



MapReduce WordCount: Execution and Effects of Altering Parameters

Dr. A.J Singh, Vibha Sarjolta

Professor, Department of Computer Science, Himachal Pradesh University, Shimla, India

Research Scholar, Department of Computer Science, Himachal Pradesh University Shimla, India

ABSTRACT: With the growing advent of new technologies, devices and communication means, a large amount of data is being generated by everything around us that forms a part of what today is known as Big Data. In order to process such colossal quantities of data Apache's Hadoop acts as a software that deals with this type of huge and unstructured data with efficiency. In this paper one of the basic programs of Hadoop, that is, MapReduce WordCount is executed in a single node setup. The changes in the size of input files and the number of reduce tasks affecting the execution time of the program is studied. This paper aims at comparing the execution time of WordCount under varying conditions.

KEYWORDS: Hadoop; WordCount; Reducer; Mapper; Execution Time

I. INTRODUCTION

Big Data is an all-encompassing term for any collection of data sets so large and complex that it becomes difficult to process them using traditional data processing applications. Although Big Data doesn't refer to any specific quantity, the term is often used when speaking about petabytes and exabytes of data.

Scores of challenges are encountered when dealing with Big Data as its processing, storing and analysing. To process the large volumes of data from different sources, for fast processing, a tool is required that will help such a large amount of data to be processed easily and this is where Hadoop steps in. The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing. The Apache Hadoop software library is a framework that uses simple programming models for the distributed processing of large data sets across clusters of computers. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage [1].

Hadoop is specially designed for two core concepts: HDFS and MapReduce. Both are related to distributed computation. Hadoop architecture is primarily a distributed master slave architecture that consists of a single master and many slaves. The Hadoop Distributed File System (HDFS) is used for storage and MapReduce for computational capabilities. The functions of Hadoop in the architecture are data partitioning and parallel computation of large datasets. Its storage and computational capabilities scale with the addition of hosts to a Hadoop cluster, and can reach volume sizes in the petabytes on clusters with thousands of hosts [2].

The MapReduce master schedules the computational work on the slave nodes and organizes where the computational work will be scheduled. The HDFS master is responsible for storing the files and partitioning the storage across the slave nodes and keeping track of where data is located.

II. RELATED WORK

Since the MapReduce WordCount is one of the basic programs in Hadoop a significant amount of work has been done in this area. In [3] the authors proposed a simple and general MapReduce performance model for better understanding the impact of each component on overall program performance, and verified it in a small cluster. The results they found indicated that the model can predict the performance of MapReduce system and its relation to the configuration. According to the model, performance can be improved significantly by modifying Map split granularity and number of reducers without modifying the framework. The authors in [4] discussed the various MapReduce applications like pi, wordcount, grep, Terasort in their paper. These applications are already present in the Hadoop folder and its code be easily accessed. They ran programs showing execution of these applications on a Hadoop cluster. In the paper, performance had been shown with respect to and the number of nodes. The paper [5] found that as the number of nodes



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

increases, execution time decreases. The whole process of MapReduce processing and building up a cost function that explicitly models the relationship between the amount of input data, the available system resources (Map and Reduce slots), and the complexity of the Reduce function for the target MapReduce job. The model parameters were learned from test runs with a small number of nodes. Based on this cost model, a number of problems can be solved, such as the optimal amount of resources that can minimize the financial cost with a time deadline or minimize the time under certain financial budget. Experimental results showed that this cost model performs well on tested MapReduce programs.

In [6] the authors describe performing a WordCount Map-Reduce Job in Single Node Apache Hadoop cluster and compressing data using Lempel-Ziv-Oberhumer (LZO) algorithm. It was found that the LZO compression format was designed considering speed as priority, it decompresses about twice as fast as gzip. The results show that the LZO file is slightly larger than the corresponding gzip file, but both are much smaller than the original uncompressed file. Additionally, the LZO file compressed nearly five times faster, and decompressed over two times faster.

