Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

Based on an analysis of Amazon S3, real service simulation techniques are used to study the latency performance of distributed storage systems

SU RIGUGE[1], DR MOHAMMAD NIZAMUDDIN INAMDAR [2a]
PhD Research Scholar in Engineering, Lincoln University College Malaysia
Professor in Lincoln University College Malaysia
Contact Details: [a] nizamuddin@lincoln.edu.my

Abstract
To create the parity nodes, current erasure codes mainly rely on data nodes. "If we can increase the number of parity nodes, we may increase our chances of restoring the original data," the more error tolerance there is. Because data nodes are frequently queried to aid in the repair of parity nodes, as the number of parity nodes increases, so will the storage overhead and the burden of repair on data nodes. For example, all data nodes in LRC [25, 26] need to be fixed if a global parity node fails. The "increasing demands on the network's data nodes" mean that processing read requests for data nodes will take longer. A program where regular data and FH HTSC, or High Failure-tolerant Hierarchical Tree Structure Code."

Keywords: Data Nodes, Data Retrieval, Hierarchical Tree Structure.

INTRODUCTION

In the past ten years, the popularity of searching, social networking, and e-commerce has all significantly increased. We generate enormous amounts of digital data every day. Researchers and industry are having a hard time coming up with affordable storage system designs. The need to create large-scale distributed storage systems stems from the explosion of data. A couple of examples are Windows Azure Storage (WAS) [3] and the Hadoop Distributed File System (HDFS) [2]. Large data, fast computing, and cloud-scale applications may all be met with great ubiquity and dependability with the help of these storage systems. A lot of low-cost, unstable storage devices are frequently used in the development of large distributed storage systems, and these individual nodes are prone to failure. There are significant benefits to Despite the scalability of these systems, failure is the norm rather than the exception. [1] We must thus prevent frequent system failures and ensure that these systems are robust and dependable. In the past ten years, the popularity of searching, social networking, and e-commerce has all significantly increased. We generate enormous amounts of digital data every day. Researchers and industry are having a hard time coming up with affordable storage system designs. The need to create large-scale distributed storage systems stems from the explosion of data. A couple of examples are Windows Azure Storage (WAS) [3] and the Hadoop Distributed File System (HDFS) [2]. Large data, fast

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

computing, and cloud-scale applications may all be met with great ubiquity and dependability with the help of these storage systems. It is typical for a vast distributed storage system to be created, a lot of cheap, unstable storage devices must be used, and each node in the system is prone to failure. Although these systems have significant scalability advantages, failure is the rule rather than the exception. [1] As a result, we must overcome frequent system failures while ensuring that these systems" are reliable and resilient.

LITERATURE REVIEW

In large-scale distributed storage systems, redundancy is achieved through replication or erasure coding, which provides a high level of failure protection.

GFS ensures that data can "be accessed reliably" by distributing it across three separate storage nodes. This simple replication approach is well-suited to Google's frequent read requirements [6]. Because of the high storage requirements for a given level of fault tolerance, "replication" ensures data availability and prevents data loss in the event of node failures.

Files with a fixed size M can "be divided into k parts (sometimes referred to as "k nodes"), each of size M, and encoded into n encoded nodes for use in generic erasure code systems." Compared to replication, the storage requirements for a given level of The erasure coding method can significantly reduce dependability. Reed Solomon (RS) codes, for example, are among the most popular and efficient storage codes due to their Maximum-Distance-Separable (MDS) feature [4]. It is a codeword-containing element in a "standard code." An MDS codeword contains n nodes, and any k of them can be used to reassemble the entire text. In addition, we call a codeword a systematic code if it contains original data nodes. Any feasible MDS codeword contains k original data nodes and an equal number of "n-k parity" nodes [5]. Nodes of a codeword are typically kept on multiple storage devices in different locations to avoid failures caused by common" reasons Any three of the six nodes in a (6,3) "MDS codeword can decode all of the information in the codeword, as shown in Fig. 1.1. The code is logical because d1–d3 are not coded. Large-scale distributed storage systems that use coding frequently use a code with a predetermined set of (n, k) parameters and a specified size for each codeword to store its data, making them easier to manage and operate. The two types of RS codes used in HDFS and GFS II are (14,10) for Face-HDFS books and (9,6) for GFS II [7, 8]. For actual large-scale distributed storage systems, a codeword consists of several files with a fixed total size. We can better investigate the storage system's. Characteristics are due to the constant coding rate.

