ANALYSIS OF PAGE RANK ALGORITHM IN DISTRIBUTED HADOOP FILE SYSTEM

CSE4001 – Parallel and Distributed Computing

PROJECT BASED COMPONENT REPORT

by:

17BCI0133 – Sujeeth ES

17BCI0147 – K Rajesh

17BCI0037 – Antik Chattopadhyay

17BCI0183 - Anik Dewanje

SUBMITTED TO:

PROF. NARAYANAN PRASANTH

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I hereby declare that the report entitled " ANALYSIS OF   PAGE RANK ALGORITHM IN DISTRIBUTED HADOOP FILE SYSTEM" submitted by me, for CSE4001 Parallel and Distributed Computing (EPJ) to VIT is a record of bonafide work carried out by me under the supervision of Dr. Narayanan Prasanth.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for any other course in this institute or any other institute or university.

Place : Vellore

Date :

Signature of the Candidate

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my Professor, Dr Narayanan Prasanth as well as Vellore Institute of Technology who gave me the opportunity to do this project on the topic *ANALYSIS OF PAGE RANK ALGORITHM IN DISTRIBUTED HADOOP FILE SYSTEM*, which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them.

ABSTRACT:

• In this project we will implement a python program to analyze the number of users based on the page rank algorithm.

The page rank algorithm can be executed on the local operating system when the number of page rank algorithms is small.

• To calculate it at a faster rate and increase the efficiency of the algorithm, we need to run this algorithm on parallel systems and distributed systems

• There are five large-scale performance engines that support page rank algorithms that contain large amounts of data: - Hadoop, MPI, Dryad, Pregel, Twister.

• We have used python for parallel page rank algorithm which is executed on HADOOP.

CONTENTS **Page No.**

List of Figures

1. INTRODUCTION

• Page rank is an important web graph algorithm that ranks Internet users based on their importance.

• Page rank ranks a set of hyperlinks and it suggests random surfers about the probability of reaching a particular hyperlink.

• Markov series is useful in understanding the concept of page rank.

• Current iteration values are dependent on previous iteration values.

• The process must be repeated until the number of iterations reaches the given maximum iteration.

• The process can be stopped if the distance of EUCLIDEAN between successive iterations is less than the predefined threshold value.

• If we want to know the page rank for millions of nodes then single computer or laptop is not suitable. Computing such a large number of nodes requires us to have parallel and distributed systems to increase the efficiency of the program.

2. LITERATURE REVIEW

SN o	Title	Author	Journal Name & Date	Key Concepts	Adv/Disadv
1	Research on PageRank Algorithm parallel computing Based on Hadoop	Pengfei Yang Liquing Zhou	4th International Conference on Mechanical Materials and Manufacturing Engineering (MMME 2016)	Pagerank Algorithm	**Adv:-**It can effectively integrate Pagerank algorithm and Mapreduce framework on Hadoop **Dis:-**Wasting time communication problem between experimental ignored cluster nodes
2	Page Rank Algorithm in Hadoop By MapReduce Framework	Ankita Sachdev1, Ankita Rawale2 , Palak Pande3 , Kajal Sawadh4	International Journal of Innovations in Engineering and Science, Vol. 3, No.6, 2018	Pagerank, Hadoop, Data Analysis.	**Adv:-**To accelerate the PageRank computation and reduced the PageRanks of spam web pages **Dis:-** Computing the PageRank score for a large web graph using in memory matrix format is certainly not feasible.
3	Improved Parallel Computation of PageRank for Web Searching	Hema Dubey1 , Nilay Khare2 and AppuKuttan KK	Indian Journal of Science and Technology, Vol 9(S1), December 2016	Indian Journal of Science and Technolog y, Vol 9(S1), December 2016	**Adv:-** The execution time of mapper significantly reduced as the number of worker node increases. **Dis:-**The method is very slow to reach

					convergence. The problem with computing PageRank for very big graph in traditional way is we need machine with high processing power and huge memory
4	Google Page Rank Algorithm and It's Updates	Priyanka Patel , Anuj Joshi	ResearchGate ,October 2018	Google, Search Engine, PageRank, Page Rank Algorithm, Search Result	**Adv:-** Google is more suitable when searching for particular keyword but in case of not getting exact match it also shows the relevant data. **Dis:-** Data relevant to page category will surely boost its ranking while irrelevant content might affect a website's rank negatively
5	Efficient Parallel Computation of PageRank	Christian Kohlsch¨utter, PaulAlexandruChirit a, and Wolfgang Nejdl	L3S Research Center / University of Hanover,2017	Search Engine, PageRank, Page Rank Algorithm, Search Result	**Adv:-** Fast re-balancing for incremental crawling **Dis:-** The method is very slow to reach convergence.
6	Real-time Twitter data analysis using Hadoop ecosystem	Anisha P. Rodrigues1 * and Niranjan N. Chiplunkar1	Cogent Engineering (2018)	Apache Flume; Apache Hive; Hadoop	**Adv:-**Pig Latin and HiveQL languages ease the complexity of writing complex MapReduce programs

					Dis: The method is very slow to reach convergence
7	A Sentiment Analysis of Twitter Data using Hadoop Framework	Jeevitha, Raksha, Vipin N	International Journal of Innovative Research in Computer & Communication Engineering,2017	Twitter, HDFS, Hadoop, Map-Reduce, Sentiment Analysis	**Adv:** It achieves considerable accuracy than the existing techniques **Dis:** The problem with computing PageRank for very big graph in traditional way is we need machine with high processing power and huge memory
8	Analysis of User Behavior for Twitter Posts on Hadoop	Priya Gupta1	International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 05 | May -2017	Big data, Sentiment Analysis, Hadoop, Twitter	**Adv:** User behavioural analysis system that predicts the behaviour of user whether the user is in drifting mode, positive or negative on the basis of the tweet_id of user on live social twitter data
9	Tweet Analysis: Twitter Data processing Using Apache Hadoop	Manoj Kumar Danthala	International Journal Of Core Engineering & Management (IJCEM) Volume 1, Issue 11, February 2015	Index Terms—BigData , Hadoop, MapReduce	**Adv:** Processing time and retrieving capabilities are made very easy **Dis:** Algorithm design for handling the problems raised by the huge data volume and the dynamic data characteristics.