III. EXECUTION OF MAPREDUCE WORDCOUNT

Word Count is the “Hello World” of Hadoop programming. **WordCount** example reads text files and counts the occurrence of each word. The input is text files and the output is also text files, each line of which contains a word and the count of how often it occurred, separated by a tab.

The MapReduce WordCount source code can be found in the examples folder of Hadoop from where it can be run. Another way of running the WordCount program, used here, is compiling the .java file and creating its jar file and then executing the program. The java source code for the WordCount program was taken from Apache official site [7] for the latest stable version of Hadoop 2.7.1. The program includes Mapper and Reducer interfaces to provide the map and reduce tasks.

Mapper

Mapper maps input key/value pairs to a set of intermediate key/value pairs. A given input pair may map to zero or many output pairs. The Hadoop MapReduce framework spawns one map task for each InputSplit generated by the InputFormat for the job [7]. All intermediate values associated with a given output key are subsequently grouped by the framework, and passed to the Reducer(s) to determine the final output. The Mapper outputs are sorted and then partitioned per Reducer. The total number of partitions is the same as the number of reduce tasks for the job.

Reducer

Reducer reduces a set of intermediate values which share a key to a smaller set of values. The number of reduces for the job can be set using `Job.setNumReduceTasks(int)`. Reducer has three primary phases: shuffle, sort and reduce [7]. Where in the shuffle phase the output of the mapper phase is collected which is then sorted and the reduce phase outputs' the value.

Steps to Run the Program

1. Starting Hadoop Daemons

The Hadoop daemons are started using the commands

```
$ start-dfs.sh  
$ start-yarn.sh
```

2. Creating a Directory in HDFS

In order to carry out any functions in Hadoop it is viable that the files that are used for input and output are stored in HDFS which was done using the following command

```
$ cd /usr/local/hadoop  
$ bin/hdfs dfs -mkdir /user
```

A directory named inputword was then created on the desktop that contains the text file [8] which served as an input in the WordCount Program. This folder was then put in HDFS using

```
$ bin/hdfs dfs -put 'inputword' /user
```

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

3. Compiling the Program

```
$ java -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.7.1.jar:$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.1.jar:$HADOOP_HOME/share/hadoop/common/lib/commons-cli-1.2.jar -d /home/hduser/Desktop/wordcount *.java
```

The path where java file is stored and the full name of the Hadoop version was given during the compilation of the program. After compilation three class files were created: main class(WordCount), TokenizerMapper and IntSumReducer which were saved together in a folder named wordcountc in wordcount directory.

4. Creating a jar

A jar file was created using the following

```
$ jar -cvf wordcountj.jar -C /home/hduser/Desktop/wordcount/wordcountc
```

5. Running the Program

```
$ bin/hadoop jar /home/hduser/Desktop/wordcount/wordcountj.jar WordCount /user/inputword outputword
```

Where, inputword is the input directory and outputword the output directory.

