Mansoura University
Faculty of Computers and Information Sciences
Department of Computer Science
First Semester- 2020-2021

[CS412P] Distributed Systems

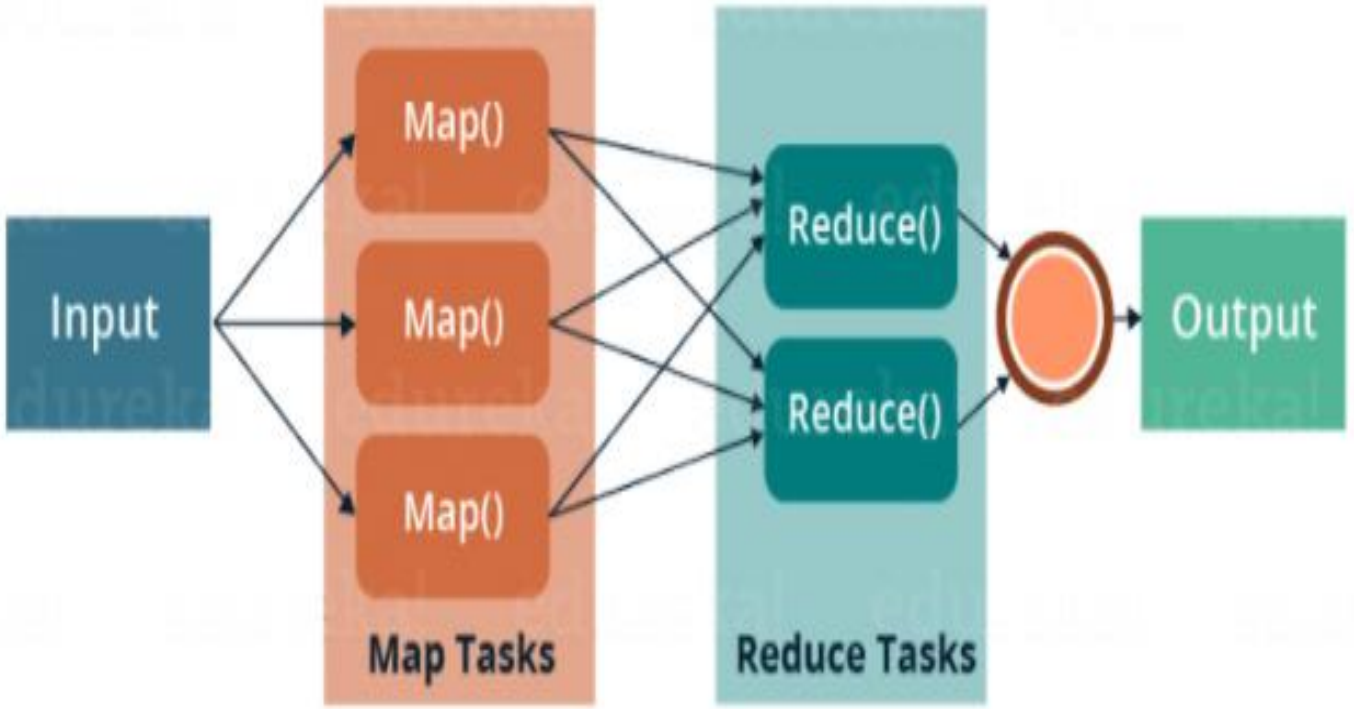
Grade : Fourth grade

By : Zeinab Awad

WHAT IS MAP REDUCE ?

-MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment.
-MapReduce consists of two distinct tasks – Map and Reduce.
As the name MapReduce suggests, the reducer phase takes place after the mapper phase has been completed.
So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs.
The output of a Mapper or map job (key-value pairs) is input to the Reducer.
The reducer receives the key-value pair from multiple map jobs.
Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.

WHAT IS MAP REDUCE ?



