A DATA RE-REPLICATION SCHEME AND ITS IMPROVEMENT TOWARD PROACTIVE APPROACH*

Thanda Shwe [1, 2] and Masayoshi Aritsugi [1]

[1]Department of Computer Science and Electrical Engineering, Graduate School of Science and Technology, Kumamoto University, Kumamoto, Japan

[2]Mandalay Technological University, Mandalay, Myanmar
e-mail: thandashwe@gmail.com, aritsugi@cs.kumamoto-u.ac.jp

Received Date: January 25, 2017

Abstract

With increasing demand for cloud computing technology, cloud infrastructures are utilized to their maximum limits. There is a high possibility that commodity servers that are used in Hadoop Distributed File System (HDFS) based cloud data center will fail often. However, the selection of source and destination data nodes for re-replication of data has so far not been adequately addressed. In order to balance the workload among nodes during re-replication phase and reduce impact on cluster normal jobs' performance, we develop a re-replication scheme that takes into consideration of both performance and reliability perspectives. The appropriate nodes for re-replication are selected based on Analytic Hierarchy Process (AHP) with the consideration of the current utilization of resources by the cluster normal jobs. Toward effective data re-replication, we investigate the feasibility of using linear regression and local regression methods to predict resource utilization. Simulation results show that our proposed approach can reduce re-replication time, total job execution time and top-of-rack network traffic compared to baseline HDFS, consequently increases the reliability of the system and reduces performance impacts on users jobs. Regarding feasibility study of prediction methods, both regression methods are good enough to predict short time future resource utilization for re-replication.

Keywords: AHP, HDFS, Re-replication of data, Resource usage prediction

Introduction

Nowadays, every online user directly or indirectly uses cloud computing technology. Energy requirement to operate the cloud infrastructure is increasing. With increasing demand for data center's energy efficiency, resources are utilized to their maximum limits. No matter whether they are used to their thresholds, failures should be considered.

Storage system for cloud computing consolidates large numbers of commodity computers and provides reliable and high performance storage service. For storing and processing large data sets in cloud data centers, Hadoop Distributed File System (HDFS) is employed. HDFS provides data block replication scheme for reliability and data availability [2]. Default replication factor in HDFS is three and HDFS allocates the first replica on one node in the local rack. The second replica is placed on a node in a remote different rack. The third replica is placed on different node in the same remote rack [3].

* This paper is based on a previous paper published in the proceedings of the AUN/SEED-NET Regional Conference on Computer and Information Engineering (RCCIE), 2016 [1].

When a node(s) failure occurs in HDFS based data center, source and destination data nodes for re-replication are selected in a random manner [3]. In HDFS based computing platform, computation flows to the place where the data is located rather than moving data to the place where the computation is performed because of big size of data. Thus, random selection of nodes for replica placement in case of node failures can lead to workload imbalance in some of the nodes. The drawback of re-replication without considering the cluster normal jobs is evident that re-replication jobs can decrease the performance of normal jobs.

To understand the challenges of random node selection for re-replication in HDFS, we allocate replicas to nodes according to HDFS replication policy by using CloudSim simulation framework [4]. The detailed simulation environment and parameters are the same as in Evaluation section. Figure 1 shows the total number of data blocks that is received by each node in the data center. Three replicas for each data block are allocated with HDFS policy which takes into consider rack aware replication. As we can see in Figure 1, the number of data blocks each node received is different. We also performed re-replication based on default HDFS replication number and replication policy by injecting node failure condition. Figure 2 represents the total number of data blocks received on each node after a single node failure. We can see that some of the re-replicated blocks are allocated to the nodes where a large number of blocks are already allocated. This can lead to workload imbalance among the nodes because of computation's flow to the data.

