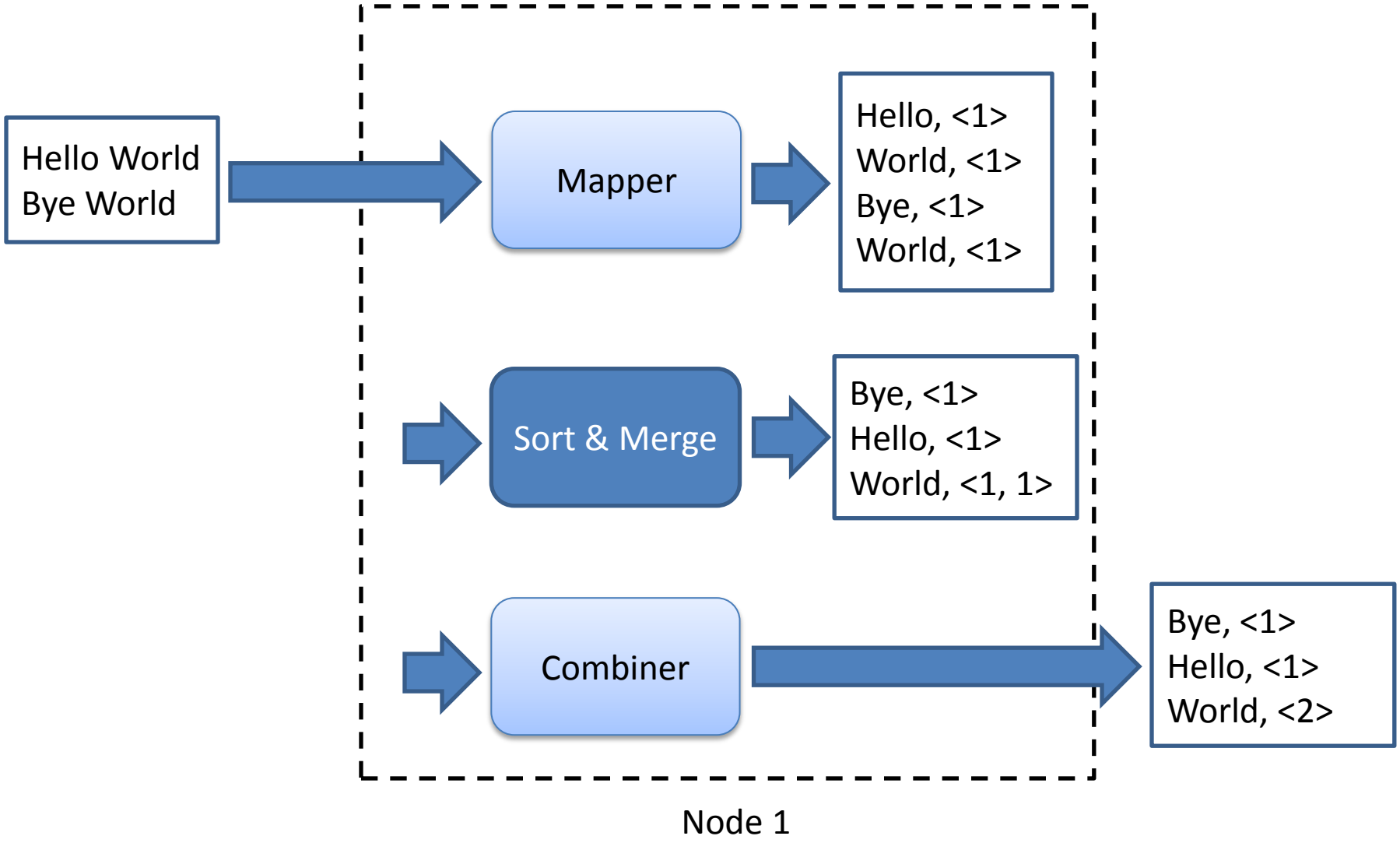
- Yahoo's Hadoop Cluster is the current winner:
  - **173 minutes**
  - 3452 nodes x (2 Quadcore Xeons, 8 GB RAM)

This is the first time that a **Java** program has won this competition.