10	Exploring Twitter networks in parallel computing environment	Bo Xu , Yun Huang	Research Gate,2016	Twitter , Parallel Computing	**Adv:** The unfollow links are interdependent,they are highly reciprocal and clustered.
11	Measuring User Reputation on Twitter Using Page Rank Algorithm	N.Abirami1 , K.Manohari	International Journal of Engineering Science Invention	PageRank Algorithm, Regression, Social media, Twitter	**Adv:** To improve the page rank algorithm achieves for user ranking problem and spamming issues. **Dis:**Algorithm design for handling the problems raised by the huge data volume and the dynamic data characteristics.
12	Social Network Analysis of Twitter to Identify Issuer of Topic using PageRank	Sigit Priyanta1 , I NyomanPrayanaTrisna	International Journal of Advanced Computer Science and Applications, Vol. 10, No. 1, 2019	Twitter ranking; social network analysis; graph based algorithm; PageRank	**Adv:** Increase the data into million as well to try other graph-based algorithm other than PageRank and its modification derivatives with more analysis with other properties and centrality methods. **Dis:**Computing the PageRank score for a large web graph using in memory matrix format is certainly not feasible
13	Sentiment analysis of	Divyesh Patel	OMICS International	Twitter, HDFS,	**Adv:** The new approach for

	twitter data using parallel write approach of replica placement in Hadoop cluster		Conference , 2016	Hadoop, Map-Reduce, Sentiment Analysis	efficient replica placement in Hadoop DFS which can improve throughout and data transfer rate. **Dis:** The total volume of tweets is tremendously high, and it takes a long time to process
14	Parallel and Improved PageRank Algorithm for GPU-CPU Colaborative Environment	PrasannChoudhari , EikshithBaikampadi , PareshPatil , SanketGadekar	(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (3) , 2016	Graphics Processing Unit, Compute Unified Device Architecture, Peripheral Component Interconnect	**Adv:** Increases the performance in terms of Time taken to compute PageRank of a given dataset. **Dis:** Convergence condition and the lack of memory coalescing and utilization of CUDA blocks and threads
15	PartitionBased Parallel PageRank Algorithm	ArnonRungsawang and BunditManaskasemsak	11th International Conference on Parallel and Distributed Systems (ICPADS'16)	Partitioning algorithms, clustering algorithms, Iterative algorithms	**Adv:** We can efficiently run on a parallel environment like the PC cluster **Dis:** If we continually increase the number of processors, the running time will also continually increase, or the overall speedup performance will be rapidly drop.

3. TECHNICAL SPECIFICTION

3.1. DESCRIPTION OF OUR PROJECT:

Our project calculates the page rank of 300000 nodes in the HADOOP DISTRIBUTED FILE SYSTEM installed on UBUNTU OS. We have used python for parallel page rank algorithm which is executed on HADOOP. We used and named the following python codes:

1. job-1reducer.py- is used to reduce operations in the initial stage.

2. gen_iter_val.py — generates the required iteration value.

3. job-2reducer.py — used to reduce operations in the second step

4. Total_user.py - Calculates total users

5. Total_zero_degree.py — It calculates the total zero degree of all nodes.

6. combine.py- It combines P_value which is generated by all reducers. It prints the sum of p difference as output.

7. Sort-1.py- It sorts the final result in descending order.

To execute operations 1 and 2, we use the shell command, which we have written in 'job1.sh'.

To execute the operations 3,4,5,6 and 7 we use the shell command we wrote in 'job 2.sh'.

3.2. PLATFORM DETAILS:

- Ubuntu system required.
- Install Hadoop on Ubuntu operating system.
- TWITTER USER ANALYSIS page is performed by Hadoop using the Python script by RANK ALGORITHM.
- Initially the data is extracted from the Twitter API using specific 3rd party applications such as FLUME.
- Execute the program in Hadoop. The data is distributed across various Hadoop servers.
- Processing takes place at each node and the combined output is displayed.

3.3. MODULE WISE DESCRIPTION:

MAP REDUCE:

MapReduce is a processing technique and a program model for distributed computing. The MapReduce algorithm has two important functions, namely, Map and Reduce. The map takes one set of data and converts it to another set of data, where individual elements are broken into tuples (key / value pairs). Second, reduce the function, which takes the output from the map as an input and combines those data tuples into a smaller set of tuples. As the name sequence of MapReduce implies, the task that is minimized after the map task is always executed. The main advantage of MapReduce is that it is easy to scale data processing on many computing nodes. Under the MapReduce model, data processing primitives are called mappers and reducers. Decomposing data processing applications into mappers and reducers is sometimes non-subjective. But, once we write an application in MapReduce form, the application scales to run hundreds, T hundred or even tens of thousands of machines in a cluster, this is just a configuration change. It is this simple scalability that has attracted many programmers to use the MapReduce model.

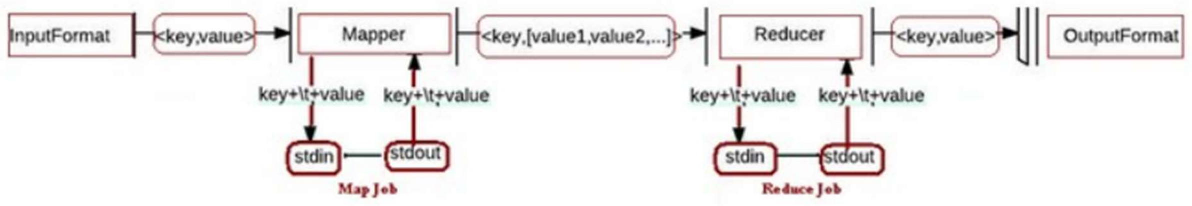


Figure 3.1

Inputs and Outputs (Python Perspective):

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a MapReduce job − (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

Table 3.1

PAGE RANK ALGORITHM:

The PageRank algorithm outputs the probability distribution used to

demonstrate the probability that the person clicking on the link will arrive at a particular page. PageRank can be calculated for a collection of documents of any size. It is believed in many research papers that the distribution is divided equally among all the documents in the collection at the beginning of the computational process. PageRank computations require multiple passes, called "iterations", to more closely adjust the estimated PageRank values through the collection to reflect the theoretical true value.

