IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

Efficient Single/Concurrent Failure Detection for Mapreduce Application in Heterogeneous Node

Akshya Ramesh[1], Akshaya S[1], Beaula K.A[1], M P Geetha[2]

UG Scholar, Department of Information Technology, Sri Krishna College of Engineering and Technology, Coimbatore, Tamil Nadu, India[1]

Assistant Professor, Department of Information Technology, Sri Krishna College of Engineering and Technology, Coimbatore, Tamil Nadu, India[2]

Abstract: Data availability is critical in distributed storage systems especially when node failures are prevalent in real life. A key requirement is to minimize the effort of recovering the lost or unavailable data of failed nodes. To ensure data availability, a storage system often introduces data redundancy via replication or erasure coding. The erasure coding is adopted in large scale storage system which achieves less redundancy overhead than normal replication under the same fault tolerance. Erasure Coded Storage system supports both single and concurrent failure recovery and aims to minimize the band width of recovering the failures. The performance of degraded reads is boosted by addressing both I/O parallelism and node heterogeneity. The temporarily unavailable data is quickly retrieved with the support of Raid Node, which retrieves the unavailable block from the surviving nodes. The performance is evaluated by Word Count Job to compare which computes the occurrences of each word in a document.

Keywords: Erasure-coded storage system, degraded reads, RAID Node.

I. INTRODUCTION

To provide high storage capacity, large scale distributed storage systems are being widely adopted in enterprises that are interconnected over a network environment. In large-scale distributed storage systems ensuring data availability is critical since node failures are frequent and diverse. To ensure data availability, storage systems usually stripe data across multiple storage nodes. Replication is traditionally used to provide data redundancy, but it introduces high storage overhead and becomes a scalability bottleneck. On the other hand, erasure coding provides space-optimal data redundancy. It operates encoding data into multiple fragments, such that subset of fragments can reconstruct the original data [1].

Practical storage systems may experience two types of node failures: permanent and temporary node failure. A node is permanently failed if it loses all its stored data. On the other hand, a node is temporarily failed if its data is not lost but is only temporarily unavailable for direct accesses. This contributes to the majority of component failures in large-scale storage systems. To preserve the required redundancy level and maintain data availability, a storage system performs failure recovery, which reconstructs all the lost data another new node. To access the unavailable data, a storage system performs a degraded read, which is slower than normal reads to retrieves data from surviving nodes and reconstructs the unavailable data [1].

The proposed Erasure Coded Storage system which supports both single and concurrent failure recovery and aims to minimize the band width of recovering the failures. Erasure Coded Storage system is based on Rotated Reed Solomon Code which stripes the data into blocks and XORs the blocks with the encoded matrix. This results in encoded form of data. The original data is recovered by decoding with the sub-matrix. The Erasure Coded Storage system achieves space optimal data redundancy and adds a new recovery scheme atop existing Regenerating Codes. To tolerate large number of failures Rotated Reed-Solomon code is used. It performs degraded reads more efficiently, but otherwise inherits the reliability and performance properties of Reed-Solomon codes.

To boost the performance of degraded reads in large scale heterogeneous erasure coded storage system, Enumerated Greedy algorithm (EG) is proposed to quickly search for an efficient solution for degraded reads. EG algorithm addresses both node heterogeneity and I/O parallelism which significantly reduces the degraded read time compared to the traditional approach.

HDFS achieves fault tolerance via replication, and stores copies for each block. These blocks are grouped into stripes and converted to redundant replicas of Erasure-Coded blocks. HDFS-RAID runs on HDFS, and leverages HDFS for the read/write operations. It adds a Raid Node to support the Name node to boost the performance of degraded reads, i.e., when a client wants to read an unavailable block, it downloads the necessary data and parity blocks from other surviving nodes and reconstructs the unavailable data blocks.

II. RELATED WORK

There have been extensive studies on improving the recovery performance of storage systems. Temporary

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

failures contribute to the majority of component failures in large-scale storage systems. The temporarily unavailable data results in degraded reads which are normally slower than normal reads. The existing system mainly works to speed up reconstruction of unavailable data. FARM (FAst Recovery Mechanism) improves recovery of lost data in large-scale storage systems using parity de-clustering. Mirrored copies of the data are placed across disk to reconstruct the lost data which will result in high storage overhead [8].

K. Greenan, X. Li, and J. Wylie, proposes new non-MDS XOR-based codes to achieve efficient recovery. To maximize the reliability, redundancy placement algorithm of Flat XOR-based erasure codes and simple analytic model was used [4].

Runhui Li, Jian Lin, Patrick P. C. Lee, Cheng Huang, proposes CORE, a fault-tolerant storage and minimum recovery solution based on regenerating code which is mainly designed for single and concurrent failure detection failure [2].