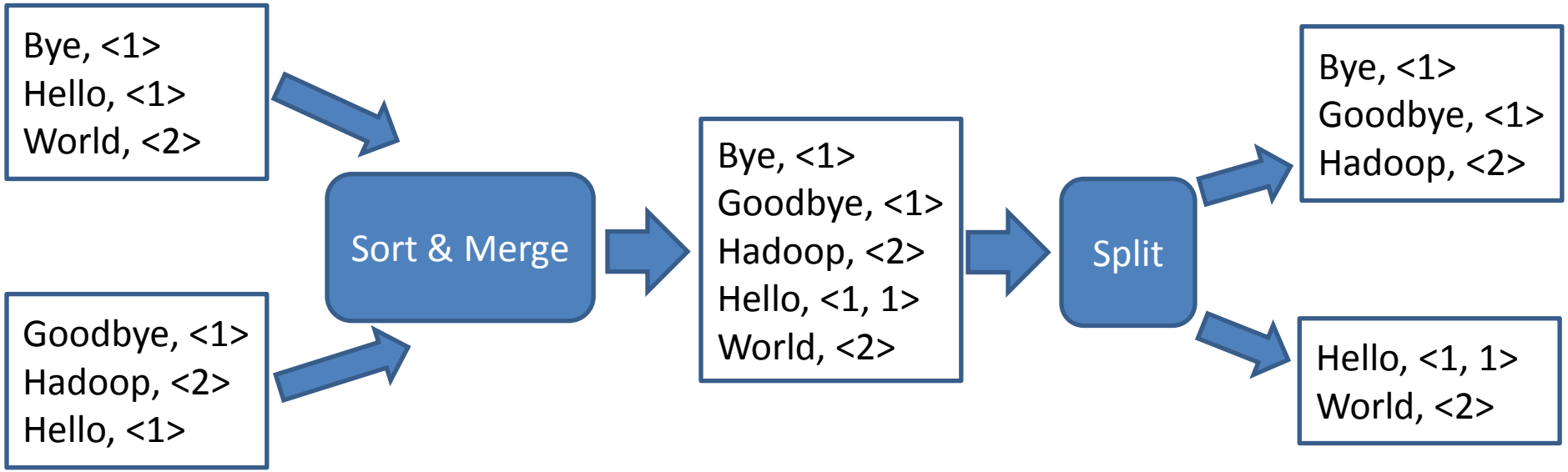
Counting Words by MapReduce



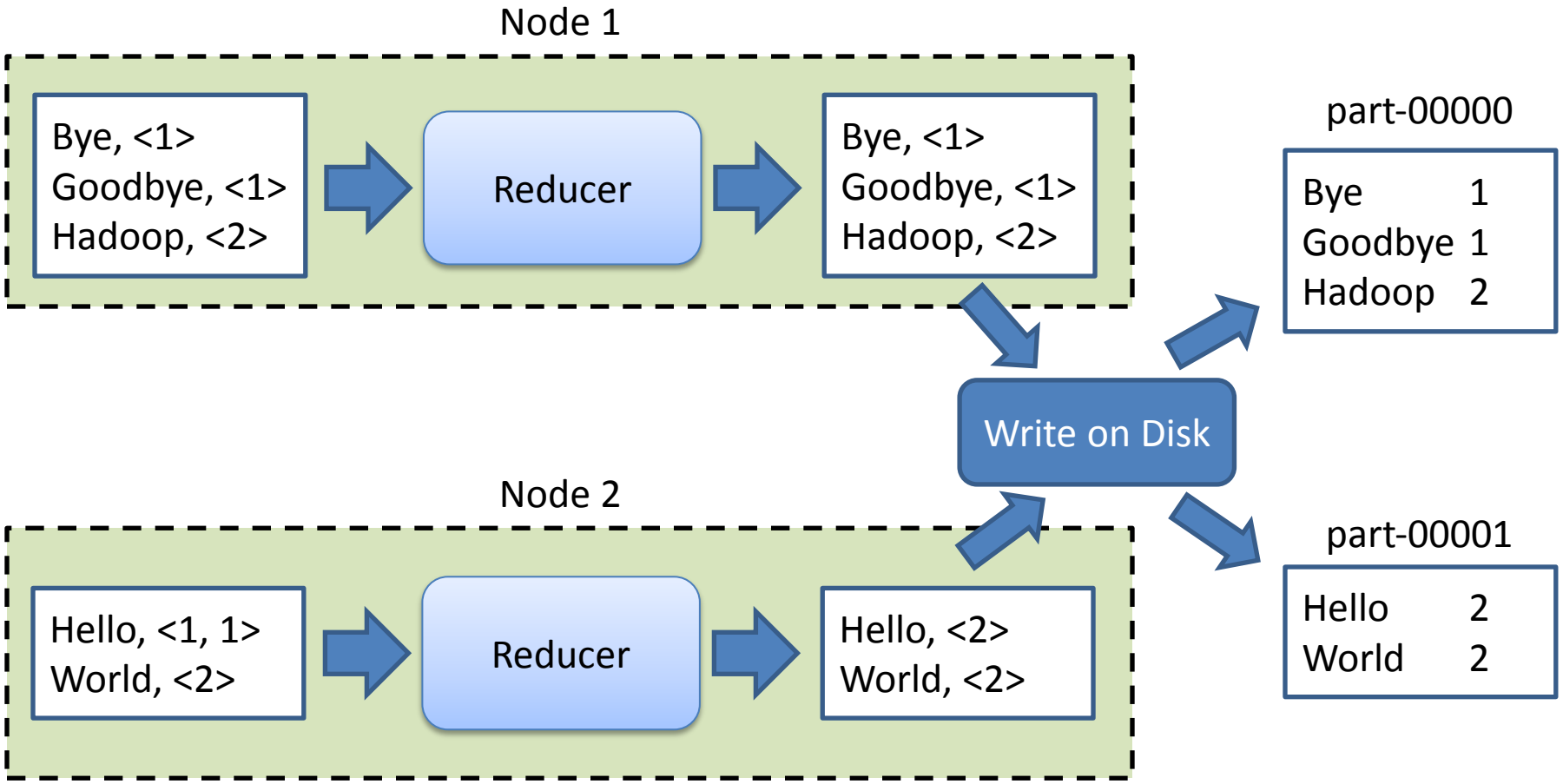
Counting Words by MapReduce



Counting Words by MapReduce



Counting Words by MapReduce



Writing Word Count in Java

- Download hadoop core (version 0.20.2):
  - http://www.apache.org/dyn/closer.cgi/hadoop/core/
- It would be something like:
  - hadoop-0.20.2.tar.gz
- Unzip the package and extract:
  - hadoop-0.20.2-core.jar
- Add this jar file to your project class path

Warning! Most of the sample codes on web are for older versions of Hadoop.

Word Count: Mapper

```
public class WordCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

    @Override
    public void map(LongWritable key, Text value, final Context context) throws IOException,
        InterruptedException {

        String line = value.toString().toLowerCase();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }

    }
}
```

Source files are available at: http://www.ics.uci.edu/~yganjisa/files/2011/hadoop-presentation/WordCount-v1-src.zip

Word Count: Reducer

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

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {

        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));

    }

}
```

Word Count: Main Class

```
public class WordCount extends Configured implements Tool {

    @Override
    public int run(String[] args) throws Exception {
        Configuration conf = new Configuration();

        Job job = new Job(conf, "Word Count");
        job.setJarByClass(WordCount.class);

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

