IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

PABR Algorithm for Improving The Data Archival Performance of aHDFS

M. Mounica[1*], A. Ananda Rao[2], P. Radhika Raju[3]

[1*] PG Scholar, Department of CSE, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India
[2]Professor, Department of CSE, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India
[3]Ad-hoc Assistant Professor, Department of CSE, JNTUA College of Engineering, Ananthapuramu, Andhra Pradesh, India

[*]*Corresponding Author: mudiyalamounika@gmaill.com, Tel.: +91-8179772662*

Available online at: www.ijcseonline.org

Accepted: 13/Jul/2018, Published: 31/July/2018

***Abstract*—**Hadoop Distributed File System (HDFS) is highly a fault-tolerant distributed file system associated with Hadoop framework. HDFS can handle a large amount of data known as big data. HDFS deals with data archival as well. Data archiving is a phenomenon which finds inactive data and moves it into a separate storage premise. Cloud-based storage facilitates it cost-effectively while Hadoop clusters provide the computational power required. However, protecting archived data is the main concern of the data owner point of view. Erasure Coding (EC) is a method which has the mechanism to regain lost data as well. Of late aHDFS was developed to have special data archival features with EC. The problem with it is that it needs similar the computational cost for data of different sizes. Towards this end, we proposed a methodology to overcome this problem. A model application has built to exhibit evidence of the idea. The empirical results revealed that the methodology presented improves the computational efficiency in rendering data archival services.

***Keywords*—** Hadoop, HDFS, Data archival system, Erasure codes

I. INTRODUCTION

Hadoop is one of the widely used fault-tolerant distributed programming frameworks. It supports the new model of programming known as MapReduce. It has both Map and Reduce tasks for handling voluminous data. Moreover, the Hadoop ecosystem includes cloud computing and data centers so as to utilize a large number of commodity computers. HDFS is associated with the Hadoop framework. It takes care of distributed storage features and all computers involved in the parallel processing of MapReduce jobs can gain access to information in HDFS. Both input and output data needs to be stored in HDFS. HDFS needs to have mechanisms to optimize storage and processing. Towards this end, it can have features to identify data replicas and the data that is not used for a long time. Such data is eligible for archiving. Before diving into an archival system and the issues associated with it is good to see the architecture of Hadoop and HDFS [1].

The Figure 1 shows that the storage process is associated with both data and metadata. Name node and data nodes play a vital role in HDFS functionality. Data nodes take care of storing actual data in the form of blocks. The client program can perform both read and write operation on the data nodes. The file system is distributed across thousands of commodity computers associated with Hadoop clusters. In this context, the problem addressed in the paper is related to data archival system with EC.

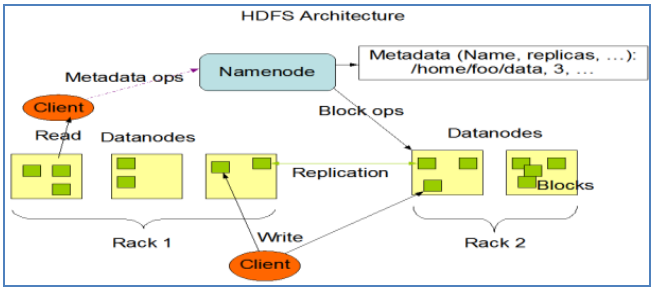

Figure 1. Overview of Hadoop Distributed File System

Many researchers contributed the work related to archival systems are explored in [1]. In this paper, we proposed a methodology for an effective data archival system that can work well to protect data besides reducing computational complexity. The aHDFS causes overhead as it needs same computational cost irrespective of the size of data. To overcome this problem, by proposing a technique and also to ensure that the computational cost is further reduced. The contributions are as follows.

1. A new EC-based approach for data archival to protect archived big data. It considers size-aware processing in dealing with data archival process and data retrieval to optimize computational cost.
2. To handle data archival process a Data size-Aware Erasure Coding (DAEC) algorithm is used that maintains the efficiency of the archival system in terms of speed and the reduction of computational complexity.
3. To exhibit evidence of the idea a model application has build. The experimental results revealed that the projected mechanism has utility in archival process and protection of archived storage.

