International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887*
*Volume 6 Issue III, March 2018- Available at www.ijraset.com*

# RSECB: A Reed Solomon Erasure Coding System for Hadoop Clusters

P. Joseph Sylvan[1], G. Guru Ramkoushik[2], S. Siddharth Gupta[3], S. Avinash[4]
[1, 2, 3] *B. Tech, CSE, TKREC, Hyderabad*
[4] *M. Tech CSE, Assistant Professor TKREC, Hyderabad*

***Abstract:*** ***In this paper we propose a coding technique known as Reed Solomon Erasure Coding using Backblaze method (RSECB) for Hadoop data archival system for Hadoop clusters where RS (m+n; m) codes are employed to archive data replicas in the Hadoop distributed file system (RSECB). We have originated two strategies for archiving RSECB Hadoop system (i.e., Grouping & Pipeline) to speed up the data archival process. RSECB-Grouping: which is based on Map Reduce data archival scheme which stores local key-value in the form of mapped intermediate output Key-Value pairs. With the local store in place, RSEC-Grouping does integration into a single key-value pair with the same key using all intermediate key-value pairs, succeeded by shuffling the single Key-Value pair to reducers to generate final parity blocks. RSECB-Pipeline uses multiple data nodes in a Hadoop cluster for forming data archival pipeline. RSECB-Pipeline conveys the merged single key-value pair to a successive node's local key-value store. In the pipeline, the rearmost node is accountable for outputting parity blocks. We implement RSECB in a real-world Hadoop cluster. The experimental results show that RSECB-Grouping and RSECB-Pipeline diminishes phases by a factor of 15 and 8, subsequently and also accelerates Threshold's shuffle. When block size is larger than 32MB, RSECB Hadoop distributed file system boosts the efficiency of EC and RAID roughly by 16.2% and 42.5%, respectively.***

***Keywords: Reed Solomon Erasure Coding, Backblaze, Hadoop, Archival, Reconstruction, Map Reduce, Grouping, Pipeline***

## I. INTRODUCTION

Now a days, existing disk-based archival storage systems are inadequate for Hadoop clusters due to the ignorance of data replicas and the map-reduce programming model. To tackle this problem, we develop an erasure-coded data archival system called RSECB, which archives rare accessed data in large-scale data centers to minimize storage cost. RSECB exploits parallel and pipelined encoding processes to speed up data archival performance in Hadoop distributed file system (HDFS) on Hadoop clusters. In particular, RSECB leverages the three-way data replicas and the map-reduce programming models in Hadoop cluster to boost the archival performance in HDFS. We show how to accelerate the encoding process in RSECB by the virtue of data locality of three replicas of blocks in HDFS. The following three factors motivate us to develop the erasure-code-based archival system for Hadoop clusters:

*A.* a pressing need to lower storage cost,
*B.* high cost-effectiveness of erasure-code storage and,
*C.* the popularity of Hadoop computing platforms.

Reducing Storage Cost. Petabytes of data are nowadays stored in large distributed storage systems [1] like the Google File System (GFS) [2], the Hadoop Distributed File System (HDFS) [3], the Windows Azure Storage [4]. In 2012, two million of search queries received per minute; by 2017, that number has more than double. Every minute, 2.5 million pieces of content shared in Facebook, 50 thousand applications downloaded by Apple users, 200 million messages sent via Emails [5]. Such a massive amount of data demands large-scale storage systems maintained in data centers. With an increasing number of storage nodes installed, the data storage cost goes up dramatically. A large number of nodes leads to a high possibility of failures caused by unreliable components, software glitches, machine reboots, maintenance operations and the like. To guarantee high reliability and availability in the presence of various types of failures, data redundancy is commonly used in cluster storage systems. Two widely adopted fault-tolerant solutions are replicating additional data blocks (i.e., 3X-replica redundancy) and storing additional information as parity blocks (i.e., erasure-coded storage). For example, the 3X-replica redundancy is employed in Google's GFS [2], HDFS [3], Amazon S3 [6] to achieve fault tolerance. Also, erasure-coded storage is widely used in cloud storage platforms (e.g., Windows Azure and Facebook HDFS [4] [7] [8]) and data centers [9] [10] [11]. One way to reduce storage cost is to convert

a 3Xreplica-based storage system into an erasure-coded storage. It makes sense to maintain 3X replicas for frequently accessed data. Importantly, managing non-popular data using erasure-coded schemes facilitates savings in storage capacity without adversely imposing performance penalty. A significant portion of data in data centers are considered as non-popular data because data have an inevitable trend of decreasing access frequencies. Evidence shows that most of data are accessed within a short duration of the data's lifetime. For example, over 90% of accesses in a Yahoo! M45 Hadoop cluster occur within the first day after data creation [12].