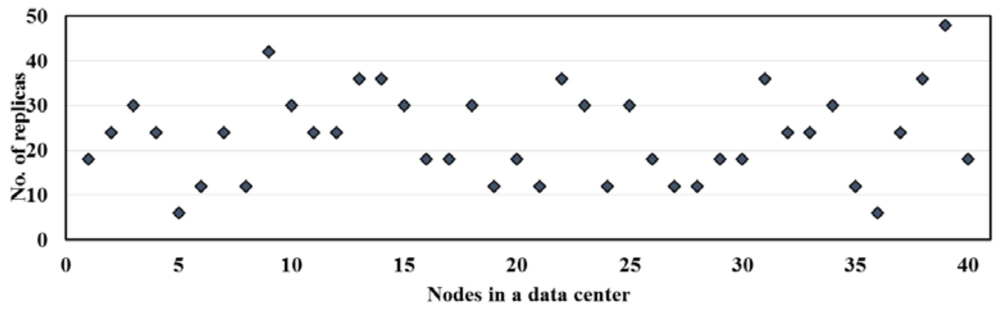

Figure 1. Replica placement on nodes based on HDFS

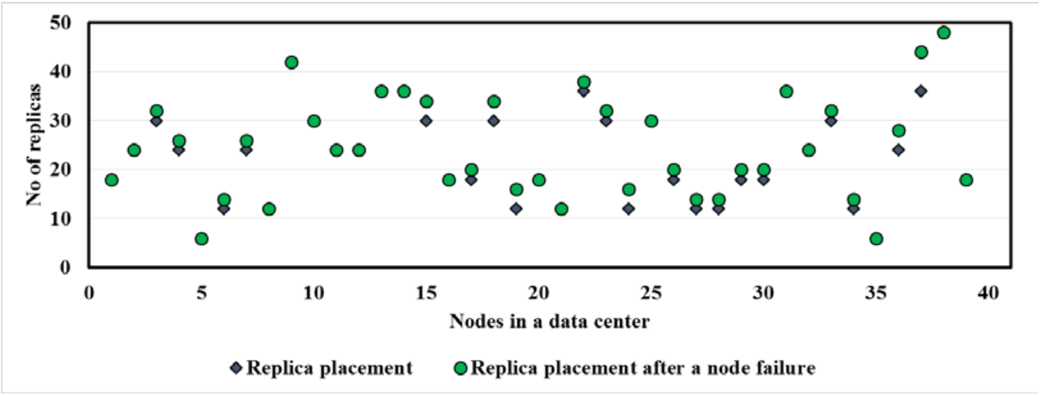

Figure 2. Replica placement on nodes based on HDFS after a node failure

The effective re-replication scheme is needed to select the appropriate nodes based on current resource utilization of the nodes. It will consider both performance and reliability issues. Moreover, while we are collecting resource utilization information for re-replication reactively, resources are continuously in use and actual resource utilization may be different with collected information at the time of actual re-replication. Thus, if we can predict resource utilization information, re-replication can be performed in proactive manner. This proactive nature can help to select the appropriate nodes for re-replication timely and to provide effective re-replication scheme.

Replica reconstruction schemes in the previous work [5] mainly focused on balancing the replication jobs among the nodes in the single rack cluster and they extended their work for the multi-rack cluster as proposed by Higai and colleagues [6]. Unfortunately, they did not address the impact on the cluster normal jobs' performance because of re-replication jobs. Our previous work [1] took into consideration of both performance and reliability perspectives and scheduled re-replication depending on the current system state and workload. The appropriate nodes for re-replication were selected with the consideration of the current utilization of resources by the cluster normal jobs. However, this approach performed re-replication reactively that means it collected resource usage information right after node failure occurred and resource usage changed at the time of actual re-replication, leading to poor re-replication scheduling. Thus, it is necessary to enhance re-replication in a proactive manner.

In terms of prediction methods, regression based prediction techniques, namely linear regression [7] and local regression [8], were applied in the previous works [9, 10]. The techniques were used on estimation of CPU utilization. They focused on how to determine whether a host was going to be overloaded or under-loaded by estimating CPU utilization and employed it in the VM migration process that aimed to reduce energy consumption of the servers. However, the authors of these studies did not directly mention how well these methods can predict utilization of resource.

We propose a re-replication scheme that selects appropriate nodes for re-replication based on AHP [11]. In the case of node failure, AHP selects appropriate nodes by ranking the nodes based on resource utilization information (i.e., CPU utilization, disk utilization, bandwidth utilization) and capacity of resources (i.e., CPU capacity, disk capacity and memory capacity). Re-replication scheduler uses the nodes that are selected by AHP to allocate the data blocks. If many data nodes fail simultaneously, e.g., the whole rack breaks down, there will be a massive amount of data blocks that are needed to be re-replicated which can occur many intra-rack and inter-rack data block transfers. In order to handle transferring of large amount of data blocks effectively, our proposed scheme applies priority grouping which allocates important data immediately and delays some portions of re-replication jobs based on current resource usage status of the system, consequently minimizes performance impacts on normal cluster jobs. In addition, in order to enhance the above mentioned reactive re-replication approach to proactive one, we investigate the feasibility of using linear regression and local regression methods to estimate short time future resource utilization which will be used in the proactive re-replication approach.

