ISSN:0975-9646

An Efficient Dynamic Data Replication for HDFS using Erasure Coding

Franklin John[#1], Suji Gopinath[#2], Elizabeth Sherly[#3]

[#1]*M.Phil. Scholar, Indian Institute of Information Technology and Management-Kerala*
Trivandrum, Kerala, India

[#2]*Research Scholar, University of Kerala*
Kerala, India

[#3]*Professor, Indian Institute of Information Technology and Management-Kerala*
Trivandrum, Kerala, India

Abstract— **The Hadoop Distributed File System (HDFS) component of Apache Hadoop helps in distributed storage of big data with a cluster of commodity hardware. HDFS ensures availability of data by replicating data to different nodes. However, the replication policy of HDFS does not consider the popularity of data. The popularity of the files tend to change over time. Hence, maintaining a fixed replication factor will affect the storage efficiency of HDFS. In this paper we propose an efficient dynamic data replication management system, which consider the popularity of files stored in HDFS before replication. This strategy dynamically classifies the files to hot data or cold data based on its popularity and increases the replica of hot data by applying erasure coding for cold data. The experiment results show that the proposed method effectively reduces the storage utilization up to 40% without affecting the availability and fault tolerance in HDFS.**

Keywords— **Big Data, Hadoop Distributed File System, Dynamic data replication.**

I. INTRODUCTION

Big Data is high-volume, high-velocity and high-variety information that demands cost-effective, innovative forms of information processing for enhanced insight and decision making [1]. The extraction, storage and processing of big data is beyond the ability of traditional data processing techniques. Therefore a more sophisticated framework is required to handle these data. Apache Hadoop [2] is one of the best known platforms for distributed storing and processing of big data across clusters of computers. The storage component of Hadoop, Hadoop Distributed File System (HDFS)[3][4] maintains a default replication factor for each file as three, which is placed in separate nodes.

HDFS provides high performance access to data by applying a static and default replication strategy. Though HDFS ensures high reliability, scalability and high availability, its static and default approach in data replication requires large amount of storage space. With a replication factor of three, a file is copied three times in different nodes. If the size of a file is 1TB then, after replication it will take 3TB of space. Furthermore, in HDFS the files are replicated without considering the popularity of the file. In real scenario, the access frequency of every file in the file system is not accessed equally. Some files are accessed frequently while some others stay idle for a long period of time. By keeping replicas for these idle files, a valuable amount of storage space will consumed unnecessarily resulting in wastage of storage space, that lead to bad effect of performance. If the number of copies can be reduced for these files the storage space can be freed and can be utilized by more frequently accessed files. But reducing the number of copies increases the chance for data loss. Therefore a data replication strategy which reduces the replicas of under-utilized file without affecting the data availability and fault tolerance has to be implemented.

In the proposed work, we present a dynamic data replication strategy which focuses on a storage efficient replication in HDFS without affecting the availability of data. In this strategy data files are classified into hot and cold based on the popularity of the data file in the Hadoop cluster. The replica of the popular file is increased while the replica of non-popular file is reduced to one and erasure coding is applied on it to prevent from data loss. The result shows that the proposed replication strategy reduces the storage space utilization significantly without affecting the availability constraint of HDFS. The remainder of this paper is structured as follows. In section 2, we discuss the related work and in section 3, we discuss the background theory. In section 4, we present the data replication strategy in detail with its architecture and algorithm. In section 5, we include the implementation and evaluation results demonstrating the storage efficiency of our proposed algorithm. Finally in section 6, we conclude with the scope of future work.

II. RELATED WORK

There have been a number of research efforts in recent years which focus on the issues of replicating large files in a network.

Wei et al. (2010) [5] proposed a cost-effective dynamic replication strategy for the cloud storage systems which is referred as CDRM. In this work the popularity of a data file is calculated to create replica for the data file. After finding the popular file the replica is placed in a suitable node considering the blocking probability and capacity of the nodes. This method concentrates on capturing the relationship between availability and replica number. They do not consider the availability of files that has low replica factors.