Applying Erasure-Coded Storage. Although 3X-replica redundancy (a.k.a., three-way replication) achieves high performance than erasure-coded schemes, 3X-replica redundancy inevitably leads to low storage utilization. Cost effective erasure-coded storage systems are deployed in large data centers to achieve high reliability at low storage cost [8] [9]. Reed-Solomon code [13] is a popular family of erasure codes used in Google's ColossusFS [14], Facebook's HDFS [9] [15], and several other storage systems [16] [17]. The Reed-Solomon code (RS) has a general 1.5x storage overhead in ColossusFS [14]; the Facebook's HDFS reduces the storage overhead down to 1.4x by adopting RS(m+n,m) (e.g., (10+4,10)), the overhead of which is half less than that of 3X-replica redundancy schemes. We are motivated to develop an economically friendly RSECB to archive data replicas in Hadoop clusters using erasure codes. Archiving Data in Hadoop. Hadoop is a MapReduce implementation for clusters, where HDFS stores data to offer high aggregate I/O bandwidth. Hadoop is a simple yet efficient parallel and distributed computing framework providing high scalability and fault tolerance [18] [19]. The popularity of Hadoop inspires us to develop data archival schemes for HDFS to maintain a low data storage cost for Hadoop clusters deployed in large-scale data centers. Facebook's HDFS-RAID implements RS encoding and decoding in the format of a distributed RAID file system running above HDFS [15]. HDFS-EC - developed by Cloudera engineers - employs erasure coding to reduce storage overhead compared to replication while maintaining high fault tolerance [20]. Our RSECB is similar to HDFS-RAID and HDFS-EC in the way that erasure coding is a promising and practical approach to saving storage cost. RSECB differs from HDFS-RAID and HDFS-EC in that RSECB is a data archival system that archives rarely accessed replicas in HDFS; archiving replicated data is out the scope of the existing HDFS-RAID and HDFS-EC systems. Determining rarely accessed replicas can be achieved by a few straightforward ways like Opass [21]. For example, the average number of accesses can be used to distinguish rarely accessed files from popular ones (see, for example, [22]). Only the files that have been accessed fewer than a specified threshold will be treated as nonpopular data to be archived. In this study, we rely on a dedicated popularity-tracking module to identify rarely accessed replicas. Because such a module is orthogonal to RSECB, we pay little attention to the impacts of the popularity tracking module on the performance of RSECB. Salient Features of RSECB. We make use of 3X-replica redundancy in HDFS to boost archival performance in clusters by the virtue of the MapReduce programming model in addition to reducing network traffic. We develop RSECB - a parallel data archival system - for Hadoop clusters. RSECB has the following five salient features.

*1)* RSECB manages multiple Map tasks across the data nodes that are archiving their local data in parallel.
*2)* Two archival schemes (i.e., parallel archiving and pipeline archiving) are seamlessly integrated into RSECB, which switches between the two techniques based on the size and locality of archived files.
*3)* Intermediate parity blocks generated in the Map phrase substantially lower the I/O and computing load during the data reconstruction process.
*4)* RSECB reduces network traffic among nodes during the course of data archiving, because RSECB takes the advantage of the good data locality of the 3X-replica technique.
*5)* We develop RSECB on the top of Hadoop system, allowing a system administrator to easily and quickly deploy our RSECB without having to modify and rebuild HDFS in Hadoop clusters.

The contributions of this study are:

*6)* We propose two efficient archival techniques - parallel archiving and pipeline archiving. These two schemes make use of the 3X-replica data layout to speed up archival performance.
*7)* We apply the MapReduce programming model to develop the RSECB system, where the two data archival techniques are implemented for Hadoop clusters.
*8)* We conduct extensive experiment to investigate the impacts of the block size, file size, the value size on the performance of RSECB.

## II. EXISTING SYSTEM

When data becomes corrupted in storage systems, erasure coded data can be reconstructed by using data stored in a disk array or distributed storage systems. Erasure coding has two salient features, namely, improving fault tolerance performance and minimizing the storage-capacity overhead in date centers. For instance, Huang et al. adopted an erasure-coded or RS codes to

build an erasure-coded storage cluster [23] to improve both data storage reliability and efficiency. RS(m+n,m) encodes source data with RS-generated-coding matrix. It is worth mentioning that the RS generated-coding matrix contains two matrices:

• Identical Matrix. This is a m × m matrix, where all the diagonal elements are set to 1 and the other elements are set to 0.