Related Work

Many researchers have studied data replication strategies in distributed, grid and cloud computing environments for reliabitiy, performance and energy efficiency. Recent studies in data replication strategies focused on energy efficiency and performance issues [12-16]. Although many studies have been carried out in replication strategies for reliability, data locality and energy efficiency, there has been little work that studied re-replication in case of node and rack failures. As our focus in this paper is on re-replication, we present closest related work with our proposed scheme in this section.

Higai et al. [5] proposed two replica reconstruction schemes, an optimization scheme and a heuristic scheme that aimed to balance the workloads of replication processes during the re-replication phase. In this work, the nodes were arranged in a virtual ring structure and data blocks were transferred based on this one-directional ring structure to minimize the difference of the amount of data transfer of each node. The source and destination data nodes were selected to balance only re-replication jobs. They demonstrated through several experiments that replica reconstruction throughput of the proposed schemes had significant improvement over default scheme.

They extended their work for multiple racks cluster [6]. In their proposed scheme, data transfer in a rack was performed based on the one-directional ring structure and inter-rack data transfer was performed in a round robin manner. Source and destination data nodes were selected to balance the load of each data node with respect to data transfer during re-replication. The re-replication jobs were scheduled by controlling the number of streams between racks and giving priority for the blocks based on consideration of the loads of networks and fault tolerance. However, the above two studies mainly intended to balance replica reconstruction workload among the nodes and did not address the impact of re-replication jobs on the cluster normal jobs. Our proposed re-replication scheme selected nodes for re-replication based on current resource utilization status and capacity status of the system and scheduled re-replication jobs accordingly, resulting in reducing performance impacts on normal cluster jobs.

Cidon et al. [17] proposed copyset replication technique which split the data nodes into copysets and stored each data block only on one copy set. They showed that under 1% of data nodes failure, copyset replication scheme protected the occurrence of data loss event whereas default HDFS almost guaranteed to lose data. In their subsequent research [18], they proposed tiered replication which divided the data nodes into primary tier and back up tier. Then, the first two replicas were allocated on primary tier and the third replica was allocated on back up tier. Under node failure condition, they studied the probability of data loss and showed that tiered replication reduced the data loss. These studies focused the day layout pattern in the cluster and under proposed allocation pattern of data, probability of data loss was successfully reduced. However, restoring data blocks on failed node to other remaining nodes based on resource utilization and capacity status of the system was not addressed.

Wang et al. [19] proposed a reliability model that took into consideration both data loss probability and bandwidth allocation for recovery process. Then, they analyzed reliability and system repair rate for different data layout schemes. Li et al. [20] proposed a reliability model to reduce storage cost and showed that only two replicas for each data block in the system assured wide range of data reliability. In contrast to these studies, we consider to re-replicate data blocks on failed node to other remaining data nodes effectively in consideration of both reliability and performance impacts on normal cluster jobs.

Proposed System

An overview of the proposed system is shown in Figure 3. As HDFS cluster consists of two types of nodes, namely, a name node and a number of data nodes, the re-replication scheduling is mainly performed by the name node. As shown in Figure 3, firstly the data nodes send resource utilization (CPU, disk, bandwidth) and capacity information (CPU, disk, memory) to the name node through heartbeat message periodically, by default every three seconds. The information for every data node in the system is collected in Resource Information Collection Module. At the same time, Data Access History keeps the history of every data block access in the system.

When a node failure occurs, the collected resource information is fed to AHP that ranks the data nodes in a decreasing order of resource utilization in combination with resource capacity.

At the same time, Data Popularity Module takes the popularity values from Data Access History and sorts them based on popularity. Then, re-replication scheduler allocates popular data on low utilization nodes based on ranked nodes lists and sorted replicas list to reduce performance impacts on normal cluster jobs.