A WORD COUNT EXAMPLE OF MAPREDUCE

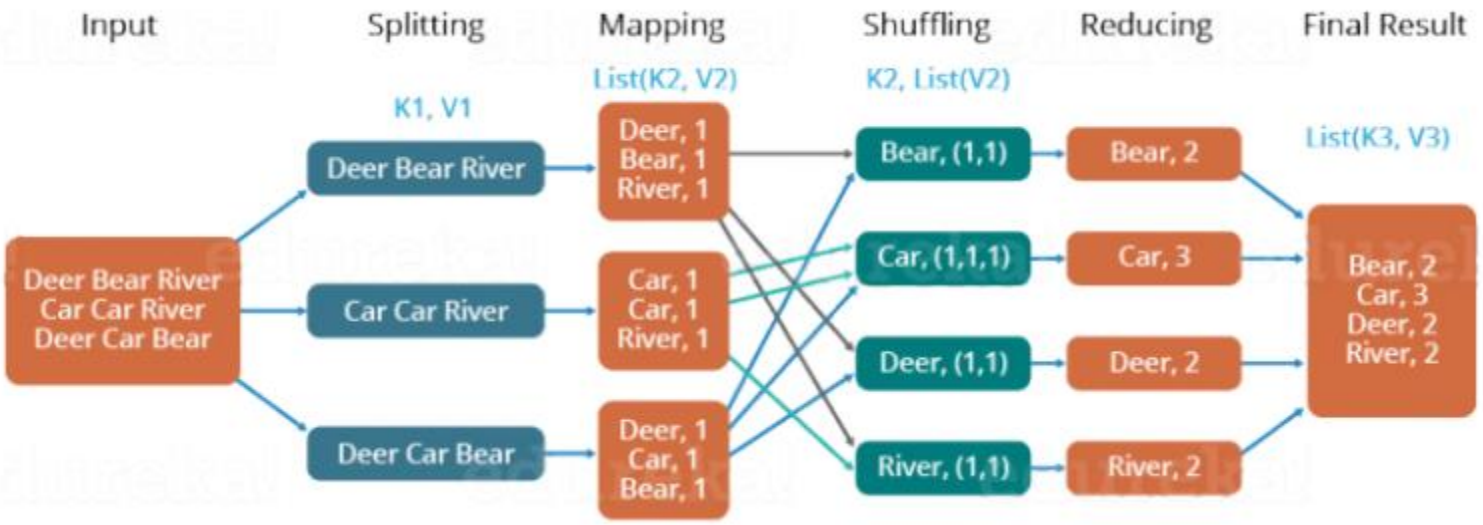
A Word Count Example of MapReduce

Let us understand, how a MapReduce works by taking an example where I have a text file called example.txt whose contents are as follows:

Dear, Bear, River, Car, Car, River, Deer, Car and Bear

Now, suppose, we have to perform a word count on the sample.txt using MapReduce. So, we will be finding the unique words and the number of occurrences of those unique words.

WORD COUNT EXAMPLE OF MAP REDUCE



First, we divide the input into three splits as shown in the figure. This will distribute the work among all the map nodes.

Then, we tokenize the words in each of the mappers and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, in itself, will occur once.

THE OVERALL MAP REDUCE WORD COUNT PROCESS

Now, a list of key-value pair will be created where the key is nothing but the individual words and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs – Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.

After the mapper phase, a partition process takes place where sorting and shuffling happen so that all the tuples with the same key are sent to the corresponding reducer.

So, after the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1].., etc.

