

IIT KHARAGPUR

IIT GANDHINAGAR


NPTEL ONLINE
CERTIFICATION COURSES

Scalable Data Science

Lecture 18: Hadoop System

Sourangshu Bhattacharya

Computer Science and Engineering

IIT KHARAGPUR

In the previous lectures:

- Outline:
  - What is Big Data and Hadoop?
  - What is Map Reduce ?
  - Example Map Reduce program.
  - HDFS – commands and internals.

In this Lecture:

- Outline:
  - Map-reduce programming in Java
  - Map reduce programming in other languages
  - Implementation details:
    - Job and tasks
    - Shuffle and sort

Hadoop Map Reduce

- Provides:
  - Automatic parallelization and Distribution
  - Fault Tolerance
  - Methods for interfacing with HDFS for colocation of computation and storage of output.
  - Status and Monitoring tools
  - API in Java
  - Ability to define the mapper and reducer in many languages through Hadoop streaming.

Wordcount program

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

Wordcount program - Main

```
public class WordCount {

public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 : 1);
} }
```

Wordcount program - Mapper

```
public static class TokenizerMapper extends Mapper<Object, Text, Text,
IntWritable>{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();

public void map(Object key, Text value, Context context )
throws IOException, InterruptedException {
        StringTokenizer itr = new StringTokenizer(value.toString());
        while (itr.hasMoreTokens()) {
                word.set(itr.nextToken()); context.write(word, one);
        }
}
}
```

Wordcount program - Reducer

```
public static class IntSumReducer extends
Reducer<Text,IntWritable,Text,IntWritable> {
private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values, Context
context )
throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
                sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
}
}
```

Wordcount program - running

```
export JAVA_HOME=[ Java home directory ]

bin/hadoop com.sun.tools.javac.Main WordCount.java

jar cf wc.jar WordCount*.class

bin/hadoop jar wc.jar WordCount [Input path] [Output path]
```

Wordcount in python

Mapper.py

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

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        print '%s\t%s' % (word, 1)
```

Wordcount in python

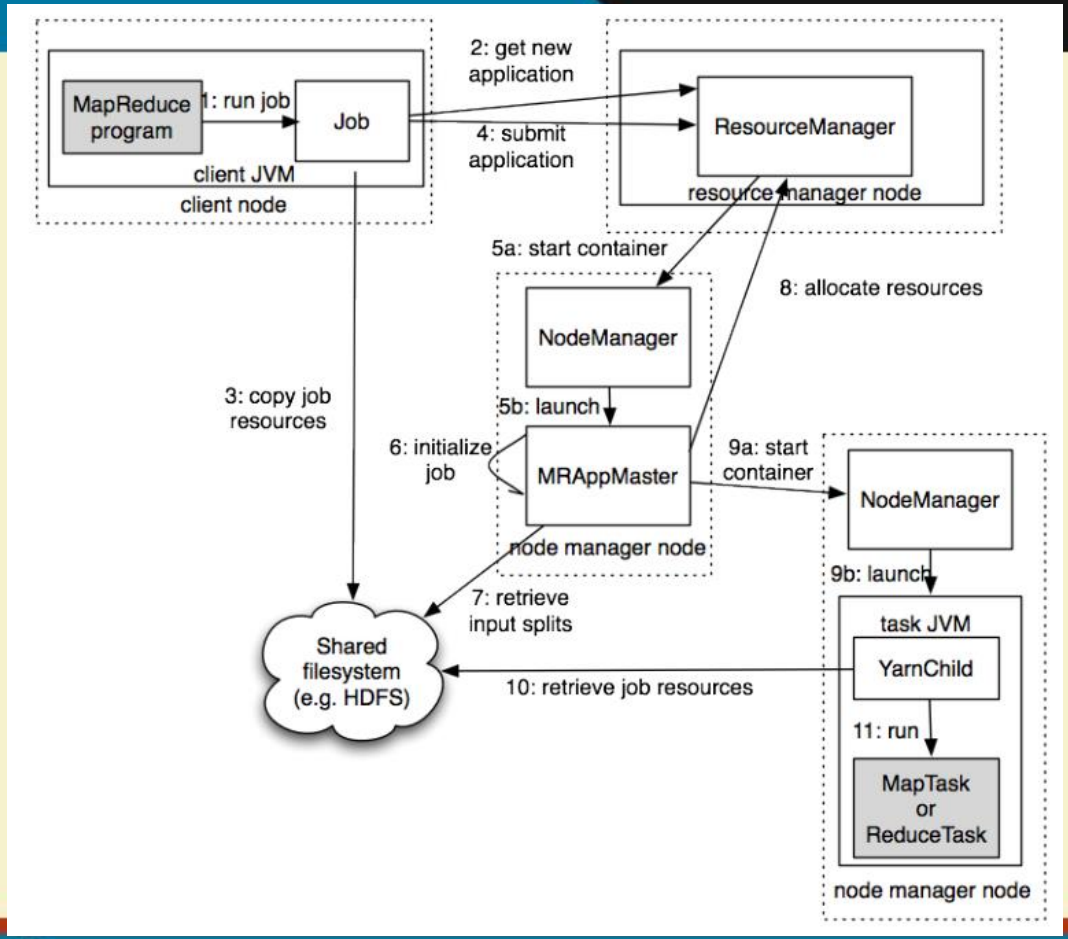
Reducer.py

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

from operator import itemgetter
import sys

# maps words to their counts
word2count = {}

# input comes from STDIN
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
        word2count[word] = word2count.get(word, 0) + count
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        pass

# sort the words lexigraphically;
#
# this step is NOT required, we just do it so that our
# final output will look more like the official Hadoop
# word count examples
sorted_word2count = sorted(word2count.items(), key=itemgetter(0))

# write the results to STDOUT (standard output)
for word, count in sorted_word2count:
    print '%s\t%s'% (word, count)
```

