Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 50 (2015) 590 – 595

Procedia Computer Science

2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)

A NOVEL APPROACH FOR REPLICA SYNCHRONIZATION IN HADOOP DISTRIBUTED FILE SYSTEMS

Miss.J.Vini[a], Rachel Nallathamby[b], C.R.Rene Robin[c]

*P.G scholar,*Research scholar,*Professor and Head of the department
Department of Computerscience, Jerusalem college of engineering, Chennai

Abstract

The Map Reduce framework provides a scalable model for large scale data intensive computing and fault tolerance. In this paper, we propose an algorithm to improve the I/O performance of the Hadoop distributed file system. The results prove that the proposed algorithm show better I/O performance with comparatively less synchronization

Peer-review under responsibility of scientific committee of 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)

Keywords: Big data, distributed file system, Map Reduce,Hadoop, Adaptive replica synchronization

Peer-review under responsibility of scientific committee of 2nd International Symposium on Big Data and Cloud Computing (ISBCC'15)
doi:10.1016/j.procs.2015.04.090

1. Introduction

To improve the performance and system scalability in the distributed file system we are using replica synchronization [1]. A replica is nothing but a copy of the original data in the file system. Replica synchronization needs to make sure that the changes in the data sent to one replica are received by other relevant replicas. In Replica synchronization, updates are made to all relevant replicas as many times as the number of write requests, so bottleneck increases.

In HDFS, we use the Meta Data Server (MDS) to manage the whole system and replica synchronization is triggered when any one of the replicas has been updated [2] and the Storage Server (SS) handles the data management.

We use MapReduce to map the input set and get the output set. The input and the output of the MapReduce job are stored in Hadoop Distributed File System (HDFS).

2. Related Work

General parallel file system (GPFS) [4] allocates the space for the multiple copies of data on the different storage server which supports the chunk replication and it writes the updates to all the location. GPFS keeps track of the file which been updated to the chunk replica to the primary storage server. Ceph [5] is the free storage platform has similar replica synchronization technique where the newly written data should be sent to all the replicas which are stored in different storage servers before responding to the client. In Hadoop File System [6] the large data are spitted into different chunks and it is replicated and stored on storage servers. In Google File System (GFS) [7], there are various chunk servers were the MDS manages the location and data layout. For the purpose of the reliability in the file system the chunk are replicated on multiple chunk servers; replica synchronization can be done in MDS. The Lustre file system [8], which is known for parallel file system, which has replication mechanism Mosa Store [9], uses dynamic replication for the data reliability. Here when one new data block is created, the block at one of the SSs is stored in the MosaStore client, and the MDS replicates the new block to the other SSs to avoid the bottleneck when the new data block is created. Replica synchronization is done in the MDS of MosaStore.

The Gfarm file system [10] the replication mechanism is used for data replication for the reliability and availability. In the distributed and parallel file system, the MDS controls the data replication and send the data to the storage servers; this makes pressure to the MDS. Data replication which has the benefits to support for better data access was the data is required and provide data consistency. In the parallel file system [11], this improves the I/O throughput, data duration and availability by data replication. The proposed mechanism, according to the cost of analysis the data pattern are analysed a data replication is done, but replication synchronization is done in the MDS.

In the PARTE file system, the metadata file parts can be replicated to the storage servers to improve the availability of metadata for high service [12]. In the PARTE file system, the metadata file parts can be distributed and replicated to the corresponding metadata into chunks on the storage servers, the file system in the client which keeps the some request of the metadata which have been sent to the server. If the active MDS crashed for any reason, then these client backup requests are used to do the work by the standby MDS to restore the metadata which was lost during the crash.

3. Proposed System Overview

Adaptive replica synchronization is used to improve the I/O throughput, communication bandwidth and performance of distributed file systems. The MDS manages the information in the distributed file system which splits the large data into chunks.

The main aim of using the adaptive replica synchronization is the storage server cannot withstand the large amount of the concurrent read requests to specific replica. The adaptive replica synchronization will be performed to satisfy heavy concurrent reads when the access frequency to the target replica is greater than the predefined

threshold. The adaptive replica synchronization mechanism among SSs intends to enhance the I/O subsystems performance.