4. DESIGN

STEP-1:

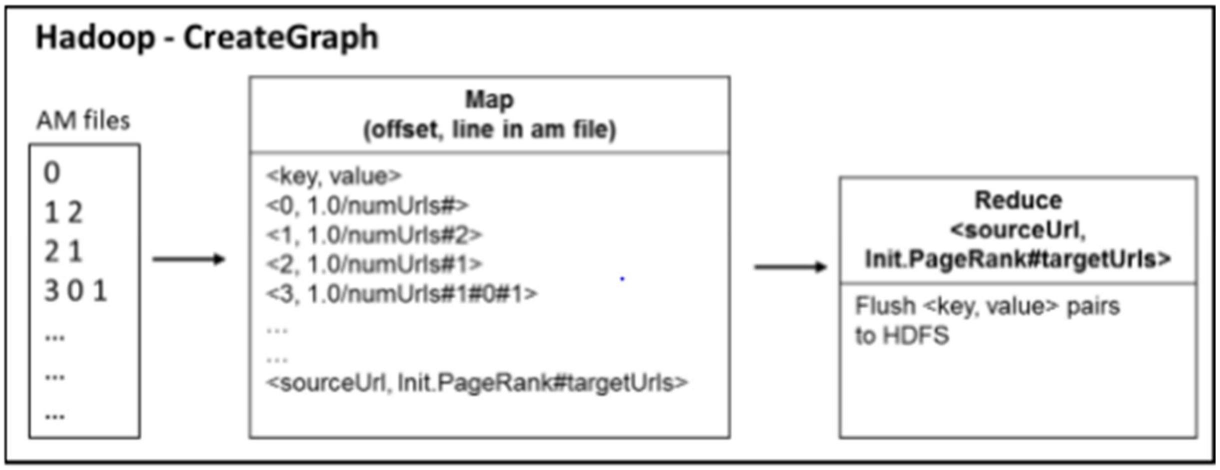


Figure 4.1

STEP-2:

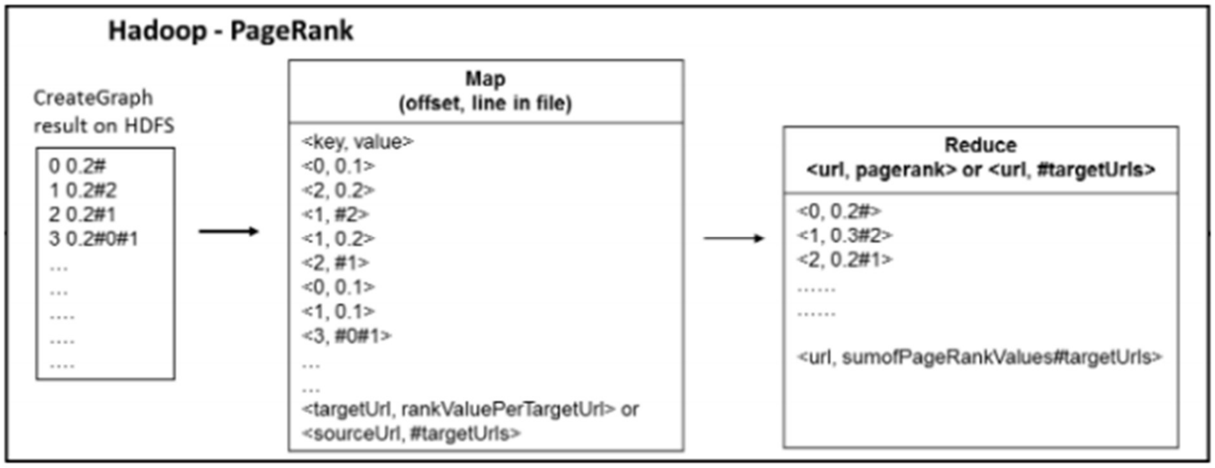


Figure 4.2

STEP-3:

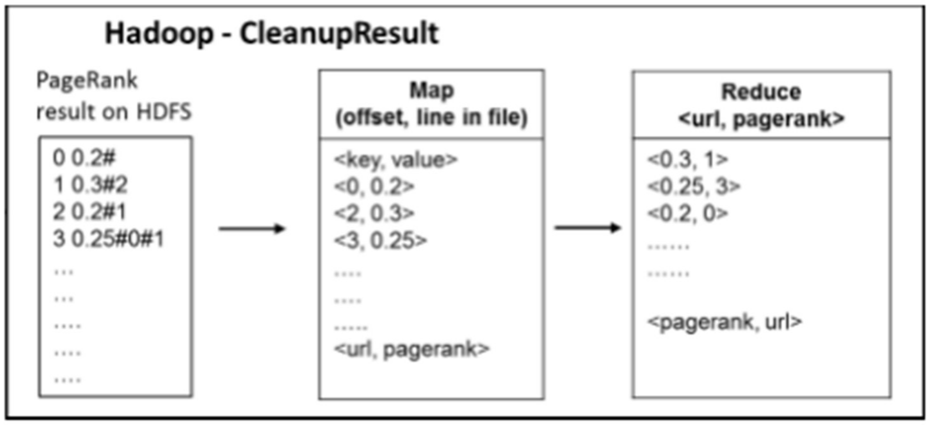


Figure 4.3

<u>HADOOP INSTALLATION:</u>

INSTALLATION OF HADOOP ON UBUNTU:

• HADOOP can be installed in three different modes. They are as follows:

1. Standalone Mode

2. Pseudo Distributed Mode

3. Distributed Mode

• Standalone mode works on a single node cluster.

• Standalone mode is suitable for understanding the basics of Hadoop.

• Pseudo distributed mode works on a single node cluster.

• Pseudo distributed mode works on a single server but emulates a multi-node environment.

• Distributed mode works on multi node clusters.

• In our project we are using HADOOP on standalone mode.

<u>PART 1:</u>

This contains installation of java and updating it.

1. sudo apt-get update

2. sudo apt-get purge openjdk*

3. sudo add-apt-repository ppa:webupd8team/java

4. sudo apt-get update

5. sudo apt-get install oracle-java8-installer

6. java -version

7. sudonano /etc/profile

A file opens. In that file we have to type the following command.

8. export JAVA_HOME=/usr

9. source /etc/profile

We have to disable IPv6 in order to make HADOOP work.

The following command are to be typed.

1. sudonano /etc/sysctl.conf

This opens a file. Move to the end of the file and type the following commands.

1.1 net.ipv6.conf.all.diable_ipv6=1

1.2 net.ipv6.conf.default.diable_ipv6=1

1.3 net.ipv6.conf.io.diable_ipv6=1

After typing these commands close the file by saving the file.

1.4 control+x+y+enter

2. sudo reboot

PART 2:

This part contains adding HADOOP USER on the ubuntu system.

1. sudoaddgrouphadoopgroup

2. sudoadduser -ingrouphadoopgrouphduser

Enter the password of your choice

Leave other tabs as default.

3. sudo apt-get install ssh1

4. sudosystemctl enable ssh

5. sudosystemctl start ssh

Switch to Hadoop user

6. su – hduser

<u>PART 3:</u>

It consists of setting and generation of RSA key

1. ssh-keygen -t rsa -P “ “

2. cat /home/hduser/.ssh/id_rsa.pub >> /home/hduser/.ssh/authorized_keys

3. cd .ssh/

4. chmod600 ./authorized_keys

5. ls -l

6. ssh-copy-id -I /home/hduser/.ssh/id_rsa.pub localhost

7. ssh localhost

The above command tests the local host.

<u>PART 4:</u>

INSTALLATION OF HADOOP:

open apache.org

download version 2.8.5 binary

copy the link location of the first link in the website.

