International Journal of Engineering & Technology, 7 (2.32) (2018) 167-169

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

An Efficient Data Replication Scheme for Hadoop Distributed File System

T. Lakshmi Siva Rama Krishna[1], J. Priyanka[2], N. Nikhil Teja[3], Sd. Mahiya Sultana[4], B. Jabber[5]

[1,2]Dept. Of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India-522502.
**Corresponding author E-mail: sivamca.mtech@kluniversity.in*

Abstract

A Distributed file system (DFS) is a storage component of a distributed system (DS). DS consists of multiple autonomous nodes connected via a communication network to solve large problems and to achieve more computing power. One of the design requirement of any DS is to provide replicas. In this paper, we propose a new replication algorithm which is more reliable than the existing replication algorithm used in DFS. The advantages of our proposed replication algorithm by incrementing nodes sequentially (RAINS) is that it distributes the storage load equally among all the nodes sequentially and it guarantees a replica copy in case two racks in a DS are down. This feature is not available in the existing DFS. We have compared existing replication algorithm used by Hadoop distributed file system (HDFS) with our proposed RAINS algorithm. The experimental results indicate that our proposed RAINS algorithm performs better when more number of racks failed in the DS.

Keywords: *Hadoop Data locality, Data Replication, Data Placement*

1. Introduction

Replication is one of the important design requirements of any distributed system (DS). Many DFSs are exist in the literature. Hadoop distributed file system (HDFS) is one of the popular open source implementations of google file system (GFS) [1]. HDFS is developed by apache foundation that provides advantages of the power of high-speed computing clusters and storage and high performance in big data storage [1-5]. HDFS provides high availability, fault-tolerance, and replication. Hadoop maintains default replication factor as three. Many IT industries are using HDFS as it supports to store, process and analyze big data applications. Data locality is an important feature to improve the performance of HDFS [1,6]. In this paper, we propose a new replication algorithm which guarantees the data even two racks fail. In the literature [1-5] There are three types of data proximities are considered (i) data local (ii) rack local and (iii) different rack.

Data local means when the data is located on the same node where the mapper program is executing or being executed. Here, the proximity of the data is close to the computation logic. Since data local is always not possible while executing the mapper programs in a Hadoop cluster. Hence we choose the different node on the same rack to compute is known as rack local. In different rack, some situations data local and rack local is not possible to choose for executing the mapper logic, in such cases we go for other alternative procedure such as choosing the node in the different rack to execute the mapper program.

In our work, we have proposed a new replication algorithm by considering different rack possibility which is a frequent case in a busy Hadoop cluster. Rest of the paper is organized as follows, section 2, we discus related work. Existing replication strategy is described in section 3. In section 4, we discuss our proposed replication algorithm. Section 5 we discuss our experimental results. Section 6, concludes the paper.

2. Related Work

There are several previous research works on data replication have been proposed in HDFS. They are mainly focused on fault tolerance to overcome unexpected failure. Recently, some of the researchers in [10-12] are focused on improving data locality for efficient execution on data replication in Hadoop and some of the researchers in [13-14] are proposed techniques to improve data locality. In paper [17] authors compared the performance of read and write operation of HDFS. Table 1 shows features of existing data replication schemes.

Table 1. Data Replication Scheme and Scheduling

	Replication Scheme	**Technique**	**Weakness**
1	dynamic data replication	access patterns	remove replicated data
2	data placement	Balancing for requirement	depend on application
3	data prefetching	prediction by log	depend on log data

In contrast to the above existing schemes, we have proposed a new replication algorithm replication algorithm by incrementing nodes sequentially (RAINS). We have considered two different scenarios for the RAINS algorithm. In the first scenario, we have not considered the nearest data node to the client for replication. In the first scenario the data nodes are placed in racks which are generally far away from clients which involve communication overhead for writing data in the data nodes thereby performance degrades. In order to overcome the above issue, we have proposed another scenario called choose nearest racks to the clients.

3. Existing Replication Strategy

In this section, we discuss the existing replication strategy used by Hadoop DFS. The components in Hadoop DFS are name node which is maintains metadata and several data nodes which actually stores the data and multiple client nodes.

