CSFC: A New Centroid Based Clustering Method to Improve the Efficiency of Storing and Accessing Small Files in Hadoop

R. Rathidevi, R. Parameswari

Abstract**:** ***In day to day life, the computer plays a major role, due to this advancement of technology collection of data from various fields are increasing. A large amount of data is produced by various fields for every second and is not easy to process. This large amount of data is called as Big data. A large number of small files also considered as Big data. It's not easy to process and store the small files in Hadoop. In the existing methods Merging technologies and Clustering Techniques are used to combine smaller files to large files up to 128 MB before sending it to HDFS in Hadoop. In the Proposed system CSFC (Clustering Small Files based on Centroid) Clustering Technique is used without mentioning the number of Clusters previously because if the clusters are mentioned before, all the files are clubbed within the limited number of clusters. In proposing system clusters are generated by depending on the number of related files in the dataset. The relevant files are combined up to 128 MB in a cluster. If any file is not relevant to the existing cluster or if the memory size reached 128MB then-new cluster will be generated and the file will be stored. It is easy to process the related files, comparing two irrelevant files. By using this method fetching data from the data node, it produces efficient result when comparing with other clustering techniques.***

Keywords : Datanode, Hadoop Distribuited File System, Hadoop, Namenode

I. INTRODUCTION

Big data can be categorized as the huge volume of data that goes beyond traditional tools, systems or process ' handling capabilities. It is a requirement for an organization that manages such data to establish techniques and architectures to tackle this huge data with the generation of this big data at such a rapid pace.

A. Hadoop

Apache Hadoop is an open-source software platform that uses a single programming model to spread large data sets through commodity computer clusters.[1] It is used to develop applications for data processing that are executed in a distributed computing environment. HADOOP-built applications operate on large data sets spread through commodity computer clusters.

Commodity computers are ubiquitous and inexpensive. These are primarily useful for achieving greater low-cost computing power. Large data sets mean the data will be stored in Tera bytes or Peta bytes. Commodity computers do not require a high-quality machine, i.e. a single programming model. The program is divided into many small fragments of work in MapReduce, each of which can run or re-execute on any node in the cluster. It has two frameworks HDFS, and Map reduce.

B. HDFS Framework

HDFS works efficiently with large files while comparing with small files. It contains two categories of nodes,ne Name node and multiple data nodes, which is in the form of Master-slave architecture. The name node holds the metadata of each file which is going to store in the data node. Every block in data node can hold 128 MB of data. Small files are not up to 128MB in size, It may be 10 KB, 20KB, like soon. [2] If each small hold a single block in the data node, then it occupies 150 bytes of memory space in the Name node to hold the metadata. If the number of files increased then the number of blocks in the data node is utilized more

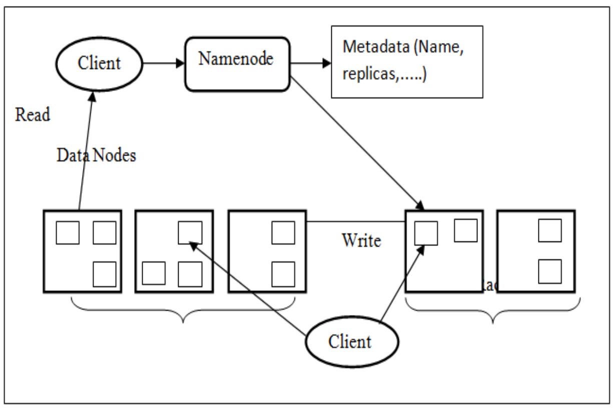
To store all the information like File folder, path, block number block location. Number of frames and slave related configuration of all the data sets in Hadoop cluster. That is, it actually knows where what data is stored. This metadata is stored in memory for quicker retrieval to minimize latency due to disk searches. Data Node is a daemon (a system running in the background) operating in the Hadoop Cluster on the' Slave Node In the HDFS, blocks are split in small chunks default size of 128MB. Such data frames are stored on the node of the slave. The real data is stored in it, Therefore, it occupies large number of disks to store data. The datanode performs read and write operations on disks.

Revised Manuscript Received on December 5, 2019.

