Storage and Rack Sensitive Replica Placement Algorithm for Distributed Platform with Data as Files

Vinay Venkataramanachary
Systems & Computer Engineering
Carleton University
vinay.venkataramanach@carleton.ca

Enrique Reveron
Systems & Computer Engineering
Carleton University
enrique.reveron@carleton.ca

Wei Shi
School of Information Technology
Carleton University
wei.shi@carleton.ca

***Abstract*—Distributed File System (DFS) is a key component in cloud and data center networking. Frequent hardware failure and network bottlenecks in the underlying infrastructure degrade the performance significantly. Replica technique provides enhanced fault tolerance by storing multiple replicas of a single data block. In the Hadoop platform, the Hadoop Distributed File System (HDFS) handles data storage and provides replica placement services. The Default HDFS replica engine adopts a simple rack aware policy and is designed to improve fault tolerance by storing data blocks in multiple racks. However, the HDFS replica engine does not consider key performance indicators of data center resources such as rack utilization and node storage utilization. Furthermore, in HDFS data is stored as uniformly divided small-sized blocks, which increases traffic flow during the entire file access, therefore degrading the response time. In this research, we propose a Storage and Rack Sensitive (SRS) replica placement algorithm that aims at improving the rack and storage utilization of data center resources. The proposed algorithm also attempts to optimize traffic flow during file access by storing data as original files instead of small uniform blocks. Experimental results of the proposed SRS algorithm are compared against the default HDFS replica distribution and significant improvement on rack-utilization and storage-utilization were observed. Furthermore, latest literature confirms that the "Data as a File" approach indeed decreases the amount of data flow caused by file access traffic.**

***Index Terms*—Hadoop distributed file system (HDFS), Replica Engine (RE), Replica Factor (RF), Data Node, Name Node, Data Locality.**

I. Introduction

Hadoop is an open-source software utility platform which enables distributed computing, Hadoop Distributed File System (HDFS) stores and manages the data objects and offers enhanced fault tolerance via replica engine. HDFS divides a file into multiple chunks known to be as data blocks of equal sizes for replica storage [1]. HDFS uses a simple rack aware policy and random replica placement strategy. For replica factor three, the replication engine places the first data block on one data node of a random rack and the second data block on a remote rack and the third data block on the same remote rack [2]. Table 1 shows replica distribution obtained for three racks environment for replica factor three and thirty five data blocks. [3]. From Table I, it is observable that HDFS replica placement is non-uniform and rack utilization percentage is erratic resulting in under-utilized nodes and due to multiple data blocks for a file, it increases traffic flow when a file is accessed [4]. Challenges associated with default HDFS replica distribution can be summarized as 1) Non-uniform replica data block distribution leading to under-utilization of racks, 2) Un-accounted storage utilization of nodes and racks, 3) High data block movements increase response time during the file access.

In this research, we propose a weighted function based Storage and Rack sensitive replica placement algorithm (SRS) to address the issues in default HDFS replica distribution and three critical performance parameters are introduced for performance evaluation purpose, followed by an experimental evaluation of the proposed algorithm using a simulated data center rack environment workload. Results obtained for replica distribution is analyzed and compared with the default HDFS replica distribution scheme, further analysis of performance parameter shows significant improvements in the proposed algorithm in addition to the benefits of data as a file approach. This paper is organized to present related work in section II, followed by SRS Algorithm in section III, algorithm implementation and results are presented in section IV, research conclusions and future work are discussed in section V.

II. Related work

Kumar P J et. al. in [5] presents a multi attribute QoS aware replica distribution algorithm for HDFS platform, proposed algorithm considers various input attributes such as incoming request metrics, available space, replica factor to generate a balanced replica distribution while satisfying the QoS needs specified by user and aims at minimal or zero QoS violation. However, this algorithm does not focus on storage utilization which is an essential cost component for data center operations.

Ibrahim et al. in [3] propose an intelligent data placement algorithm IDPM, a heuristic rack aware policy which uses the number of free nodes in a rack with least load iterations to distribute the replica blocks achieving uniform distribution with a high data node utilization. Results observed show balanced replica distribution of data blocks across all the nodes of the racks across the cluster. However, IDPM does not consider storage utilization performance parameter and traffic flow issues during file access.