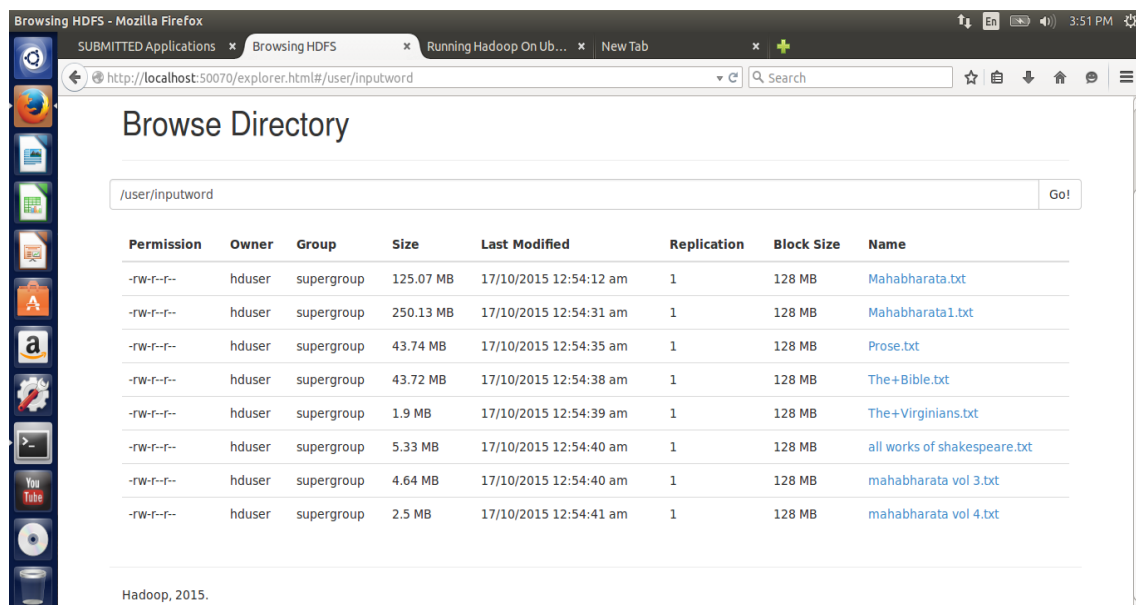


Figure 1. Input Files stored in HDFS

6. Output

The output of the WordCount program can be viewed either on the NameNode web interface, <http://localhost:50070> or using the following command in Ubuntu terminal

```
$ /usr/local/hadoop/bin/hadoop fs -cat outputword/part-r-00000
```

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

Where, outputword/part-r-00000 is the folder that contains the output of the program.

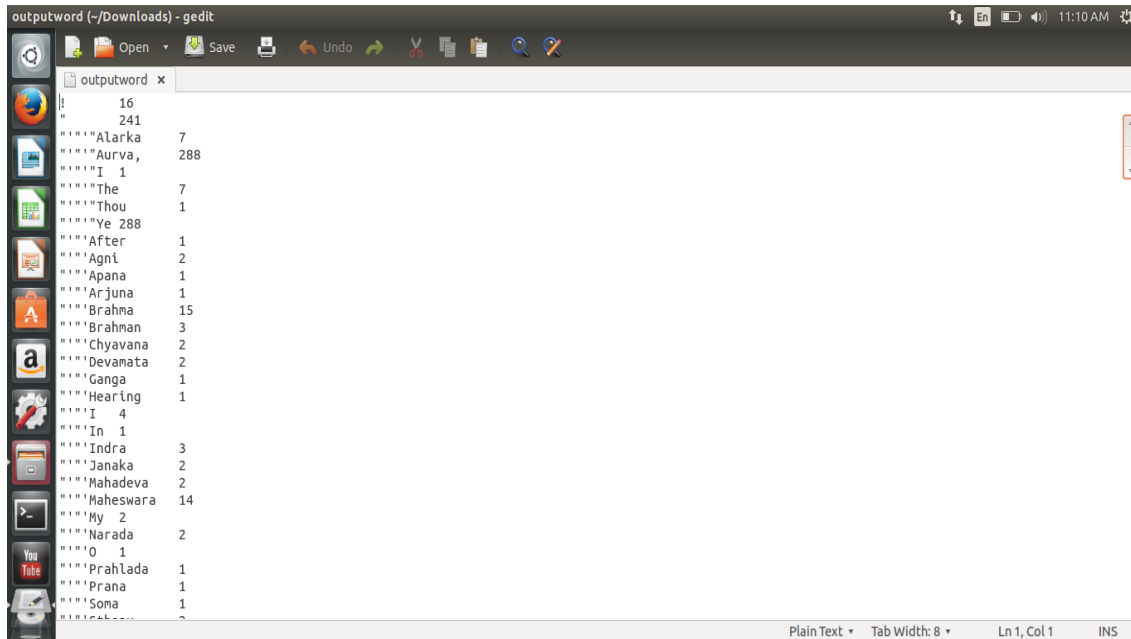


Figure 2. Output of the WordCount program

IV. SET UP AND RESULTS

The programs were run in Single node Hadoop Cluster using Ubuntu 14.04LTS as a platform which was run in a virtualized environment inside VirtualBox 4.3.2 Virtual Machine. Operating System: 64-bit Windows 7 Ultimate with Intel Core i3 CPU M 350 @ 2.27 GHz and 3GB of RAM. The version of Hadoop used is 2.7.1 and Sun JAVA jdk-1.7.0.