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

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

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

        return (job.waitForCompletion(true) ? 0: 1);
    }

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

}
```

My Small Test Cluster

- 3 nodes
  - 1 master (ip address: 50.17.65.29)
  - 2 slaves
- Copy your jar file to master node:
  - Linux:
    - scp WordCount.jar john@50.17.65.29:WordCount.jar
  - Windows (you need to download pscp.exe):
    - pscp.exe WordCount.jar john@50.17.65.29:WordCount.jar
- Login to master node:
  - ssh john@50.17.65.29

Counting words in U.S. Constitution!

- Download text version:

  wget http://www.usconstitution.net/const.txt

- Put input text file on HDFS:

  hadoop dfs -put const.txt const.txt

- Run the job:

  hadoop jar WordCount.jar edu.uci.hadoop.WordCount const.txt word-count-result

Counting words in U.S. Constitution!

- List my files on HDFS:
  - Hadoop dfs -ls

```
john@ip-10-118-55-229:~$ hadoop dfs -ls
Found 2 items
-rw-r--r--   1 john supergroup      45119 2011-02-23 05:13 /user/john/const.txt
drwx------   - john supergroup          0 2011-02-23 05:54 /user/john/word-count-result
```

- List files in word-count-result folder:
  - Hadoop dfs -ls word-count-result/

```
john@ip-10-118-55-229:~$ hadoop dfs -ls word-count-result/
Found 1 items
-rw-r--r--   1 john supergroup      15644 2011-02-23 05:54 /user/john/word-count-result/part-r-00000
```

Counting words in U.S. Constitution!

- Downloading results from HDFS:

