**What is Hadoop?**

Hadoop, in general, is an ecosystem of tools that help deal with Big Data. Core Hadoop is more specifically known for its Data Storage and Data Computation abilities, which are provided by HDFS and MapReduce respectively.

**What are the components of Hadoop?**

Hadoop core has two main components, which are Data Storage and Data Computation.

Data Storage is provided by Hadoop Distributed File System (HDFS). This protects against data loss from hardware failure, creates and helps with parallelization and can deal with data in the petabytes range and above.

MapReduce provides the Data Computation in Hadoop core. This is designed well to work on distributed systems and has inbuilt fault tolerance. This consists of the Mapper phase and the Reducer phase.

**Why do we need Hadoop?**

As the size of the problem becomes a part of the problem that we are trying to solve, we need something to be able to deal with this type of Big Data. Hadoop is an ecosystem built to be able to deal with all kinds of Big Data and can even achieve close to real time analytics of this data. Hadoop provides an environment where the user is unaware as to the location and the complexity of the file storage (one file multiple blocks, across multiple data warehouses, etc.) without comprising on the ease of access to the file system.

**What problems does it solve?**

It solves problems where the size of the data itself is part of the problem. It gives the ability to process massive amounts of data, possibly even at real-time. It helps with business analytics where a clients’ decision yesterday could be used to influence a decision tomorrow. It also helps with the storage of massive petabytes of information across multiple “blocks” or “Data Warehouses” and for seamless access to this data. It also helps in reducing the cost of storing and processing Big Data.

**What is HDFS?**

Hadoop Distributed File System (HDFS) is the storage part of Hadoop. It is a file system designed to store very large files.

* Protects in case of data loss from hardware failure
* It creates and helps with parallelization opportunities
* Capable of dealing with data in the range of Petabytes and higher
* Provides high performance streaming data access
* Can be implemented on any hardware
* Works with noticeable delay even in the case of failures
* Automatically stores replicas of blocks to recover from data failure

**What is the HDFS architecture?**

HDFS is made of two types of nodes: Namenodes & Datanodes.

An HDFS cluster consists of 1 NameNode (master) and Many DataNodes (Workers who serve the Master node).

1. HDFS files follow the write once read many (WORM) model
2. Metadata (of the file locations and job locations) are stored on the NameNode and are updated by multiple worker nodes concurrently
3. HDFS uses block structured file system
4. Individual files are broken into blocks of a fixed size
5. Blocks are stored in one or more DataNodes
6. This gives the ability for a file to be of any size as it can be stored across multiple machines
7. HDFS comes with its own utilities for file management
8. These HDFS file blocks are managed by DataNode services
9. Each block is replicated across multiple machines to deal with data loss from hardware failures

**What does the Namenode do? How many are there in a cluster?**

It is the Master Node. There is usually just one for each cluster. Manages the file system namespace and its tree. Maintains the metadata for all the files and directories on the tree. Stores the File Names, Permissions and the Data Location of each block of each file. This is stored on the main memory for fast data access. NameNode helps rectify hardware failures.

**What does the Datanode do? How many are there in a cluster?**

These are the workers nodes. There are multiple per cluster. They serve the requests of the Master node. They can store and retrieve blocks on demand. They report to the NameNode with the list of the blocks that they are storing. They Compute CheckSums and report any errors to the NameNode.

**What is the advantage of using HDFS in a Hadoop cluster instead of using networked storage?**

* HDFS can use the power of multiple machines where as NFS is restricted to 1.
* The Size of the file system is quickly extensible in HDFS whereas it is fixed for NFS.
* HDFS offers inbuilt reliability whereas NFS does not
* Both have clients contending for service but in HDFS the load is distributed across multiple servers.
* Clients need not copy data before processing it in HDFS
* HDFS supports very large file sizes

**What is MapReduce?**

It provides the Data Computation capability of Hadoop. Provides Distributed Computation power. It leverages Data-Locality for higher performance. It tries to bring the computation to the data by assigning the jobs to the node, which holds the data.

**Where does a MapReduce program run?**

A MapReduce program runs on the DataNodes and is the job is allocated and tracked by the Job Tracker. It can be configured to run on a single server or on a distributed network.