1. wget “paste the link location”

2. cd ..

To get out of given directory

Extract the downloaded Hadoop file

3. tar -xvf Hadoop-2.8.5.tar.tz “tab”

Move the extracted content to the following folder

4. cd /home/hduser/

5. ls

6. sudo mv /hadoop-2.8.5 /usr/local/

7. sudo ln -sf /usr/local/hadoop-2.8.5/ /usr/local/hadoop

8. sudochown -R hduser:hadoopgroup /usr/local/hadoop-2.8.5/

PART 5:

It includes the configuration of hadoop.

1. su -hduser

2. nano ./.bashrc

It opens a file. Move to the end of the file and make the following changes.

3. #hadoop configuration

```
export HADOOP_PREFIX = /usr/local/hadoop
export HADOOP_HOME = /usr/local/hadoop
export HADOOP_MAPRED_HOME = ${HADOOP_HOME}
export HADOOP_COMMON_HOME = ${HADOOP_HOME}
export HADOOP_HDFS_HOME = ${HADOOP_HOME}
export YARN_HOME = ${HADOOP_HOME}
export HADOOP_CONF_DIR = ${HADOOP_HOMES}/etc/hadoop
```

#NATIVE PATH

```
export HADOOP_COMMON_LIB_NATIVE_DIR = ${HADOOP_HOME}/lib/native
export HADOOP_OPTS = “-Djava.library.path = $HADOOP_PREFIX/lib/native”
```

#JAVA PATH

```
export JAVA_HOME=”/usr”
```

#OS PATH

```
export PATH = $PATH : $HADOOP_HOME/bin:$
JAVA_HOME/bin:$HADOOP_HOME/sbin
```

Press control+x+y+enter

PART 6:

1. source ./.bashrc

2. nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh

Go to the end of the file and type the following command

3. export JAVA_HOME=”/usr”

4. press control + x + y +enter

5. cd /usr/loca/hadoop/

6. cd etc/hadoop

PART 7:

THIS INCLUDES CHANGING THE CONFIGURATION OF SOME XML FILES

1. nano core-site.xml

Go to the configuration and make the following changes

```
<configuration>

<property>

<name>fs,default.name</name>

<value> hdfs://localhost:9000</value>

</property>

</configuration>
```

Press control + x + y +enter

2. nano hdfs-site.xml

```
<configuration>

<property>

<name>dfs.replication</name>

<value> 1 </value>

</property>
```

```
<property>
<name>dfs.name.dir</name>
<value> file:/usr/local/hadoop/hadoopdata/hdfs/namenode </value>
</property>
<property>
<name>dfs.data,dir</name>
<value> file:/usr/local/hadoop/hadoopdata/hdfs/datanode </value>
</property>
</configuration>
```

Press control + x + y + enter

3. nanomapred-site.xml.template

Go to the end of the file

```
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value> yarn </value>
</property>
</configuration>
```

Press “control + x + y”

