Yatish Kelkar TE IT 8001

Assignment 3 : Access Log

* Aim : Design a distributed application using MapReduce which processes a log file of a system. List out the users who have logged maximum times of the system. Use simple log file and process it using pseudo distribution mode on Hadoop platform.

* Theory :

Q1) Explain job execution in Hadoop.

Ans : - MapReduce is a programming model designed to process huge amounts of data by dividing the job into independent local tasks.

- When user submits a mapreduce job to Hadoop, the local job client prepares the job for submission & hands it off to the job tracker.

- The job tracker schedules the job and distributes the map work amongst multiple task trackers for parallel processing.

- Each task tracker issues a map task. These tasks are assigned with task IDs. Job initialization & job cleanup task are created & run by these task trackers.

- Once mapping phase results are available, job tracker distributes the

reduce work among the task trackers for parallel processing.

- Each task tracker issues a reduce task to perform the work. Job tracker recieves progress information from task trackers. Job client keeps polling the job tracker for progress.

- Once job is completed cleanup task gets processed. Task tracker sends the job completion status to the job tracker. Job tracker sends job completion message to the client. The total process causes Job Client's waitForJobTO Complete method to return.

Q2) Explain following classes:

1) IntWritable

Ans: i) IntWritable is the wrapper class in Hadoop which is similar to Integer class in Java. It is optimized to provide serialization in Hadoop.

ii) It implements Comparable, Writable & WritableComparable interfaces.

2) Iterable

Ans: i) Java iterable interface represents a collection of objects which can be iterable. A class implementing this interface can have its elements iterated.

eg: Iterable < String > = new Iterable [ ];

3) Context

Ans: i) The context object allows the Mapper / Reducer to interact with the rest of the Hadoop system.

ii) It includes configuration for the job and provides functions to write to an area of memory the outputs of various tasks.

eg: Context con;

con. write (key – val pair);

* Conclusion: Map Reduce application to process log file is successfully implemented.

Mapper class : LogMap.java

```
package Logs;

import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;

public class LogMap extends Mapper<LongWritable, Text, Text, IntWritable>
{
    public void map(final LongWritable key, final Text value, final Mapper.Con
text con) throws IOException, InterruptedException {

        final String line = value.toString();

        final StringTokenizer tokenizer = new StringTokenizer(line);

        con.write((Object)new Text(tokenizer.nextToken()), (Object)new IntWrit
able(1));
    }
}
```

Reducer class : LogReduce.java

```
package Logs;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class LogReduce extends Reducer<Text, IntWritable, Text, IntWritable>
```

```
{
    public void reduce(final Text word, final Iterable<IntWritable> values, fi
nal Context con) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }
        con.write(word, new IntWritable(sum));
    }
}
```

Driver class : LogDriver.java

```
package Logs;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class LogDriver {

    public static void main(String[] args) throws Exception {

        Configuration con = new Configuration();
        Job job = new Job(con, "Log count");

        job.setJarByClass(Logs.LogDriver.class);
        job.setMapperClass(Logs.LogMap.class);
        job.setReducerClass(Logs.LogReduce.class);

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

```
        FileInputFormat.addInputPath(job, new Path(args[1]));
        FileOutputFormat.setOutputPath(job, new Path(args[2]));

        job.waitForCompletion(true);

        FileSystem fs = FileSystem.get(con);
        FileStatus[] status = fs.listStatus(new Path("hdfs://localhost:9000"+a
rgs[2]));
        FSDataInputStream fd = fs.open(status[1].getPath());


        String string = null;
        string = fd.readLine();

        float max = -9999, count;
        String maxIP = null;

        while(string != null) {
            String [] tokens = string.split("\t");
            count = Integer.parseInt(tokens[1]);

            if(count > max) {
                max = count;
                maxIP = tokens[0];
            }

            string = fd.readLine();
        }
        System.out.println("The IP with most logs is " + maxIP + " with " + Ma
th.round(max) + " number of logs");
    }
}
```

Output Screenshot