Osama Khan, Randal Burns, James Plank, William Pierce, Presents an algorithm to find the optimal number of codeword symbol needed for recovery of XOR-based erasure code. The use of Reed-Solomon (RS) Code inherits the ability to perform fast degraded reads by the ability to tolerate large number of failures [3].

The basic approach to solve the heterogeneous model for degraded reads is to download the data and parity blocks from exactly k nodes, such that the downloaded blocks can be used to reconstruct the lost blocks correctly. In the case of single failure recovery, it is assumed that the basic approach reads the blocks from other k -1 surviving data nodes.

III. PROPOSED WORK

Whenever the client requests a process to the name node, the job client submits the job to the data node and monitors them. The data node sends heartbeat message to the name node to reassure that the job tracker is still alive. When the data is unavailable to access the name node looks for the next data node to access the data. This results in degraded reed which takes longer time to search the data. To reduce the low latency degraded read the use of raid node is proposed which contains the Meta data and supports the name node to perform fast degraded read. Enumerated Greedy algorithm is proposed for degrade reads to efficiently reconstruct the lost data.

There is temporary failure if its stored data is not lost but is temporarily unavailable for direct accesses. To achieve this fault tolerance the data is replicated and stored using Erasure coding. The data is broken into fragments and encoded with redundant data and stored across a set of different location to reduce storage overhead. The data is encoded using cyclic Reed-Solomon codes code to reduce the storage overhead.

VI. SYSTEM DESIGN

4.1 Architecture

The architecture describes about the components that are used. The architecture consists of the following components: Name Node, Data Node, Raid Node, Erasure Coded Storage System and Client. The Map Reduce framework consists of a single master Job Tracker and single slave Task Tracker per cluster-node. The master is responsible for resource management, tracking resource consumption/availability and scheduling the jobs component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves Task Tracker execute the tasks as directed by the master and provide task-status information to the master periodically. Name Node is a master server that controls the Hadoop cluster. It executes operations of the file system like opening, closing and renaming files and directories. Data Node is a node that stores HDFS blocks and acts as a slave to the Name Node. Data Node is required to send heartbeat and block report to the Name Node in a regular interval.

When the client requests to access the data, the name node sends a command to the Data node to perform read/write operation. When the data is temporarily unavailable it will results in degraded reads. The performance of degraded reads is boosted with the help of Raid Node which will support the Name node to quickly recover the lost data from the surviving nodes.

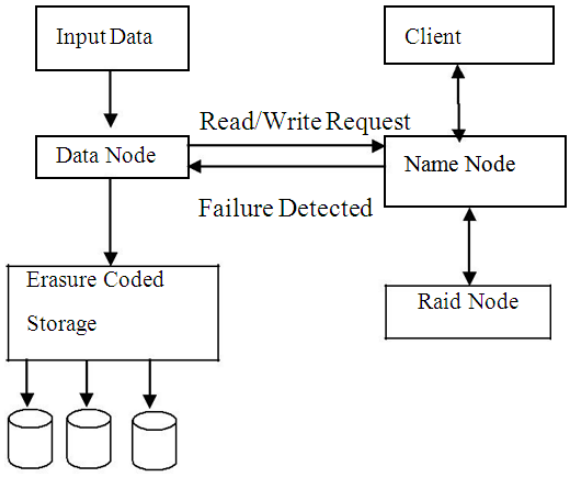

Fig.1 System Architecture

V. IMPLEMENTATION PROCEDURES

The tasks are described in the following four phases

1. Data node Processing.
2. Coded Storage System.
3. Evaluation.
4. Recovery phase

5.1 Data Node Processing:

This phase mainly aims to place the data in the Data Node. A block to be written into HDSF is initially cached in a temporary local file, and then a write request is sent to the

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 4, April 2016

Name Node. The Name Node generates block id, finds and returns a list of Data Nodes to the Client for saving the data. The Client then writes the block to the Data Node. In a regular time interval Name Node keep receiving a Block report and a Heartbeat from each of the Data Nodes in the cluster.

5.2 Coded Storage System:
Erasure coding in Hadoop Distributed File System (HDFS) enhances the performance for degraded reads through disk-oriented reconstruction of lost data to optimize the degraded read and parallelize the read operation. Erasure coded storage systems add redundancy for fault tolerance. The file is divided into blocks and replicated to avoid loss of data.

Specifically, a system of n disks is partitioned into k disks that hold data and m disks that hold coding information. The coding information is calculated from the data using an erasure code technique. For practical storage systems, the erasure code typically has two properties. First, it must be Maximum Distance Separable (MDS), which means that if any m of the n disks fails, their contents may be recomputed from the k surviving disks. Second, it must be systematic, which means that the k data disks hold encoded data. The Rotated Reed-Solomon code encodes the data by converting the one-row code into a multi-row code, and then split across adjacent rows. The Enumerated Greedy algorithm determines which blocks to be downloaded from surviving nodes, so as to efficiently fulfil degraded read requests.

