	December Library and extens) Jews and proving from proton
	Assignment 3: Access Log
dal	The state of the s
4	Aim: Oscion a distributed and settle is a set
	Court a distributed application USIM MapReduce which acresses
- Since	by the of a system. List out the users who have loosed maximum
^	times of the system. Use simple log file and process it using pseudo distribution mode on Hadasa clatters
53	distribution mode on Hadoop platform.
- 90	to at denium statemens of deliverations a hours del e 2000
*	Theory:
,	
91)	Explain job execution in Hadoop.
1	
Ans:	- MapReduce is a programming model designed to process huge amounts
al an	activity the 100 into independent local tasks.
	When user submits a magreduce of 1 llad a 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
9	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 social of the social control of the social soci
-	The job tracker schedules the job and distributes the map work amongst
	The job tracker schedules the job and distributes the map work amongst multiple task trackers for parallel processing.
-	Each tosk tracker issues a map task. These tasks are assigned with
	task 10s. Job initialization & job cleanup task are created & run by
ded)	trest task trackers.
_	Once mapping phase results are available, job trocker distributes the
	pring priore results are available, job draight aistributes tile

FOR EDUCATIONAL USE

(Sundaram)

	reduce work among the tost tracters for parallel processing.
	TO THE PROPERTY OF THE PROPERT
_	Each task tracker issues a reduce task to perform the work. Job
	tracker recieves progress Information from task trackers. Job
2 121010	tracker recieves progress Information from task trackers. Job client keeps polling the job tracker for progress.
MAIN THE SECTION	Once up is completed cleanup took acts processed. Task tracker sends
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	the job smalletion status to the 10h tracker Job tracker
	condo in completion message to the client The total process
	Once job is completed cleanup took gets processed. Task tracker sends the job completion status to the job tracker. Job tracter sends job completion message to the client. The total process couses Job Client's waitfor Job To Complete method to return.
	couses septiments workers see the control of the co
Q2)	Explain following classes: 900 billion nellosses de molegia 100
,)	IntWritable Large la labora principality and soutoffeld and and
C 442 (4) (1)	sent lose tachares has one de at particular par and to
A	i) IntWritoble is the Wrapper class in Hadoop which is similar to
Ans:	Totales class in Tays It is optimized to provide
	Integer class in Java. It is optimized to provide serialization in Hadoop.
13/14	Germanzation III materials
. \	It implements Comparable, Writable & Writable Comparable
	interfaces and
	interfaces parasony following and authority deat objettion is
1	Therable event sent had your a every reshort sent that +
(1) 2)	Interable and shall general day & nodustation date 201 day
pd Au	i) Java iterable interface represents a collection of objects which
	can be iterable. A class implementing this interface can have its
	elements iterated
90) e	elements lierared
	FOR EDUCATIONAL USE
Sundaram	

	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 - vol pair);
*	Conclusion: Map Reduce application to process log file is successfully
	implemented
	Test conferences strapported y Williams Leading New Property Services and the Conference of the Confer
Sundaram	FOR EDUCATIONAL USE

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