The rest of the paper is organized as follows. Segment II provides the review of literature related to EC and it usage in distributed programming frameworks. Segment III presents the proposed methodology. Segment IV presents the implementation and experimental results while segment V closes the paper and provides bearings for future work.

II. RELATED WORK

This section provides review of literature on Hadoop and its variants and related to archival with distributed programming frameworks. Hadoop Distributed File System (HDFS) and its architectural elements that play vital role in distributed computing are explored in [1]. An email-system with tracking of related messages with automatic linking is studied in [2]. For security of data in [3], erasure codes with files stored in cloud is focused in [4]. Big data analytics in the distributed environment is studied in [5], while the research in [6] focused on an archival system and efficient query using Hadoop. An archival system with the concept of deep storage is investigated in [7]. In case of large archival systems, the concept of disk scrubbing is explored in [8]. Analyzing Hadoop workloads, optimizations and implications are the main focus on [9] while polynomial codes for security and the efficiency of quantcast file system are studied in [10] and [11] respectively.

The concept of pipelined encoding process to improve archival system is explored in [12]. A novel system for archival storage [13], novel erasure codes concept [14], a data placement scheme in distributed environment for data-intensive applications [15], decentralized erasure coding for efficient data archival system [16], erasure coding with parity logging in cluster storage environment [17], the utility of Software Defined Network (SDN) to decouple data and control planes in distributed environments, and graph based optimizations in Hadoop framework for improving functions like Swappiness and Hugepage [18] are other important contributions found in the literature.

Data encryption and security dynamics with HDFS of Hadoop are studied in [19]. MapReduce programming paradigm with entity resolution based on parallel progressive approach is investigated in [20]. The work which is closer to that of this paper is in [21] where a variant of Hadoop DFS known as aHDFS is designed with erasure codes for efficient archival system. From the literature it is understood that the reconstruction process when data blocks are lost is crucial for the archival system. In this paper a framework with underlying algorithm was proposed to achieve this.

III. PROPOSED METHODOLOGY

The proposed work is to cope up the block failure issues on Hadoop clusters through the reconstructive system. It is planned to apply the Hadoop based data archival strategies, namely aHDFS-Grouping and aHDFS-Pipeline Schemes on the reconstructive system to accelerate the process of reconstruction. The methodology to achieve the DAEC system is to make use of Hadoop, one of the distributed programming frameworks, for parallel processing of the data archiving process. It has two phases for demonstrating the process of archival and also the process of recovering data from block failures. When certain blocks fail in storing data, such data can be reconstructed by utilizing erasure codes. An algorithm is known as Parallel Archival and Block Failure Recovery (PABFR) algorithm is defined.

Algorithm: Parallel Archival and Block Failure Recovery (PABFR)
Inputs: Files to be archived F={f1, f2, ..., f_n}, set of failed blocks B={b1, b2, ..., b_n}, set of surviving blocks S{s1, s2, ..., s_n}
Output: Efficient archival and block failure recovery

```
Initialize
Parallel Archival Process
For each file f in F
  Divide file into K blocks
  Organize K blocks into K key/value pairs
End For
For each key/value pair in K key/value pairs
  Assign key/value pair to a mapper (commodity computer)
  Generate RS code
End For
Grouping for Reducing Number of Keys
For each k in K
  Group similar content into one key/value pair
  Add k to K'
End For
Reduce all key/value pairs into final archival blocks
Store them in cloud server with corresponding parity values as a reconstruction matrix
Block Failure Recovery
Divide B into the set of key/value pairs
Assign them to mappers
Mapper computes data from reconstruction matrix and survival matrix
Reducers produce the final output
The recovered blocks are saved to the server
```

Algorithm 1. Parallel Archival and Block Failure Recovery

The algorithm is, a DAEC based algorithm deals with enhancing the performance of Hadoop further to handle archival system with block failure recovery phase. The workflow of the MapReduce framework can be compared with the flow given in Figure 1. It shows how a word count MapReduce application works. Similarly, the proposed algorithm makes use of Map and Reduce phases for achieving parallel archival and block failure recovery with erasure codes using RS approach.