Change the name of the file to mapred-site.xml

Press “enter” and press “y

4. nano yarn-site.xml

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
```

```
</configuration>
```

Press “control + x + y + enter”

PART 8:

1. cd /usr/local/hadoop/bin

2. hd

3. hdfsnamenode -format

Look for “ INFOCommon.Storage:” directory and check whether it is successfully formatted or not.

4. cd /home/hduser/

PART 9:

IT INCLUDES STARTING THE SERVICES.

1. Start-dfs.sh

2. Start-yarn.sh

3. Jps

4. Ls

5. hadoop fs -ls

It gives the lists of all hadoop file systems.

EXISTING NORMAL DISTRIBUTION SYSTEM DESCRIPTION:

• Connecting a computer running on different operating systems is not as easy as it appears.

• Working with operating systems with different architecture causes confusion.

• Scheduling tasks for maximum utilization of available resources.

• Dynamic determination of threads. This means that we have to use threads whenever needed. And these should be distributed effectively between tasks.

• Scalability is another major issue with distributed computing.

• Synchronization of tasks and data racing is a major problem.

• Current methods of synchronization such as semiconductors, monitors, intercepts, remote process calls, object method invocation and message passing are not well-scaled in normal delivery systems.

• Communication on the local network of computers should be made secure to prevent any data theft.

• Distribution of tasks on computers with multiprocessors and multicores requires improvement in scheduling tasks.

• Availability of source code with parallel classes in case of other complex computation problems.

5. PROPOSED SYSTEM

• In this project we will be implementing a python program to provide PAGERANK ALGORITHM.

• APACHE HADOOP Software Library is a framework. It uses simple programming models to compute large data sets in clusters of computers.

• It is mainly used for computing programs on distributed systems.

• Since Twitter data has a large amount of data, we may need multiple processors and computers to run the program. The best option is to use HADOOP.

• Hadoop distributed file system provides a single server to thousands of machines or nodes. Each machine provides its own local calculation and storage.

• Hadoop increases the run time efficiency of a program. It has the capability of scheduling very complex mathematical problems and big data on its servers using the MAP-REDUCE concept.

NOVELTY OF THE PROJECT:

- The pagerage algorithm can be executed on the local operating system when the number of page rank algorithms is small.
- But there are millions of users and local systems are not suitable for calculating the page rank of each user.
- To calculate it at a faster rate and increase the algorithm efficiency, we need this algorithm on parallel systems and distributed systems.
- There are mainly five job execution engines that support page rank algorithms that contain large amounts of data.
- There are as follows:

1.Dryad

2.Hadoop

3.MPI

4.PREGEL

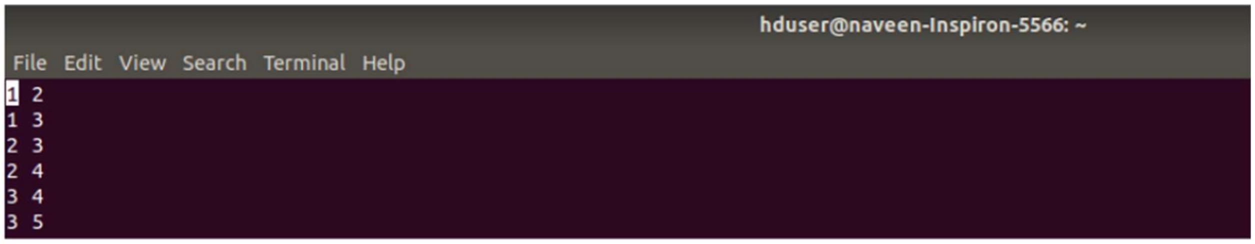
5.TWISTER

- So, for analyzing large amount of data HADOOP plays an important role because it caneasily handle tera bytes of data.
- Basically, Hadoop is mainly created for batch processing of data.
- It means that it is not pretty suitable for stream processing which are in real time systems.

- Because of advancement in present days technology and developments, Hadoop can handle.
- the stream processing with combining with some projects like STORM-YARN, SPARK,
- IMPALA, APACHE-DRILL etc.

6. <u>RESULTS:</u>

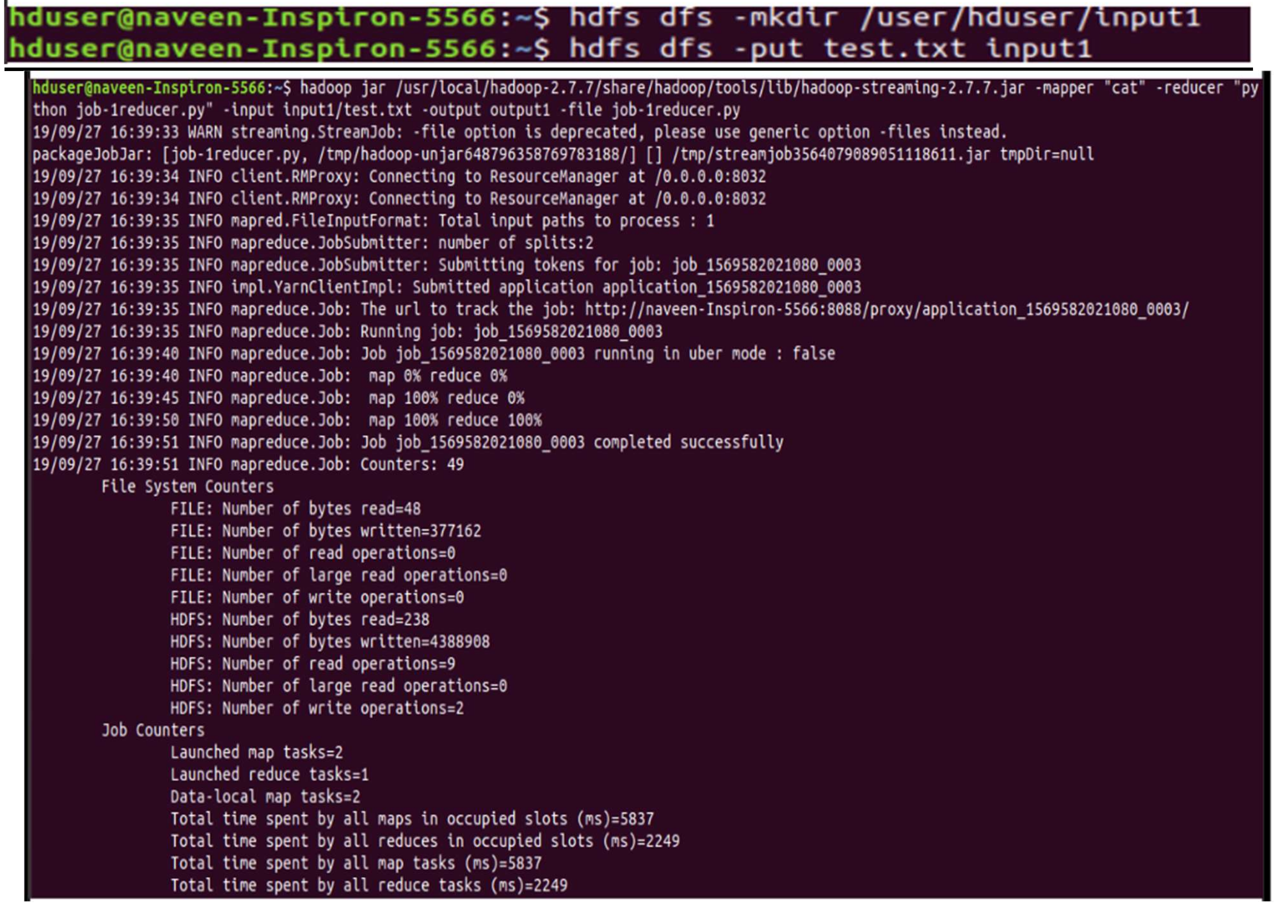
INPUT1:


hduser@naveen-Inspiron-5566: ~
File Edit View Search Terminal Help

```
1 2
1 3
2 3
2 4
3 4
3 5
```

Figure 6.1

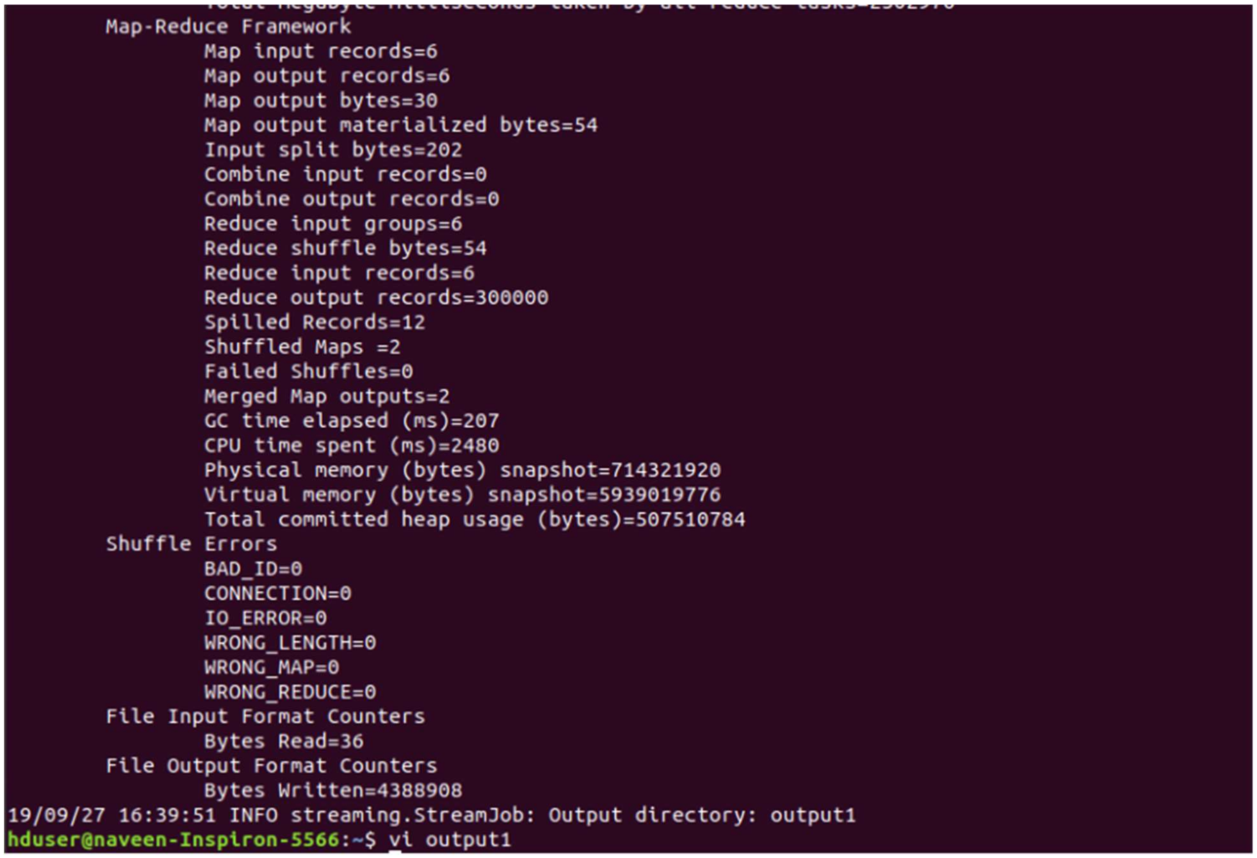
EXECUTION OF JOBREDUCER1.py:



```
hduser@naveen-Inspiron-5566:~$ hdfs dfs -mkdir /user/hduser/input1
hduser@naveen-Inspiron-5566:~$ hdfs dfs -put test.txt input1
```

```
hduser@naveen-Inspiron-5566:~$ hadoop jar /usr/local/hadoop-2.7.7/share/hadoop/tools/lib/hadoop-streaming-2.7.7.jar -mapper "cat" -reducer "py
thon job-1reducer.py" -input input1/test.txt -output output1 -file job-1reducer.py
19/09/27 16:39:33 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [job-1reducer.py, /tmp/hadoop-unjar648796358769783188/] [] /tmp/streamjob3564079089051118611.jar tmpDir=null
19/09/27 16:39:34 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/09/27 16:39:34 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/09/27 16:39:35 INFO mapred.FileInputFormat: Total input paths to process : 1
19/09/27 16:39:35 INFO mapreduce.JobSubmitter: number of splits:2
19/09/27 16:39:35 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1569582021080_0003
19/09/27 16:39:35 INFO impl.YarnClientImpl: Submitted application application_1569582021080_0003
19/09/27 16:39:35 INFO mapreduce.Job: The url to track the job: http://naveen-Inspiron-5566:8088/proxy/application_1569582021080_0003/
19/09/27 16:39:35 INFO mapreduce.Job: Running job: job_1569582021080_0003
19/09/27 16:39:40 INFO mapreduce.Job: Job job_1569582021080_0003 running in uber mode : false
19/09/27 16:39:40 INFO mapreduce.Job:  map 0% reduce 0%
19/09/27 16:39:45 INFO mapreduce.Job:  map 100% reduce 0%
19/09/27 16:39:50 INFO mapreduce.Job:  map 100% reduce 100%
19/09/27 16:39:51 INFO mapreduce.Job: Job job_1569582021080_0003 completed successfully
19/09/27 16:39:51 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=48
                FILE: Number of bytes written=377162
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=238
                HDFS: Number of bytes written=4388908
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=5837
                Total time spent by all reduces in occupied slots (ms)=2249
                Total time spent by all map tasks (ms)=5837
                Total time spent by all reduce tasks (ms)=2249
```

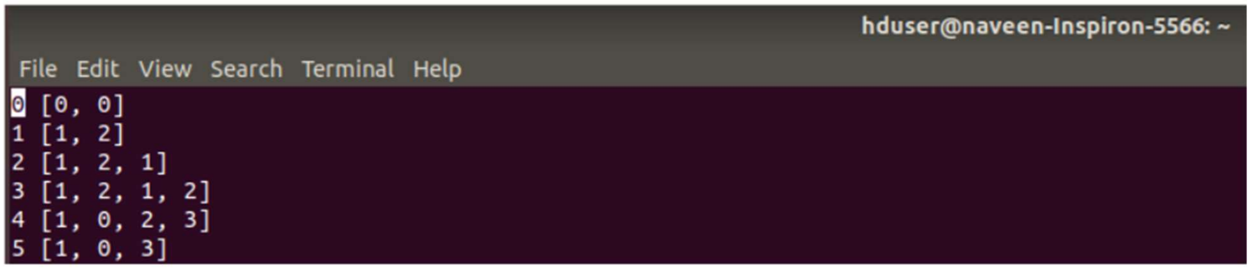
Figure 6.2



```
        Map-Reduce Framework
                Map input records=6
                Map output records=6
                Map output bytes=30
                Map output materialized bytes=54
                Input split bytes=202
                Combine input records=0
                Combine output records=0
                Reduce input groups=6
                Reduce shuffle bytes=54
                Reduce input records=6
                Reduce output records=300000
                Spilled Records=12
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=207
                CPU time spent (ms)=2480
                Physical memory (bytes) snapshot=714321920
                Virtual memory (bytes) snapshot=5939019776
                Total committed heap usage (bytes)=507510784
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=36
        File Output Format Counters
                Bytes Written=4388908
19/09/27 16:39:51 INFO streaming.StreamJob: Output directory: output1
hduser@naveen-Inspiron-5566:~$ vi output1
```

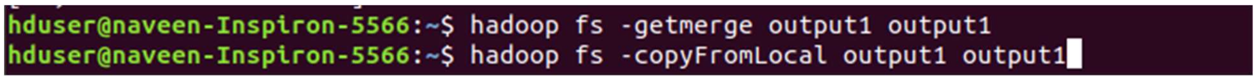
Figure 6.3

OUTPUT1:


hduser@naveen-Inspiron-5566: ~
File Edit View Search Terminal Help

```
0 [0, 0]
1 [1, 2]
2 [1, 2, 1]
3 [1, 2, 1, 2]
4 [1, 0, 2, 3]
5 [1, 0, 3]
```

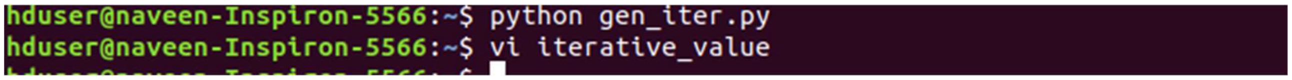
Figure 6.4