• Redundancy Matrix. This is a m × n matrix generating parity blocks.

With a simple linear algebra calculation, the parity data blocks are derived from the m × n redundancy matrix. In a data archival scenario, we simply apply the redundancy matrix to obtain calculate r parity blocks, which can be employed to reconstruct blocks for data failures. A RS(m+n,n) code scheme can sustain up to n numbers of concurrent blocks failures.

Existing work provides high resilience and speedup the data archival performance on cold data but performance degrades for hot data. So, in existence, this system is limited cold data.

## III. PROPOSED SYSTEM

In our proposed system, we address the problem of providing of high resilience to hot data with low storage overhead in a distributed system and also, we will investigate a way of choosing intermediate parity blocks to be kept in the local key value store to optimize the reconstruction performance. Recent research showed that it is feasible to use erasure coding for hot data as well. Our proposed work is two stage process – Data Archival and Data Reconstruction.

Cold data is the data which less frequently accessed by the particular user whereas hot data is the most frequently used data by the user. Even though the existing system makes use of Reed Solomon Erasure Coding using XORing which makes the system expensive and adds onto storage overhead and works only on cold data, our proposed system makes use of Reed Solomon Erasure Coding using Backblaze method of matrix inversion.

## IV. ARCHITECTURE

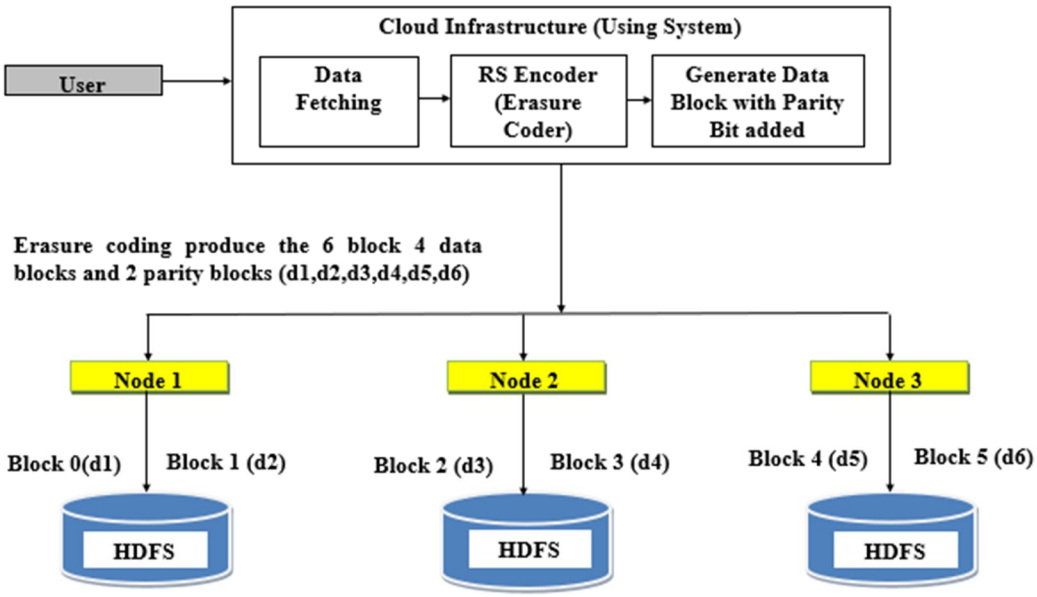


Fig. 1: Architecture

In, this architecture, when the user uses our application he has to first upload a file which is part of *data fetching* and upon confirmation of uploading the user applies the RS Encoding which splits the file into 6 erasure coded blocks: two parity blocks and four data blocks. These erasure coded blocks are then distributed among different nodes known as servers when the user clicks on distribute. This describes the archival part of the system. Now the reconstruction part makes uses of pipelining, where the user first requests for the desired file, upon sending the request to the system, the cloud searches the related data blocks among all the nodes or servers and once retrieved, the user clicks on RS Decode to get the required original file back. In case if an error occurs, i.e., one or many of the servers go down or gets corrupted, the RS erasure coding retrieves the lost files back making use of the backblaze method using matrix inversion technique by creating data replicas in the initial archival stage.