978-1-7281-3187-9/20/$31.00 ©2020 IEEE

TABLE I: Default HDFS Replica placement for three rack environment with Replica Factor =3 [3]

HDFS Replica Distribution (Data as Blocks)	Rack 0				
Rack Node ID / Data Node ID	0	1	2	3	4
1	4	26	20	15	22
2	29	31	4	29	30
3	20	16	1	17	0
4	13	33		13	9
5	17	34			31
6	12	12			26
7	8	1			1
8		8			
9		3			
10		2			
11		7			

HDFS Replica Distribution (Data as Blocks)	Rack 1		
Rack Node ID / Data Node ID	0	1	2
1	10	25	19
2	4	23	24
3	18	6	5
4	9	21	9
5	28	2	28
6	19	16	6
7	8	22	1
8	32	10	27
9	11		3
10			27

HDFS Replica Distribution (Data as Blocks)	Rack 2						
Rack Node ID / Data Node ID	0	1	2	3	4	5	6
1	15	23	20	15	24	21	25
2	14	30	5	30	14	29	26
3	34	18	18		5	16	21
4	13	7	31		6		34
5	23	28	17		25		0
6	0	33	12				27
7	11		11				24
8	32		3				33
9	3		2				2
10	1						
11	10						

Wei Dai et al. in [6] propose a partition replica placement policy where all available nodes are divided into three sections. Section 1 will have about two-thirds of nodes and the first two replicas will be placed in this section and the remaining one-third of the nodes will form the second section where the third replica will be distributed to achieve even distribution of data blocks. High rack utilization is observed. However, since data is considered as blocks, traffic flow issue due to file access still exists.

III. Storage and Rack Sensitive Replica Placement Algorithm

General description: Proposed Storage and Rack Sensitive (SRS) replica placement algorithm is a combination of rack aware policy and storage sensitive policy. The fundamental strategy of the proposed heuristic algorithm is to consider the data as a file itself to reduce traffic flow during file access and since files will be of different sizes, the algorithm employs storage sensitivity to achieve uniform storage utilization. Rack sensitivity is achieved by computing the free data node and free racks in real-time. Storage sensitivity is achieved by computing overall storage for each rack and all the nodes in real-time. A weighted function is introduced to combine rack sensitivity and storage sensitivity for replica placements.

Weighted function: A mathematical model to perform uniform distribution with weighted control inputs based on cluster information and allows influencing a specific input to achieve an even replica distribution in an uneven storage rack environment for uneven file size. In the proposed SRS algorithm, racks of uneven storage capacity, free nodes and node storage capacity are the control inputs, end-user will be able to key in and modify the weighted function according to the local environment setup. Eq 1. describes the weighted function for simulated rack environment workload described in section IV where n is the number of files, f is replica factor, a is free rack storage utilization defined in eq 2, b is free node available in racks defined in eq 3, and c is free node storage utilization defined in eq 4.

$$SRS = \frac{\sum_{n=0}^{nf-1} \mathrm{a} * 2\mathrm{b} * \mathrm{c}}{4} \quad (1)$$

(a) **free rack storage utilization:**

$$a = \left(\frac{(\text{free\_rack\_capacity} - \text{file\_size})}{\text{free\_rack\_capacity}}\right) * 100 \quad (2)$$

(b) **free nodes:**

$$b = \left(\frac{(\text{free\_rack\_nodes} - 1)}{\text{free\_rack\_nodes}}\right) * 100 \quad (3)$$

(c) **free node storage utilization:**

$$c = \left(\frac{(\text{free\_node\_capacity} - \text{file\_size})}{\text{free\_node\_capacity}}\right) * 100 \quad (4)$$

Rack & node selection criteria: A rack is avoided and is not considered for a third replica placement if the first and the second replica was placed in the same rack, a rack is avoided if any of the nodes in a rack do not have enough storage individually to store the file, A rack and data node is selected if it returns maximum value for the weighted function. Replicas are assigned to nodes from left to right with the lowest node number, first, free node available with storage greater than the file size is chosen.

Algorithm: Rack and node Selection function

1) for each rack [0 . . . number_of_racks - 1]
2) calculate free rack and node storage capacity
3) calculate weighted function, end for
4) best_rack is rack with maximum weighted function
5) current_node = returned by node selection criteria
6) update cluster
7) return best_rack and current_node

IV. Algorithm Implementation & Results

Simulation workload data: Test workload considered for evaluation is as follows, 1) Three rack environments: R0, R1, R2 with 5, 3 and 7 internal nodes respectively for each rack, 2) Total rack storage: R0: 5 TB, R1: 1.8 TB, R2: 3.5 TB, 3) Internal node storage: R0: 1.0 TB per node, R1: 0.6 TB per node, R2: 0.5 TB per node, 4) Input files: total files:35, varying file sizes generated by a random generator, 5) Replica factor is 3. Results obtained for the said workload is presented below. Table II represents SRS replica distribution, Table III presents analysis of per rack storage and overall rack utilization of proposed SRS v/s default HDFS replica distribution, Table IV represents per node storage utilization comparison of SRS v/s default HDFS, and Fig 1 presents analysis of per node storage utilization for SRS v/s default HDFS replica distribution.