**What is data locality optimization?**

Data Locality optimization is the logic used to schedule the computation of the Mapper as close as possible to the Node in which the Data is contained.

Same node – distance – 0

Same Rack – distance – 2

Same Data Warehouse – distance – 4

Difference Data Warehouse – distance – 6

This is computed using Rack Mapping Tree Structure.

**Does data locality optimization apply equally to Map task and Reduce tasks?**

No this applies primarily to the Map Task and generally not to the Reduce Tasks.

**How does Hadoop decide where a Map task should run (distance computation)?**

The Admin runs a script and sets up the configuration so that Hadoop is aware of the cluster topology. This allows it to create a Rack Mapping Tree Structure. The Distance between two blocks is calculated as the distance to the nearest node in the Tree where they meet.

Same node – distance – 0

Same Rack – distance – 2

Same Data Warehouse – distance – 4

Difference Data Warehouse – distance – 6

**How is a MapReduce program different from a regular Java, Python, C/C++ program?**

A regular Java, Python or C/C++ program normally run on the local and on one machine whereas MapReduce is designed to run on multiple machines.

**How is a MapReduce program scheduled such that it runs optimally?**

It is scheduled with Data Locality in mind.

**What is the advantage/disadvantage of using speculative execution?**

**Adv –** It is helpful when some tasks are running too slow and are slowing down overall completion time. It is an optimization for well built code that works most of the time and rarely leads to slowness.

**DisAdv -** When the code contains a bug then the tasks are going to be slow when the bug is encountered and this would make Speculative Execution start another task with the same bug. This leads to counter the optimization. Reducer tasks have to fetch the outputs from all its mappers and this intensely uses up the bandwidth.

**What does a Mapper do?**

Maps the key-value input pairs to possibly other key-value output pairs after some processing of the input. It drops bad/erroneous records. It Sorts and group key-value pairs by key before sending to the Reducer.

**What does Mapper input data look like?**

(line\_offset\_number, line\_contents)

**Where does Mapper input data come from?**

It can be configured to get the input data from many sources. The most common input data source is a text file. The user specifies this.

**What are Map tasks?**

Map tasks are the set of tasks that run on each of the input splits on the Worker Nodes and are managed by the JobTracker. The Mapper takes an input and maps to some (key,value) pair and sorts and groups by key and sends this to the reducer.

**Where does Map task output get written?**

Writes its output to the shared HDFS where the Reducer picks up the output and processes.

**What does a Reducer do?**

Takes the Key/Value input pairs from the Mapper or Combiner and maps it to other key/value pairs that are the output. Usually iterated over the key/value pairs from Mapper/Combiner and picks out some values/one value/creates a value using these, and uses this to create a key/value output.

**What does Reducer input data look like?**

(mapper\_produced\_key, mapperOrCombiner\_produced\_list\_of\_values)

**Where does Reducer input data come from?**

Mapper or Combiner Outputs to shared HDFS. Reducer picks it up from there.

**What are Reduce tasks?**

Reduce tasks are the set of tasks that run on the group of the Mapper tasks outputs that worked on the input splits. These are managed by the JobTracker. Takes the Key/Value input pairs from the Mapper or Combiner and maps it to other key/value pairs that are the output.

**Where does Reduce task output get written?**

The reduce task output is usually written back to the local file system as this is the analysis that is required by the user.

**What happens during the shuffle phase?**

The shuffle phase is where the Mappers key/value pair outputs are sorted by key before being given to the Reducer.

**Is Pig Case Sensitive ?**

Mixed rules – Operators and Commands are not case-sensitive. Aliases and function names are case-sensitive.

**When we submit a Pig program, are there MapReduce jobs created?**

Yes, Underneath it all Pig generates and runs MapReduce programs

**Can a Pig program read input from HDFS?**

Yes Pig launches jobs and interacts with HDFS from my workstation.

**Can a Pig program output data to HDFS?**

Yes Pig launches jobs and interacts with HDFS from my workstation.

**How do we define a schema for a Pig table?**

A schema is optional in PIG.