The stages that are considered to have much significance in this are as follows:

*A. Archival Stage*

1) *Data fetching:* The user selects a file to store in cloud server which are taken as input to our program.
2) *Reed Solomon Encoder:* The Input File is first split into multiple data blocks. Each Data blocks are encoded with RS Encoding. Reed Solomon Code generates the parity block and takes data blocks as the input and produces coded blocks among them two are used as parity blocks and remaining are data blocks.
3) *Distribution Phase:* The Erasure coded data blocks are distributed across the multiple nodes to provide high reliability. Nodes receive the data blocks and saves them in Hadoop File System(HDFS).

*B. Reconstruction Stage*

1) *User Request:* The user issues a request for retrieving the file.
2) *Cloud Server:* Upon receiving the request, the cloud searches across the nodes for respective data blocks that make up the file User request.
3) *RS Decoder:* Once it finds all the corresponding blocks the reconstruction process started by applying RS decoding to file block and reconstructed file it sends to the user.
4) *Error Phase:* The Erasure coded data blocks are distributed across the multiple nodes to provide high reliability. Nodes receive the data blocks and saves them in Hadoop File System(HDFS).

## V. ALGORITHM

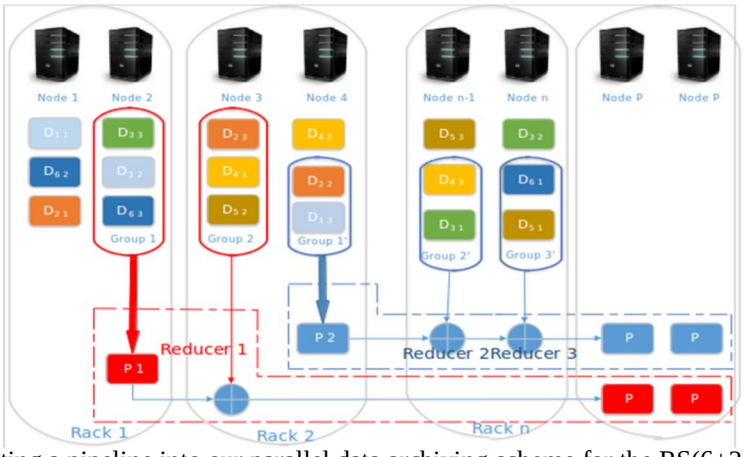


Fig. 2. Incorporating a pipeline into our parallel data archiving scheme for the RS(6+2,6)-coded storage.

To boost the performance of the parallel data archiving scheme, we design a pipeline and incorporate the pipelining technique into our RSECB system. Fig. 2 shows a way of incorporating a pipeline into our parallel data archiving scheme for the RS (6+2,6)-coded storage. In the parallel archiving scheme, intermediate parity blocks created by mappers running on a node are delivers to reducers. Unlike this process, the pipelined data archiving scheme delivers intermediate parity blocks (see blocks P1 and P2 in Fig. 2) to subsequent nodes (for example, node 3 and node n−1 in Fig. 2). Next, the subsequent nodes repeatedly deliver their intermediate parity blocks to their subsequent nodes (see, for example, node n is the subsequent one of node n−1). Finally, the last node that has no subsequent node writes its parity blocks to the file system (i.e., HDFS).

Algorithm for RSECB Pipeline Archiving :

Input: Map-Register, Map-Counter-Key, Val-Buffer

Map-Register: spawned Mapper tasks on each node

Map-Counter-Key: amount of mapper tasks which have already stored the key's value into data server

Val-Buffer: store the value of Key-Value pairs

1) If Val-Buffer.get ($KEY_k$) = = null then
2) Val-Buffer.put ($KEY_k$, VALUE)
3) Map-Counter-$KEY_k$++
4) Else
5) Return Val-Buffer.get($KEY_k$) to Mapper
6) Mapper executes exclusive or calculation to returned value and value of $KEY_k$ emitted

*7)* Map-Counter-$KEY_k$++
*8)* If Nodes is first node && Map-Counter-$KEY_k$ = = Map-Register then
*9)* Write calculated result to subsequential node
*10)* Else if node is not last node && Map-Counter-$KEY_k$ = = Map-Register+1 then
*11)* Write calculated result to subsequential node
*12)* Else if node is not last node && Map-Counter-$KEY_k$ = = Map-Register+1 then
*13)* Write calculated result to reducer
*14)* Val-Buffer remove $KEY_k$
*15)* Else
*16)* Store calculated result to data server
*17)* Map-Counter-$KEY_k$++
*18)* End
*19)* End