TABLE II: Storage and Rack sensitive replica distribution results for data as files of different size

SRS Replica distribution (Data as Files of different size)	Rack 0				
Rack Node ID / Data Node ID	0	1	2	3	4
1	0	1	2	3	4
2	4	5	5	6	6
3	7	7	8	9	9
4	10	10	11	11	12
5	12	13	14	15	16
6	16	17	17	18	18
7	19	19	20	21	21
8	22	23	23	24	25
9	25	26	26	27	28
10	28	29	29	30	30
11	31	31	32	33	34

SRS Replica distribution (Data as Files of different size)	Rack 1		
Rack Node ID / Data Node ID	0	1	2
1	1	2	4
2	5	6	7
3	8	8	9
4	10	11	16
5	17	18	19
6	20	20	26
7	27	27	28
8	29	30	

SRS Replica distribution (Data as Files of different size)	Rack 2						
Rack Node ID / Data Node ID	0	1	2	3	4	5	6
1	0	0	1	2	3	3	12
2	13	13	14	14	15	15	21
3	22	22	23	24	24	25	31
4	32	32	33	33	34	34	

TABLE III: Analysis of per rack storage and overall rack utilization of proposed SRS v/s default HDFS replica distribution

Racks	Rack 0	Rack 1	Rack 2
Per Rack Storage Utilization (HDFS)	50%	96%	102%*
Overall Rack Utilization (HDFS)	58%	90%	68%
Per Rack Storage Utilization (SRS)	85%	80%	72%
Overall Rack Utilization (SRS)	100%	96%	96%

* Indicates Over Utilization and requires additional storage

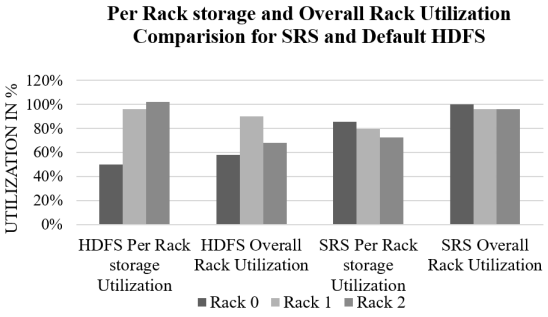

TABLE IV: Per node storage utilization for proposed SRS v/s default HDFS replica distribution

Racks	Rack 0					Rack 1			Rack 2						
Data Nodes	0	1	2	3	4	0	1	2	0	1	2	3	4	5	6
Per node Storage Utilization (HDFS) Max: 160%, Min 12%	46%	100%	12%	34%	57%	112%*	100%	75%	160%*	115%*	131%*	40%	58%	49%	164%*
Per node Storage Utilization (SRS) Max: 100, Min 62%	81%	81%	76%	90%	100%	76%	78%	85%	75%	75%	61%	67%	84%	83%	62%

* Indicates Over Utilization and requires additional storage

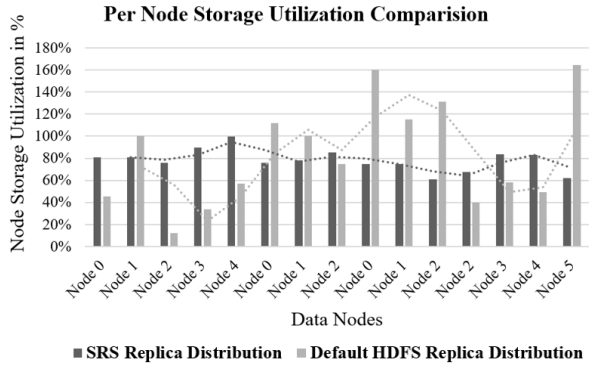

Fig. 1: Analysis of per node storage utilization of proposed SRS v/s default HDFS replica placement algorithm.