```
hadoop dfs -cat word-count-result/part-r-00000 > word-count.txt
```

- Sort and view results:

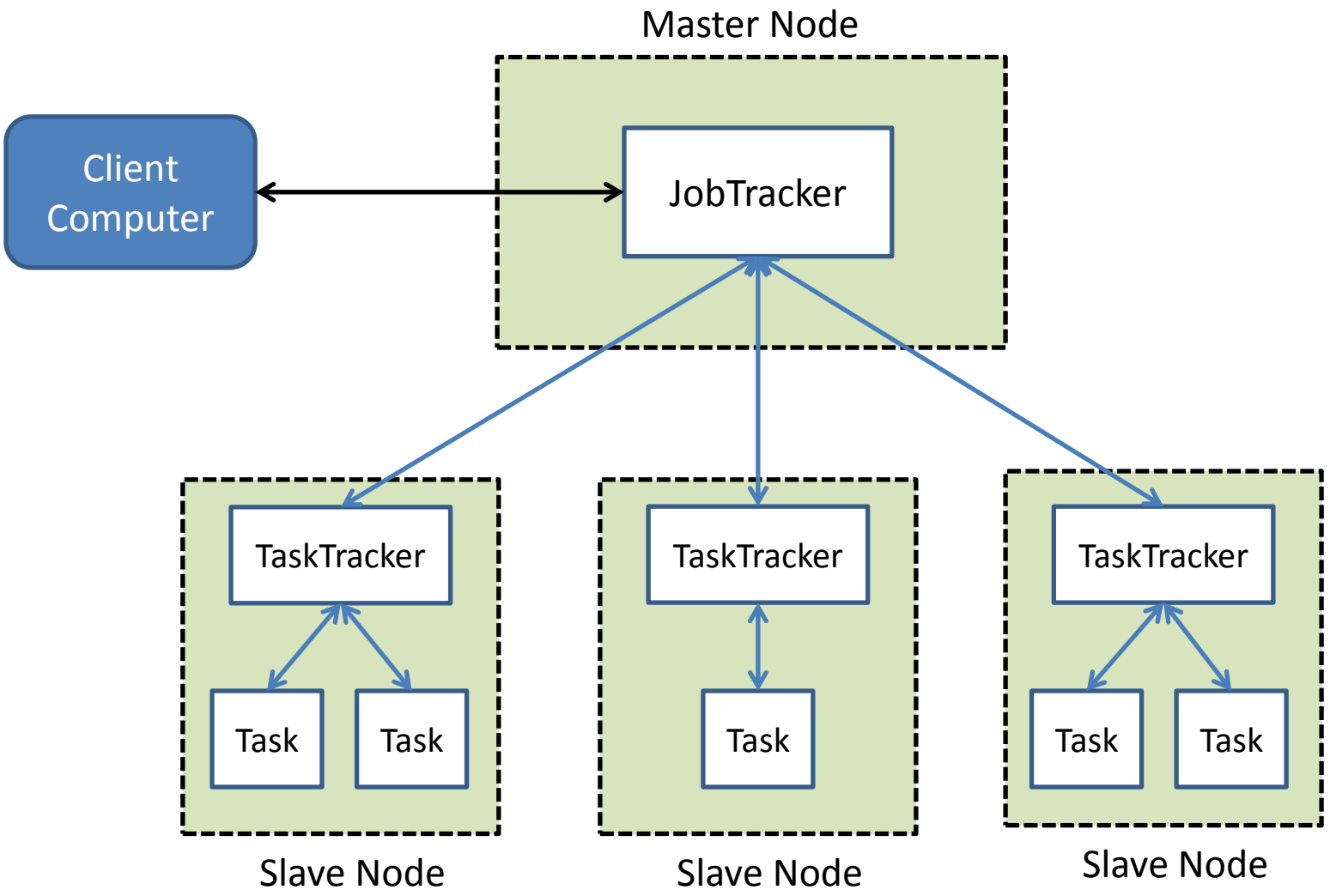
```
sort -k2 -n -r word-count.txt | more
```