According to data placement governed by the layout of HDFS, all nodes can be divided into multiple groups, each of which handles data archiving operations in a pipelined manner. For example, Fig. 2 shows that node group 1 consists of nodes 2 and 3, whereas node group 2 contains nodes 4, n−1, and n. In this example, RSECB builds an archiving pipeline between nodes 2 and 3 in node group 1; RSECB constructs another archiving pipeline among nodes 4, n−1, and n in node group 2. These two archiving pipelines perform data archival operations in parallel. Multiple data archiving pipelines are capable of simultaneously carrying out encoding processes for separate data sets, thereby delivering high data-archival performance through improved archival parallelism.

## VI. RESULT ANALYSIS

Now we provide an overall performance evaluation of RSECB by comparing it with the two existing systems, namely HDFS-RAID [15] and HDFS-EC [20]. Fig. 3(a) shows the data archiving times of our RSECB-Grouping and RSECB-Pipeline with the two alternatives under various block size. The experimental results show that when block size is larger than 32MB, RSECB improves the performance of HDFS-RAID and HDFS-EC by approximately 31.8% and 15.7%, respectively. If the block size is as small as 16 MB, RSECB shorten the execution time of HDFS-RAID and HDFS-EC by 7.3% and 9.6%. No noticeable improvement is observed if the block size is below 8 MB. Two factors contribute to the performance improvements. First, RSECB significantly shortens the shuffle and reduce phases in the large-block-size cases, because RSECB combines all intermediate output Key-Values into a single Key-Value on each node. Second, RSECB optimizes performance through the grouping (i.e., local key-value stores) and pipelining techniques. Fig. 3(b) reveals the archiving performance of RSECB, HDFS-RAID, and HDFS-EC as a function of file size. The performance trend observed in Fig. 3(b) is similar to that shown in Fig. 3(a). Fig. 3(c) reveals that the grouping technique of RSECB coupled with a pipeline significantly reduces network traffic for data archival workload. For example, RSECB lowers the network traffic of HDFS-RAID and HDFS-EC by 90.5% and 50.9%, respectively.

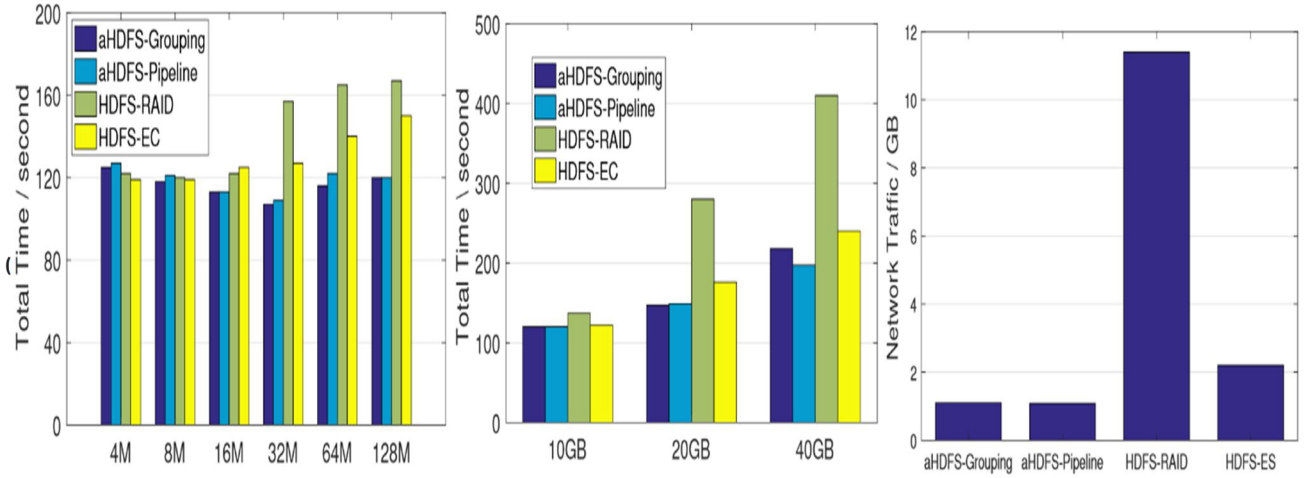


Fig. 3: An overall performance comparisons of RSECB, HDFS-RAID, and HDFS-EC.

