ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 15, March 2016

An Efficient Degraded Reads for Map Reduce in Heterogeneous Erasure Coded Storage Clusters

M.A. Mohamed Razak
PG Scholar, M.E., CSE (Final Year)
Anna University Regional Campus, Coimbatore
mdrazakme@gmail.com

L. Viji
Assistant Professor/IT
Velalar College of Engg and Tech, Erode
vijibtechit20@gmail.com

ABSTRACT: Degraded reads have become performance critical operations, due to the fact that temporary errors account for the majority of failures in modern storage systems. To boost the performance of degraded reads in practical erasure-coded storage systems, it is necessary to take into account parallel I/Os and node heterogeneity when performing degraded reads. The System observation is that while the tasks are running, the MapReduce job does not completely utilize the available network resources. The proposes degraded-primary scheduling, whose main idea is to schedule several degraded tasks in advance stages of a MapReduce job and allow them to download data first using the unused network resources. To reduce the redundancy in the storage due to replication, erasure coding can be used. It conducts mathematical analysis and discrete event simulation to show the performance gains of degraded-first scheduling.

KEYWORDS*:* Degraded reads; boost the performance; erasure coded storage systems; MapReduce Jobs; degraded first scheduling.

1 INTRODUCTION

Distributed storage systems, such as GFS and Azure, have been widely adopted in enterprises to provide large-scale storage services. Nevertheless, component failures are frequent and diverse in large-scale storage systems. To make sure data availability, storage systems generally stripe data redundancy across multiple storage nodes (or servers). Replication is traditionally used to provide data redundancy, yet it introduces high storage overhead and becomes a scalability bottleneck. Alternatively, erasure coding provides space-optimal data redundancy while achieving the same fault tolerance as replication. It operates by encoding data into multiple fragments, such that any subset of fragments can sufficiently reconstruct the original data. An erasure coding has been widely deployed and evaluated in large-scale storage systems by both commercial and intellectual community.

Big data technologies are important in providing more accurate analysis, which may guide to more concrete decision-making resulting in better operational efficiencies, cost reductions, and reduced risks for the big business. To exploit the power of big data, you would require an infrastructure that can manage and process enormous volumes of structured and unstructured data in real-time and it can protect data privacy and security. There are different technologies in the market from different vendors

including Amazon, IBM, Microsoft, etc., to handle big data. Although looking into the technologies that handle big data, we examine the following two classes of technology.

Generally MapReduce pattern is based on sending the computer to where the data resides. MapReduce program executes in two stages, namely map stage and reduce stage.**[1]**Map stage: The map or mapper's job is to process the input data. In general the input data is in the form of file and is stored in the Hadoop distributed file system. The input file is passed to the mapper task line by line and processes the data. It creates several small chunks of data. **[**2]Reduce stage: It is the combination of the shuffle stage and the reduce stage. The Reducer's job is to processes the data that comes

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 15, March 2016

from the mapper. In later processing, it produces a new set of output, which will be stored in the HDFS. Replication provides a easy and robust form of redundancy to shield against most failure scenarios. It also ease scheduling compute tasks on locally store data blocks by providing multiple replicas of each block. During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster. The framework manages all the details of data-passing such as issuing tasks and verifying task around the cluster between the nodes.

2 DESIGN REQUIREMENTS

Transient failed nodes are degraded, as the unavailable data to be reconstructed from the unused surviving node. Our goal is reduce the redundancy in storage and data transfer cost. The following requirements to be assist to design efficient degraded read solution.

Single Node Hadoop Installation: Hadoop is a framework written in Java for running applications on large clusters of product hardware and incorporates features similar to those of the Google File System (GFS) and the MapReduce computing paradigm. HDFS is a highly fault-tolerant distributed file system and, like Hadoop in common, designed to be deployed on low-cost hardware. It provides high throughput access to application data and is appropriate for applications that have large data sets. A DataNode stores data in the Hadoop File System. A functional file system has in excess of one DataNode, with the data replicated across them. The NameNode is the centre piece of an HDFS file system. It keeps the index of all files in the file system, and tracks where across the cluster the file data is kept. It does not store up the data of these file itself. The Jobtracker is the service within hadoop that farms out MapReduce to exact nodes in the cluster, ideally the nodes that have the data, or atleast are in the same rack. TaskTracker is a node in the cluster that accepts tasks Map, Reduce and Shuffle operations from a Jobtracker. Secondary Namenode whole function is to have a checkpoint in HDFS. It is just a helper node for namenode.