R. Rathidevi, Research Scholar, Department of Computer Science, Vels Institute of Science, Technology & Advanced Studies, Chennai 600 043, India. rathirajsuresh@gmail.com

Dr. R. Parameswari, Associate Professor, Department of Computer Science, Vels Institute of Science, Technology & Advanced Studies, Chennai 600043, India dr.r.parameswari16@gmail.com

HDFS Architecture



.Fig. 1 : HDFS Architecture

C. Map Reduce Framework

Map Reduce is Hadoop's core component for data processing. Hadoop Map Reduce is an easy-to-write a software application that processes the large amount of structured and unstructured data stored in the Hadoop Distributed File System (HDFS). MapReduce works by splitting data into two phases: Map phase and Reduce phase. MapReduce is an application-writing computational method and computer platform running on Hadoop. Such MapReduce programs can process massive data on large clusters of computing nodes in parallel.

D. Small files problem in Hadoop

A large number of small files are stored in Hadoop will occupy Namenode memory separately for holding the metadata. The size of Namenode is limited, suppose if 1 million of small files want to store means, then for each file 150 bytes of memory allocated separately in Name node and moreover single block in data node also allocated separately for each and every small file. Data node stores every file redundantly 3 times for security purpose. So accessing files from datanode takes more time due to the large number of blocks occupied in datanode.

If we want access a file first we have to refer HDFS name node for metadata. In name node no difference of memory allocation for small files and big file, while accessing small file we need to access the name node frequently, because each and every small files are stored in separate data nodes. So it takes much time.

E. Clustering

Clustering is the method of splitting the entire data into groups based on some similarity of data. This is another famous clustering application. Multiple documents and you need to bring together related documents. Clustering allows us to organize these documents into the same clusters.

II. THE RELATED WORK

To solve small file problem in Hadoop,Small files are merged by using some techniques and then they are processed to produce good results in accessing the files.

[1]Sequence File System. Sequence File is a binary file system provided by Hadoop . It is possible to store the multiple small files in a unified manner. A sequence of binary key / value is composed of its data structure. The name of the file is called the value of key, and the result of the file is called the value ,separated by the operation of the MapReduce block and independent storage. It merges a large amount of small files[2] Hadoop Archive is a tool used to archive the small files in HDFS, Which reduces the utilization of Name node memory. In this Archive technique once HAR file created it cannot be modified, addition and deletion of files are not possible in HAR file. Have to recreate the HAR file again, so it’s not easy to use this method.[3] Vorapongkitipun et al proposed two ways to improve HDFS and HAR. Fundamental changes are made first to Hadoop HAR's indexing mechanism. [1]HAR uses 2-level indexed files to keep track of the files stored in the data nodes ' physical storage locations. The paper suggested a transition to a single index file with several partitions from a 2-level indexing technique

Passent M ElKafrawy et.al[4] proposed method is compared by using the two simulation tool kits with the initial HDFS. The cluster's output was calculated using two parameters first the amount of main memory that name node uses to write and the use of name node memory to store metadata. Secondly, the number of requests and the time taken to evaluate the name node overhead and the SPOF problem Merging technique is used to reduce the size of small files to reduce the name node memory consumption in an efficient manner. The metadata is stored in the form of Indexed linear file with hashing technique. While accessing the small files index file is used to access the particular file. Mohd Abdul Ahadet.al[5][6]proposed a method DM-sfs (Dynamic Merging based Small Files Storage) . In this method all the small are categorized based on the size and the they are combined based on the type of files. Merged files are stored in Name node and then they are processed.In that method the block are not utilized up to its level.

The revised version of HAR called "NHAR" [7]to improve the use of metadata by memory and to provide space for files to be added to existing HAR. "Hadoop Archive (HAR)" combines in a single large unit the multiple small files. The resulting big archive consists of an index file and the small initial data. HAR's main drawback is that it does not allow file attachments.

Ahad, Mohd Abdul, and Biswas, Ranjit[7] in their proposed method they created methods to handle a large number of small files efficiently. First, they analyze the size of each file by using a file size analyzer algorithm and then the File type analyzer algorithm is used to combine files based on its type of dynamic merging strategy. Then the merged files are encrypted for security purpose, and then it's sent to HDFS. In this merging system files are arranged according to its size and then they are merged to overcome this step in our proposed system clustering technique is used to combine the related files using centroid.