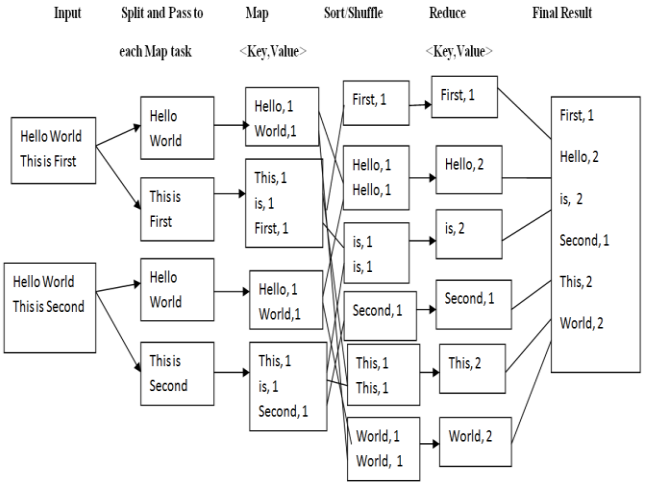

Figure 2. Workflow of MapReduce WordCount benchmark

It is obvious that there are numerous stages engaged with MapReduce programming paradigm as plotted in Figure 2. The functionalities of the system are legitimately divided into the following modules. They are known as RS Coding Module, Mapper Module, Reducer Module and Block Recovery Module.

RS Coding Module: This module is responsible to convert data blocks into corresponding erasure codes using Reed – Solomon codes. The erasure codes are used to reconstruct data in case of block failure. This module precisely generates reconstruction matrix to the data that has been archived.

Mapper Module: It is capable to generate intermediate output while performing archival process or block failure recovery process. It acts on key/value pairs of intended purpose.

Reducer Module: It is capable to generate final output while performing archival process or block failure recovery process. It acts on key/value pairs with an intended purpose while generating the final output.

Block Failure Recovery Module: This module performs recovery of failed blocks. It makes use of survival, archived blocks and reconstruction matrix in order to generate data for failed blocks.

With the proposed implementation, the system is capable of producing failed blocks using erasure codes. Since it makes use of grouping process, it reduces computational cost as it needs less number of commodity computers in the distributed environment. It provides the guarantee of data recovery pertaining to archive data. It can avoid data replication that needs more infrastructures with respect to hardware and storage.

IV. EXPERIMENTAL RESULTS

Experiments are made with a Cloudera environment where Hadoop implementation is provided. The performance of the DAEC system is compared with that of the existing system with different file sizes by observing the execution time.

Table 1. Shows execution times of Map phase

File Size	Execution Time (seconds) of Map Phase				
	Baseline	***aHDFS-Grouping***	***aHDFS-Pipeline***	***DAEC-Grouping***	***DAEC-Pipeline***
640M	5	17	19	18	17
1280M	5	19	19	18	17
2560M	6	20	20	19	18
5120M	7	20	20	19	18
10G	8	20	20	19	18
20G	10	19	19	18	17

The file sizes and execution times of Map phase for all approaches are mentioned in Table 1.

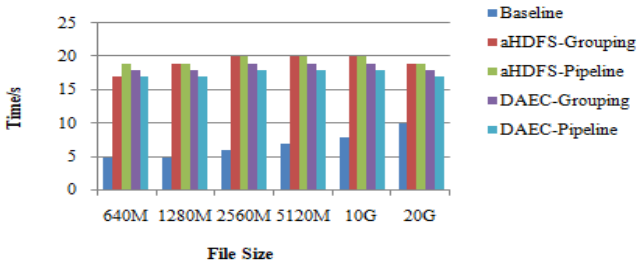

Figure 3. Shows the performance comparison in Map phase

The Map phase performance of the DAEC system is plotted in Figure 3 is better than that of existing methods.