**Load**

Loads a bag with the data from some user-specified input.

raw = LOAD 'excite.log' USING PigStorage('\t') AS (user, time, query);

**filter .. by**

Loads the new bag with the values from the specified bag that follow the filter.

clean1 = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query);

**foreach .. generate**

Runs through the contents on the specified bag and generates something to new bag.

clean2 = FOREACH clean1 GENERATE user, time, org.apache.pig.tutorial.**ToLower(**query) as query;

**dump**

Dump allows us to view the contents of a bag.

Dump clean2;

**Store**

Allows us to store the contents of a bag to a file.

STORE ordered\_uniq\_frequency INTO '/tmp/tutorial-results' USING PigStorage();

**Join**

Joins two or more relations.

**Group**

Groups the data in a single relation.

**order .. by**

Sorts a relation by one or more fields.

**Distinct**

Removes duplicate rows from a relation

**Limit**

Limits the size of the relation to a maximum number of tuples.

**Union**

Combines two or more relations into one

**Describe**

Prints a relations schema

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

public class MaxTemperature {

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

if (args.length != 2) {

System.err.println("Usage: MaxTemperature <input path> <output path>");

System.exit(-1);

}

JobConf conf = new JobConf(MaxTemperature.class);

conf.setJobName("Max temperature");

FileInputFormat.addInputPath(conf, new Path(args[0]));

FileOutputFormat.setOutputPath(conf, new Path(args[1]));

conf.setMapperClass(MaxTemperatureMapper.class);

conf.setReducerClass(MaxTemperatureReducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

JobClient.runJob(conf);

}

}

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class MaxTemperatureMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

private static final int MISSING = 9999;

public void map(LongWritable key, Text value,

OutputCollector<Text, IntWritable> output, Reporter reporter)

throws IOException {

String line = value.toString();

String year = line.substring(15, 19);

int airTemperature;

if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs

airTemperature = Integer.parseInt(line.substring(88, 92));

} else {

airTemperature = Integer.parseInt(line.substring(87, 92));

}

String quality = line.substring(92, 93);

if (airTemperature != MISSING && quality.matches("[01459]")) {

output.collect(new Text(year), new IntWritable(airTemperature));

}

}

}

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class MaxTemperatureReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text key, Iterator<IntWritable> values,

OutputCollector<Text, IntWritable> output, Reporter reporter)

throws IOException {

int maxValue = Integer.MIN\_VALUE;

while (values.hasNext()) {

maxValue = Math.max(maxValue, values.next().get());

}

output.collect(key, new IntWritable(maxValue));

}}

input :

000000011111111222233333333333333333333333333333333333333333333333333333333333333333333+44445555555555

000000011111111222233333333333333333333333333333333333333333333333333333333333333333333+44445555555555

000000011111111666633333333333333333333333333333333333333333333333333333333333333333333+77775555555555

000000011111111666633333333333333333333333333333333333333333333333333333333333333333333+77775555555555

000000011111111222233333333333333333333333333333333333333333333333333333333333333333333+44445555555555

000000011111111666633333333333333333333333333333333333333333333333333333333333333333333+77775555555555

000000011111111222233333333333333333333333333333333333333333333333333333333333333333333+44445555555555

000000011111111888833333333333333333333333333333333333333333333333333333333333333333333+44445555555555

000000011111111999933333333333333333333333333333333333333333333333333333333333333333333+44445555555555

output :

2222 4444

6666 7777

8888 4444

9999 4444

**TWEET**

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

public class TweetCounter {

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

if (args.length != 2) {

System.err.println("Usage: TweetCounter <input path> <output path>");

System.exit(-1);

}

JobConf conf = new JobConf(TweetCounter.class);

conf.setJobName("Tweet Word Counter");

FileInputFormat.addInputPath(conf, new Path(args[0]));

FileOutputFormat.setOutputPath(conf, new Path(args[1]));

conf.setMapperClass(TweetCounterMapper.class);

conf.setReducerClass(TweetCounterReducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

JobClient.runJob(conf);

}

}

import java.io.IOException;

import org.apache.commons.lang.StringUtils;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class TweetCounterMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {

public void map(LongWritable key, Text value,

OutputCollector<Text, IntWritable> output, Reporter reporter)

throws IOException {

String [] wordsToCheck = { "Chicago", "Dec", "Java", "hackathon"};