Ananthanarayanan et al. (2011) [6] proposed an off-line system called Scarlett, which periodically replicates popular files using prediction method based on the historical usage and jobs submitted for execution. It also distributes the replicas among clusters with the goal of minimizing hotspots. This method uses the concept of aging for replicas to give space for new replicas. This may cause loss of some files.

Abad et al. (2011) [7] proposed an adaptive data replication for efficient cluster scheduling (DARE), which replicates the data files dynamically to increase data locality. The number of replicas to be created for each file and the node to place the replica is determined based on the probabilistic sampling and a competitive aging algorithm independently at each node. The replication decision is based on probability and does not consider the trends of data utilization.

Kaushik et al. (2011) [8] proposed a predictive data replication policy for GreenHDFS which proactively create and delete replicas based on the file heat predictions. The file's heat is obtained based on the total number of access to the file and the file's hot lifespan. Replicas are created for the hot files and the replica of cold file gets deleted. This method fails to consider the management of cold data efficiently and the directory structure is an important factor in the anticipation for file accesses.

M. Bsoul et al. (2011) [9] proposed a replication strategy for data grids which considers the factors like frequency of requests, size of file, etc for replicating the data. This strategy does not consider the scenario of varying user behavior.

Cheng et al. (2012) [10] proposed an elastic replica management system (ERMS), which uses an active/standby model for the storage of data in HDFS. The data is classified into hot or cold by using the complex event processing engine and replica is created dynamically based on this classification. Erasure code is applied to unpopular data to save the storage space.

Kousiouris et al. (2013) [11] proposed a proactive data management in Hadoop clusters which is based on predictive data activity patterns. The method will predict the future data demand in the Hadoop cluster using Fourier series analysis [12]. In this method file is classified only in limited replication scenarios.

Bui et al. (2016) [13] proposed an adaptive replication management in HDFS based on supervised learning. This method replicate the data files based on the predictive analysis.The popularity of each data file is predicted using probability theory and replicate the high potential files. Erasure coding is applied to low potential files to ensure reliability.

Qu et al. (2016) [14] proposed a dynamic replication strategy (DRS) based on Markov Model for HDFS. In this method a transition probability matrix is constructed based on the accessing of files over time and then calculates the stationary probability distribution of the system. Using the results obtained data is classified as hot or cold. Extra replica is created for hot data and replica of cold data is deleted. This method is not considering the effective management of cold data resulting in a probability of data loss.

III. BACKGROUND THEORY

The Hadoop framework plays an important role in the handling and processing of big data. Two main components of Hadoop are MapReduce and Hadoop Distributed File System (HDFS). The MapReduce is an algorithm which helps in the processing of large data by implementing parallel processing. MapReduce consist of two parts, a Map task and Reduce task. These two tasks combined together perform the processing tasks. The HDFS is the distributed storage system which handles the storage of files in Hadoop.

HDFS provides a reliable and fault tolerant architecture to store files. It follows a rack based clustering, in which nodes are stored in racks and a cluster is formed combining these racks. A file entered to the HDFS is divided into blocks of equal size except the last block. These blocks are replicated and stored in separate nodes. By default HDFS creates three copies for each block. HDFS manages the placement of these replicas in such a way that two blocks are stored in the same rack and one in a separate rack. By following this method HDFS ensures the availability of a block even if a node fails or even if an entire rack goes down.

The operations in HDFS consist of two types of nodes DataNodes and NameNode. Namenode manages the the operation in the HDFS cluster. Datanodes are the nodes in which the blocks are stored. Namenode holds the information about each datanode in the cluster and the details of blocks stored in it. When a client wants to read or write to HDFS, it first communicates with the namenode and the namenode provides the information regarding the blocks and nodes. After acquiring these information the client communicates directly to the datanode for reading or writing.