Table 2. Shows execution times of Shuffle phase

File Size	Execution Time (seconds) of Shuffle Phase			
	Baseline	***aHDFS-Grouping***	***DAEC-Grouping***	***DAEC-Pipeline***
640M	5	10	8	7
1280M	10	20	18	17
2560M	30	20	18	17
5120M	60	20	18	17
10G	90	20	18	17
20G	200	40	35	30

The file sizes and execution times of Shuffle phase for all approaches are mentioned in Table 2.

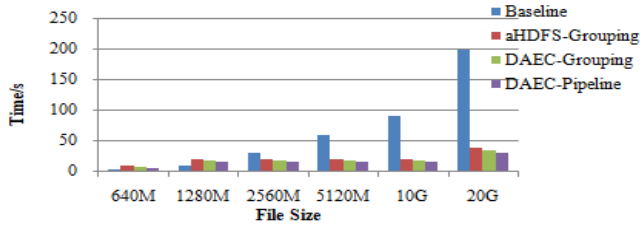

Figure 4. Shows the performance comparison in Shuffle phase

The performance of the DAEC system is plotted in Figure 4, is better than that of existing methods.

Table 3: Shows execution times of Reduce phase

File Size	Execution Time (seconds) of Reduce Phase			
	Baseline	***aHDFS-Grouping***	***DAEC-Grouping***	***DAEC-Pipeline***
640M	8	10	9	8
1280M	9	12	11	10
2560M	18	8	7	6
5120M	25	7	6	5
10G	35	15	14	13
20G	65	10	9	8

The file sizes and execution times of Reduce phase for all approaches are mentioned in Table 3.

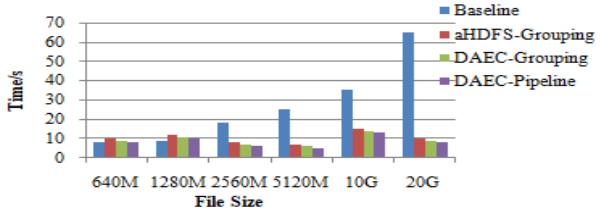

Figure 5. *Shows the performance comparison in Reduce phase*

The results of the DAEC system are plotted in Figure 5, is compared and the performance of the DAEC system is better than that of the existing methods.

Table 4: Shows total execution times of all phases

File Size	Total Execution Time (seconds) of All Phases				
	Baseline	***aHDFS-Grouping***	***aHDFS-Pipeline***	***DAEC-Grouping***	***DAEC-Pipeline***
640M	30	40	40	35	30
1280M	35	40	40	35	30
2560M	50	35	40	35	30
5120M	95	45	40	35	30
10G	150	60	55	50	45
20G	300	70	65	60	55

The file size and the total execution time of all phases mentioned in Table 4.

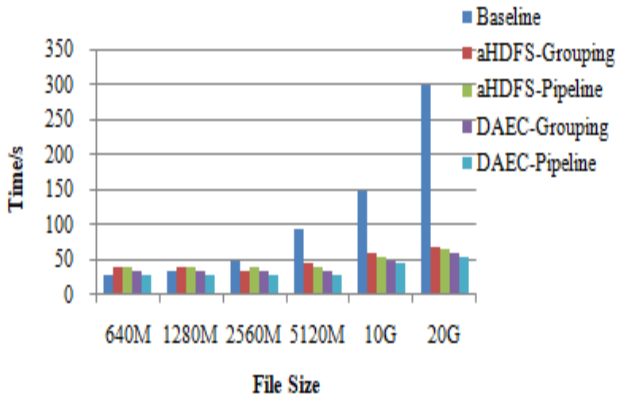

Figure 6. Shows the performance comparison in all phases

The performance of the DAEC system is plotted in Figure 6, is better than that of existing methods.