In case of whole rack failure, there are massive amount of data blocks that need to be re-replicated. In order to balance reliability and performance, re-replication scheduler divides data blocks that need to be re-replicated into four priority groups based on the number of remaining replicas and popularity. Then, re-replication scheduler sets the re-replication schedule according to the priority group—schedules high priority group first as fast as possible and does low priority group replicas depending on the workload of the current system.

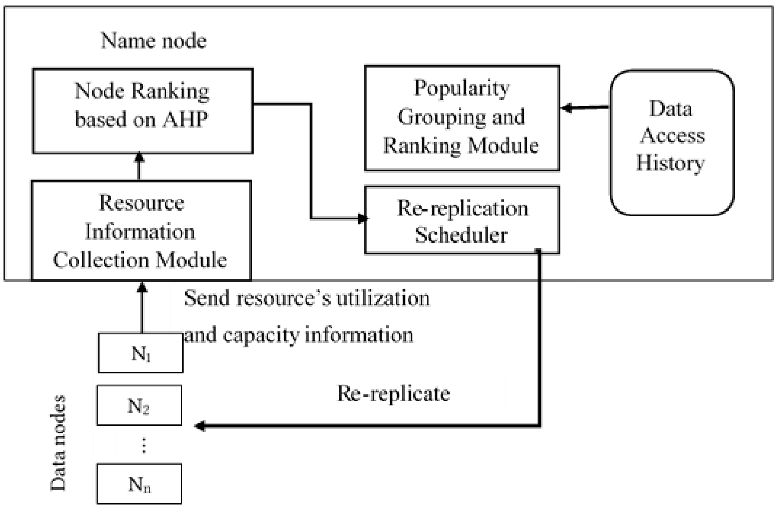
Name node
Node Ranking based on AHP
Popularity Grouping and Ranking Module
Data Access History
Resource Information Collection Module
Re-replication Scheduler
Send resource's utilization and capacity information
Re-replicate
N_1
N_2
N_n
Data nodes

Figure 3. Overview of the proposed system

Node ranking for re-replication based on Analytic Hierarchy Process (AHP)

Selection of nodes for re-replication is a complicated task because there are many alternative nodes in the data center with various utilization and capacities. For effective re-replication, appropriate nodes should be selected by considering the factors such as CPU utilization, bandwidth utilization, disk utilization, disk capacity, CPU capacity and memory capacity.

Analytic Hierarchy Process (AHP) is a method for ranking decision alternatives and selecting the best one when the decision maker has multiple objectives, or criteria and for optimizing decision making when one is faced with qualitative and quantitative criteria. The principle of AHP can be explained in the following steps:

(a) Define the problem
(b) Define criteria or factors and structure the decision hierarchy with criteria into levels and sublevels.
(c) Construct a set of pairwise comparison matrices of each factor with respect to each other.
(d) Synthesize the ranks of alternatives until the final choice is made. [11]

To select the best nodes, AHP method ranks the nodes based on criteria, namely, CPU utilization, disk utilization, CPU capacity, disk capacity and memory capacity giving the available nodes and user's preferences on the selection criteria. Then, re-replication scheduler selects the nodes that are ranked by AHP and performs re-replication.

Consideration for popularity of replicas

In HDFS based systems, during the re-replication phase, the source and destination data nodes are selected in a random manner [3]. Intuitively, some of the popular replicas may be allocated in a highly overloaded node so that it will result in workload imbalance among the data nodes because computation is basically performed on the node where the data is located. This proposed re-replication scheme considers the popularity of the replicas.

Based on the assumption that popular files in the past will continuously be accessed more frequently than the others in the future, the popularity of the replicas is determined by analyzing the history of access to these replicas. We need to define a threshold value to decide whether a replica is popular or not. In [21], they presented an approach to distinguish the popular data files. The average number of accesses ($\overline{\mathrm{NOA}}$) is used as the threshold. $\overline{\mathrm{NOA}}$ can be calculated as $\overline{\mathrm{NOA}} = \frac{1}{|\mathrm{H}|}\sum_{\mathrm{h}\in\mathrm{H}} \mathrm{NOA}\ (\mathrm{h})$, where $|\mathrm{H}|$ is the number of records in H that is the access history table, and each record h in H indicates the number of accesses NOA (h) for data block h. After determining popular replicas, the proposed system allocates the popular replicas on low utilization nodes during the re-replication phase.