Erasure coding (EC)[15] is a method of data protection in which data is broken into fragments, expanded and encoded with redundant data pieces and stored across a set of different locations on storage media [4]. Reed-Solomon (RS) encoding is a popular erasure coding method. In this method the data is divided into equal blocks and a set of parity blocks are added to it. So even if some of the blocks went missing the original file can be recreated with the help of these parity blocks. Reed-Solomon (10,4) configuration splits the file into 10 blocks consisting of 6 data blocks and 4 parity blocks. The proposed work combines the erasure coding with the files in HDFS to provide fault tolerance and availability. Reed-Solomon (10, 4) erasure coding is used in this work, since it can survive four block failures. This assures the prevention of data loss while cold data is stored as erasure coded file.

IV. STORAGE-EFFICIENT DYNAMIC DATA REPLICATION

In HDFS, in order to ensure data availability and to reduce the chance of data loss, each file is replicated across a number of machines. The default replication factor in HDFS is to create three replicas for each file. HDFS replication strategy will not consider whether a particular file is popular or not.

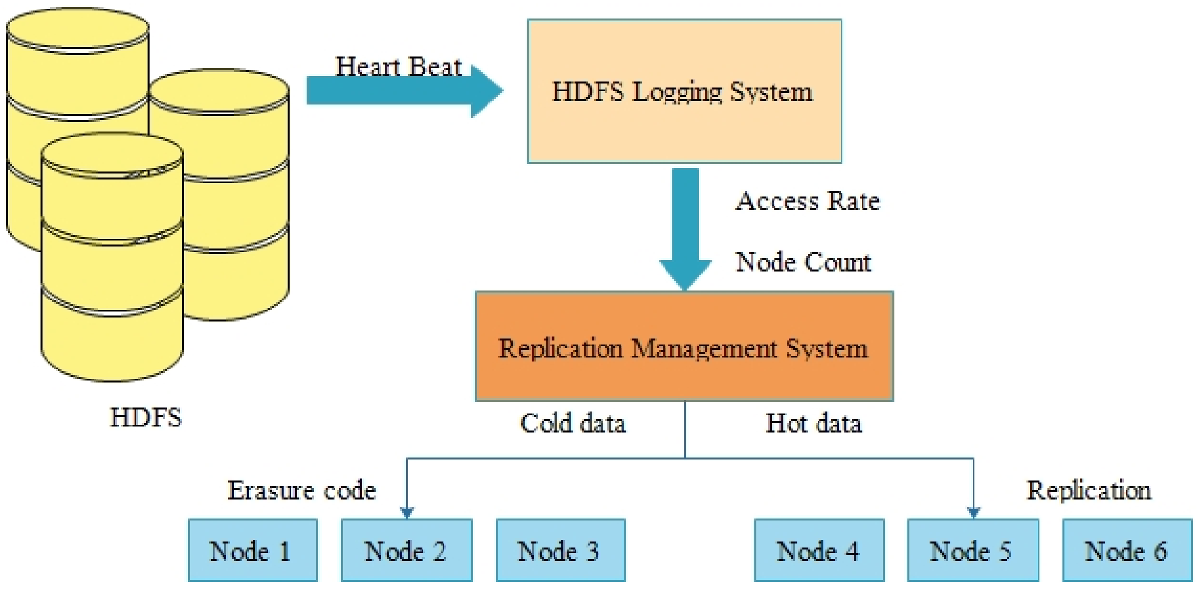


Fig. 1 Architecture of the proposed system.

Unnecessary replication of non-popular file will result in storage overhead. In the proposed strategy, a dynamic data replication algorithm is used to manage the replicas in HDFS.