Table 5. Shows execution times of Map phase for reconstruction

File Size	Execution Time (seconds) of Map Phase for Reconstruction		
	Baseline	***DAEC-Grouping***	***DAEC- Pipeline***
640M	5	18	17
1280M	5	18	17
2560M	6	19	18
5120M	7	19	18
10G	8	19	18
20G	10	18	17

The observations regarding file size and execution time of Map phase of all approaches are recorded in Table 5 while performing reconstruction using the DAEC system.

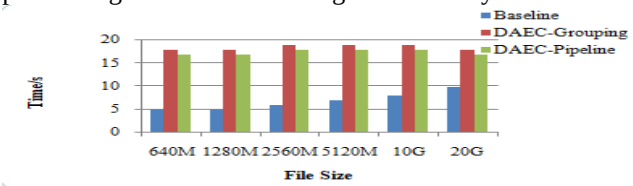

Figure 7. Shows performance comparison in Map phase

The performance of the DAEC system is plotted in Figure 7, and is better while performing the reconstruction.

Table 6: Shows execution times of Shuffle phase for reconstruction

File Size	Execution Time (seconds) of Shuffle Phase for Reconstruction		
	Baseline	***DAEC-Grouping***	***DAEC-Pipeline***
640M	5	8	7
1280M	10	18	17
2560M	30	18	17
5120M	60	18	17
10G	90	18	17
20G	200	35	30

The file size and execution time of Shuffle phase of all approaches are mentioned in Table 6. The observations are recorded while performing reconstruction using the DAEC system.

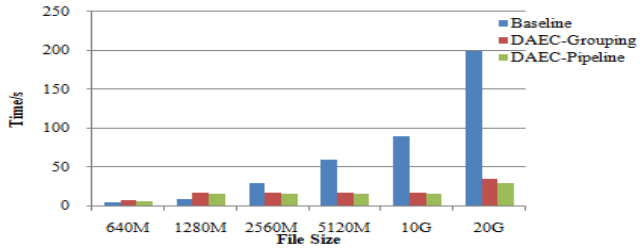

Figure 8. Shows the performance comparison in Shuffle phase

The times taken for execution of Shuffle phase of the DAEC system is plotted in Figure 8.The results are compared and the performance is better than that of existing methods while performing the reconstruction.

Table 7: Shows execution times of Reduce phase for reconstruction

File Size	Execution Time (seconds) of Reduce Phase for Reconstruction		
	Baseline	***DAEC-Grouping***	***DAEC-Pipeline***
640M	8	9	8
1280M	9	11	10
2560M	18	7	6
5120M	25	6	5
10G	35	14	13
20G	65	9	8

The observations of all phases are recorded as shown in Table 7 while performing reconstruction using the DAEC system.

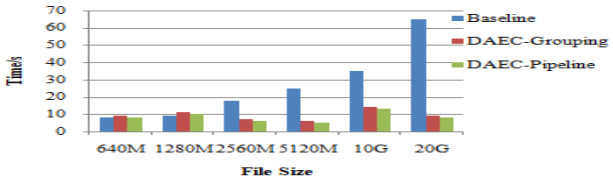

Figure 9. Shows the performance comparison in Reduce phase

The performance of the DAEC system is better than that of existing methods while performing the reconstruction is plotted in Figure 9.

The file size and total execution time of all phases for approaches are mentioned in Table 8.The observations are recorded while performing reconstruction using the DAEC system.

Table 8. Shows execution times of Reduce phase for reconstruction

File Size	Total Execution Time (seconds) of All Phases for Reconstruction		
	Baseline	***DAEC-Grouping***	***DAEC-Pipeline***
640M	30	35	30
1280M	35	35	30
2560M	50	35	30
5120M	95	35	30
10G	150	50	45
20G	300	60	55

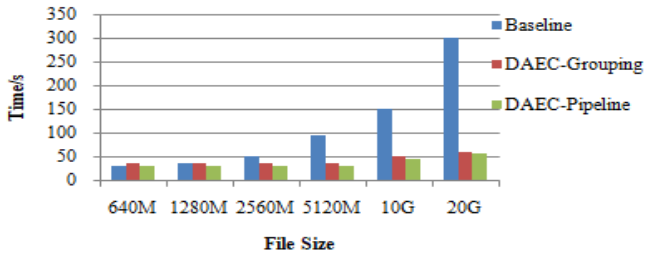

Figure 10. Shows the performance comparison in all phases

The results plotted in Figure 10, are compared and the performance of the DAEC system is better while performing reconstruction.