```
hduser@naveen-Inspiron-5566:~$ hadoop fs -getmerge output1 output1
hduser@naveen-Inspiron-5566:~$ hadoop fs -copyFromLocal output1 output1
```

Figure 6.5

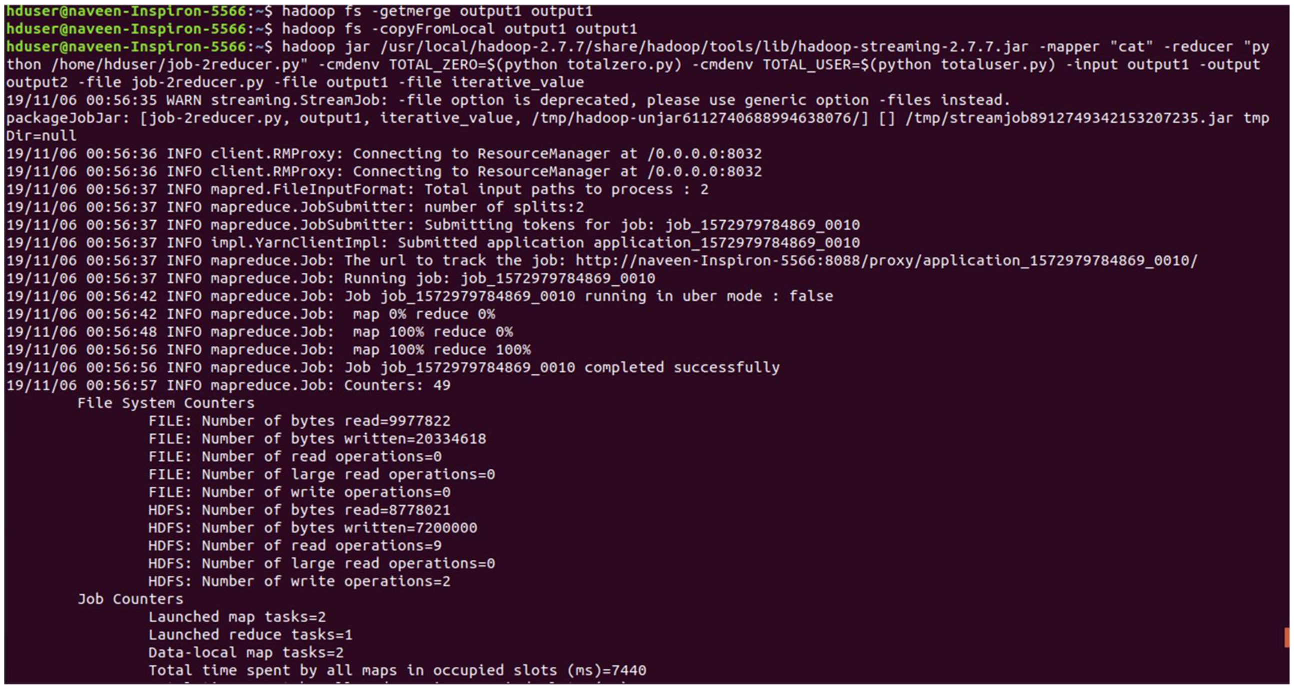
python iterator generator:



```
hduser@naveen-Inspiron-5566:~$ python gen_iter.py
hduser@naveen-Inspiron-5566:~$ vi iterative_value
```

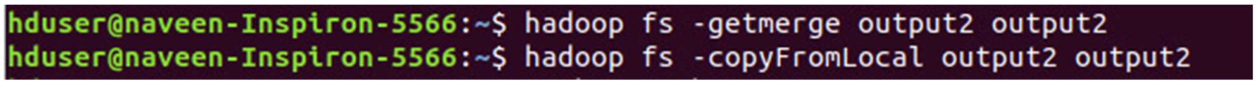
Figure 6.6

Output2 command:-



```
hduser@naveen-Inspiron-5566:~$ hadoop fs -getmerge output1 output1
hduser@naveen-Inspiron-5566:~$ hadoop fs -copyFromLocal output1 output1
hduser@naveen-Inspiron-5566:~$ hadoop jar /usr/local/hadoop-2.7.7/share/hadoop/tools/lib/hadoop-streaming-2.7.7.jar -mapper "cat" -reducer "py
thon /home/hduser/job-2reducer.py" -cmdenv TOTAL_ZERO=$(python totalzero.py) -cmdenv TOTAL_USER=$(python totaluser.py) -input output1 -output
output2 -file job-2reducer.py -file output1 -file iterative_value
19/11/06 00:56:35 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [job-2reducer.py, output1, iterative_value, /tmp/hadoop-unjar6112740688994638076/] [] /tmp/streamjob8912749342153207235.jar tmp
Dir=null
19/11/06 00:56:36 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/11/06 00:56:36 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
19/11/06 00:56:37 INFO mapred.FileInputFormat: Total input paths to process : 2
19/11/06 00:56:37 INFO mapreduce.JobSubmitter: number of splits:2
19/11/06 00:56:37 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1572979784869_0010
19/11/06 00:56:37 INFO impl.YarnClientImpl: Submitted application application_1572979784869_0010
19/11/06 00:56:37 INFO mapreduce.Job: The url to track the job: http://naveen-Inspiron-5566:8088/proxy/application_1572979784869_0010/
19/11/06 00:56:37 INFO mapreduce.Job: Running job: job_1572979784869_0010
19/11/06 00:56:42 INFO mapreduce.Job: Job job_1572979784869_0010 running in uber mode : false
19/11/06 00:56:42 INFO mapreduce.Job:  map 0% reduce 0%
19/11/06 00:56:48 INFO mapreduce.Job:  map 100% reduce 0%
19/11/06 00:56:56 INFO mapreduce.Job:  map 100% reduce 100%
19/11/06 00:56:56 INFO mapreduce.Job: Job job_1572979784869_0010 completed successfully
19/11/06 00:56:57 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=9977822
                FILE: Number of bytes written=20334618
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=8778021
                HDFS: Number of bytes written=7200000
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=7440
```

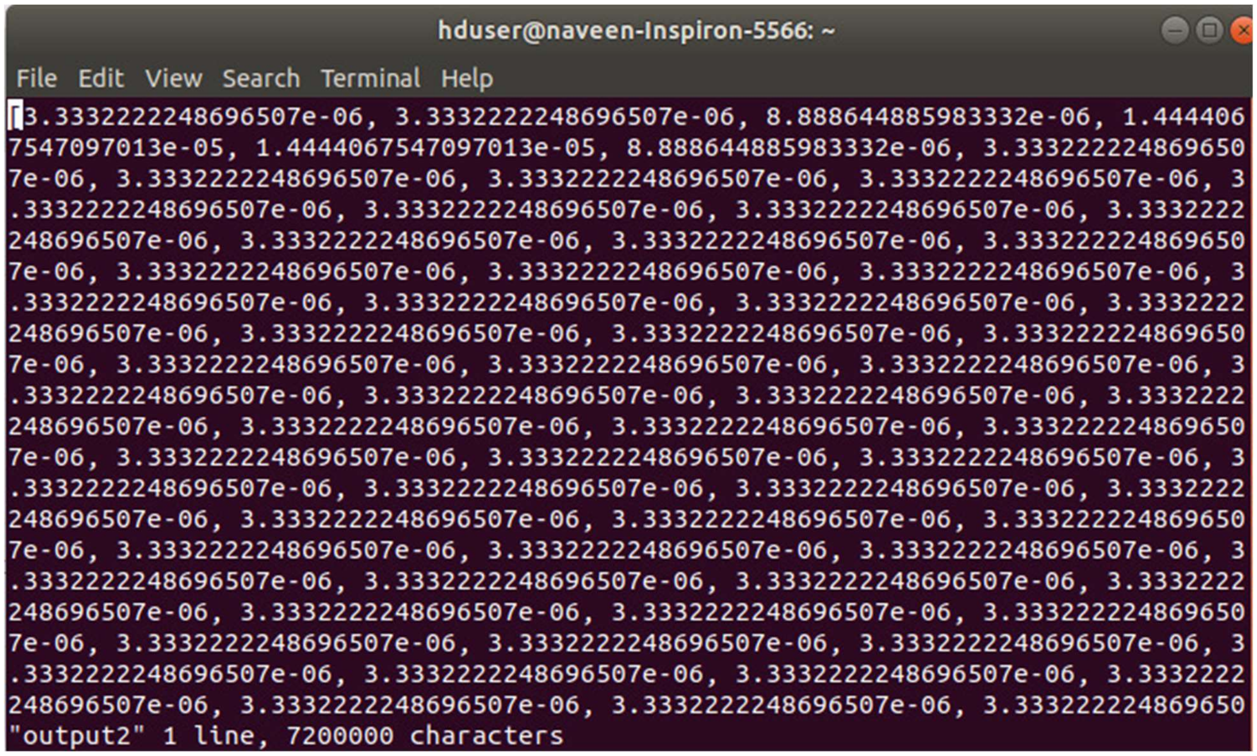
Figure 6.7