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

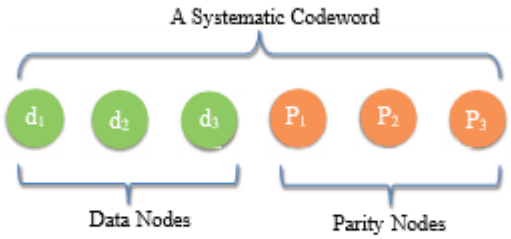

Figure 1.1 A codeword of systematic (6,3) MDS code.

STATEMENT OF THE PROBLEM

In distributed storage systems, access latency can have a significant impact on the user experience. Erase codes have long been known to be more reliable than replication for the same storage cost, but it has only recently been demonstrated that "coding" may reduce access latency [9]. Chinese researchers proved for the first time that codes can reduce queueing delays [10]. Following that, a lot of research has focused on the Redundant Scheme (RedS), which suggests that sending redundant requests to storage systems can reduce latency. While some of the research [11] is theoretical, others [12] use trace-driven simulations to test their findings. Shah et al. [13] describe how duplicate queries can help to reduce latency. In [14, 15], RedS and RanS latency-cost tradeoffs will be investigated. Coding outperforms replication in terms of delay for the same storage cost, according to [16].

Despite these efforts, "when evaluating latency performance, certain important practical requirements of distributed storage systems" are ignored.

When users request files of "varying sizes from a codeword," [17, 18] can only handle cases in which each request reads the entire codeword. When a replication system receives a request for files from multiple data nodes at the same time, no comparison will be made between replication and coding, as was done in the case of single node reads in [19]. A more generalized version of the "second situation is more in line with current large-scale storage." systems. Most previous studies [20, "21, 22] assumed pure exponential service times, but our real-world observations on Amazon S3 show that this assumption falls far short of the real-world reality, as demonstrated by Liang et al.

In addition to data retrieval, distributed storage systems frequently repair failures [23]. GFS, Amazon "S3," and WAS are three examples of distributed storage systems that rely heavily on faulty hardware. Recovery procedures necessitate approximately 180TB of data transfer between racks in Facebook HDFS per day, and there are multiple instances of high repair rates each day

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

[24]. Increased frequency of read requests will undoubtedly lead to an increase in access latency under the same conditions because there are fewer nodes.
to store the information requested by a read request. When a storage node fails, no data is lost, and repair requests are prioritized over read/write operations to prevent data loss. Repair requests are prioritised over read requests, which may have a significant impact on access times. As a result, in this study, we will look into the impact of repair requests on read request latency.

The study aims
to identify the most effective methods for direct readings to reduce latency.
Research Questions
• What are the best methods for direct readings to minimize latency?

RESEARCH METHODOLOGY

Erasure codes and replications in distributed storage systems are one method for dealing with system failures [29]. Codes commonly used in practice are systematic codes, which means that each codeword contains a copy of the original data. Erasure coding can also be used in Windows Azure storage (WAS) systems, but only when a file reaches a certain size (for example, 3GB). If you only need a portion of the file, the storage nodes will be able to retrieve it from one of the codeword's massive files, which are frequently extremely large in practice (we refer to these as direct read requests) [31, 32, 33]. Requests for k-access reads, in which each request must read the entire file in a codeword and access Another type of request involves at least k nodes. Latencies in a distributed storage system vary according to the number of direct and k-access reads performed. To our knowledge, this is the first time that direct readings have been thoroughly investigated in any previous study [30].
Latency performance has been deemed "crucial in distributed storage systems, and some studies claim that codes can minimize latency in data centers, while many other strategies have been proposed to reduce latency in distributed storage" infrastructures. Previous research largely ignored direct readings and focused solely on k-access accesses. There has been no research into how RedS can speed up direct readings."RedS sends requests to all n nodes for each k-access read, the Random Scheme (RanS) Requests are only sent randomly to those k nodes. Compared to RanS, RedS necessitates a greater investment of time and resources. When it comes to practical distributed storage systems, RanS is a popular choice because it is simple to implement and does not require any additional information or resources.

RESEARCH DESIGN

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

Redundant Request Technique (RedS) is a recently popular read scheme for (n, k) MDS-coded storage systems [34–36]. "Regardless of how many files a read request requests in a codeword, RedS divides it into n jobs and distributes them to each of the n nodes. When k nodes out of n have finished providing their services, the request is considered complete, and the remaining n k jobs are terminated immediately.

We created a RedS-based solution that can handle requests for files of various sizes while reducing access latency. We refer to it as the Flexible Redundant Scheme (FRedS).