Delay recovery of replicas to balance reliability and performance

In case of whole rack failure, many intra-rack and inter-rack data block transfers will be occurred. Intuitively, the network between racks may become the bottleneck if network bandwidth is occupied for re-replication jobs. At that time, if the system is busy with cluster normal jobs, re-replication jobs can decrease the performance of normal cluster jobs. If we only consider reducing performance impacts on cluster normal jobs, we can adjust to decrease re-replication jobs by changing HDFS configuration parameters (dfs.namenode. replication.work.multiplier.per.iteration and dfs.namenode.replication.max-streams), resulting in slow down of re-replication to finish. On the other side, according to the HDFS triple replication policy, if we have whole rack failure, one or two replicas of each data block will remain in the system. If we cannot perform quick re-replication of blocks that remains only one copy, another sub-sequent failure will cause permanent data loss.

Thus, an effective re-replication scheme is required to balance the system from both performance perspective, that is, reducing performance impacts on cluster normal jobs because of massive re-replication jobs and reliability perspective, that is, avoiding permanent data loss.

Our proposed system takes into consideration performance and reliability issues and schedules re-replication according to priority groups. The main reason behind grouping replicas is to schedule the data blocks in group order based on the current system state. The replicas that need to be re-replicated are divided into four different priority groups as shown in Table 1.

Table 1. Priority Group of Replicas

Group Name	Data Popularity	No. of. Replicas Left
G 1	√	1
G 2	×	1
G 3	√	2
G 4	×	2

The priority is based on the number of current replicas available in the system and popularity of the replicas. There are 4 different priority groups: G1 is for blocks which are popular and only one replica remains in the system. G2 is for blocks which are unpopular and one replica remains in the system. Then G3 is for blocks which are popular and two replicas remain in the system. And the last group, G4 is for blocks which are unpopular and two replicas remain in the system. G1 is the most important group and it is assigned as first priority group because the data blocks in this group are popular and only one replica remains, thus it needs to be re-replicated urgently. In case of whole rack failure, high priority groups, G1 and G2, are re-replicated quickly and low priority groups are delayed to be re-replicated later depending on the workload of the current system.

If whole rack failure is detected in the data center, except priority grouping is employed to effectively handle massive amount of data blocks to re-replication, data block allocation by selecting appropriate nodes based on AHP is the same for node failure and rack failure.

Toward Proactive Re-replication Approach

The system gathers resource utilization information in order to select the appropriate nodes for re-replication. While gathering resource information of the nodes, these resources are continuously in use during that time and there may be changes in values at the time of re-replication. Even though we select the appropriate nodes for re-replication, as it takes time in resource utilization collection phase before we perform re-replication, at the time of actual re-replication, selected nodes for re-replication may be overloaded. This can lead to performance degradation for both re-replication jobs and cluster normal jobs.

The re-replication approach could be enhanced if there is a predictive aspect in collecting resource utilization information for re-replication. This predictive aspect can help to prepare the necessary things to be prepared in advance or to provide better information for re-replication. In order to predict future resource utilization using historical resource usage, we apply linear regression [7] and local regression [8] prediction methods in this paper. Historical resource utilization information can be obtained periodically from the Heartbeat message. If a node failure occurs, the name node uses the resource usage record in recent one hour and predicts the future resource utilization for each node. The resource utilization prediction results will be used instead of collecting resource utilization by re-replication scheduler to perform re-replication in a proactive manner.

Linear regression

The two main objectives of linear regression are to forecast and to find the relationship between the variables. Linear regression can be used to fit a predictive model to an observed data set of y and x values. After developing such a model, if an additional value of x is then given, the fitted model can be used to make a prediction of the value of y [7] [22].

In this paper, the linear regression models the relationship between current resource utilization x and future resource utilization which is described by the following equation.

$$y = mx + c \tag{1}$$

where c and m represent the regression coefficients which can be obtained by minimizing the residual between prediction output and actual output over previous recorded utilization values using least square method. Least square is the popular method to minimize the error between the prediction output and actual output for each previous utilization value. After calculating regression coefficients using historical resource usage values in recent one hour, predicted utilization can be obtained from the equation mentioned above.