```
the       727
of        494
shall     293
and       262
to        201
be        178
or        157
in        147
by        101
a         97
united    85
for       82
any       79
president         72
```

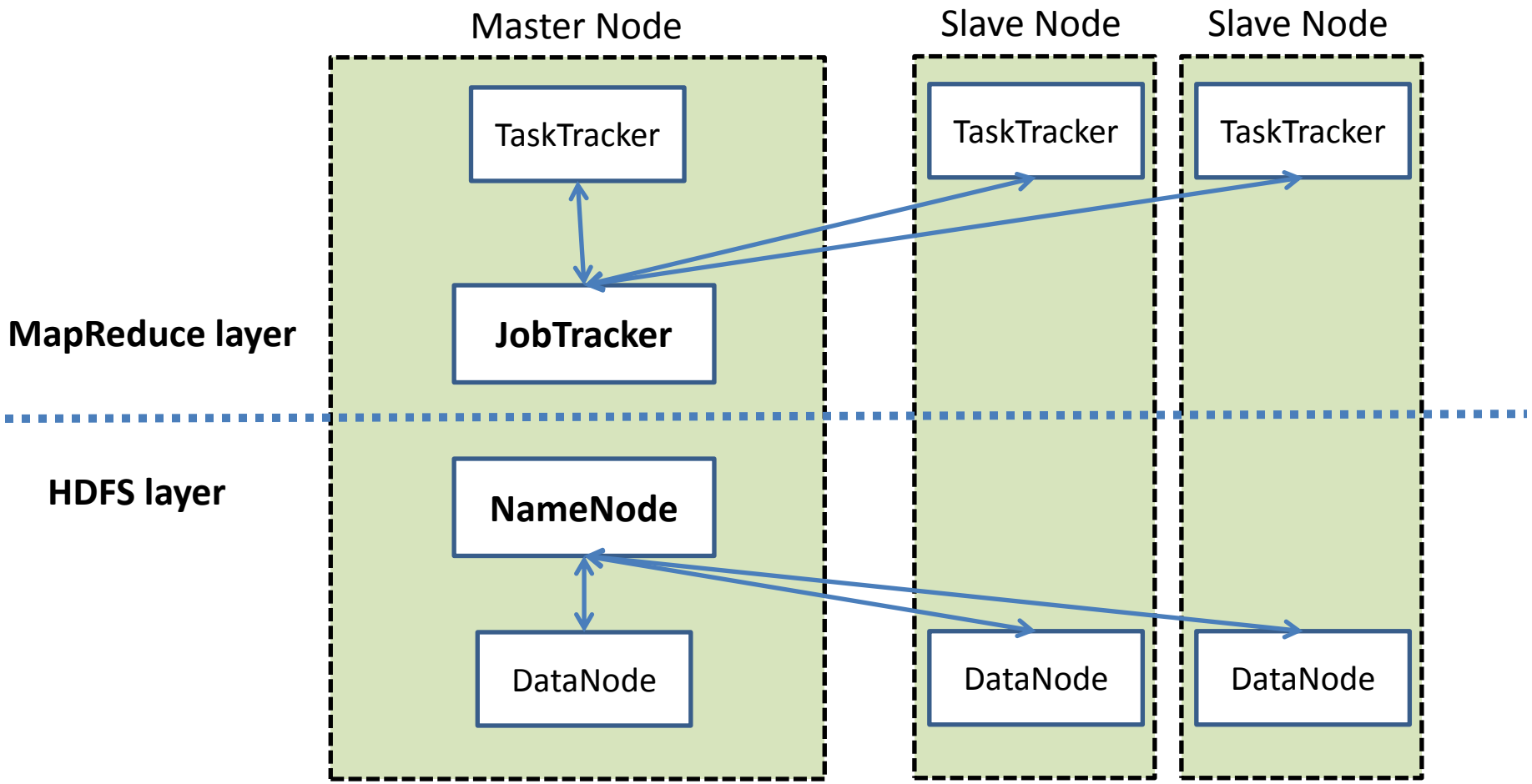
Hadoop Map/Reduce - Terminology

- Running “Word Count” across 20 files is one ***job***
- ***Job Tracker*** initiates some number of ***map tasks*** and some number of ***reduce tasks.***
- For each map task at least one ***task attempt*** will be performed… more if a task fails (e.g., machine crashes).

High Level Architecture of MapReduce



High Level Architecture of Hadoop



Web based User interfaces

- JobTracker: http://50.17.65.29:9100/
- NameNode: http://50.17.65.29:9101/

Hadoop Job Scheduling

- FIFO queue matches incoming jobs to available nodes
  - No notion of fairness
  - Never switches out running job

- Warning! Start your job as soon as possible.

Reporting Progress

If your tasks don't report anything in 10 minutes they would be killed by Hadoop!