Changing the size of input file

The WordCount program had been run using input files of different data sizes and its effect on the execution time of the WordCount program is noted. Here, four files were taken as below:

- 500MB-containing 8 small files of varying lengths
- 1GB-containing 16 files of varying lengths
- 4GB-containing 7 files of varying lengths
- 8GB-single text file

INPUT FILE SIZE	EXECUTION TIME(in seconds)
500MB	366
1GB	655
4GB	1891
8GB	3593

Table 1. WordCount of varying file sizes with their respective execution time

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

From the table it can be observed that as the size of input in a WordCount program increase, the time taken also increases, but not in the same proportion. A 1GB file takes 655seconds to execute and if the time taken were to increase in the same proportion, then a 4GB file should have run in 2620seconds but instead it takes just 1891 seconds. This is approximately 27% less. Similarly, for an 8GB file the time is reduced by 31% if we considered the execution time to be eight times for what it was for input file is 1 GB.

Changing the number of Reduce tasks

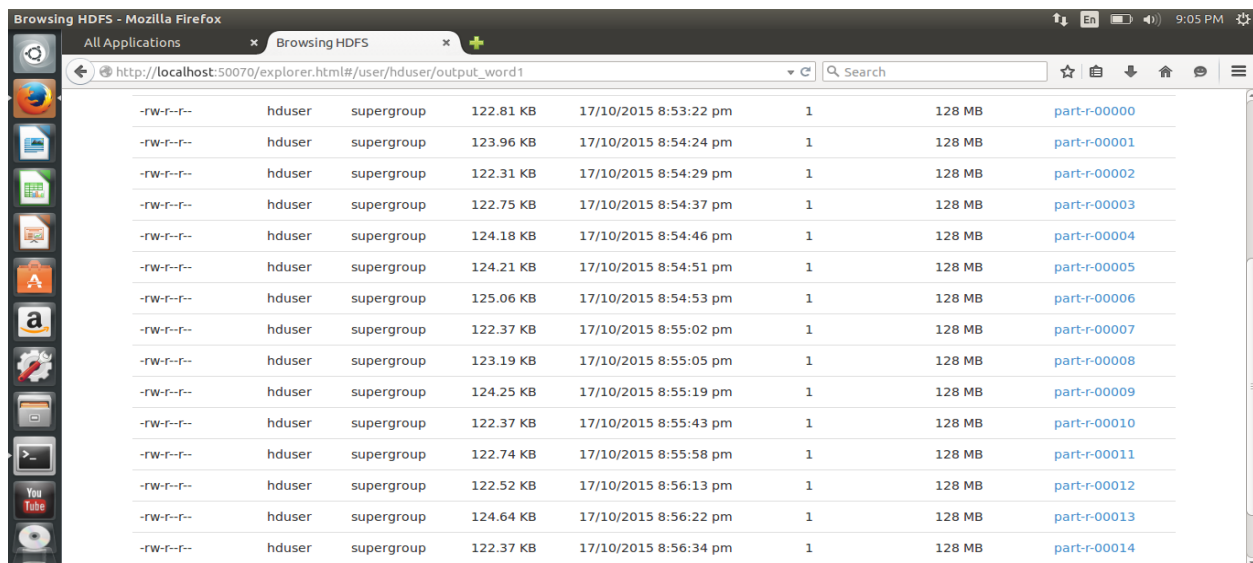
Choosing the right size for the tasks for your job can immensely change the performance in Hadoop. Increasing the number of tasks increases the framework overhead, but increases load balancing and lower the cost of failures [9].