## VII. CONCLUSION

We present an erasure-coded data archival system - RSECB - in the realm of Hadoop cluster computing. We proposed two archiving strategies called RSECB-Grouping and RSECB-Pipeline to speed up archival performance in Hadoop distributed file system or HDFS. Both the archiving schemes adopt the MapReduce-based grouping strategy, which wraps up multiple intermediate key−value pairs sharing same key into one key−value pair on each node. RSECB-Grouping transfers the single key−value pair to a reducer, whereas RSECB-Pipeline delivers this key−value pair to the subsequent node in the archiving pipeline. We implemented these two archiving strategies, which were compared against the conventional MapReduce-based archiving strategy referred to as Baseline. The experimental results show that RSECB-Grouping and RSECB-Pipeline can improve the overall archival performance of Baseline by a factor of 4. In particular, RSECB-Grouping and RSECB Pipeline speed up Baseline's shuffle and reduce phases by a factor of 10 and 5, respectively. In addition, RSECB-Grouping and RSECB-Pipeline significantly lower the network I/O traffic by 87% and 89%, respectively.

## VIII. FUTURE ENHANCEMENTS

As a future research direction, we will develop a data reconstruction system to deal with block failure issues on Hadoop clusters. We plan to apply the grouping and pipelining strategies to the reconstruction system to speed up the reconstruction process. To optimize reconstruction performance, we will investigate a way of choosing intermediate parity blocks to be kept in the local key-value store so that we can make use of dynamic Reed Solomon Erasure coding (m+n;n) using backblaze method of matrix conversion.

## REFERENCES

[1] 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.
[2] S. Ghemawat, H. Gobioff, and S.-T. Leung, The google file system in ACM SIGOPS operating systems review, vol. 37, no. 5. ACM, 2003, pp. 29–43.
[3] D. Borthakur, The hadoop distributed file system: Architecture and design, 2007 Apache Software Foundation, 2012.
[4] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci et al., Windows azure storage: a highly available cloud storage service with strong consistency in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 143–157.
[5] 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.
[6] D. Robinson, Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services enables you to reach business goals faster. Emereo Pty Ltd, 2008.
[7] 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.
[8] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin et al., Erasure coding in windows azure storage. in Usenix annual technical conference. Boston, MA, 2012, pp. 15–26.
[9] A. Thusoo, Z. Shao, S. Anthony, D. Borthakur, N. Jain, J. Sen Sarma, R. Murthy, and H. Liu, Data warehousing and analytics infrastructure at facebook in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 1013– 1020.
[10] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, Availability in globally distributed storage systems. in OSDI, 2010, pp. 61–74.
[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] 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.
[13] 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.
[14] Colossus: Successor to the google file system (gfs) http://www.highlyscalablesystems.com/3202/ colossus-successor-to-google-file-system-gfs/.
[15] 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.
[16] K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, A hitchhiker's guide to fast and efficient data reconstruction in erasure-coded data centers in Proceedings of the 2014 ACM conference on SIGCOMM. ACM, 2014, pp. 331–342.
[17] F. B. Schmuck and R. L. Haskin, Gpfs: A shared-disk file system for large computing clusters. in FAST, vol. 2, 2002, p. 19.
[18] J. Dean and S. Ghemawat, Mapreduce: simplified data processing on large clusters Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
[19] 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.
[20] M. Xia, M. Saxena, M. Blaum, and D. A. Pease, A tale of two erasure codes in hdfs into appear in Proceedings of 13th Usenix Conference on File and Storage Technologies, 2015.
[21] J. Yin, J. Wang, J. Zhou, T. Lukasiewicz, D. Huang, and J. Zhang, Opass: Analysis and optimization of parallel data access on distributed file systems in

2015 IEEE Int'l Parallel and Distributed Processing Symp. IEEE, 2015, pp. 623–632.

[22] M. Tang, B.-S. Lee, X. Tang, and C.-K. Yeo, The impact of data replication on job scheduling performance in the data grid Future Generation Computer Systems, vol. 22, no. 3, pp. 254–268, 2006.

[23] J. Huang, F. Zhang, X. Qin, and C. Xie, Exploiting redundancies and deferred writes to conserve energy in erasure-coded storage clusters ACM Transactions on Storage (TOS), vol. 9, no. 2, p. 4, 2013.

[24] 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.

[25] J. Huang, X. Liang, X. Qin, P. Xie, and C. Xie, Scale-rs: An efficient scaling scheme for rs-coded storage clusters Parallel and Distributed Systems, IEEE Transactions on, vol. 26, no. 6, pp. 1704–1717, 2015.