5.3 Evaluation:
The performance is evaluated with the following job:
Word Count, which computes the number of occurrence of each word in the dataset.

Fig 2, shows the execution time for Word Count. It has been observed that the performance of the degraded reads is boosted than the traditional approach.

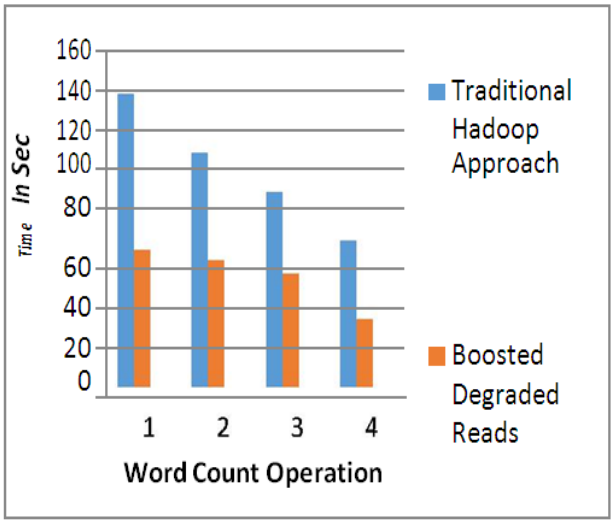

Fig. 2 Execution time of traditional hadoop approach and boosted degraded performance for Word count operation.

5.4 Recovery Phase
When the Client wants to read a file, it sends a read request to the Name Node. The Name Node locates the file and finds the Data Nodes having these blocks. The Client communicates with the nearest Data Nodes to retrieve data. If it cannot retrieve the data from the nearest Data Node due to temporary unavailability, the Raid node recovers failure and schedule the Name Node to retrieve the data from the next nearest Data Node. Thus the Raid Node supports the Name Node to recover the unavailable data.

5.5 Erasure Coding
Erasure coding in HDFS has reduced the storage overhead by approximately 50% compared to the traditional methods. Normally in HDFS it replicates its block three times to ensure data availability in case of node failure. This incurs a 200%overhead in storage space. The use of Erasure Coded system in place of replication uses less storage space while still provides the same level of fault tolerance. Given an object, Erasure Coded System first divides it into d fragments, then produces another e fragments based on a generator matrix, and these e + d fragments are defined as an Erasure Coding set. With this technology the system can tolerate the loss of data and reconstruct the original data out of d + e fragments. Thus Erasure Coding has advantage in both recovering a failure and saving storage space [11].

The RAID node in a HDFS cluster is responsible for parity file generation and block fixing. It generates a corresponding parity file for each source file. The RAID node also periodically asks HDFS for corrupted file. For each corrupted data, it will file its corresponding parity bit and fix the missing block by decoding the source block and its parity block. Thus the RAID node will avoid the chance of data unavailability [12].

VI. CONCLUSION

Erasure Coded Storage System reduces the storage overhead of data by converting it into coded format. HDFS file systems distribute the coded blocks on a different set of storage nodes. This strategy provides load balance and incremental scalability in the data centre. It also prevents correlated failures from resulting in data loss and mitigates the effect that any single failure has on a data set or application.

It recovers each stripe independently from a different set of disks. Also 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 Erasure-Coded Storage Systems, parallel I/Os and node heterogeneity is performed. Implementing the changes has outperformed the naive approach of data read by Fast Degrade Reads. The performances of degraded reads are boosted for the temporarily unavailable data with the support of Raid Node.

REFERENCES

[1] Yunfeng Zhu, Jian Lin, Patrick P. C. Lee, and YinlongXu "Boosting Degraded Reads in Heterogeneous Erasure – Coded Storage Systems", IEEE Transaction on Computer Vol.64, August 2015.

[2] Runhui Li, Jian Lin, Patrick P. C. Lee, Cheng Huang "CORE: Augmenting Regenerating Coding-Based Recovery For Single And Concurrent Failures In Distributed Storage Systems", International Journal of Computer Science and Information Technologies,Vol.5(3) , 4526-4530 , June 2014.

[3] O. Khan, R. 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 Proc. 10th USENIX Conf. File Storage Technol., 2012, pp. 251–264.

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

[5] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems", IEEE Trans. Inf. Theory, Vol. 56, No. 9, pp. 4539–4551, June 2010.

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

[7] Qi Xin ,Ethan L Miller ,"Evaluation of Distributed Recovery in Large-Scale Storage System", University of California, July 2008.

[8] 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. 172– 181.

[9] http://www.michaelnoll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster.

[10] http://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf.

[11] http://www.quora.com/any-real-life-usecase-of-apache- hadoop.

[12] http://www.tutorialspoint.com/hadoop/hadoop_mapreduce.ht ml.

[13] http://hortonworks.com/hadoop-tutorial/hello-world-an-introduction-to-hadoop- catalog-hive-and-pig/.