Erasure Coded Generating: Erasure coded storage systems add redundancy for fault tolerance. Specifically, a system of n disks is partitioned in to k disks that hold data and m disks that hold coding information. The coding information is calculated from the data using an erasure code. For realistic storage systems, the erasure code typically has two properties. Earliest, it must be Maximum Distance Separable (MDS), which means that if any m of the n disks fails, their contents possibly recomputed from the k surviving disks. Next, it must be systematic, which means that the k data disks hold un-encoded data.

An erasure coded storage system is partitioned into stripes, which are collections of disk blocks from each of the n disks. The blocks are partitioned into symbols, and there is a fixed number of symbols for each disk in each stripe. Here denote this quantity r. The stripes perform encoding and decoding as independent units in the disk system. Therefore, to improve hot spots that can occur because the coding disks may need more activity than the data disks, one can spin the disks' identities on a stripe-by-stripe basis.

Recovering Failure Pattern: The proposed expresses an virtual symbol as a function of real symbols by solving a system of equations. However, we note that for some failure patterns (i.e., the set of failed nodes), the system of equations cannot return a unique solution. A failure pattern is said to be good if we can uniquely express the virtual symbols as a function of the real symbols, or bad otherwise. Our goal is to reduce the recovery bandwidth even for bad failure patterns. We now extend our baseline approach of proposed to deal with the bad failure patterns, with an objective of reducing the recovery bandwidth over the conventional recovery approach.

We evaluate the recovery performance. For a given (n, k), we configure our HDFS testbed with n DataNodes, one of which also deploys the RaidNode for striping the encoded data. We prepare a kGB of original data as our input. By our observation, the input size is large enough to give a steady throughput. HDFS first stores the file with the default 3-replication scheme. Then the RaidNode stripes the replica data into encoded data using either RS codes or IA codes. The encoded data is stored in n DataNodes. We rotate node identities when we place the blocks so that the parity blocks are evenly distributed across different DataNodes to achieve load balancing. We fix the symbol size at 8KB. We use the default HDFS block size at 64MB, but for some (n, k), we alter the block size slightly to make it a multiple of the strip size (which is $(n - k)\times 8KB$) for IA codes. Then we manually delete all blocks stored on t DataNodes to mimic t failures, where t = 1, 2, 3. Since we rotate node identities when we stripe data, the lost blocks of the t

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 15, March 2016

failed DataNodes include both data and parity blocks. The RaidNode recovers the failures and uploads reconstructed blocks to new DataNodes (similar as the unsuccessful DataNodes in our evaluation). Here, we deploy the RaidNode in individual of the new DataNodes for the recovery operation. We measure the recovery throughput as the total size of lost blocks divided by the total recovery time.

Map Reduce Jobs: Here that a MapReduce job is composed of four parts: Setup, Map, Reduce, and Cleanup, among which only the task of Map involves, degraded reads. Therefore, mainly improves the Map tasks. It also brings benefits for execution of Reduce tasks as the Map tasks can return the intermediate results faster. Run three MapReduce applications: (i) WordCount, which computes the occurrence frequency of each word in the dataset; (ii) Dedup, which removes duplicate lines in the dataset and outputs all unique lines; and (iii) Grep, which extracts similar strings from text files and counts their occurrences.

3 EXISTING SYSTEM

There have been extensive studies on improving the recovery performance of erasure-coded storage systems. The existing workflow parallelization is speed up reconstruction. Optimal recovery schemes have been existing system proposed for different RAID-6 codes, and achieve around 25 percent of I/O savings compared to simply reconstructing all original data. FastDR, addresses node heterogeneity and exploits I/O parallelism, so as to boost the performance of degraded reads temporarily unavailable data. It incorporates a greedy algorithm that seeks to reduce the data transfer cost of reading surviving data for degraded reads, whereas allowing the search of the efficient degraded read solution to be completed in a well-timed manner. To implement a FastDR prototype, and conduct extensive evaluation through simulation studies on top of testbed experiments on a Hadoop cluster with 10 storage nodes.