```
public enum MyCounters {
    MAPPER_LINE_COUNTER, REDUCER_INVOCATION_COUNTER
}
```

```
@Override
public void map(LongWritable key, Text value, final Context context) throws IOException,
    InterruptedException {

    String line = value.toString().toLowerCase();
    StringTokenizer tokenizer = new StringTokenizer(line, " \"()-.[],:;/?");
    while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        context.write(word, one);
    }

    context.getCounter(MyCounters.MAPPER_LINE_COUNTER).increment(1);

}
```

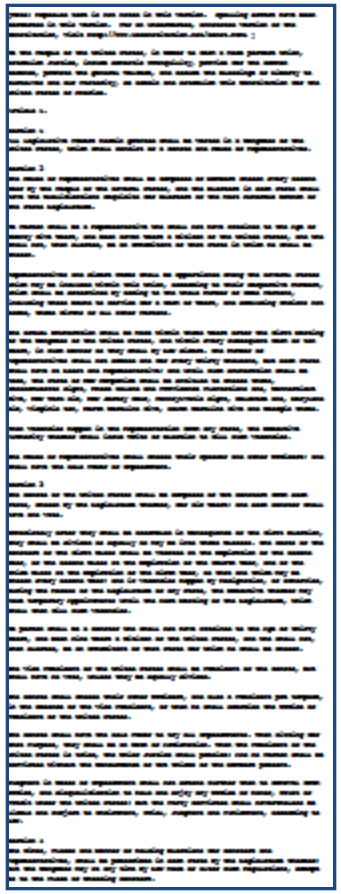
Source files are available at: http://www.ics.uci.edu/~yganjisa/files/2011/hadoop-presentation/WordCount-v2-src.zip

Distributed File Cache

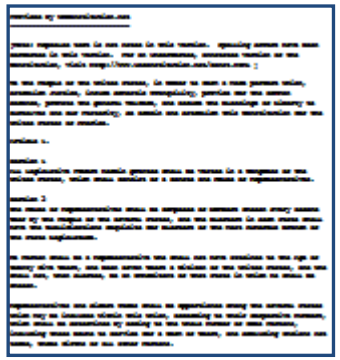
- The Distributed Cache facility allows you to transfer files from the distributed file system to the local file system (for reading only) of all participating nodes before the beginning of a job.

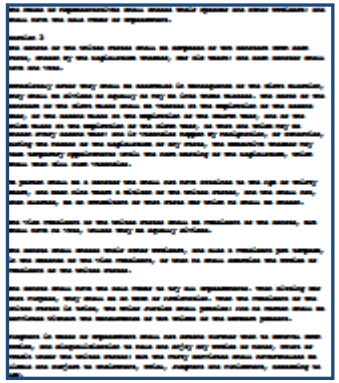
```
DistributedCache.addCacheFile(new URI("/user/john/crawler4j.properties"), conf);
```

TextInputFormat



Split





LineRecordReader

<offset$_1$, line$_1$>

<offset$_2$, line$_2$>

<offset$_3$, line$_3$>

For more complex inputs,
You should extend:

- InputSplit
- RecordReader
- InputFormat

Part 2: Amazon Web Services
(AWS)

What is AWS?

- A collection of services that together make up a **cloud computing** platform:
  - S3 (Simple Storage Service)
  - EC2 (Elastic Compute Cloud)
  - Elastic MapReduce
  - Email Service
  - SimpleDB
  - Flexibile Payments Service
  - …

Case Study: 

- Yelp uses Amazon S3 to store daily logs and photos, generating around 100GB of logs per day.
- Features powered by Amazon Elastic MapReduce include:
  - People Who Viewed this Also Viewed
  - Review highlights
  - Auto complete as you type on search
  - Search spelling suggestions
  - Top searches
  - Ads
- Yelp runs approximately 200 Elastic MapReduce jobs per day, processing 3TB of data.

Amazon S3

- Data Storage in Amazon Data Center
- Web Service interface
- 99.99% monthly uptime guarantee
- Storage cost: $0.15 per GB/Month

- S3 is reported to store more than 102 billion objects as of March 2010.

Amazon S3

- You can think of S3 as a big HashMap where you store your files with a unique key:
  - HashMap: key -> File

References

- Hadoop Project Page: http://hadoop.apache.org/

- Amazon Web Services: http://aws.amazon.com/