Hooda, Hanu, Nandal, Rainu [9] in their proposed work k-means algorithm is to cluster the related files. In k-means algorithm cluster should be declared while starting the clustering technique after that centroid point is identified,then data sets are added which is close to centroid. In that, all the files are clustered within that predefined number of clusters, but we are not sure that all the data sets are related to each other and they are fit within that predefined number of clusters to overcome this problem CSFC algorithm is generated.

III. THE PROPOSED WORK: CLUSTERING SMALL FILES BASED ON CENTROID (CSFC)

The CSFC clustering technique: CSFC (Clustering Small files based on centroids): In the proposed system The huge number of small files is combined before sending it to the HDFS.

The combining technique is carried out by clustering CSFC based on the type of file and they are combined up to 128 MB of size because the block size is 128MB. If the file size is more than 128MB that will not create a problem because large files can be handled efficiently in Hadoop, but to keep the metadata efficiently the small files are combin ed up to 128 MB as large file.

A. Text file converter:

This is responsible for identifying the file type. If the file type is text, then it is converted to numerical format of data for clustering the data are converted to numerical values to find the centroid value.

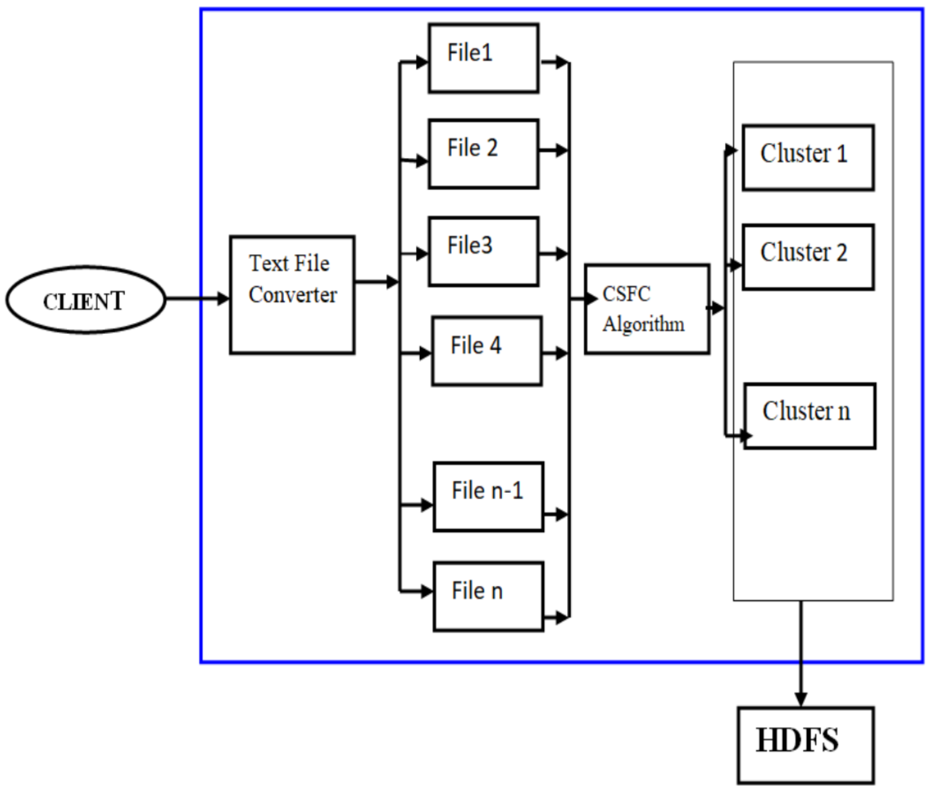


Fig 2. Architecture of CSFC Approach

B. File cluster

By using the CSFC clustering algorithm the files are combined by using the centroid randomly and clusters are generated, Membership matrix is used to update the centroid for the number of times to get the cluster result same after many iterations. The cluster size will be checked before adding a file to an existing cluster, if the cluster size less than 128 then that file will be added otherwise new cluster will be generated. In this algorithm number of clusters is not predefined because in CSFC clusters generated based on the related files by using the centroid. The files are stored in the form of a Sequence File (key-value pair) in order of their sizes in the newly partitioned memory block.

Contiguous memory block allocation to the individual files in the SequenceFile minimizes internal fragmentation. Nonetheless, the Sequence File's last block will be slightly unused (in some cases) which may be ignored in view of the performance improvements of the proposed solution. The SequenceFile is then translated to a MapFile and transferred to the encoding module.