String line = value.toString();

line = StringUtils.lowerCase(line);

for (int i =0; i< wordsToCheck.length; i++){

int count = StringUtils.countMatches(line, wordsToCheck[i].toLowerCase());

output.collect(new Text(wordsToCheck[i]), new IntWritable(count));

}

}

}

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class TweetCounterReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text key, Iterator<IntWritable> values,

OutputCollector<Text, IntWritable> output, Reporter reporter)

throws IOException {

int cnt =0;

while (values.hasNext()) {

int num = values.next().get();

System.out.println("key = " + key + ", num = " + num);

cnt = cnt + num;

}

output.collect(key, new IntWritable(cnt));

}

}

input:

9-Dec-14,5:00PM,‏#Hackatopia,Tribeca Film Hackathon: Code As A New Language For Content Creators

28-Oct-13,7:00PM,‏#NYCHadoop,Hadoop-NYC Strata/Hadoop World Meetup at AppNexus NYC

31-Dec-14,3:00PM,‏#Hackatopia,Designers, Developers, Doers, don't miss this upcoming Chicago hackathon

output:

Chicago 1

Dec 2

Java 0

hackathon 2

**PAGERANK**

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

public class PageRank {

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

if (args.length != 2) {

System.err.println("Usage: TweetCounter <input path> <output path>");

System.exit(-1);

}

JobConf conf = new JobConf(PageRank.class);

conf.setJobName("Page Rank");

FileInputFormat.addInputPath(conf, new Path(args[0]));

FileOutputFormat.setOutputPath(conf, new Path(args[1]));

conf.setMapperClass(PageRankMapper.class);

conf.setReducerClass(PageRankReducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(Text.class);

JobClient.runJob(conf);

}

}

import java.io.IOException;

import org.apache.commons.lang.StringUtils;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class PageRankMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {

public void map(LongWritable key, Text value, OutputCollector<Text, Text> output,

Reporter reporter) throws IOException {

// TODO Auto-generated method stub

String line = value.toString().trim();

String [] arr = line.split(" ");

String pointsTo = "";

System.out.println(line);

for(int i =1;i<arr.length-1;i++){

double pr = Double.parseDouble(arr[arr.length-1].trim()) / (arr.length-2);

System.out.println("key=" + arr[i].trim() + ", " + "value=" + arr[0].trim() + ", " + pr);

output.collect(new Text("key=" + arr[i].trim() + ", " ), new Text("value=" + arr[0].trim() + ", " + pr));

pointsTo += arr[i].trim() + " ";

}

output.collect(new Text("key=" + arr[0].trim() + ", " ), new Text("value=" + pointsTo.trim()));

}

}

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class PageRankReducer extends MapReduceBase implements Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterator<Text> values,

OutputCollector<Text, Text> output, Reporter reporter) throws IOException {

String k = key.toString();

k = k.split("=")[1];

String pointsTo = "";

double cnt = 0;

while (values.hasNext()) {

String t = values.next().toString();

System.out.println(t);

String [] arr = t.split(", ");

if(arr.length==1){

pointsTo += arr[0].split("=")[1];

}

else{

cnt += Double.parseDouble(arr[1].trim());

}

}

output.collect(new Text(k.split(",")[0].trim()), new Text(pointsTo.trim() + " " + cnt));

}

}

A C F 0.166667

B D E F 0.166667

C A B 0.166667

D A B C E F 0.166667

E F 0.166667

F B C 0.166667

A C F 0.1166669

B D E F 0.20000040000000002

C A B 0.20000040000000002

D A B C E F 0.05555566666666667

E F 0.08888906666666667

F B C 0.3388895666666667

**PAGERANK EXTRA CREDIT**

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.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapreduce.Job;

public class PageRank {

public static void main(String[] args) throws IOException, InterruptedException {

if (args.length != 2) {

System.err.println("Usage: TweetCounter <input path> <output path>");

System.exit(-1);

}

JobConf conf = new JobConf(PageRank.class);

conf.setJobName("Page Rank Extra Credit 1");

FileInputFormat.addInputPath(conf, new Path(args[0]));