Execution code

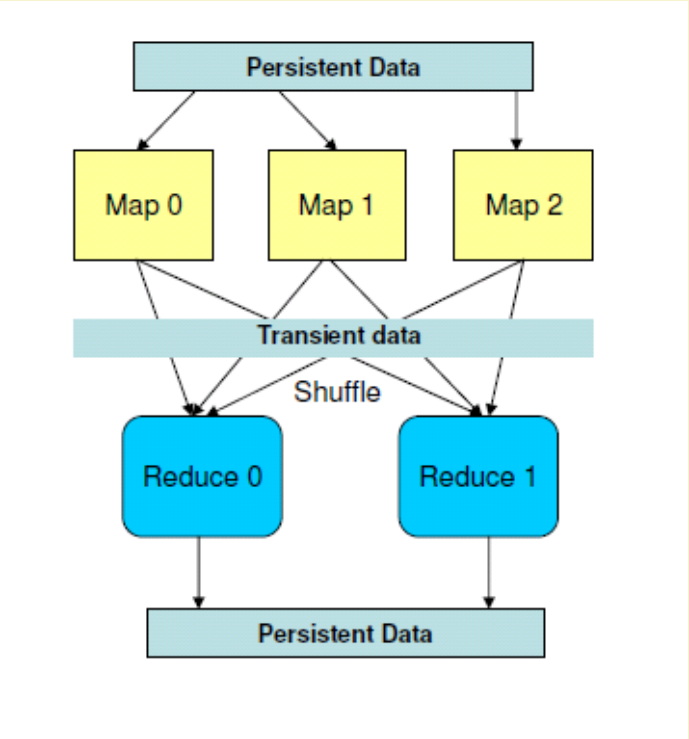
```
bin/hadoop dfs -ls

bin/hadoop dfs –copyFromLocal example example

bin/hadoop jar contrib/streaming/hadoop-0.19.2-streaming.jar -file
wordcount-py.example/mapper.py -mapper wordcount-
py.example/mapper.py -file wordcount-py.example/reducer.py -reducer
wordcount-py.example/reducer.py -input example -output java-output

bin/hadoop dfs -cat java-output/part-00000

bin/hadoop dfs -copyToLocal java-output/part-00000 java-output-local
```