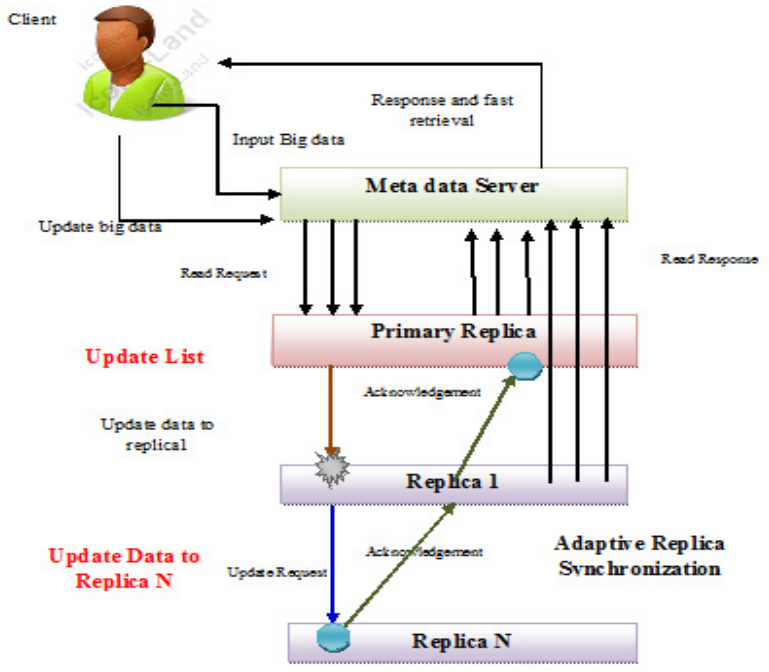

Fig 1: Architecture of replica synchronization mechanism

3.1. Big data Preparation and Distributed data Storage

Configure the storage server in distributed storage environment. The Hadoop distributed file system consists of big data, Meta Data Servers (MDS), number of replica, Storage Server (SS). Configure the file system based on the above mentioned things with proper communication. Prepare the social network big data. It consists of respected user id, name, status, updates of the user. After the data set preparation, it should be stored in a distributed storage server.

3.2 Data update in distributed storage

The user communicates with distributed storage server. After that, user accesses the big data using storage server (SS). Based on user query, update the big data in distributed storage database. By updating the data we can store that in the storage server.

3.2 Chunk list replication to storage servers

The chunk list consists of all the information about the replicas which belongs to the same chunk file and stored in the SSs. The primary storage server which has the chunk replica that is newly updated to conduct the adaptive replica synchronization , when there is a large amount of the read request which concurrently passes in a

short while with minimum overhead to satisfy this that mechanism is used.

3.4 Adaptive replica synchronization

The replica synchronization will not perform synchronization when one of the replicas is modified at the same time. The proposed mechanism Adaptive replica synchronization which improve the I/O subsystem performance by reducing the write latency and the effectiveness of replica synchronization is improved because in the near future the target chunk might be written again, we can say that the other replicas are necessary to update until the adaptive replica synchronization has been triggered by primary storage server.

In the distributed file system the adaptive replica synchronization is used to increase the performance and reduce the communication bandwidth during the large amount of concurrent read request. The main work of the adaptive synchronization is as follows: The chunk is saved in the storage servers are initiated first. In second step the write request is send one of the replicas after that the version and count are updated. Those SS update corresponding flag in the chunk list and reply an ACK to the SS. On the next step read/write request send to other overdue replicas .On other hand it should handle all the requests to the target chunk and the every count is incremented according to the read operation and frequency is computed. In addition, the remaining replica synchronization for updated chunks, which are not the hot spot objects after data modification, will be conducted while the SSs are not as busy as in working hours. As a result, a better I/O bandwidth can be obtained with minimum synchronization overhead. The proposed algorithm is shown in algorithm.

ALGORITHM: Adaptive replica synchronization

Precondition and Initialization:

1) MDS handles replica management without synchronization, such as creating a new replica;
2) Initialize *[Replica Location]* [*flag*], [*count*], and [*version*] in Chunk List when the relevant chunk replicas have been created

Iteration:

1: **while** active storage server
2: **if** request send to chunk **then**
3: **if** [*flag*] == 1 **then**
4: Return the *Replica Status*;
5: break;
6: **end if**
7: **if** request for write is received **then**
8: *I/O request ID* ← *version* ;
9: show *Update Chunk List Request*;
10: do write operation;
11: **ACK** received
12: read count initiated
13: [*count*] ← 1;
14: **else**
15: Undo the write operation;
16: own *Chunk List recovered*;
17: **end if**
18: break;
19: **end if**
20: **if** Read request received **then**
21: do read operation;

22: **if** [*count*] > 0 **then**
23: [*count*] ← [*count*] + 1;
24: [*Frequency*] is computed
25: **if** [*Frequency*] >= *Config Threshold* **then**
26: adaptive replica synchronization is done;
27: **end if**
28: **end if**
29: **end if**
30: **else**
31: **if** *updated list of chunk Request* received **then**
32: Update chunk List and ACK
33: [*flag*] ← 1;
34: break;
35: **end if**
37: **if** *Syn Request* received **then**
38: replica synchronization done;
39: **end if**
40: **end if**

4. Performance Results

The replica in the target chunk has been modified by the primary SSs which will retransmits the updated data to the other relevant replicas, the required time need for the write latency for the each write is done by proposing new mechanism adaptive replica synchronization the write latency is measured by writing the data size given in fig. 2