C.Working on CSFC Approach

It gets input from the client and then it is pre-processed to check whether the data is in numerical data or text file, If it is text file then it converted to numerical data, then the data is combined by using CSFC approach.
Converting text files to numerical data set:

- Input file
- Check the file type
- If file type is text then
- For every column in table do
- Assign initial value of=0
- For every row in column i do
- Calculate Val=val+ data [i] [j]
- Calculate average =val/len(data)
- Result=append(val)
- Forward the converted numerical dataset to CSFC Algorithm.

In the proposed algorithm It calculates the membership matrix for all the Filesets and then randomly center points are chosen . Based on the data difference ratio between the center and file, they are combined and the cluster will be generated . The number of clusters is not pre-defined in this algorithm. Depending upon the types of files and size of file's clusters are generated.
CSFC Algorithm

- Calculate membership matrix
- For every data value in file do
- Assign Random number list= random (data values in every column)
- Generate average for random number list as member ship_mat
- Assign curr=0
- Repeat
- Calculate cluster centers
- Calculate Membership Matrix
- Increment curr
- Until Curr<= Maximum iteration
- If cluster size is less than 128 MB
- Add files into clusters
- Else, generate new cluster

The generated clusters are sent to name node for storing the metadata. In data node it maintains details of files stored in each block of data node for easy access. In data node files are stored redundantly minimum three times in the rack for security purpose.

```
# Number of Clusters
k = 0
# Maximum number of iterations
MAX_ITER = 20
# Number of data points
n = len(df)
# Fuzzy parameter
m = 2.00
df_full = pd.read_csv("tinput.csv")
columns = list(df_full.columns)
features = columns[:len(columns)-1]
class_labels = list(df_full[columns[-1]])
df = df_full[features]
```

In the above coding the number of clusters are not predefined so depending upon the file type the files are clustered and if it exceeds the size 128MB then-new cluster will be generated. So all files in clusters are related one and they are easily fetched from the data node efficiently..

D. DDR comparison

Table I: Data Difference Ratio between various algorithms.

Algorithm	Average of DDR	Time taken in nanoseconds	No. of Clusters
CSFC	2.328947368	2.5066853	3
k-means	2.374774775	4.9157381	2
Merging Algorithm	27.0201386	18.6666536	3

Data Difference ratio: It represents the data difference ratio values between each file and centroid value in the cluster. A centroid is chosen randomly from the existing dataset and then other data sets are placed based on that centroid in the cluster. This is repeated for the maximum number of iterations to get the same result without any change. Data difference the average of a difference value between the centroid and data set.

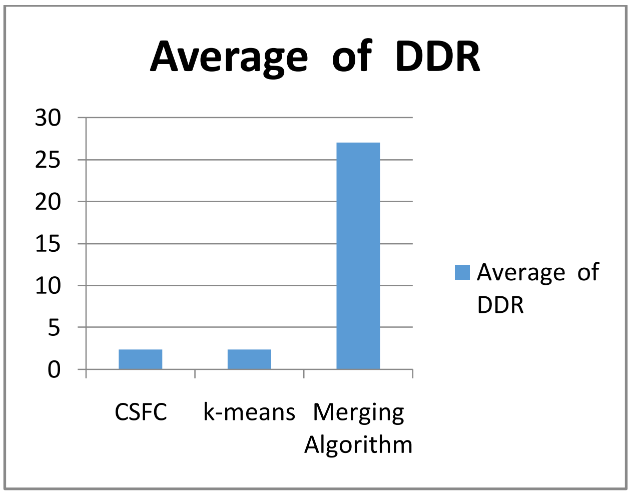


Fig 3. Average of DDR value X-axis - Algorithms ,y-axis- values to show the average.

Comparatively it is less in CSFC algorithm than K-means and Merging algorithm.It represents how the files are related to each other. In CSFC the cluster size also represented to 128MB. If the size exceeds the 128MB then immediately new cluster will be formed. If the DDR value decreases it increased the relationship between the data sets. Clustering time represents the time taken for generating clusters by various approaches in Nano seconds.If the time decreases then automatically processing speed increases. In DDR value and Time consumption our proposed CSFC is the efficient way compared with other algorithms. Totally 922 data sets are used with various file sizes from 1KB to 50MB. Number of files in the cluster will be different from algorithm to algorithm. The DDR value represents the related data sets because of that data set will be accessed easily without going number of searches in datanode.