Hadoop(v2) MR job

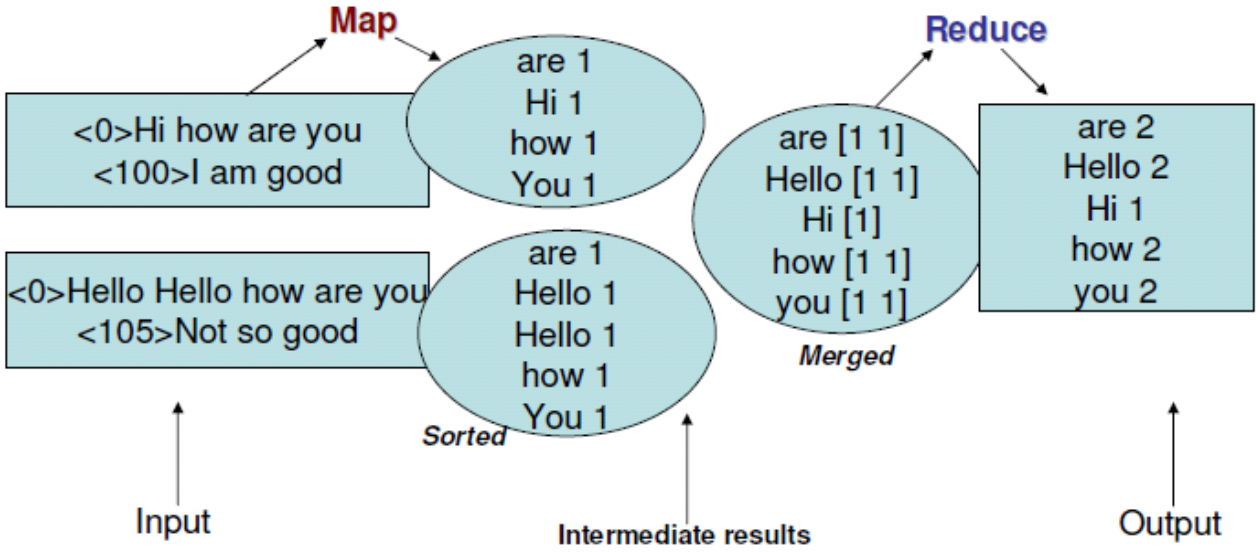


Source: Hadoop: The Definitive Guide

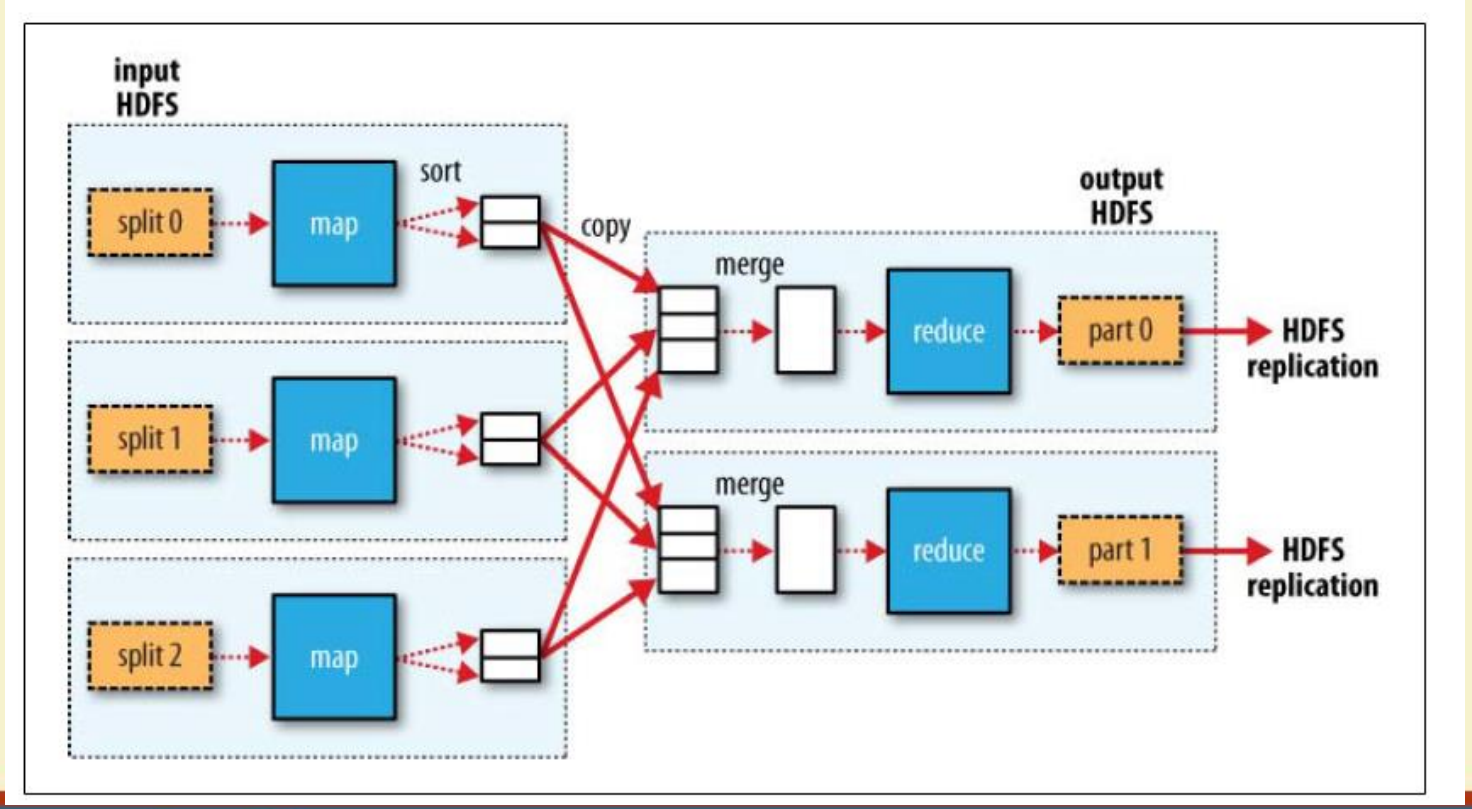
Map Reduce Data Flow



Data: Stream of keys and values

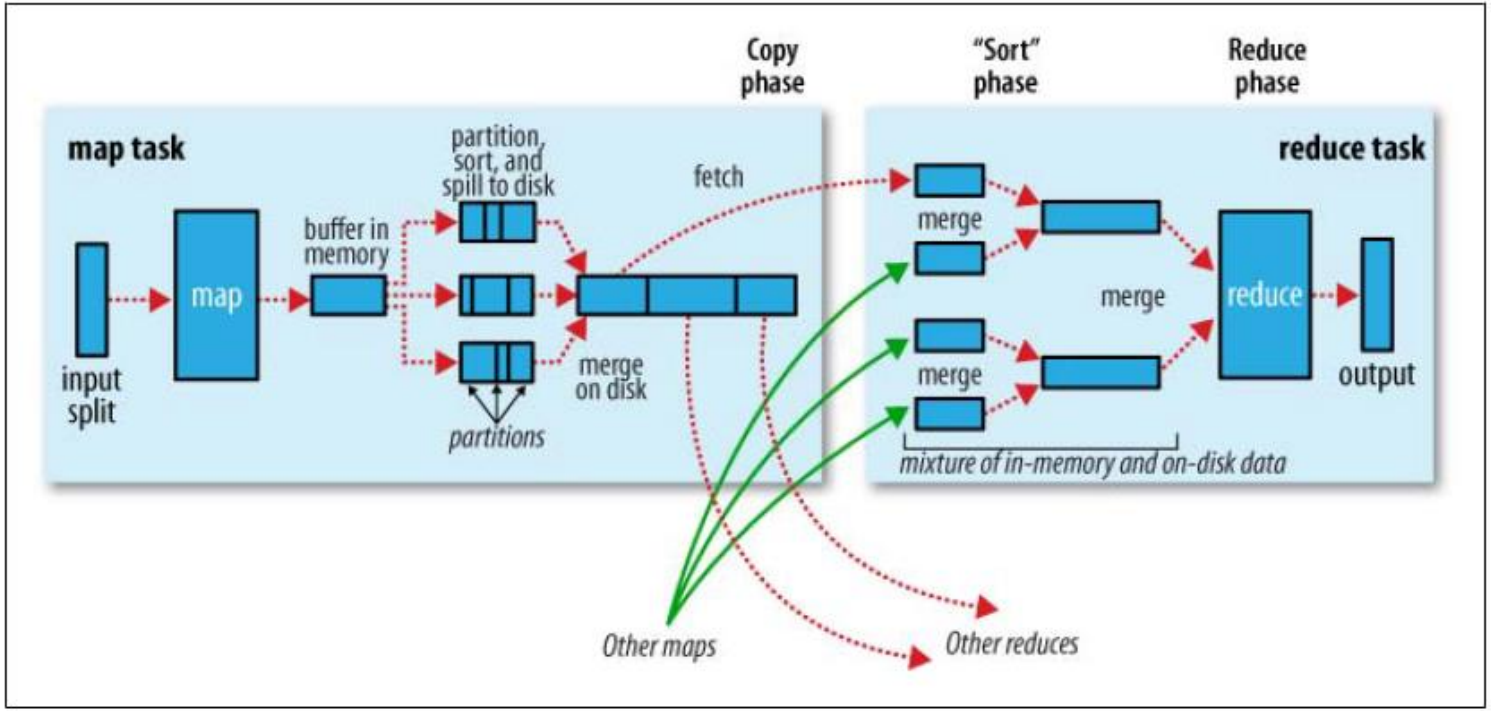


Hadoop MR Data Flow



Source: Hadoop: The Definitive Guide

Shuffle and sort



Source: Hadoop: The Definitive Guide

Data Flow

- **Input and final output are stored on a distributed file system (FS):**
  - Scheduler tries to schedule map tasks "close" to physical storage location of input data
- **Intermediate results are stored on local FS of Map workers.**
- **Output** of **Reduce** workers are stored on a **distributed file system**.
- **Output is often input to another MapReduce task**

Conclusion:

- We have seen:
  - Map-reduce programming in Java
  - Map reduce programming in other languages
  - Implementation details:
    - Job and tasks
    - Shuffle and sort

References:

- Jure Leskovec, Anand Rajaraman, Jeff Ullman. **Mining of Massive Datasets.** *2nd edition. - Cambridge University Press.* *http://www.mmds.org/*

- Tom White. **Hadoop: The definitive Guide.** Oreilly Press.

Thank You!!