In MapReduce, the WordCount program was altered using the `mapreduce.job.reduces` parameter. The input files in this case were the same as those used in the previous setup. Here, comparison was made between programs when the number of reduce tasks were altered from 5 to 15. This was done using the command given below:

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar  
wordcount -D mapreduce.job.reduces=5 /user/inputword /user/hduser/output_word
```

Here, the number of reduce jobs is specified as 5, similarly, the reduce tasks can be changed to 15 as:

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar  
wordcount -D mapreduce.job.reduces=15 /user/inputword /user/hduser/output_word
```



Permissions	User	Group	Size	Timestamp	Block Count	Size	File Name
-rw-r--r--	hduser	supergroup	122.81 KB	17/10/2015 8:53:22 pm	1	128 MB	part-r-00000
-rw-r--r--	hduser	supergroup	123.96 KB	17/10/2015 8:54:24 pm	1	128 MB	part-r-00001
-rw-r--r--	hduser	supergroup	122.31 KB	17/10/2015 8:54:29 pm	1	128 MB	part-r-00002
-rw-r--r--	hduser	supergroup	122.75 KB	17/10/2015 8:54:37 pm	1	128 MB	part-r-00003
-rw-r--r--	hduser	supergroup	124.18 KB	17/10/2015 8:54:46 pm	1	128 MB	part-r-00004
-rw-r--r--	hduser	supergroup	124.21 KB	17/10/2015 8:54:51 pm	1	128 MB	part-r-00005
-rw-r--r--	hduser	supergroup	125.06 KB	17/10/2015 8:54:53 pm	1	128 MB	part-r-00006
-rw-r--r--	hduser	supergroup	122.37 KB	17/10/2015 8:55:02 pm	1	128 MB	part-r-00007
-rw-r--r--	hduser	supergroup	123.19 KB	17/10/2015 8:55:05 pm	1	128 MB	part-r-00008
-rw-r--r--	hduser	supergroup	124.25 KB	17/10/2015 8:55:19 pm	1	128 MB	part-r-00009
-rw-r--r--	hduser	supergroup	122.37 KB	17/10/2015 8:55:43 pm	1	128 MB	part-r-00010
-rw-r--r--	hduser	supergroup	122.74 KB	17/10/2015 8:55:58 pm	1	128 MB	part-r-00011
-rw-r--r--	hduser	supergroup	122.52 KB	17/10/2015 8:56:13 pm	1	128 MB	part-r-00012
-rw-r--r--	hduser	supergroup	124.64 KB	17/10/2015 8:56:22 pm	1	128 MB	part-r-00013
-rw-r--r--	hduser	supergroup	122.37 KB	17/10/2015 8:56:34 pm	1	128 MB	part-r-00014

Figure 3. Screenshot showing the output of 1GB File when number of reducers is 15