4 PROPOSED SYSTEM

Degraded reads have become performance critical operations, due to the fact that temporary errors account for the majority of failures in modern storage systems. To boost the performance of degraded reads in practical erasure-coded storage systems, it is necessary to take into account parallel I/Os and node heterogeneity when performing degraded reads.

.

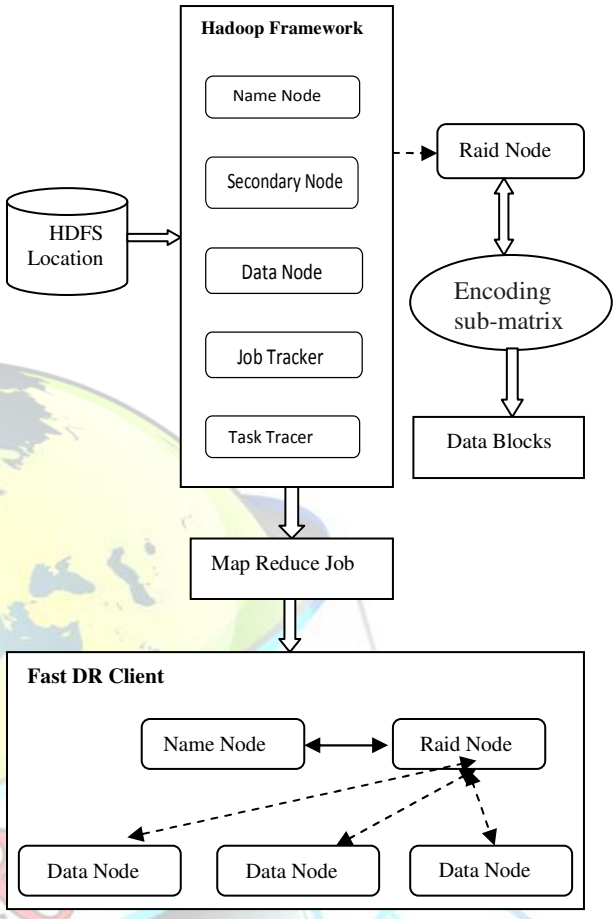

Fig1. Proposed System Architecture Diagram .

The System observation is that while local tasks are running, the MapReduce job does not entirely utilize the available network resources. The proposed degraded-primary scheduling, whose main idea is to schedule some degraded tasks at in advance stages of a MapReduce work and allow them to download data first using the unused network resources. To reduce the redundancy overhead owing to replication, erasure coding can be used. It conduct mathematical analysis and separate event simulation to show the performance gains of degraded-first scheduling

5 SIMULATION AND RESULTS

The simulation studies entail the proposed algorithm is implemented with hadoop. We evaluate the computational overhead and the degraded read performance of three approaches The degraded read operation performs two steps: (i) reading data blocks and parity blocks from the surviving storage nodes, and

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 15, March 2016

(ii) reconstructing the normal blocks and the lost blocks. In our simulation studies, only the running time of the block read part is evaluated. Our justification is that in a distributed environment, the performance bottleneck is due to network transmission instead of computations.

In Fig. 1 is shows the percentage reduction of degraded read time in triple fault tolerant codes. Here CRS is Cauchy Reed-Solomon codes=3. In Fig. 2 represent the percentage reduction of degraded read time in double fault tolerant codes. Table.1 is show the difference between enumeration traverse time and enumeration greedy algorithm traverse time. Using the proposed algorithm to accomplish reduce the response time and data transfer cost.

Our simulation all are conducted under commodity under configurations. a Linux-based desktop computer with Intel(R) i3 @3.2 GHz CPU and 2 GB RAM. The operating system is Ubuntu 12.04.

Algorithm	No of Blocks				
	2	4	6	8	10
FastDR	21.45	18.73	15.70	14.10	14.05
Hybird Recovering any Failure Pattern	19.13	17.21	14.65	12.43	10.87

Table1.Reduction of Degraded Time vs Block

FastDR is used to addressing the node heterogeneity and I/O parallelism. When using the testbed experiment to evaluate the existing system resulting is low compared than the hybrid recovering any failure pattern. Propose a new hybrid recovery approach for single and multi disk failure which can reduce the number of disk reads and therefore improves system recovery performance.