DATA ANALYSIS

In general, "To save your data using HTSC(D) or FH HTSC, you must combine your files into a single large one of size M, say 1 to 3 GB, and then divide it into K parts (D, h). The fixed size M can be calculated by combining the available storage space at each node with the parameters of HTSC(D, h) or FH FH HTSC. (D, h). Users are frequently only interested in a subset of a "file's uncoded systematic component," which is stored in one of the K nodes. Previous research assumed that readers would want access to the entire contents of a "Considering that every bit of information is now stored in the K-tree, the use of a codeword To put it simply, it is a significant change. However, this oversimplifies the situation rather than reflecting reality. In WAS, for instance, erasure coding is available only for files beyond a certain size threshold (say, 3GB) [31]. Most people only use a small portion of the 3GB available, so it's understandable that it's a waste. This corresponds to the HTSC's intended functionality (D, h). As a result, we will focus on read requests from customers who are only interested in a subset of the information stored in one of the K data nodes. "Inferences drawn from this research

CONCLUSION

In general, "To save your data using HTSC(D) or FH HTSC, you must combine your files into a single large one of size M, say 1 to 3 GB, and then divide it into K parts (D, h). The fixed size M can be calculated by combining the available storage space at each node with the parameters of HTSC(D, h) or FH FH HTSC. (D, h). Users are frequently only interested in a subset of a "file's encoded systematic component," which is stored in one of the K nodes. Previous research assumed that readers would want access to the entire contents of a "Considering that every bit of information is now stored in the K-tree, the use of a codeword to Describe it as a significant shift. However, this oversimplifies the situation rather than reflecting reality. In WAS, for example, erasure coding is only available for files larger than a certain size threshold (say, 3GB) [31]. Most people only use a small portion of the 3GB available, so it's understandable that it's a waste. This corresponds to the HTSC's intended functionality (D, h). As a result, we will focus on read requests from

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

customers who are only interested in a subset of the information stored in one of the K data nodes." Inferences drawn from this research.

LIMITATIONS OF THE STUDY

Because nodes and connections "must be protected," it is difficult to ensure effective security in distributed systems. Data and messages may be lost in the network as they move between nodes. In comparison to a single user system, the database for distributed systems is extremely complex and difficult to manage. If all of the distributed system's nodes attempt to "communicate data at once," the network may become overloaded.

REFERENCES

[1]. 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.

[2]. M. Foley, "High availability HDFS," in 28th IEEE Conference on Massive Data Storage, MSST, vol. 12, 2012.

[3]. 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 ATC, 2012, pp. 15–26.

[4]. N. B. Shah, K. Lee, and K. Ramchandran, "The MDS queue: Analysing the latency performance of erasure codes," in IEEE International Symposium on Information Theory (ISIT), 2014, pp. 861–865.

[5]. B. Y. Kong, J. Jo, H. Jeong, M. Hwang, S. Cha, B. Kim, and I.-C. Park, "Low- complexity low-latency architecture for matching of data encoded with hard systematic error-correcting codes," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 7, pp. 1648–1652, 2014.

[6]. K. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchan- dran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the Facebook warehouse cluster," in Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems. USENIX, 2013.

[7]. A. Fikes, "Storage architecture and challenges," Talk at the Faculty Summit, 2010.

[8]. D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V.-A. Truong, L. Barroso,C. Grimes, and S. Quinlan, "Availability in globally distributed storage sys- tems." in OSDI, 2010, pp. 61–74.

[9]. A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchan- dran, "Network coding for distributed storage systems," IEEE Transactions on Information Theory, vol. 56, no. 9, pp. 4539–4551, 2010.

[10]. K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," IEEE Transactions on Information Theory, vol. 57, no. 8, pp. 5227–5239, 2011.

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

[11]. V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh, "Asymp- totic interference alignment for optimal repair of MDS codes in distributed data storage," 2011.

[12]. N. B. Shah, K. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in Information Theory Workshop (ITW), 2010 IEEE. IEEE, 2010, pp. 1–5.

[13]. V. R. Cadambe, S. A. Jafar, and H. Maleki, "Distributed data storage with minimum storage regenerating codes-exact and functional repair are asymp- totically equally efficient," arXiv preprint arXiv:1004.4299, 2010.

[14]. N. B. Shah, K. Rashmi, P. V. Kumar, and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," IEEE Transactions on Information Theory, vol. 58, no. 4, pp. 2134–2158, 2012.

[15]. A. Duminuco and E. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," in 29th IEEE International Conference on Dis- tributed Computing Systems. IEEE, 2009, pp. 376–384.