Local Regression

The main idea of the method of local regression is to divide the data points into localized subsets and to build up a function that describes the deterministic part of the variation in the data, point by point. The fitting of the model at any point is weighted toward the nearest data point. The observations are assigned neighborhood weights using the tri-cube weight function which is described below [8, 23].

$$w(x) = (1 - |d|^3)^3 \tag{2}$$

where d is the distance of given data value from the data value that is being fitted. For prediction of future resource utilization, local regression firstly assigns weight value to each historical resource utilization value using above mentioned weight function. After assigning weight value to each previous utilization value, the remaining calculation is the same with the procedure for linear regression.

Evaluation

Simulation setup

We used CloudSim[4] to evaluate our proposed re-replication system as CloudSim top benefits are flexibility, easy to customize, widely used and easy to integrate simulation framework. CloudSim is an extensible simulation toolkit that enables modeling and simulation of cloud computing systems and application provisioning environments. CloudSim framework supports both system and behavior modeling of cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. CloudSim is written in Java.

In order to evaluate our re-replication approach in case of node(s) and whole rack failures, fault injection is the key operation for testing by creating abnormal behavior to the system. Although our extended fault injection module is a separate module of original CloudSim codes, we also made some modifications in original codes for handling node failures that happened in the data center. For fault injection, we followed the outline described in [24]. The detailed modifications are as follows: providing Datacenter class for handling node and rack failures' fault event, providing Broker class with additional methods for re-replication, and providing modification on file and host class for defining the replica popularity. We set the simulation parameters based on [9, 10, 24]. The detailed simulation configurations are shown in Table 2.

We used two types of workload in the experiments, namely, random workload which is synthetic workload generated in CloudSim and workload traces from Planetlab which can be imported to the CloudSim simulator in order to simulate both extreme and realistic conditions.

Our proposed re-replication approach used AHP which is multi-criteria decision making method to select appropriate nodes for re-replication. In some of the experiments, we also used Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [25] which is also another multi-criteria decision making method to see the effective of AHP. Note that AHP and TOPSIS are not comparative studies of the proposed approach. The proposed re-replication approach applied AHP or TOPSIS method to select appropriate nodes for re-replication.

Table 2. Simulation Parameters

Parameter	**Value**
Total Data	300* 64MB
Nodes/rack	8
No. of. Racks	5
Replication Factor	3
Disk/Ram/CPU Capacity	Five different configurations for disk, memory and CPU (Disk-320G,500G,1T,2T,4T Ram-512MB,1G,2G,4G,8G CPU-750,1000,1500,2250,2500 MIPS rating)
Network	1Gbps

In all experiments, we compare our re-replication scheme with baseline HDFS. The reasons behind comparing the evaluation results only with baseline HDFS is that there are only a few studies that we presented in related work section to address re-replication problem in HDFS. However, these studies focused on balancing of re-replication workloads among the nodes whereas our proposal focused to select appropriate nodes for re-replication based on current resource utilization status of the system and took into consideration of balancing reliability and performance perspectives by applying priority groupings. Although related studies and our proposal tried to solve re-replication problems, the goal we are o to get is different. Related studies tried to improve the performance of re-replication jobs whereas our proposal tried to reduce impacts on cluster normal jobs by re-replication jobs and to balance reliability and performance. To highlight the effectiveness of our proposal in this paper, we compare with HDFS only.

Results

In this section, we evaluate the performance of our proposed re-replication approach. We first provide the comparison between baseline HDFS and our proposed approach in terms of average re-replication time, bandwidth cost and workload execution time. Various simulations are carried out with different number of node failures and different number of workloads.

Figure 4 demonstrates the comparison of three different re-replication strategies, namely, baseline HDFS, proposed scheme based on AHP, and proposed scheme based on TOPSIS. As baseline HDFS allocates the replica in a random manner, the number of replicas that each node received is different that could lead to workload imbalance among the nodes in data center. Among them, proposed scheme based on AHP allocates replicas more stable than the others. As job scheduling flows to data, it can balance the workload among the nodes and reduce the impact of re-replication jobs on cluster normal jobs.

Figure 5 shows the execution time of re-replication. In the simulation, we injected different number of node failures and observed the difference between baseline HDFS and our proposed approach in terms of average re-replication time.

Clearly, the Figure shows that the average re-replication time in HDFS is higher than our proposed approach because our proposed approach gives higher priority to intra-rack block transfers than inter-rack block transfers and considers utilization state of nodes (CPU, disk and bandwidth) to select the appropriate nodes for re-replication so that it can avoid overloading condition on the nodes.