Evaluation Parameters: To evaluate the results obtained, three performance parameters are introduced and defined which directly relates to resource costs and issues identified in HDFS. **Per Node storage utilization (NS_u)**

$$NS_u = \left(\frac{\text{Total storage used on a node}}{\text{Storage capacity of the node}}\right) * 100 \quad (5)$$

Per Rack storage utilization (RS_u)

$$RS_u = \left(\frac{\text{Total storage used on a rack}}{\text{Total storage capacity of Rack}}\right) * 100 \quad (6)$$

Overall Rack utilization (OR_u)

$$OR_u = \left(\frac{\text{Total node} - \text{Non-Utilized node}}{\text{Total node cells}}\right) * 100 \quad (7)$$

Result analysis: Table II shows replica distribution results obtained from the proposed SRS replica placement algorithm which shows significant improvement in the reduction of non-utilized data nodes as compared to HDFS replica distribution in Table I. In default HDFS replica placement, rack 0 has four un-utilized rack nodes (0,2,3,4) and least utilized rack node has only three files but can accommodate maximum of eleven files, also rack 0 has eight un-utilized data nodes(4,5,6,7,8,9,10,11) where data nodes 8,9,10,11 are least utilized with just one file but can store maximum of five files, rack 1 and rack 2 has a total of thirty-five un-utilized data nodes. However, results obtained in the proposed SRS replica placement algorithm shows that rack 0 has full utilization and two un-utilized data node in rack 1 and rack 2 respectively, projecting significant improvement in proposed SRS algorithm.

From the results obtained for evaluation parameters and analysis on the same in Table IV and Fig 1, it is evident to notice that the proposed SRS algorithm has near to uniform distribution and uniform storage utilization within the rack and across the data center indicating optimal system resource utilization. Rack storage utilization of greater than 100 percent in default HDFS distribution indicates that for a given workload, HDFS requires additional rack and node capacity to store the replicas as compared to the storage infrastructure required by proposed SRS replica distribution to store the same replicas.

Analysis of Data as a file approach: Since all data is considered as files (not divided into multiple blocks), data locality for a given file will be maximum [7] due to decreased access overheads which primarily address the challenges in default HDFS algorithm. Improved data locality factor accounts to decrease of traffic flow across cluster nodes and also increases the data access response time by avoiding index overhead delays at name node. Ciritoglu et. al. in [8] and [9] proposes a heterogeneity-aware replica deletion scheme which attempts to optimize resources by resizing replica factor. Data as a file approach in such a setting would be effective in computing data popularity related metrics.

V. Conclusion & Future Work

In this research, we proposed a Storage and Rack sensitive smart replica placement algorithm which addresses three main challenges of the default HDFS replica placement techniques, 1) Non-uniform replica data block distribution, 2) Non-uniform utilization of storage resources and 3) Increased file access traffic. Three performance evaluation parameters were introduced to quantify and compare the results with default HDFS replica placement algorithm and analysis shows significant improvement in utilization of per node storage, per rack storage and overall rack. Analysis and optimization of compute overheads for SRS are considered for future work.

Acknowledgement

The authors gratefully acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant No. RGPIN-2015-05390.

References

[1] D. Borthakur *et al.*, "Hdfs architecture guide," *Hadoop Apache Project*, vol. 53, no. 1-13, p. 2, 2008.

[2] A. K. Karun and K. Chitharanjan, "A review on hadoop—hdfs infrastructure extensions," in *2013 IEEE conference on information & communication technologies*. IEEE, 2013, pp. 132–137.

[3] I. A. Ibrahim, W. Dai, and M. Bassiouni, "Intelligent data placement mechanism for replicas distribution in cloud storage systems," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2016, pp. 134–139.

[4] R. K. Grace and R. Manimegalai, "Dynamic replica placement and selection strategies in data grids—a comprehensive survey," *Journal of Parallel and Distributed Computing*, vol. 74, no. 2, pp. 2099–2108, 2014.

[5] P. Kumar and P. Ilango, "Bmaqr: balanced multi attribute qos aware replication in hdfs," *International Journal of Internet Technology and Secured Transactions*, vol. 8, no. 2, pp. 195–208, 2018.

[6] W. Dai, I. Ibrahim, and M. Bassiouni, "A new replica placement policy for hadoop distributed file system," in *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE, 2016, pp. 262–267.

[7] N. Mansouri and M. M. Javidi, "A hybrid data replication strategy with fuzzy-based deletion for heterogeneous cloud data centers," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 5349–5372, 2018.

[8] H. E. Ciritoglu, J. Murphy, and C. Thorpe, "Hard: a heterogeneity-aware replica deletion for hdfs," *Journal of big data*, vol. 6, no. 1, p. 94, 2019.

[9] H. E. Ciritoglu, T. Saber, T. S. Buda, J. Murphy, and C. Thorpe, "Towards a better replica management for hadoop distributed file system," in *2018 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2018, pp. 104–111.