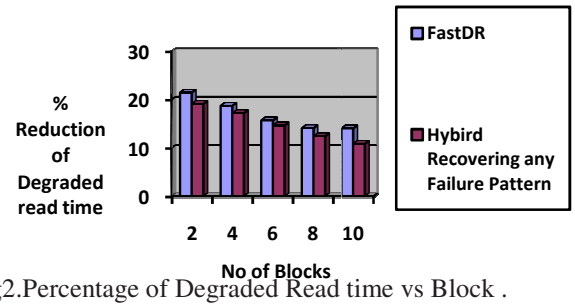

Fig2.Percentage of Degraded Read time vs Block .

Algorithm	Read Size							
	3	5	8	11	14	17	20	23
Basic	19	38	43	50	54	59	61	64
FastDR	20	31	50	56	62	74	81	100
Hybird Recovering any Failure Pattern	26	39	58	63	69	77	88	109

Table2.Reductrion of Degraded Read Time vs Read size

Here, the basic which is represents the ordinary traverse time in distributed storage system. FastDR is addressing the node for heterogeneity using in existing algorithm. Hybrid recovering any failure pattern, reduce the number of disk reads. Table 2 shows the differentiate between Basic, fastDR and Hybrid recovering any failure traverse time.

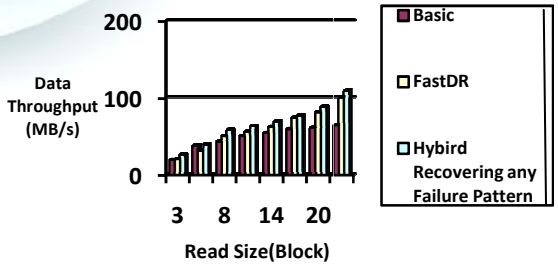

Fig3. Percentage of De-graded Read Time vs Read Size.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 15, March 2016

In Fig3. Expresses the percentage of de-graded read size comparision of basic, fastDR and hybrid recovering any failure pattern. It always shows the throughput and response time significally. Throughput, which is request the process for accessing the data. Block size is calculated as KB,MB,GB and TB. If the system response the request to take time related for size of data. Basically, requesting data is very high the response time will be increased. Our goal is reduce the response time and data transfer cost. Hence, the hybrid recovering any failure pattern to access the data in minimum response time and data transfer cost is very low compare than existing system.

Algorithm	No of Reduce Task		
	1	2	4
Basic	51.24	47.45	45.32
FastDR	27.57	17.39	12.23
Hybird Recovering any Failure Pattern	22.44	15.32	10.54

Table3. Number of Reduced Task vs Time(Word Count).

The number of task reduced always time will be decreased. Table 3, show the time by perform the word count job. It is a job for calculating the performance of degraded reads response time in distributed storage system.

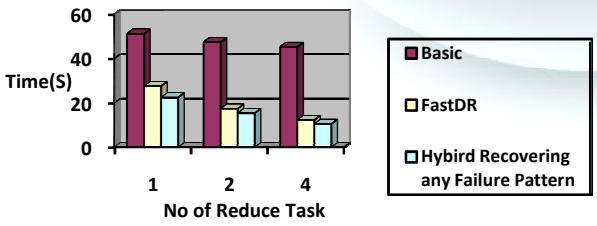

Fig4. Percentage of Reduced Task vs Time .

Basically, the number of task is low automatically to perform the request by using low response time. The proposed system to reduce the disk while searching the appropriate tasks in distributed storage system. The main goal is reduced the data transfer cost and degraded reads failure. Hybrid recovering any failure pattern, accessing the data in low response time and reduce the data transfer cost compare than basic and FastDR technique. The proposes, always to recover the data from the surviving nodes with the help of RAID nodes.

6 CONCLUSIONS

The proposed use of regenerating codes is providing the storage is fault tolerant and minimize the bandwidth of data transfer during recovery. Propose a system which generalizes existing optimal single-failure-based regenerating codes to support the recovery of both single and concurrent failures. Here theoretically show that proposed system minimizes the reconstruction bandwidth in most concurrent failure patterns. Our current implementation of the update manager has a centralized design.

REFERENCES

[1] Yunfeng Zhu, Jian Lin, Patrick P. C. Lee, and Yinlong Xu(2016)," Boosting Degraded Reads in Heterogeneous Erasure Coded Storage System' IEEE Trans. Inf. Theory, Vol. 64, No. 8, pp. 0018–9340.