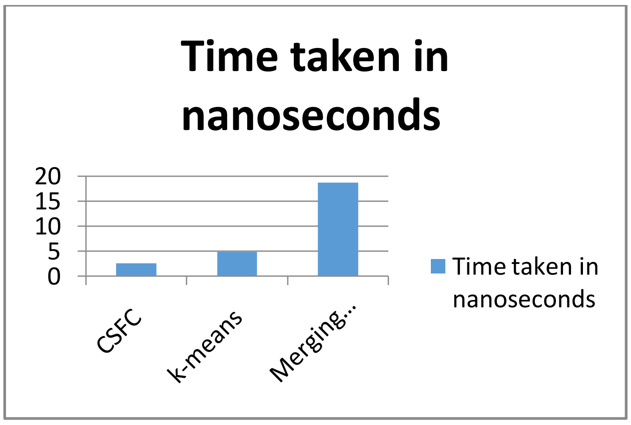


Fig 4. X-axis time in nano seconds, Y-axis Algorithms

Time consumption for generating clusters for small files using various approaches.in nano seconds. CSFC and Merging algorithm generates three clusters, K-means algorithm generates two clusters but the Data Difference value is comparatively more so its not much efficient when comparing with other algorithms

IV. CONCLUSION:

In advancement of technologies, data are generated in large number, but all the generated data are not large in size. For each and every second data are generated IOT devices using sensors. Handling this type of small files in Hadoop hasn't been easy to overcome this problem we developed a Clustering technique CSFC. In this approach the file types are identified and they are clustered up to 128 MB in each cluster if the files are related ,otherwise new cluster will be generated. In this approach clusters are not predefined because of this the data are not clubbed in limited clusters. The related files are combined, if they are not related new cluster will be generated so it is easy to store and access the files in data node easily. The usage of Name node memory is reduced effectively by clustering and they are stored and easily because of related files in the data nodes. The execution time also reduced effectively with this approach when compared with existing work. In future enhancement this approach should be implemented for images and videos and for performing real time data analytics. In future work combined files are encrypted and send to datanode for storing the data.

REFERENCES

1. A. Mehmood, M. Usman, W. Mehmood, and Y. Khaliq, “Performance efficiency in Hadoop for storing and accessing small files,” *7th Int. Conf. Innov. Comput. Technol. INTECH 2017*, no. Intech, pp. 211–216, 2017.
2. W. Cheng, M. Zhou, B. Tong, and J. Zhu, “Optimizing small file storage process of the HDFS which based on the indexing mechanism,” *2017 2nd IEEE Int. Conf. Cloud Comput. Big Data Anal. ICCCBDA 2017*, pp. 44–48, 2017.
3. W. Tao, Y. Zhai, and J. Tchaye-Kondi, “LHF: A new archive based approach to accelerate massive small files access performance in HDFS,” *Proc. - 5th IEEE Int. Conf. Big Data Serv. Appl. BigDataService 2019, Work. Big Data Water Resour. Environ. Hydraul. Eng. Work. Medical, Heal. Using Big Data Technol.*, pp. 40–48, 2019.
4. K. Bok, J. Lim, H. Oh, and J. Yoo, “An efficient cache management scheme for accessing small files in Distributed File Systems,” *2017 IEEE Int. Conf. Big Data Smart Comput. BigComp 2017*, pp. 151–155, 2017.
5. M. A. Ahad and R. Biswas, “Dynamic Merging based Small File Storage (DM-SFS) Architecture for Efficiently Storing Small Size Files in Hadoop,” *Procedia Comput. Sci.*, vol. 132, pp. 1626–1635, 2018.
6. X. Ren, X. Geng, and Y. Zhu, “An Algorithm of Merging Small Files in HDFS,” *2019 2nd Int. Conf. Artif. Intell. Big Data, ICAIBD 2019*, pp. 24–27, 2019.
7. A. Ahmed Shah and M. C. N, “Improving Hadoop Performance in Handling Small Files.”
8. H. Hooda and R. Nandal, “Implementation of k-Means Clustering Algorithm in CUDA,” *Int. J. Enhanc. Res. Manag. Comput. Appl.*, vol. 3, no. 9, pp. 829–833, 2014.
9. B. Mao, S. Wu, and H. Jiang, “Improving Storage Availability in Cloud-of-Clouds with Hybrid Redundant Data Distribution,” *Proc. - 2015 IEEE 29th Int. Parallel Distrib. Process. Symp. IPDPS 2015*, pp. 633–642, 2015.
10. R. Rathidevi and R. Parameshwari, “4 th International Conference on Cyber Security A Systematic Approach for Merging Small Files in Hadoop using Prolonged HDFS Framework.”
11. B. Gupta, R. Nath, G. Gopal, and K. K, “An Efficient Approach for Storing and Accessing Small Files with Big Data Technology,” *Int. J. Comput. Appl.*, vol. 146, no. 1, pp. 36–39, 2016.
J. N. Zacharias, “Fuzzy C - Means What is Clustering ?”
12. S. Bhandari, “An approach to solve a Small File problem in Hadoop by using Dynamic Merging and Indexing Scheme,” vol. 5, no. 04, pp. 227–230, 2017.