A. System Architecture

The architecture of the system is illustrated in the Fig.1. The Replication Management System in the proposed algorithm manages the replication of files in HDFS. This module classifies the data files into hot data or cold data. After classifying the data, the replication factor is increased for the hot data and its replica is placed in data node. For placement of replicated data Hadoop's random placement strategy is used. Erasure coding is applied for cold data to ensure availability. Replication Management System does these tasks with the help of HDFS Logging System. The logging system provide details such as the number of files accessed, their source, the nodes which accessed them, frequency of access for each file, etc. The Logging system obtains all these information from HDFS and provides it to the Replication management System.

B. Replication Strategy

The algorithm divides each data into two categories as hot data or cold data based on their popularity in the Hadoop cluster. To determine the popularity of a file, different parameters like number of accesses to that file, number of nodes accessing the file and the replication factor of the file are considered. To obtain these values the log files in HDFS are analyzed. After analyzing and extracting the required values the popularity for each file is calculated. The Popularity Index (PI_i) of a file $f_{i,}$ is calculated as follows:

$$PI_i = (ac_i * nc_i) / rf_i$$

where, ac_i is the number of access received to a file f_i, nc_i denotes the number of nodes which accessed f_i and rf_i represents the current replication factor of f_i in HDFS. After calculating the popularity of each file, a threshold value, T, is obtained by calculating the mean of popularity values as follows:

$$T = \frac{\sum_{i=1}^{n} PI_i}{n}$$

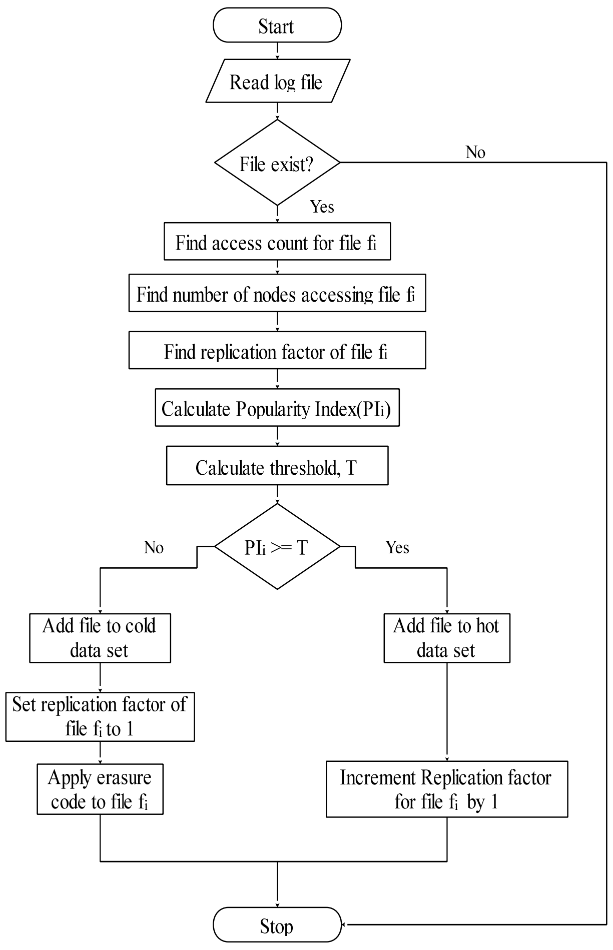
The value of T is used to classify the files in HDFS. Popularity Index of each file is compared with the obtained threshold value. The file with popularity greater than or equal to T is classified as hot and the file with popularity less than threshold is classified as cold. The popularity is comparatively low for the files classified as cold. Maintaining three copies for these least popular files is wastage of space. So the replication factor of these files is set to 1. This change in replication count will make these files vulnerable for threats like data loss and unavailability. To overcome this risk and to provide data availability, the concept of erasure code is implemented. Reed-Solomon (10,4) erasure coding is applied to cold files. Reed-Solomon (10,4) divides a file into 10 equally sized blocks with 6 data blocks and 4 parity blocks. All the files classified as cold are encoded using Reed-Solomon. The above process is iterative and depending on the popularity of data over time, a data file will move from hot data to cold data and vice versa.