1. Hdfs client Write requests to the Name Node
2. Name Node has to perform various checks like (file existence, and permissions) .
3. Name node sends response to hdfs client list of data nodes available.
4. Hdfs client writes the data to the available data nodes
5. Once data written into the data nodes send the acknowledgement to the name node

The default replication factor in the existing algorithm is three. And they assumed multiple racks are available to store and maintain replicas. Here we assume each rack consists of 20 to 40 data nodes. We also assume that writing client is also part of the Hadoop cluster. Hence, first replica placement is the node which is writing the data. For placing the second replica we choose a data node which is in the different rack than first replica and the third replica is in the same rack which is used to place the second replica but in a different node.

We have explained the existing replica algorithm with the help of below

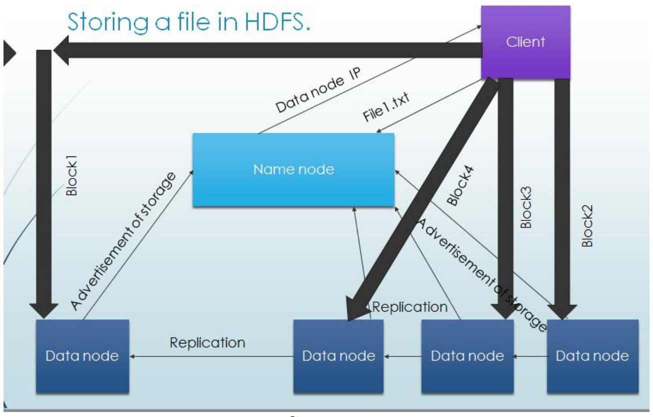


figure 1.

4. Proposed Efficient Data Replication Scheme

In this section, we have proposed a new replication algorithm by incrementing nodes sequentially (RAINS). We have considered two different scenarios for the RAINS algorithm. In the first scenario, we have not considered the nearest data node to the client for replication. Data nodes are placed in racks which are generally far away from clients which involve communication overhead for writing data in the data nodes thereby performance degrades. In order to overcome the above issue, we have proposed another scenario called choose nearest racks to the clients.

In the second scenario, we took the nearest racks to the client that includes we have taken only the racks which have minimum cost from the client. The disadvantage of the first scenario is cost of communication is more from client to the rack and performance degrades. The advantage for the second technique is reduces the cost from client to rack and also the processing time.
If the data nodes are not available in one racks place the data in the another rack and the load balance will be minimum.

So in this way second technique is a better approach for the data writing on the data nodes and also for replication.

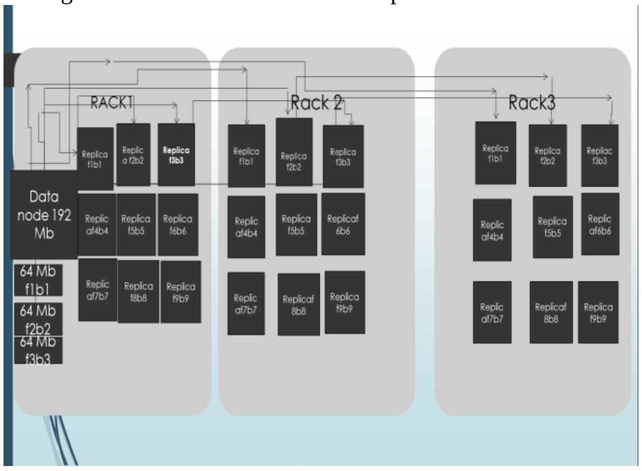


Figure 2.

Proposed RAINS Algorithm:

4.1 Notations and assumptions used in the proposed algorithm Assumptions

1.We assume that always there is sufficient storage space is available to store data in the data nodes.
2. We also assume that the size of each block is 64MB
3. Number of blocks to be replicated is represented with *nbr*
4. File size to be replicated is represented with *fs*
    nbr = *fs*/64MB.

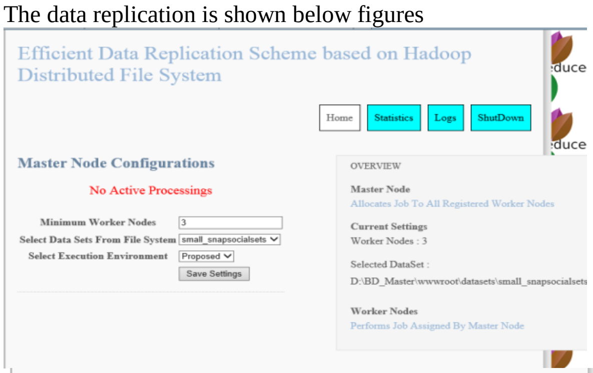
4.2 Replica Placement Algorithm

Input: file 'f'
Output: File is successfully stored in the Data Nodes
with 3 replicas
Step 1. Selection of a rack to store the replica
Step 2. Selection of a node to store the replica
place the first replica on the first rack
/*rack1selected = first replica placed*/
Step 3. Place the second replica on the second rack.
/*rack2selected = second replica
placed*/
Step 4.place the third replica on third rack
/*rack3selected = third replica placed*/
Step 5. if replication factor is more than three(3)circulate the replicas from first rack onwards as this process goes on if it contains more than three replicas.
Step 6. else replication process stops
Note that, the implementation of this algorithm is useful since gigabit network are available nowadays hence we can overcome communication overhead problem if exists.