13. J. Chen, D. Wang, L. Fu, and W. Zhao, “An improved small file processing method for HDFS,” *Int. J. Digit. Content Technol. its Appl.*, vol. 6, no. 20, pp. 296–304, 2012.
14. R. Rathidevi and S. Srinivasan, “Small files problem in Hadoop - A Survey,” *Int. J. Pure Appl. Math.*, vol. 119, no. 15 Special Issue B, pp. 2833–2841, 2018.
15. A. N. Approach, T. Undestand, S. Files, and P. In, “A Review on Small Files in HADOOP,” no. 5, pp. 6585–6588, 2017.
16. Deepika, “An Optimized Approach for Processing Small Files in HDFS,” *Int. J. Sci. Res.*, vol. 6, no. 6, pp. 402–405, 2017.
17. R. Thangaselvi, S. Ananthbabu, S. Jagadeesh, and R. Aruna, “Improving the efficiency of MapReduce scheduling algorithm in Hadoop,” *Proc. 2015 Int. Conf. Appl. Theor. Comput. Commun. Technol. iCATccT 2015*, pp. 63–68, 2016.
18. Y. Fan, Y. Wang, and M. Ye, “An improved small file storage strategy in ceph file system,” *Proc. - 14th Int. Conf. Comput. Intell. Secur. CIS 2018*, pp. 488–491, 2018.
19. P. V. Subba Reddy, “Fuzzy mapreduce data mining algorithms,” *iFUZZY 2018 - 2018 Int. Conf. Fuzzy Theory Its Appl.*, pp. 304–310, 2018.
20. E. Alshammari, G. Al-Naymat, and A. Hadi, “A New Technique for File Carving on Hadoop Ecosystem,” *Proc. - 2017 Int. Conf. New Trends Comput. Sci. ICTCS 2017*, vol. 2018-Janua, pp. 72–77, 2017.
21. C. K. Leung, C. S. H. Hoi, A. G. M. Pazdor, B. H. Wodi, and A. Cuzzocrea, “Privacy-Preserving Frequent Pattern Mining from Big Uncertain Data,” *Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018*, pp. 5101–5110, 2019.
22. M. Stratford, “Welcome to Mendeley: Quick Start Guide,” 2018.
23. https://www.google.com/search?q=Mapreduce+content&oq=Mapreduce+content+&aqs=chrome..69i57.8567j0j8&sourceid=chrome&ie=UTF-8

AUTHORS PROFILE

R .Rathidevi, is research scholar at Department of Computer science, Vels Institute of Science, Technology and Advanced studies, Chennai. She received her M.Sc.,(C.S) degree from University of Madras. M.Phil. from Periyar University, M.Tech. from SRM University. She has published two papers in International journals. Her research interest is Big Data Analytics, Cloud computing.

Dr.R.Parameswari, is working as Associate Professor in Department of Computer Science, Vels Institute of Science, Technology and Advanced Studies, Chennai. She had 13 years of teaching experience. She has completed Ph.D in Computer science from St.Peter’s University, Chennai. She is presently guiding 8 Ph.D scholars and 1 M.Phil Scholar. She has produced 3 M.Phil Scholar’s. She has published 26 papers in various International Journals including journals indexed in Scopus. She has presented many papers in International conferences and attended many seminars and workshops conducted by various educational Institutions. She is acting as editor and reviewer in many International Journals. Her research interest lies in the area of cloud computing. Big data Analytics, Internet of things.