The files in hot category are the files with high popularity. These files will be accessed more than the cold files. So it is important to maintain more than one copy for these files. By providing more copies for the popular files, computation performance of the Hadoop system can be improved. In order to ensure the availability and to reduce the overall performance time, an additional copy of the popular file is created by incrementing the replication factor of the hot file by one.

The detailed flowchart and algorithm of the proposed replication strategy is given in Fig.2 and Fig.3 respectively. The notations used in the algorithm are given in Table I.

TABLE I
NOTATIONS USED IN THE PROPOSED ALGORITHM

Notation	Description
f_i	Accessed file
log	The input log file
ac_i	Access count of file F_i
nc_i	No.of nodes accessed F_i
rf_i	Replication factor of F_i
PI_i	Popularity Index of F_i
N	no. of files
T	Threshold
hd	Set of hot data
cd	Set of cold data



Start
Read log file
File exist?
No
Yes
Find access count for file fi
Find number of nodes accessing file fi
Find replication factor of file fi
Calculate Popularity Index(PIi)
Calculate threshold, T
PIi >= T
No
Yes
Add file to cold data set
Add file to hot data set
Set replication factor of file fi to 1
Apply erasure code to file fi
Increment Replication factor for file fi by 1
Stop

Fig. 2 Flowchart of proposed replication algorithm.

Proposed Dynamic Data Replication Algorithm

```
Input: log
Begin
1.  Set time interval
2.  For each time interval
    {
      i. read logfile
     ii. for each file $f_i$
         {
            a. Find $ac_i$, $nc_i$, $rf_i$
            b. Calculate popularity index($PI_i$) of each
               file
                      $PI_i = (ac_i * nc_i) / rf_i$
         }
    iii. Calculate the threshold,
                      $T = \frac{\sum_{i=1}^{n} PI_i}{n}$
    iv. For each file $f_i$
              Compare threshold T
                 If $PI_i >= T$
                      hd ← $f_i$
                 Else
                      cd ← $f_i$
    v. For each $f_i$ in hd,
                 Increment $rf_i$ by 1.
    vi. For each $f_i$ in cd
                 Set $rf_i$ to1
                 Encode $f_i$ using Reed-Solomon erasure
                 code
    }end for
End
```

Fig. 3 Dynamic Data Replication Algorithm

V. IMPLEMENTATION AND EVALUATION

To test the proposed algorithm, a Hadoop cluster was setup comprising of 10 nodes. The physical Hadoop cluster comprises of one master node and nine slave nodes and the version of a Hadoop distribution is 2.7. The master node acts as both namenode and datanode, thus a cluster of ten data nodes is formed. Each node is equipped with Intel Core i5 (3.30GHz) CPU and 8 GB RAM. Files were copied into HDFS from the local file system. Files of different size and types are considered for the experimentation of the algorithm. Various types of files including text, audio, video files with sizes ranging from The 600MB to 4GB were considered for this purpose. The files were divided into blocks by HDFS with default block size of 128MB. Initially, the replications for the files were three, which is the default replication count in HDFS. These files were accessed randomly from different nodes, at different time intervals. The log files were analyzed. Then in regular intervals the algorithm is executed in the HDFS cluster. The algorithm checks the access count for each file and calculates their popularity. Based on this popularity value and threshold, files are classified into two – hot or cold. Replication count for hot files is incremented and cold files are encoded using Reed-Solomon erasure code. The performance of the algorithm was analyzed comparing the result obtained by using Hadoop default replication strategy and the proposed data replication algorithm.

Five sample files of various file types were used for the evaluation purpose. The difference in the storage space used for the sample files while using default Hadoop replication strategy and erasure coding is showed in following table.