5. Performance Results

We have implemented this algorithm using JAVA language. Initially we have implemented this algorithm in JAVA in future we plan to implement this algorithm in actual Hadoop cluster with multiple data nodes and single name node. First we have taken data nodes which are also called worker nodes we divide the work among all the three worker nodes. First data is replicated in first worker node and second data replicated in second worker node and third data in third worker node. Experimental results indicate that our proposed RAINS algorithm performs better when more number of racks failed in the DS

The data replication is shown below figures



Efficient Data Replication Scheme based on Hadoop Distributed File System

Master Node Configurations

No Active Processings

Minimum Worker Nodes	3
Select Data Sets From File System	small_snapsocialsets
Select Execution Environment	Proposed

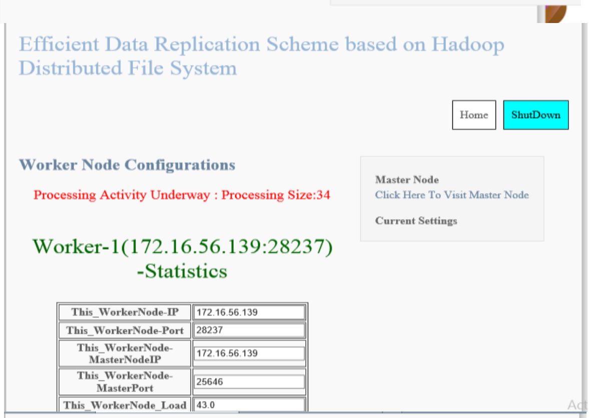
OVERVIEW

Master Node
Allocates Job To All Registered Worker Nodes

Current Settings
Worker Nodes : 3

Selected DataSet :
D:\BD_Master\wwwroot\datasets\small_snapsocialsets

Worker Nodes
Performs Job Assigned By Master Node



Efficient Data Replication Scheme based on Hadoop Distributed File System

Worker Node Configurations

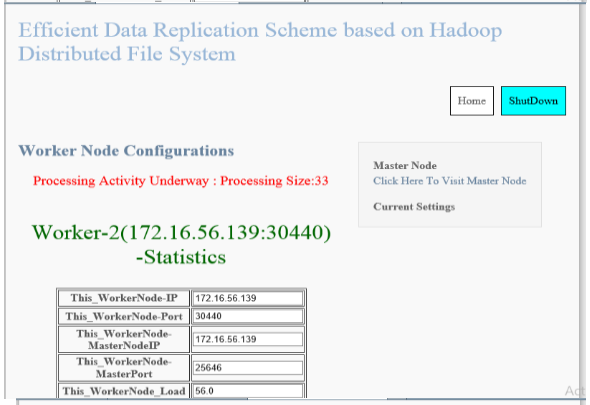
Processing Activity Underway : Processing Size:34

Master Node
Click Here To Visit Master Node

Current Settings

Worker-1(172.16.56.139:28237) -Statistics

This_WorkerNode-IP	172.16.56.139
This_WorkerNode-Port	28237
This_WorkerNode-MasterNodeIP	172.16.56.139
This_WorkerNode-MasterPort	25646
This_WorkerNode_Load	43.0



Efficient Data Replication Scheme based on Hadoop Distributed File System

Worker Node Configurations

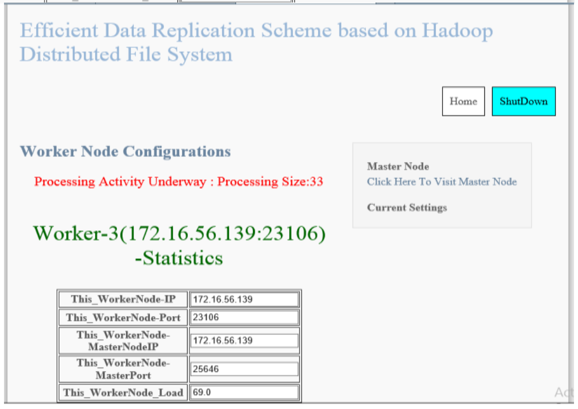
Processing Activity Underway : Processing Size:33