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

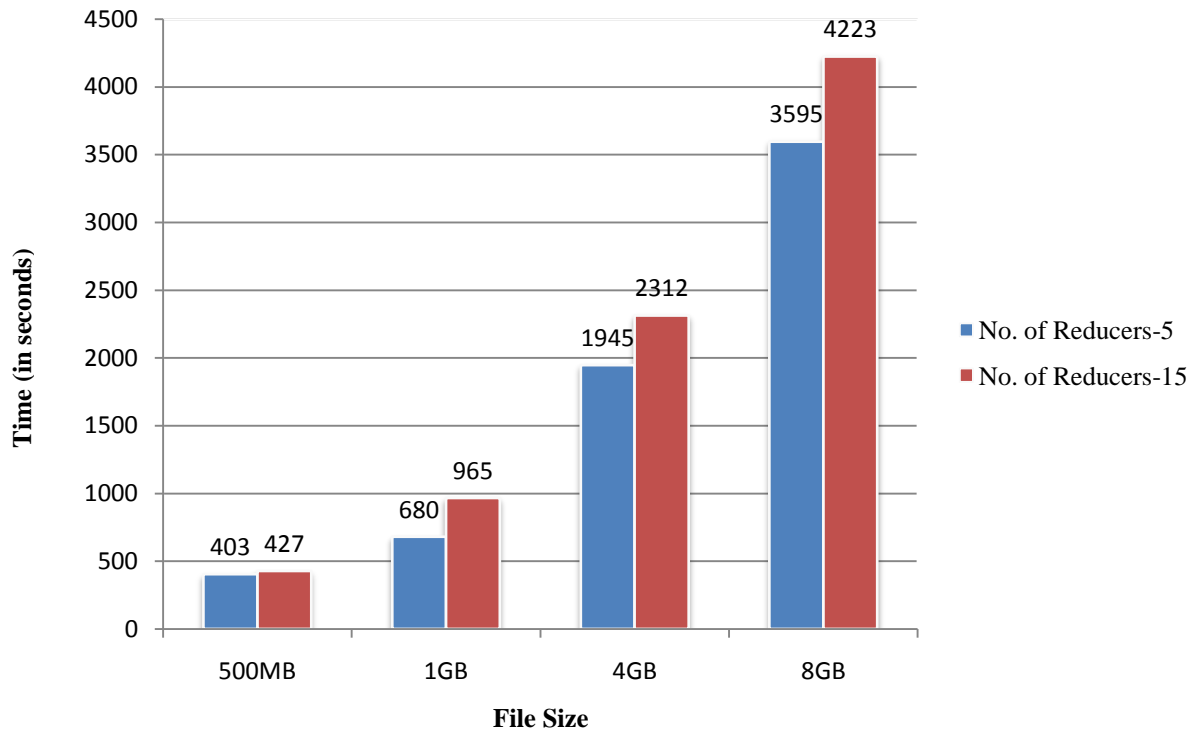


Figure 4. Comparison of WordCount program Execution time with different Reducers

Figure 4 represents the time taken by Files of sizes, 500MB, 1GB, 4GB and 8GB when the number of reduce tasks are increased from 5 to 15. The figure shows a slight increase in the time when the reducer task are increased. Although, the increase for smaller files is not very significant but for larger file, increasing the Reducers causes a significant change in its execution time.

V. CONCLUSION AND FUTURE WORK

The execution results showed that the time needed by a WordCount program for execution increases as the size of the input files is increased. It is observed that although increasing the size causes an increase in the execution time of the program, but large number of small files takes longer to execute as compared to a single larger file. It is also observed that the increase in time is not proportional and it decreases as the files are increased in size. Also, an increase in the number of reducers causes an increase in the time taken for the completion of WordCount program. We can also increase the input file to a much larger size like peta and zettabytes to observe the change in time and also analyze the performance. The setup used here is only a single node Hadoop cluster but it can also be extended to a multi node cluster for further research.

REFERENCES

1. Apache Hadoop, <https://hadoop.apache.org/>, Retrieved: 15-Oct-2015.
2. Alex Holmes, "Hadoop in practice", Manning Publications, Second edition, 2014.
3. Xiao Yang and Jianling Sun, "An analytical performance model of MapReduce", International Conference on Cloud Computing and Intelligence Systems, IEEE, 2011.
4. Maurya, M., Mahajan, S., "Performance analysis of MapReduce programs on Hadoop cluster", World Congress on Information and Communication Technologies (WICT), 2012.
5. Fengguang Tian and Keke Chen, "Towards Optimal Resource Provisioning for Running MapReduce Programs in Public Clouds", International Conference on Cloud Computing (CLOUD)", IEEE 2011.



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

6. Nandan Mirajkar, Sandeep Bhujbal , Aaradhana Deshmukh, “ *Perform wordcount Map-Reduce Job in Single Node Apache Hadoop cluster and compress data using Lempel-Ziv-Oberhumer (LZO) algorithm*” , ArXiv , July 2013.
7. MapReduce Tutorial, <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html> , Retrieved: 15-Oct-2015
8. <https://github.com/> Retrieved: 16-Oct-2015
9. Hadoop Wiki, <https://wiki.apache.org/hadoop/HowManyMapsAndReduces>, Retrieved: 16-Oct-2015

BIOGRAPHY

Vibha Sarjolta, Student pursuing M.Tech in Computer Science Engineering from Himachal Pradesh University, Department of Computer Science, Shimla, India.

Professor Dr. A.J Singh, is a Professor in the Department of Computer Science, Himachal Pradesh University, Shimla, India.