TABLE III
COMPARISON OF STORAGE SPACE UTILIZATION WITH HADOOP STRATEGY AND ERASURE CODING

	Original Size (MB)	**Hadoop 3 replica size**	**Erasure coded Size**
File 1	735.7	2207.1	1169.4
File 2	1126.4	3439.2	1790
File 3	633.4	1900.2	1056
File 4	344.8	1034.4	575
File 5	224.6	673.8	374
Total	3064.9	9254.7	4964.4

The storage space utilization of both default replication and the proposed algorithm was measured for each time interval by using the same test data. The graphical representation of the storage space utilized while experimenting with the Hadoop's default three replication and proposed replication algorithm using the sample data set over a specified time interval is given in Fig.4. and Fig.5 respectively.

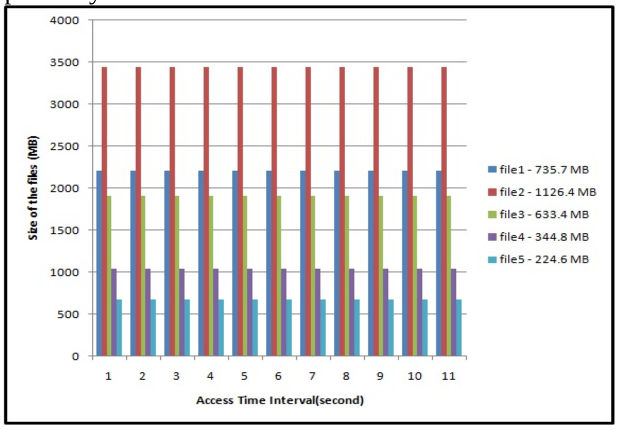


Fig. 4 space utilization for Hadoop replication strategy

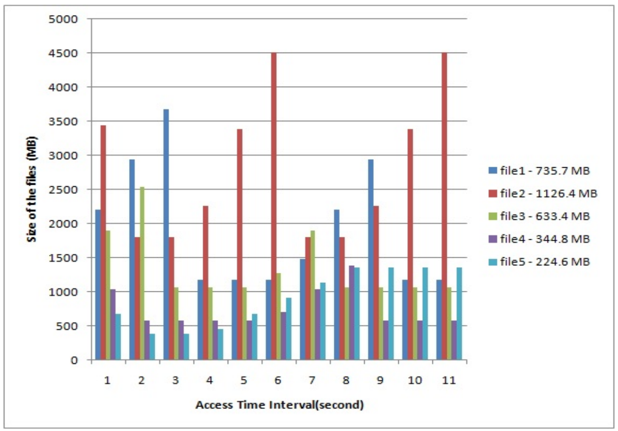


Fig. 5 Storage space utilization of the proposed replication algorithm

Based on the result of the evaluation, a comparison was done for the total storage space utilized in each time interval by the existing replication strategy of Hadoop and the proposed replication strategy. The graphical representation of the result obtained is given in Fig.6.

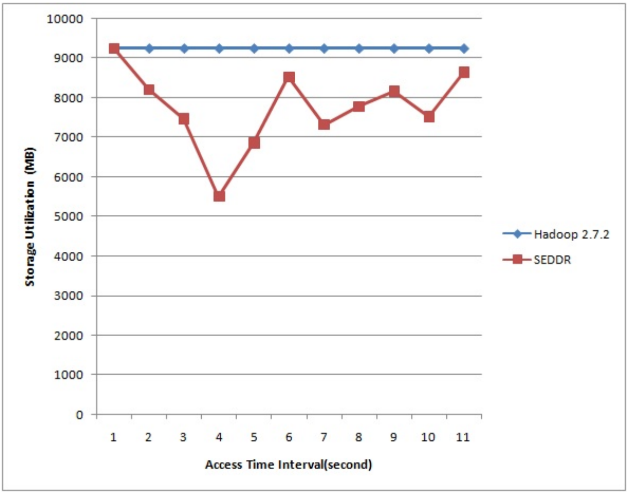


Fig. 6 Comparison of total storage utilization by Hadoop and the proposed algorithm

The result shows a significant reduction in storage space utilization in different time intervals while applying the proposed replication strategy. The proposed strategy effectively reduces the storage utilization up to 40%.