Master Node
Click Here To Visit Master Node

Current Settings

Worker-2(172.16.56.139:30440) -Statistics

This_WorkerNode-IP	172.16.56.139
This_WorkerNode-Port	30440
This_WorkerNode-MasterNodeIP	172.16.56.139
This_WorkerNode-MasterPort	25646
This_WorkerNode_Load	56.0



Efficient Data Replication Scheme based on Hadoop Distributed File System

Worker Node Configurations

Processing Activity Underway : Processing Size:33

Master Node
Click Here To Visit Master Node

Current Settings

Worker-3(172.16.56.139:23106) -Statistics

This_WorkerNode-IP	172.16.56.139
This_WorkerNode-Port	23106
This_WorkerNode-MasterNodeIP	172.16.56.139
This_WorkerNode-MasterPort	25646
This_WorkerNode_Load	69.0

6. Conclusion

In this paper, we proposed a new replication algorithm (RAINS). The advantages of our proposed replication algorithm by incrementing nodes sequentially (RAINS) is that it distributes the storage load equally among all the nodes sequentially and it guarantees a replica copy in case two racks in a DS are down. This feature is not available in the existing DFS. We have compared existing replication algorithm used by Hadoop distributed file system (HDFS) with our proposed RAINS algorithm. The experimental results indicate that our proposed RAINS algorithm performs better when more number of racks failed in the DS.

References

[1] J. Dean and S. Ghemawat, "MapReduce: simplified data pro-cessing on large clusters", Communications of the ACM, vol. 51, no. 1, (2008), pp.107-113.
[2] White, Tom. Hadoop: The definitive guide. O'Reilly Media, Inc., (2012).
[3] D. Borthakur, HDFS architecture guide, HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf, (2008).
[4] Thomasian and J. Menon, RAID5 performance with distrib-uted sparing, Parallel and Distributed Systems, IEEE Transactions on, vol. 8, no. 6, (1997), pp. 640-657.
[5] J. Dean and S. Ghemawat, MapReduce: simplified data pro-cessing on large clusters, Communications of the ACM, vol. 51, no. 1, (2008), pp. 107-113.
[6] K. Shvachko, The hadoop distributed file system, Mass Stor-age Systems and Technologies (MSST), 2010 IEEE 26th Symposi-um on. IEEE, (2010).
[7] S. Mahadev, A survey of distributed file systems, Annual Review of Computer Science, vol. 4, no. 1, (1990), pp. 73-104.
[8] Q. Wei, CDRM: A cost-effective dynamic replication manage-ment scheme for cloud storage cluster, ClustComputing (CLUS-TER), 2010 IEEE International Conference on. IEEE, (2010).
[9] J. Xiong, Improving data availability for a cluster file system through replication, Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on. IEEE, (2008).
[10] Abad, L. Cristina, Y. Lu, and R. H. Campbell, DARE: Adap-tive data replication for efficient cluster scheduling, Cluster Com-puting (CLUSTER), 2011 IEEE International Conference on. Ieee, (2011).
[11] Khanli, L. Mohammad, A. Isazadeh, and T. N. Shishavan, PHFS: A dynamic replication method, to decrease access latency in the mul-ti-tier data grid, Future Generation Computer Systems, vol. 27, no. 3, (2011), pp. 233-244.
[12] S. Seo, HPMR: Prefetching and pre-shuffling in shared MapReduce computation environment, Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on. IEEE, (2009).
[13] X. Zhang, An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environ-ments. Cloud and Service Computing (CSC), 2011 International Conference on. IEEE, (2011).
[14] M. Zaharia, Delay scheduling: a simple technique for achiev-ing locality and fairness in cluster scheduling, Proceedings of the 5th European conference on Computer systems. ACM, (2010).
[15] T. White, Hadoop: The definitive guide, O'Reilly Media, In(2012).
[16] Krishna, Talluri Lakshmi Siva Rama, Thirumalaisamy Ragun-athan, and Sudheer Kumar Battula. Performance evaluation of read and write operations in hadoop distributed file system. Parallel Archi-tectures, Algorithms and Programming (PAAP), 2014 Sixth Inter-national Symposiumon.IEEE,2014.