FileOutputFormat.setOutputPath(conf, new Path(args[1]));

conf.setMapperClass(PageRankMapper.class);

conf.setReducerClass(PageRankReducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(Text.class);

JobClient.runJob(conf).waitForCompletion();

JobConf conf1 = new JobConf(PageRank.class);

conf1.setJobName("Page Rank Extra Credit 2");

FileInputFormat.addInputPath(conf1, new Path("./src/output/part-00000"));

FileOutputFormat.setOutputPath(conf1, new Path("./src/output1"));

conf1.setMapperClass(PageRankMapper.class);

conf1.setReducerClass(PageRankReducer.class);

conf1.setOutputKeyClass(Text.class);

conf1.setOutputValueClass(Text.class);

JobClient.runJob(conf1).waitForCompletion();

JobConf conf2 = new JobConf(PageRank.class);

conf2.setJobName("Page Rank Extra Credit 3");

FileInputFormat.addInputPath(conf2, new Path("./src/output1/part-00000"));

FileOutputFormat.setOutputPath(conf2, new Path("./src/output2"));

conf2.setMapperClass(PageRankMapper.class);

conf2.setReducerClass(PageRankReducer.class);

conf2.setOutputKeyClass(Text.class);

conf2.setOutputValueClass(Text.class);

JobClient.runJob(conf2).waitForCompletion();

}

}

import java.io.IOException;

import org.apache.commons.lang.StringUtils;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class PageRankMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, Text> {

public void map(LongWritable key, Text value, OutputCollector<Text, Text> output,

Reporter reporter) throws IOException {

// TODO Auto-generated method stub

String line = value.toString().trim();

String [] arr = line.split("[ \t]");

String pointsTo = "";

System.out.println(line);

for(int i =1;i<arr.length-1;i++){

double pr = Double.parseDouble(arr[arr.length-1].trim()) / (arr.length-2);

System.out.println("key=" + arr[i].trim() + ", " + "value=" + arr[0].trim() + ", " + pr);

output.collect(new Text("key=" + arr[i].trim() + ", " ), new Text("value=" + arr[0].trim() + ", " + pr));

pointsTo += arr[i].trim() + " ";

}

output.collect(new Text("key=" + arr[0].trim() + ", " ), new Text("value=" + pointsTo.trim()));

}

}

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class PageRankReducer extends MapReduceBase implements Reducer<Text, Text, Text, Text> {

public void reduce(Text key, Iterator<Text> values,

OutputCollector<Text, Text> output, Reporter reporter) throws IOException {

String k = key.toString();

k = k.split("=")[1];

String pointsTo = "";

double cnt = 0;

while (values.hasNext()) {

String t = values.next().toString();

System.out.println(t);

String [] arr = t.split(", ");

if(arr.length==1){

pointsTo += arr[0].split("=")[1];

}

else{

cnt += Double.parseDouble(arr[1].trim());

}

}

output.collect(new Text(k.split(",")[0].trim()), new Text(pointsTo.trim() + " " + cnt));

}

}

A C F 0.166667

B D E F 0.166667

C A B 0.166667

D A B C E F 0.166667

E F 0.166667

F B C 0.166667

A C F 0.1166669

B D E F 0.20000040000000002

C A B 0.20000040000000002

D A B C E F 0.05555566666666667

E F 0.08888906666666667

F B C 0.3388895666666667

A C F 0.11111133333333334

B D E F 0.2805561166666667

C A B 0.23888936666666669

D A B C E F 0.06666680000000001

E F 0.07777793333333334

F B C 0.22500045000000002

A C F 0.13277804333333335

B D E F 0.24527826833333338

C A B 0.1813892516666667

D A B C E F 0.09351870555555557

E F 0.10685206555555557

F B C 0.2401856655555556

**PIG**

-- create pigClass4

-- move sample.txt from local

-- pig

-- this is written with a lot of intermediate tables for my understanding (if I were to refer to this in the future) :)

rawClass4 = LOAD 'sample.txt' using PigStorage() AS (line);

hackathonLines = FOREACH rawClass4 GENERATE 'hackathon' as key, ((org.apache.pig.builtin.LOWER(line) matches '.\*hackathon.\*')?1:0) as number;

decLines = FOREACH rawClass4 GENERATE 'Dec' as key, ((org.apache.pig.builtin.LOWER(line) matches '.\*dec.\*')?1:0) as number;

chicagoLines = FOREACH rawClass4 GENERATE 'Chicago' as key, ((org.apache.pig.builtin.LOWER(line) matches '.\*chicago.\*')?1:0) as number;

javaLines = FOREACH rawClass4 GENERATE 'Java' as key, ((org.apache.pig.builtin.LOWER(line) matches '.\*java.\*')?1:0) as number;