V. CONCLUSION and Future Scope

In this paper, we planned a methodology to have improvements in aHDFS which is native Hadoop's extension in terms of performance in archival system. Archival systems help in preserving data and help in using data as ready reference. Grouping and pipelining strategies are implemented in MapReduce paradigm in order to improve its performance in all phases. Moreover, an algorithm is defined to handle the reconstruction of blocks. The algorithm is named as Parallel Archival and Block Failure Recovery (PABFR). It takes files to be archived, set of failed blocks and set of the surviving blocks. Then it performs efficient archival and exhibits block recovery failure. A model application is designed to demonstrate proof of the idea. The pragmatic results revealed that the projected framework is able to improve the performance in all phases and in the process of reconstruction as well. As a future scope, another framework for architectural decision modelling can be build to improve Hadoop's performance further.

REFERENCES

[1] D. Borthakur, "*The hadoop distributed file system: Architecture and design, 2007,"* Apache Software Foundation, 2012

[2] S. S. Miller, M. S. Shaalan, and L. E. Ross, "*Correspondent-centric management email system uses message-correspondent relationship data table for automatically linking a single stored message with its correspondents*," Sep. 2 2003, uS Patent 6,615,241

[3] N. Madhusudhana Reddy, Dr. C. Nagaraju, Dr. A. AnandaRao, "*Toward Secure Computations in Distributed Programming Frameworks: Finding Rogue nodes through Hadoop logs*", JATIT (Journal of Theoritical and Applied Information Technology),ISSN No: 1992-8645, Vol95, No 23, December 2017, Page Nos: 6398-6409

[4] O. Khan, R. C. Burns, J. S. Plank, W. Pierce, and C. Huang, "*Rethinking erasure codes for cloud file systems: minimizing i/o for recovery and degraded reads.*" in FAST, 2012, p. 20

[5] R. T. Kaushik and K. Nahrstedt, "*T: a data-centric cooling energy costs reduction approach for big data analytics cloud,*" in Proceedings of the International Conference on High Performance

Computing, Networking, Storage and Analysis. IEEE Computer Society Press, 2012, p. 52

[6] R. Gupta, H. Gupta, U. Nambiar, and M. Mohania, "*Efficiently querying archived data using hadoop*," in Proceedings of the 19th ACM international conference on Information and knowledge management. ACM, 2010, pp. 1301–1304.

[7] L. L. You, K. T. Pollack, and D. D. Long, "*Deep store: An archival storage system architecture*," in Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on. IEEE, 2005, pp. 804–815.

[8] T. J. Schwarz, Q. Xin, E. L. Miller, D. D. Long, A. Hospodor, and S. Ng, "*Disk scrubbing in large archival storage systems*," in Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.(MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on. IEEE, 2004, pp. 409–418

[9] Z. Ren, J. Wan, W. Shi, X. Xu, and M. Zhou, "*Workload analysis, implications, and optimization on a production hadoop cluster: A case study on taobao,"* Services Computing, IEEE Transactions on, vol. 7, no. 2, pp. 307–321, 2014

[10] I. S. Reed and G. Solomon, "*Polynomial codes over certain finite fields,"* Journal of the society for industrial and applied mathematics, vol. 8, no. 2, pp. 300–304, 1960.

[11] M. Ovsiannikov, S. Rus, D. Reeves, P. Sutter, S. Rao, and J. Kelly, "*The quantcast file system*," Proceedings of the VLDB Endowment, vol. 6, no. 11, pp. 1092–1101, 2013.