[2] Blomer J, Kalfane M, Karp R, Karpinski M, M. Luby, and Zuckerman D.(1995), 'An XOR-Based Erasure-Resilient Coding Scheme', International Computer Sciences Institute, Berkeley, California, Technical Report TR-95-048.

[3] Calder B, Wang J, Ogus A, and Nilakantan N (2011), 'Windows Azure storage: A highly available cloud storage service with strong consistency', in Proc. 23rd ACM Symp. Operating Syst. pp. 143–157

[4] Dimakis A, Godfrey P, Wu Y, and K. Ramchandran. (2010), 'Network coding for distributed storage systems', IEEE Trans. Inf. Theory, Vol. 56, No. 9, pp. 4539–4551.

[5] Esmaili K.S, Pamies-Juarez L, and Datta A. (2013), 'The CORE storage primitive: Cross-object redundancy for efficient data repair &

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 15, March 2016

access in erasure coded storage," arXiv preprint arXiv: 1302.5192.

[6] Greenan K, Li X, and Wylie J. (2010), 'Flat XOR-based erasure codes in storage systems: Constructions, efficient recovery, and tradeoffs', in Proc. IEEE 26th Symp. Mass Storage Syst. Technol., pp. 1–14.

[7] Greenan K, Miller E, and Wylie J.(2008), ' Reliability of Flat XOR-based erasure codes on heterogeneous devices', in Proc. IEEE Int. Conf. Dependable Syst. Netw. FTCS DCC, pp. 147–156.

[8] Huang C and Xu L. (2008), 'STAR: An efficient coding scheme for correcting triple storage node failures', IEEE Trans. Comput., Vol. 57, No. 7, pp. 889–901.

[9] Khan O, Burns R, and Plank J S. (2012), 'Rethinking erasure codes for cloud file systems: Minimizing I/O for recovery and degraded reads', in Proc. 10th USENIX Conf. File Storage Technol., pp. 251–264.

[10] Li R, Lin J, and Lee P.P. (2013), 'CORE: Augmenting regenerating coding-based recovery for single and concurrent failures in distributed storage systems', in Proc. IEEE 29th Conf. Mass Storage Syst. Technol., pp. 1–6.

[11] M. Abd-El-Malek, W. Courtright II, C.Cranor, and G. Ganger, 'Ursa Minor: Versatile cluster-based storage,' in Proc. 4th Conf. USENIX Conf. File Storage Technol., Dec.2005, p. 5.

[12] G. Ananthanarayanan, S. Agarwal, and S. Kandula, 'Scarlett: Coping with skewed content popularity in MapReduce clusters', in Proc. ACM 6th Conf. Comput. Syst., 2011, pp. 287–300.

[13] R. Bhagwan, K. Tati, Y. Cheng, S. Savage, and G. Voelker, 'Total Recall: System support for automated availability management', in Proc. 1st Conf. Symp. Netw. Syst. Des. Implementation, 2004, p. 25.

[14] Q. Xin, E. Miller and S. Schwarz, 'Evaluation of distributed recovery in large-scale storage systems', in Proc. 13th IEEE Int. Symp.High Perform. Distrib. Comput., 2004, pp. 172181.

[15] S. Xu, R. Li, P. Lee, Y. Zhu, L. Xiang, Y. Xu, and J. Lui, 'Single disk failure recovery for X-code-based parallel storage systems', IEEE Trans. Comput., vol. 63, no. 4, pp. 995–1007, Apr. 2014.

[16] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, 'Improving MapReduce performance in heterogeneous environments', in Proc. 8th USENIX Conf. Operating Syst. Des. Implementation, 2008, pp. 29–42.

[17] Y. Zhu, P. Lee, Y. Hu, L. Xiang, and Y. Xu, 'On the speedup of single-disk failure recovery in XOR-Coded storage systems: Theory and practice', in Proc. IEEE 28th Symp. Mass Storage Syst. Technol., 2012, pp. 1–12.

[18] Y. Zhu, P. Lee, L. Xiang, Y. Xu, and L. Gao, 'A cost-based heterogeneous recovery scheme for distributed storage systems with RAID-6 codes', in Proc. IEEE 42nd Annu. Int. Conf. Dependable Syst. Netw., 2012, pp. 1–12