--dump hackathonLines;

--dump decLines;

--dump chicagoLines;

--dump javaLines;

hackathonGroup = GROUP hackathonLines BY key;

decGroup = GROUP decLines BY key;

chicagoGroup = GROUP chicagoLines BY key;

javaGroup = GROUP javaLines BY key;

--dump hackathonGroup;

--dump decGroup;

--dump chicagoGroup;

--dump javaGroup;

hackathonSum = FOREACH hackathonGroup GENERATE group, SUM(hackathonLines.number);

decSum = FOREACH decGroup GENERATE group, SUM(decLines.number);

chicagoSum = FOREACH chicagoGroup GENERATE group, SUM(chicagoLines.number);

javaSum = FOREACH javaGroup GENERATE group, SUM(javaLines.number);

--dump hackathonSum;

--dump decSum;

--dump chicagoSum;

--dump javaSum;

result = union hackathonSum, decSum, chicagoSum, javaSum;

STORE result INTO './output.out' USING PigStorage();

9-Dec-14,5:00PM,‏#Hackatopia,Tribeca Film Hackathon: Code As A New Language For Content Creators 28-Oct-13,7:00PM,‏#NYCHadoop,Hadoop-NYC Strata/Hadoop World Meetup at AppNexus NYC 31-Dec-14,3:00PM,‏#Hackatopia,Designers, Developers, Doers, don't miss this upcoming Chicago hackathon

**Get Tweets**

import java.io.BufferedReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.List;

import twitter4j.ResponseList;

import twitter4j.Status;

import twitter4j.Twitter;

import twitter4j.TwitterException;

import twitter4j.TwitterFactory;

import twitter4j.User;

import twitter4j.auth.AccessToken;

import twitter4j.auth.RequestToken;

/\*\*

\* Shows one single user.

\*

\* @author Yusuke Yamamoto - yusuke at mac.com

\*/

public final class GetTweetsAndSaveToFile {

/\*\*

\* Usage: java twitter4j.examples.user.ShowUser [screen name]

\*

\* @param args message

\* @throws IOException

\*/

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

if (args.length < 1) {

System.out.println("Usage: java twitter4j.examples.user.ShowUser [screen name]");

System.exit(-1);

}

try {

Twitter twitter = new TwitterFactory().getInstance();

twitter.setOAuthConsumer("men2JyLEaAsxcbfmgzOAwUnTp", "2AGN0ie9TfCDJyWeH8qhTLtMhqRvRlNBtQU3lAP2M8k3Xk1KWl");

RequestToken requestToken = twitter.getOAuthRequestToken();

System.out.println("Authorization URL: \n" + requestToken.getAuthorizationURL());

AccessToken accessToken = new AccessToken("2811255124-zigkuv8MwDQbr5s9HdjLRSbg8aCOyxeD2gYGMfH",

"D7jFABWHQa8QkTWwgYj1ISUbWP8twdfbzNgYkXI3jwySR");

twitter.setOAuthAccessToken(accessToken);

/\*

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

while (null == accessToken) {

System.out.println("Open the following URL and grant access to your account:");

System.out.println(requestToken.getAuthorizationURL());

System.out.print("Enter the PIN(if available) and hit enter after you granted access.[PIN]:");

String pin = br.readLine();

try {

if (pin.length() > 0) {

accessToken = twitter.getOAuthAccessToken(requestToken, pin);

} else {

accessToken = twitter.getOAuthAccessToken(requestToken);

}

} catch (TwitterException te) {

if (401 == te.getStatusCode()) {

System.out.println("Unable to get the access token.");

} else {

te.printStackTrace();

}

}

}

\*/

System.out.println("Got access token.");