[12] J. Huang, Y. Wang, X. Qin, X. Liang, S. Yin, and C. Xie, "Exploiting *pipelined encoding process to boost erasure-coded data archival,"* Parallel and Distributed Systems, IEEE Transactions on, vol. 26, no. 11, pp. 2984–2996, 2015

[13] S. Quinlan and S. Dorward, "*Venti: A new approach to archival storage."* in FAST, vol. 2, 2002, pp. 89–101.

[14] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "*Xoring elephants: Novel erasure codes for big data,"* vol. 6, no. 5, pp. 325–336, 2013.

[15] J. Wang, P. Shang, and J. Yin, "*Draw: A new data-grouping-aware data placement scheme for data intensive applications with interest locality*," in Cloud Computing for Data-Intensive Applications. Springer, 2014, pp. 149–174

[16] L. Pamies-Juarez, F. Oggier, and A. Datta, "*Decentralized erasure coding for efficient data archival in distributed storage systems,"* in Distributed Computing and Networking. Springer, 2013, pp. 42–56.

[17] J. C. Chan, Q. Ding, P. P. Lee, and H. H. Chan, *"Parity logging with reserved space: Towards efficient updates and recovery in erasurecoded clustered storage,"* in Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14), 2014, pp. 163–176.

[18] Sunita Choudhary; Preeti Narooka "*Hugepage & Swappiness functions for optimization of the search graph algorithm using Hadoop framework"*: 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC).

[19] Youngho Song; Young-Sung Shin; Miyoung Jang; Jae-Woo Chang "*Design and implementation of HDFS data encryption scheme using ARIA algorithm on Hadoop"* : 2017 IEEE International Conference on Big Data and Smart Computing (BigComp)

[20] Yasser Altowim; SharadMehrotra "*Parallel Progressive Approach to Entity Resolution Using MapReduce*": 2017 IEEE 33rd International Conference on Data Engineering (ICDE).

[21] Yuanqi Chen, Yi Zhou, Shubbhi Taneja, Xiao Qin, Senior Member, IEEE, Jianzhong Huang, "*aHDFS: An Erasure-Coded Data Archival Systemfor Hadoop Clusters*", IEEE Transactions on Parallel and Distributed Systems, 2017

Authors Profile

Ms.M Mounica received Bachelor of Technology in computer science and engineering from Narayana Engineering College affiliated by Jawaharlal Nehuru Technological University Ananthapuramu, in 2016, Andhra Pradesh, India. She is pursuing Master of Technology from Jawahralal Nehuru Technological University, Ananthapuramu, Andhra Pradesh , India in 2018.

Dr.A.Ananda rao received B. Tech. Degree in computer science and engineering from University of Hyderabad, erstwhile Andhra Pradesh, India and M. Tech. Degree in A.I &Robotics from University of Hyderabad, erstwhile Andhra Pradesh, India. He received his Ph. D. Degree from Indian Institute of Technology Madras, Chennai, India. He is a professor of Computer Science & Engineering Department and currently working as a Director, Research & Development, JNT University Anantapur, Ananthapuramu, Andhra Pradesh, India. Dr. Rao has published more than 160 publications in various national and international journals/conferences. He received one Best Research Paper award, one Best Paper award for his papers. He also received best educationist award, Bharat VidyaShiromani Award, RashtriyaVidyaGaurav Gold Medal Award, Best Computer Teacher Award and Best Teacher Award from the Andhra Pradesh chief minister for the year 2014. His main research interest includes software engineering and Databases. He is Sessnsior Member IAENG.

P.Radhika Raju received B.Sc. (Comp. Sci) from Sri Krishnadevaraya University, Ananthapuramu; MCA degree from Indira Gandhi National Open University (IGNOU), India and M.Tech degree in Computer Science from Jawaharlal Nehru Technological University Anantapur, Ananthapuramu, A.P, India. She Completed her Ph.D degree from JNT University Anantapur. She authored a text book and has publications in National and International Journals/Conferences. Her research interest include Software Engineering and Databases. She is now working as a Lecturer in department of Computer Science and Engineering, JNTUA College of Engineering, Ananthapuramu, A.P.