[16]. A. Duminuco and E. W. Biersack, "Hierarchical codes: A flexible trade-off for erasure codes in peer-to-peer storage systems," Peer-to-peer Networking and Applications, vol. 3, no. 1, pp. 52–66, 2010.

[17]. M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali,S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," in Proceedings of the 39th international conference on Very Large Data Bases. VLDB Endowment, 2013, pp. 325–336.

[18]. J. Li and B. Li, "Erasure coding for cloud storage systems: A survey," Ts- inghua Science and Technology, vol. 18, no. 3, pp. 259–272, 2013.

[19]. A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," Proceedings of the IEEE, vol. 99, no. 3, pp. 476–489, 2011.

[20]. A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, "Ef- ficientRijndael encryption implementation with composite field arithmetic," in Cryptographic Hardware and Embedded Systems?CHES 2001. Springer, 2001, pp. 171–184.

[21]. J. Brutlag, "Speed matters for Google web search," Google. June, 2009.

[22]. L. Huang, S. Pawar, H. Zhang, and K. Ramchandran, "Codes can reduce queueing delay in data centers," in IEEE International Symposium on Infor- mation Theory (ISIT), 2012, pp. 2766–2770.

[23]. N. B. Shah, K. Lee, and K. Ramchandran, "When do redundant requests re- duce latency?" in the 51st Annual Allerton Conference on Communication, Control, and Computing. IEEE, 2013, pp. 731–738.

[24]. G. Liang and U. C. Kozat, "TOFEC: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes," in Proceedings of INFOCOM. IEEE, 2014, pp. 826–834.

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)

[25]. G. Joshi, Y. Liu, and E. Soljanin, "On the delay-storage trade-off in content download from coded distributed storage systems," IEEE Journal on Selected Areas in Communications, vol. 32, no. 5, pp. 989–997, 2014.

[26]. Y. Xiang, T. Lan, V. Aggarwal, and Y. F. R. Chen, "Joint latency and cost optimization for erasurecoded data center storage," ACM SIGMETRICS Per- formance Evaluation Review, vol. 42, no. 2, pp. 3–14, 2014.

[27]. B. Li, A. Ramamoorthy, and R. Srikant, "Mean-field-analysis of coding versus replication in cloud storage systems," in Proceedings of INFOCOM. IEEE, 2016.

[28]. G. Liang and U. C. Kozat, "Fast Cloud: Pushing the envelope on delay perfor- mance of cloud storage with coding," IEEE/ACM Transactions on Network- ing, vol. 22, no. 6, pp. 2012–2025, 2014.

[29]. G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica,D. Harlan, and E. Harris, "Scarlett: coping with skewed content popularity in mapreduce clusters," in Proceedings of the sixth conference on Computer systems. ACM, 2011, pp. 287–300.

[30]. A. Kala Karun and K. Chitharanjan, "A review on hadoophdfs infrastructure extensions," in Conference on Information & Communication Technologies (ICT). IEEE, 2013, pp. 132–137.

[31]. M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action. Cambridge University Press, 2013.

[32]. H. A. David and H. N. Nagaraja, Order statistics. Wiley Online Library, 1981.

[33]. M. Rahman and L. Pearson, "Moments for order statistics in shift parameter exponential distribution," Journal of Statistical Research, vol. 36, no. 1, pp. 75–83, 2002.

[34]. S. B. Wicker and V. K. Bhargava, Reed-Solomon codes and their applications. John Wiley & Sons, 1999.

[35]. Q. Shuai, V. O. K. Li, and Y. Zhu, "Performance models of access latency in cloud storage systems," in Fourth Workshop on Architectures and Systems for Big Data, 2014.

[36]. M. Blaum, J. Brady, J. Bruck, and J. Menon, "Evenodd: An efficient scheme for tolerating double disk failures in raid architectures," IEEE Transactions on Computers, vol. 44, no. 2, pp. 192–202, 1995.

[37]. L. Xu and J. Bruck, "X-code: Mds array codes with optimal encoding," IEEE Transactions on Information Theory, vol. 45, no. 1, pp. 272–276, 1999.

[38]. P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, andS. Sankar, "Row-diagonal parity for double disk failure correction," in Pro- ceedings of the 3rd USENIX Conference on File and Storage Technologies, 2004, pp. 1–14.

Available online at www.jomaar.com

JOURNAL OF MANAGEMENT AND ARCHITECTURE RESEARCH
E ISSN: 2689 3541; P ISSN 2689 355X
Volume:06; Issue:01 (2024)