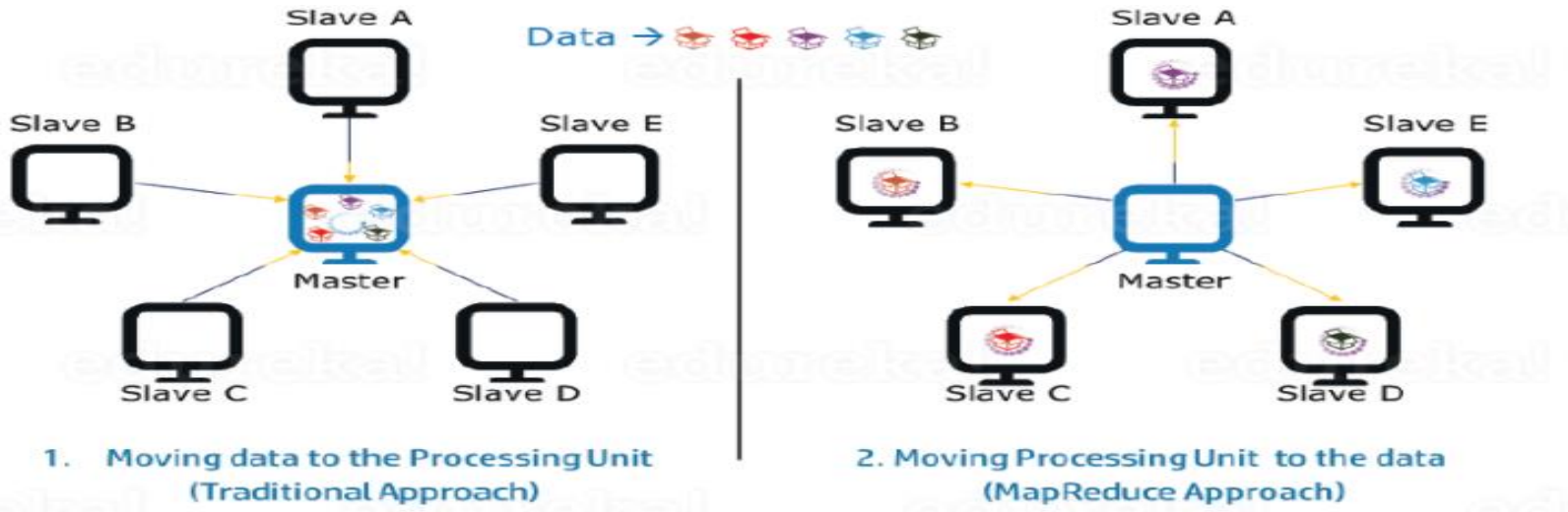
Now, each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as – Bear, 2.

Finally, all the output key/value pairs are then collected and written in the output file.

ADVANTAGES OF MAP REDUCE

1. Parallel Processing:

In MapReduce, we are dividing the job among multiple nodes and each node works with a part of the job simultaneously. So, MapReduce is based on Divide and Conquer paradigm which helps us to process the data using different machines. As the data is processed by multiple machines instead of a single machine in parallel, the time taken to process the data gets reduced by a tremendous amount as shown in the figure below (2).



ADVANTAGES OF MAP REDUCE

2. Data Locality:

Instead of moving data to the processing unit, we are moving the processing unit to the data in the MapReduce Framework. In the traditional system, we used to bring data to the processing unit and process it. But, as the data grew and became very huge, bringing this huge amount of data to the processing unit posed the following issues:

Moving huge data to processing is costly and deteriorates the network performance.

Processing takes time as the data is processed by a single unit which becomes the bottleneck.

The master node can get over-burdened and may fail.

Now, MapReduce allows us to overcome the above issues by bringing the processing unit to the data. So, as you can see in the above image that the data is distributed among multiple nodes where each node processes the part of the data residing on it. This allows us to have the following advantages:

ADVANTAGE OF MAP REDUCE

It is very cost-effective to move processing unit to the data.
The processing time is reduced as all the nodes are working with their part of the data in parallel.
Every node gets a part of the data to process and therefore, there is no chance of a node getting overburdened.

EXAMPLE :WORD COUNT

WordCount example reads text files and counts the frequency of the words. Each mapper takes a line of the input file as input and breaks it into words. It then emits a key/value pair of the word (In the form of (word, 1)) and each reducer sums the counts for each word and emits a single key/value with the word and sum

WORD COUNT MAPREDUCE ON JAVA

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;
```

WORD COUNTMAP REDUCE ON JAVA

```
public class WordCount
 {
// Map function public static class WordMapper extends
Mapper<LongWritable, Text, Text, IntWritable>
{
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException
 {
 // Splitting the line on spaces
String[] stringArr = value.toString().split("\\s+");
 for (String str : stringArr)
{
word.set(str);
context.write(word, one);
 }
 }
 }
```

WORD COUNT MAPREDUCE ON JAVA

```
// Reduce function
public static class CountReducer extends Reducer<Text, IntWritable,
Text, IntWritable>
{
private IntWritable result = new IntWritable();
public void reduce(Text key, Iterable values, Context context)
throws IOException, InterruptedException
{
int sum = 0;
 for (IntWritable val : values)
 { sum += val.get(); }
result.set(sum);
context.write(key, result); }
 }
```

WORD COUNT MAP REDUCE ON JAVA

```
public static void main(String[] args) throws Exception
{
Configuration conf = new Configuration();
Job rags = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(WordMapper.class);
job.setReducerClass(CountReducer.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);
}
 }
```

MAP REDUCE FRAMEWORK

Thanks