System.out.println("Access token: " + accessToken.getToken());

System.out.println("Access token secret: " + accessToken.getTokenSecret());

User user = twitter.showUser(args[0]);

if (user.getStatus() != null) {

System.out.println("@" + user.getScreenName() + " - " + user.getStatus().getText());

} else {

// the user is protected

System.out.println("@" + user.getScreenName());

}

FileWriter file = new FileWriter("./"+user.getScreenName()+"\_Tweets.txt");

List<Status> list = twitter.getHomeTimeline();

for (Status each : list) {

file.write("Sent by: @" + each.getUser().getScreenName()

+ " - " + each.getUser().getName() + "---" + each.getText()

+ "\n");

}

file.close();

System.exit(0);

} catch (Exception te) {

te.printStackTrace();

System.exit(-1);

}

}

}

Sent by: @TechCrunch - TechCrunch---HP Pops 4.7%, Picks Up More Than $3B In Market Cap On Breakup News http://t.co/rgEqJ0mJkO by @alex Sent by: @NewYorker - The New Yorker---How Will Haiti Reckon with the Duvalier Years? http://t.co/DHcFSxzo0g Sent by: @TechCrunch - TechCrunch---Pinterest curates its own content with new weekly collections called "Pin Picks" http://t.co/uw6giXxTH4 Sent by: @WIRED - WIRED---Not on a social network? Sorry, but you've still got a privacy problem: http://t.co/n7msgzN7QB Sent by: @RWW - ReadWrite---The Shellshock bug is bad news: At least two servers for Yahoo Games were allegedly breached in a hack http://t.co/NzPfaWcVDI Sent by: @TechCrunch - TechCrunch---Apple sapphire supplier GT Advanced Technologies files for bankruptcy http://t.co/RVzWGomm5t Sent by: @WIRED - WIRED---This cool, augmented-reality climbing wall guides your ascent http://t.co/uugcZl4j5l http://t.co/GGZ6T6ncoV Sent by: @funnyordie - Funny Or Die---We have to ask … the fuck was happening with 'Grease'? http://t.co/sQPOKWoAD3 http://t.co/0yPqddWKBt Sent by: @TheNextWeb - The Next Web---ICYMI: Facebook completes its $19 billion WhatsApp acquisition http://t.co/PEDBzspPMx Sent by: @TechCrunch - TechCrunch---Announcing the agenda for TechCrunch Disrupt Europe in London http://t.co/QcISrj2E1X http://t.co/wEPSn50H8v Sent by: @TechCrunch - TechCrunch---The Washington Post Is Now Free On Amazon Kindles http://t.co/XEFXMOsY8E by @sarahbuhr Sent by: @TechCrunch - TechCrunch---Facebook closes WhatsApp acquisition, Jan Koum to match Zuckerberg's $1 annual salary http://t.co/Vm8SI8DuUg Sent by: @RWW - ReadWrite---"Believe in your selfie", and other depressing signs of the times http://t.co/0Bo6teVbEE Sent by: @TechCrunch - TechCrunch---Foundation: Foursquare’s Product Future http://t.co/gfiD13bGyY by @kevinrose Sent by: @WIRED - WIRED---Thanks to fans, the best "Zelda" game is getting a 2-D makeover http://t.co/gqnsGfYHJO Sent by: @TechCrunch - TechCrunch---HP is in the process of pushing buttons and pulling levers and trying things and hoping something sticks http://t.co/KBzQhvmhMe Sent by: @PlayStation - PlayStation---RT @ClanLuckyStrike: #DriveClub doesn't release on @PlayStation until tomorrow but you can watch it LIVE right now! http://t.co/HI7ViM6IfQ … Sent by: @NewYorker - The New Yorker---Read “Scheherazade,” Haruki Murakami’s short story in this week’s issue: http://t.co/U41zDjN5dO http://t.co/hfsYB4ZGgF Sent by: @WIRED - WIRED---Beyond the Nobel Prize: What scientists are learning about navigation and your brain http://t.co/uggS1hAT46 Sent by: @TheNextWeb - The Next Web---ICYMI: Media streaming app Plex is coming to Xbox One and Xbox 360 http://t.co/6utMpid4Cb