Data and Parity block Placement Policy to enhance storage efficiency and utilization

Dayeon Kim[1], and Dongryul Shin[2]
[12]Department of Electrical and Computer Engineering, SungKyunKwan University, Su-won, Korea

Abstract - *HDFS which is known as Hadoop Storage manages fault-tolerance by data block replication. Replicas of data blocks are transmitted to other datanodes in a rack awareness manner. However prior HDFS storage management method wastes storage spaces and makes the situation of unbalanced cluster state. Proposed solution in this paper can save the storage space and maintain the balanced state of the entire cluster.*

Keywords: HDFS, Parity block, Balancer

1 Introduction

HDFS(Hadoop Distributed File System) is a typical storage of the Hadoop. Hadoop Storage is a block structured file system. It splits each file into the block of a fixed size and saves each block in the distributed storage node called "data node." Hadoop picks out the datanode to save the block at random. HDFS maintains the fault-tolerance in a way that makes a number of replica and saves them into the separate datanodes[1]. For example, in case of a file that consists of 6 data blocks and the replica parameter that is set to 3, additional storage spaces as much as 12 block sizes are needed. In other words, maintaining fault-tolerance based on replication makes storage waste. It is particularly inefficient when approachless file is saved in the storage.

To save the storage space, It is possible to replace the block replicas with parity block[2]. It is possible to save storage space using parity block. But additional policy is needed to maintain fault-tolerance characteristic of prior block replica based method and balanced state of the entire cluster. Data block and Parity block placement policy is proposed in this paper to save the HDFS storage space efficiently and maintain balanced state of the cluster.

2 Related Work

2.1 HDFS

Data file is divided into fixed size(default 64MB) of data blocks. HDFS maintains fault-tolerance by copying these data blocks. Data block and block replicas are allocated at the runtime instead of prior rule of block allocation. Hadoop selects the data nodes to save each data blocks randomly. It makes duplications of data block according to data block replica parameter. HDFS block placement uses rack awareness for fault-tolerance by placing one block replica on a different rack. To maintain balanced state of the HDFS cluster, It is essential to reassign the data blocks at the run time.

2.2 Replacement of block replica with parity block.

To maintain the fault-tolerance using parity block, stripe configuration has to be preceded. A stripe is set using the data blocks which is located in same datanode. This configuration has the advantage and disadvantage. The advantage is that low network costs are needed when encoding the data blocks to make parity block. Because parity blocks are encoded using data blocks which are located in same datanode, there is a problem that all parity blocks have to be renewed when updating or deleting the file. Hadoop is designed to process the Write-only-Read-many jobs, so stripe is configured using data blocks in the same data node. Figure 1 shows the stripe configuration of each file A, file B, and file C.

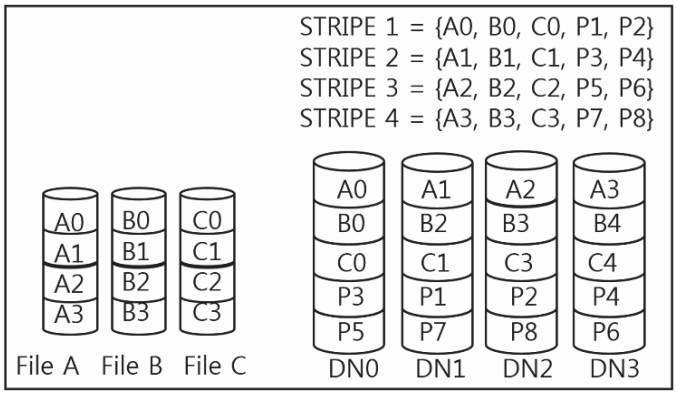

Figure 1 Stripe configuration

Stripe is set of data blocks and parity blocks. Parity block is encoded by data blcoks in same data nodes. It is possible to restore the broken data blocks as many as the number of parity blocks that consists same stripe.

3 Proposed solution

When finishing the allocation of data blocks to the HDFS, parity block is encoded using the data blocks in the same node. After encoding, namenode transmit "Block Move" instruction to datanode to distribute the blocks. To restore the data blocks when node failure occurs, there are 3 block placement policies[3]. Firstly, don't allocate the original data blocks that composes same stripe in the same data node. Secondly, don't allocate the original data block and parity block that composes same stripe in the same data node. Lastly, don't allocate the parity blocks that composes same stripe in the same data node.

When namenode gives "Move Block" command, it evaluates the utilization of each data nodes as proposed in [4] to overcome the uneven block distribution scenario. Proposed solution is represented in a mathematical standpoint as follows :

Assumptions

- U_i and T_i are used space and total capacity of a datanode respectively.
- β_i and β_c are datanode and cluster utilization respectively.
- U_c and T_c are used space and total capacity of cluster.
- Δ is Threshold value.

Datanode and cluster utilization can respectively be expressed as follows. If Data node is not the state of overutilization, it is added to the target nodes. Distinction of overutilization is decided using formula (3)

$$\beta_i = \frac{U_i}{T_i} \quad (1)$$

$$\beta_c = \frac{U_c}{T_c} \quad (2)$$

$$\beta_i = \beta_c + \Delta \quad (3)$$

Figure 2 depicts the flow of proposed solution. When namenode gets the block encode command, namenode collects the block list from each data nodes and construct the stripe. Then it transmits the encode command with stripe information to the datanodes. Datanode encodes each stripe and make a parity block. When Encoding process is done, namenode evaluates the utilization of each datanodes and select the target nodes to move the blocks. And it order "Move block" command according to block place policies.

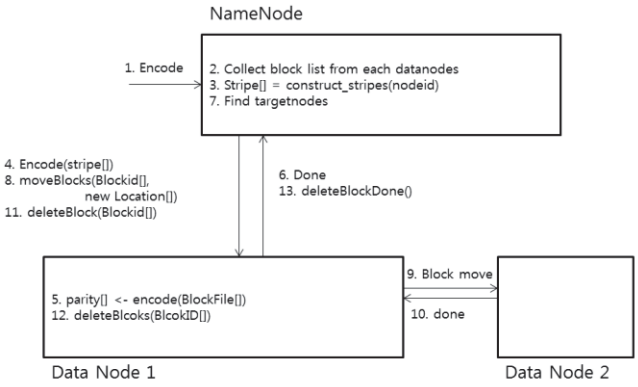

Figure 2 flow of encoding and block reallocation

f

4 Conclusion

When Parity block is used to maintain fault-tolerance, Only (P/G)*100% of Additional space to save the parity block is needed (G = the number of data blocks that composes the stripe, P = the number of created parity blocks). In case of HDFS, it needs 200% of additional space to save the duplications of original data block. So it saves the storage space efficiently. Also, proposed solution helps to keep the cluster in a balanced state when an HDFS client is trying to write data.

References

[1] Shvachko, K. et al. "The Hadoop Distributed File System," IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp.1-10, 2010

[2] Park, Chan-Ik. "Efficient placement of parity and data to tolerate two disk failures in disk array systems." *Parallel and Distributed Systems, IEEE Transactions on* 6.11 (1995): 1177-1184.

[3] 안후영, 이경하, 이수호, 이윤준, 이상민, 김영균. (2013). [분산 데이타베이스] Hadoop 분산 파일 시스템의 효과적인 저장 공간 절약 기법. 정보과학회논문지 : 컴퓨팅의 실제 및 레터, 19(3), 144-148.

[4] Nchimbi Edward Pius, Liu Qin, Fion Yang, Zhu Hong Ming. "Optimizing Hadoop Block Placement Policy & Cluster Blocks Distribution." International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:6, No:10, 2012, 1282-1288