```
hduser@naveen-Inspiron-5566:~$ hadoop fs -getmerge output2 output2
hduser@naveen-Inspiron-5566:~$ hadoop fs -copyFromLocal output2 output2
```

Figure 6.8

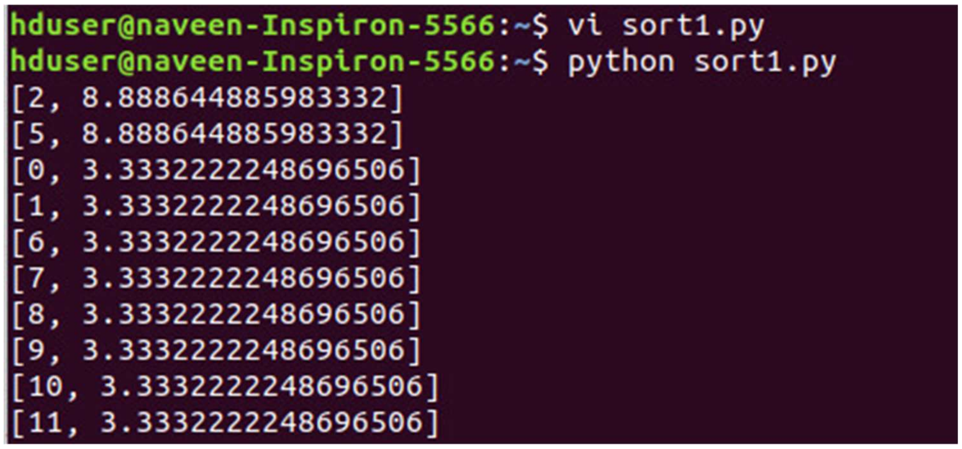
Output 2:


hduser@naveen-Inspiron-5566: ~
File Edit View Search Terminal Help

```
[3.3332222248696507e-06, 3.3332222248696507e-06, 8.888644885983332e-06, 1.444406
7547097013e-05, 1.4444067547097013e-05, 8.888644885983332e-06, 3.33322224869650
7e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3
.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222
248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.33322224869650
7e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3
.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222
248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.33322224869650
7e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3
.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222
248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.33322224869650
7e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3
.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222
248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.33322224869650
7e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3
.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222
248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.33322224869650
7e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3
.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.3332222
248696507e-06, 3.3332222248696507e-06, 3.3332222248696507e-06, 3.33322224869650
"output2" 1 line, 7200000 characters
```

Figure 6.9

Sorted Rank Values:



```
hduser@naveen-Inspiron-5566:~$ vi sort1.py
hduser@naveen-Inspiron-5566:~$ python sort1.py
[2, 8.888644885983332]
[5, 8.888644885983332]
[0, 3.333222248696506]
[1, 3.333222248696506]
[6, 3.333222248696506]
[7, 3.333222248696506]
[8, 3.333222248696506]
[9, 3.333222248696506]
[10, 3.333222248696506]
[11, 3.333222248696506]
```

Figure 6.10

7. <u>CONCLUSION</u>

In this project, we have analyzed the twitter user by page rank algorithm using map reduce concept in Hadoop by explaining the various advantages of Hadoop over normal distributed systems. We have described how Hadoop can be effectively managed and used by combining with other projects. We have also showcased how the distributed systems work. We have executed the data in parallel over the Hadoop network of computers provided by Hadoop. So Large companies that are dealing with heavy data can use Hadoop and make the analysis easier and faster.Some companies need to process large amount of data but they may not have supercomputers.For solving such issues with minimal cost and with less run time efficiency, Hadoop is very useful. In this project, we have executed a python program to analyze the number of users based on page rank algorithm. We used python for parallel page rank algorithm that is executed in Hadoop. So by using parallel as well as distributed systems we can increase the efficiency of the program and can find the page rank for millions of nodes in a single computer. We also got to know about how the performance is affected by parallel overheads.

REFERENCES

[1] Lawrence Page, Sergey Brin, Rajeev Motwani and Terry Winograd, “The PageRank Citation Ranking: Bringing Order to the Web,” StanfordInfoLab, No. 1999-66, Nov. 1999.

[2] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” OSDI 04: 6th Symposium on Operating Systems Design and Implementation, 2004.

[3] Mr. Swapnil A. Kale, Prof.SangramS.Dandge, Understanding the Big Data problems and their solutions using Hadoop MapReduce, ISSN 2319 – 4847,Volume 3.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop Distributed File System,” in the 26th IEEE Symposium on Mass Storage Systems and Technologies, pp. 1-10, May 2010.

[5] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Communications of the ACM, Vol. 51, Iss. 1, pp. 107-113, January 2008.

[6] M. Jamali and H. Abolhassani, “Different aspects of social network analysis,” in 2006 IEEE/WIC/ACM International Conference on WebIntelligence (WI 2006 Main Conference Proceedings)(WI’06), Dec 2006, pp. 66–72.