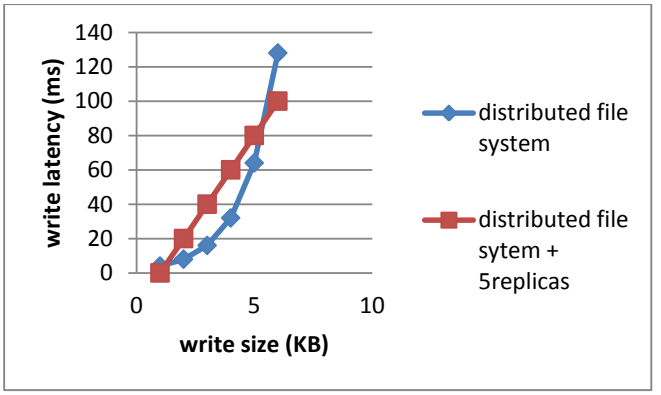

Fig:2 Write latency

By the adaptive replica synchronization we can get the throughput of the read and write bandwidth in the file system. We will perform both I/O data rate and the time processing operation of the metadata.

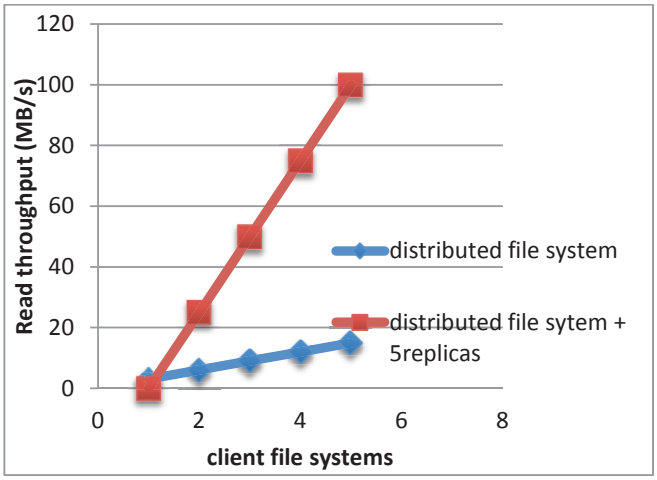

Fig.3.I/ O data throughput

5. Conclusion

In this paper we have presented an efficient algorithm to process the large amount of the concurrent request in the distributed file system to increase the performance and reduce the I/O communication bandwidth. Our approach that is adaptive replica synchronization is applicable in distributed file system that achieves the performance enhancement and improves the I/O data bandwidth with less synchronization overhead. Furthermore the main contribution is to improve the feasibility, efficiency and applicability compared to other synchronization algorithm. In future, we can extend the analysis by enhancing the robustness of the chunk list

REERENCES

[1] Jianwei Liao, Li Li, *Member, IEEE*, Huaidong Chen, and Xiaoyan Liu "Adaptive Replica Synchronization for Distributed File Systems" University of China, Chongqing 400715, China.
[2] N. Nieuwejaar and D. Kotz, "The galley parallel file system," *Parallel Comput.*, vol. 23, no. 4/5, pp. 447–476, Jun. 1997.
[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, *"The Hadoop distributed file system*," in Proc. 26th IEEE Symp. MSST, 2010, pp. 1–10,
[4] M. P. I. Forum, "Mpi: A message-passing interface standard," 1994.
[5] F. Schmuck and R. Haskin, "*GPFS: A shared-disk file system for large computing clusters,"* in Proc. Conf. FAST, 2002, pp. 231–244, USENIX Association.
[6] S. Weil, S. Brandt, E. Miller, D. Long, and C. Maltzahn, "Ceph: A scalable,high-performance distributed file system," in Proc. 7th Symp. OSDI, 2006, pp. 307–320, USENIX Association.
[7] W. Tantisiriroj, S. Patil, G. Gibson, S. Son, and S. J. Lang, "On the duality of data-intensive file system design: Reconciling HDFS and PVFS," in Proc. SC, 2011, p. 67.
[8] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in Proc. 19th ACM SOSP, 2003, pp. 29–43.
[9] The Lustre file system. [Online]. Available: http://www.lustre.org
[10] E. Vairavanathan, S. AlKiswany, L. Costa, Z. Zhang, D. S. Katz, M. Wilde, and M. Ripeanu, "A workflow-aware storage system: An opportunity study," in Proc. Int. Symp. CCGrid, Ottawa, ON, Canada, 2012, pp. 326–334.
[11] GfarmFileSystem.[Online].Available:http://datafarm.apgrid.org/
[12] A. Gharaibeh and M. Ripeanu, "Exploring data reliability tradeoffs in replicated storage systems," in Proc. HPDC, 2009, pp. 217–226.
[13] J. Liao and Y. Ishikawa, "Partial replication of metadata to achieve high metadata availability in parallel file systems," in Proc. 41st ICPP, 2012, pp. 168–1.