VI. CONCLUSIONS

HDFS is equipped with a mechanism that uniformly replicates every file without considering the popularity of the file. However, this replication strategy still remains a critical drawback with regards to the storage aspect. To overcome this drawback a storage efficient dynamic data replication strategy is implemented which can dynamically adapt to changes in data popularity. In this work, the storage space is optimized by reducing replication factor of the cold files and applying erasure code. Application of this strategy results in substantial storage-cost savings in hardware expenditure. This work can be improved further by optimizing the placement strategy of replicas in Hadoop cluster. Also a comparison can be made based on the performance of proposed strategy and existing method in Hadoop.

REFERENCES

[1] The Gartner website [Online]. Available: http://www.gartner.com/it-glossary/big-data/.
[2] The Apache Hadoop website. [Online]. http://hadoop.apache.org/.
[3] The Apache Hadoop Guide website. [Online]. http://hadoop.apache.org/common/docs/stable/hdfs_design.html.
[4] Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010, May). The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on* (pp. 1-10). IEEE.
[5] Wei, Q., Veeravalli, B., Gong, B., Zeng, L., & Feng, D. (2010, September). CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster. In *Cluster Computing (CLUSTER), 2010 IEEE International Conference on* (pp. 188-196). IEEE.

[6] Ananthanarayanan, G., Agarwal, S., Kandula, S., Greenberg, A., Stoica, I., Harlan, D., & Harris, E. (2011, April). Scarlett: coping with skewed content popularity in mapreduce clusters. In *Proceedings of the sixth conference on Computer systems* (pp. 287-300). ACM

[7] Abad, C. L., Lu, Y., & Campbell, R. H. (2011, September). DARE: Adaptive data replication for efficient cluster scheduling. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on* (pp. 159-168). IEEE

[8] Kaushik, R. T., Abdelzaher, T., Egashira, R., & Nahrstedt, K. (2011, July). Predictive data and energy management in GreenHDFS. In *Green Computing Conference and Workshops (IGCC), 2011 International* (pp. 1-9). IEEE.

[9] "Bsoul, M., Al-Khasawneh, A., Abdallah, E. E., & Kilani, Y. (2011). Enhanced fast spread replication strategy for data grid. *Journal of Network and Computer Applications*, *34*(2), 575-580.

[10] Cheng, Z., Luan, Z., Meng, Y., Xu, Y., Qian, D., Roy, A., ... & Guan, G. (2012, September). Erms: An elastic replication management system for hdfs. In *Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on* (pp. 32-40). IEEE.

[11] Kousiouris, G., Vafiadis, G., & Varvarigou, T. (2013, October). Enabling proactive data management in virtualized hadoop clusters based on predicted data activity patterns. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on* (pp. 1-8). IEEE.

[12] Papoulis, A. (1977). *Signal analysis* (Vol. 191). New York: McGraw-Hill.

[13] Bui, D. M., Hussain, S., Huh, E. N., & Lee, S. (2016). Adaptive Replication Management in HDFS based on Supervised Learning. *IEEE Transactions on Knowledge and Data Engineering*, *28*(6), 1369-1382.

[14] Qu, K., Meng, L., & Yang, Y. (2016, August). A dynamic replica strategy based on Markov model for hadoop distributed file system (HDFS). In *Cloud Computing and Intelligence Systems (CCIS), 2016 4th International Conference on* (pp. 337-342). IEEE.

[15] Reed, Irving S.*;* Solomon, Gustave *(1960),* Polynomial Codes over Certain Finite Fields*, Journal of the Society for Industrial and Applied Mathematics (SIAM),* 8 (2): 300–304, doi*:*10.1137/0108018