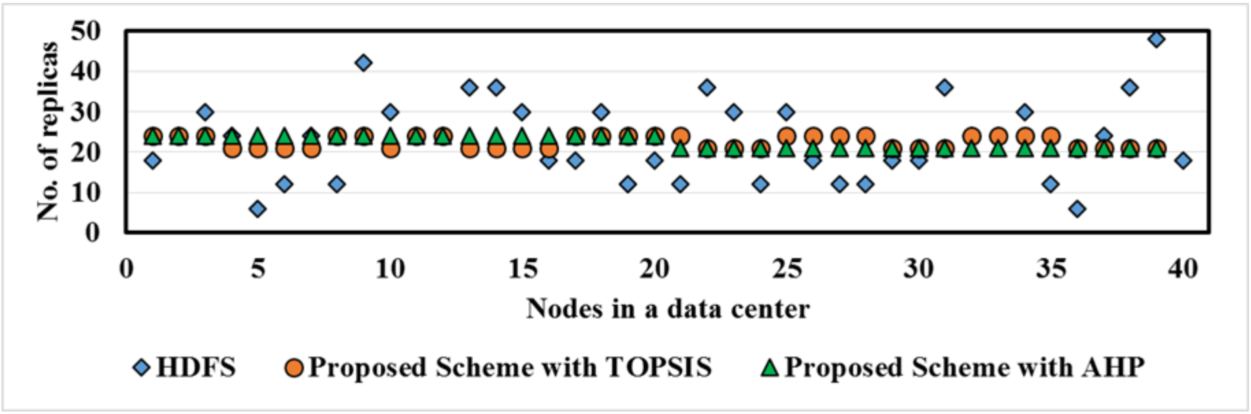

Figure 4. Replica placement with different re-replication schemes

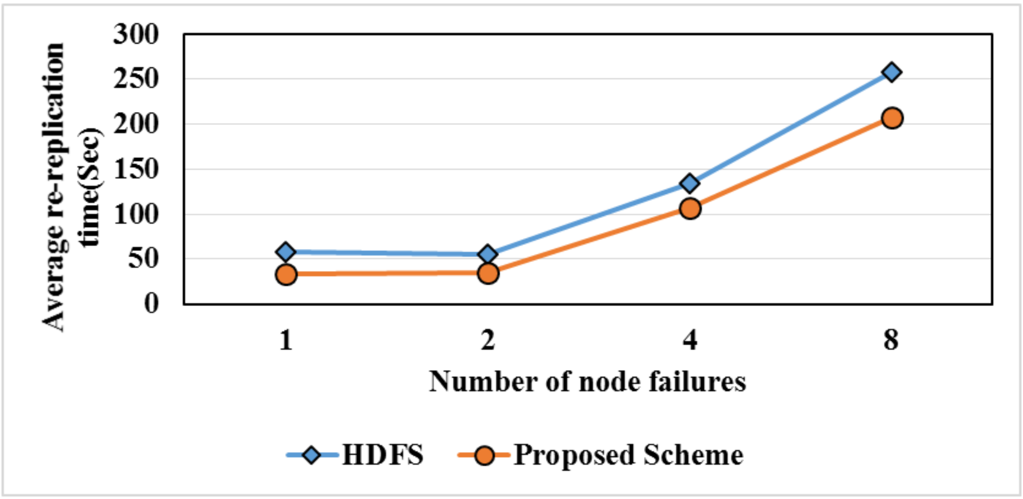

Figure 5. Average re-replication time

Total number of intra-rack and inter-rack block transfers during re-replication for baseline HDFS and our proposed scheme is shown in Figure 6. We can clearly see that during the re-replication, number of inter-rack transfer blocks that are replicated with HDFS scheme is more than that in our proposed scheme. The reason is that HDFS selects the source and destination nodes for re-replication randomly. In contrast, our proposed scheme gives the priority to intra-rack transfer in order to reduce network traffic on top of rack switch.

Figure 7 compares the simulation results of total execution time with varying number of jobs. In the simulation, we injected one node failure in order to investigate the impacts of re-replicated jobs on cluster normal job performance. We can see that re-replication with proposed scheme based on AHP has low impact on the response time of cluster normal jobs. This is because our proposed scheme based on AHP selects the appropriate nodes for replica allocation and gives priority to intra-rack block transfer, resulting in faster execution time of the job.

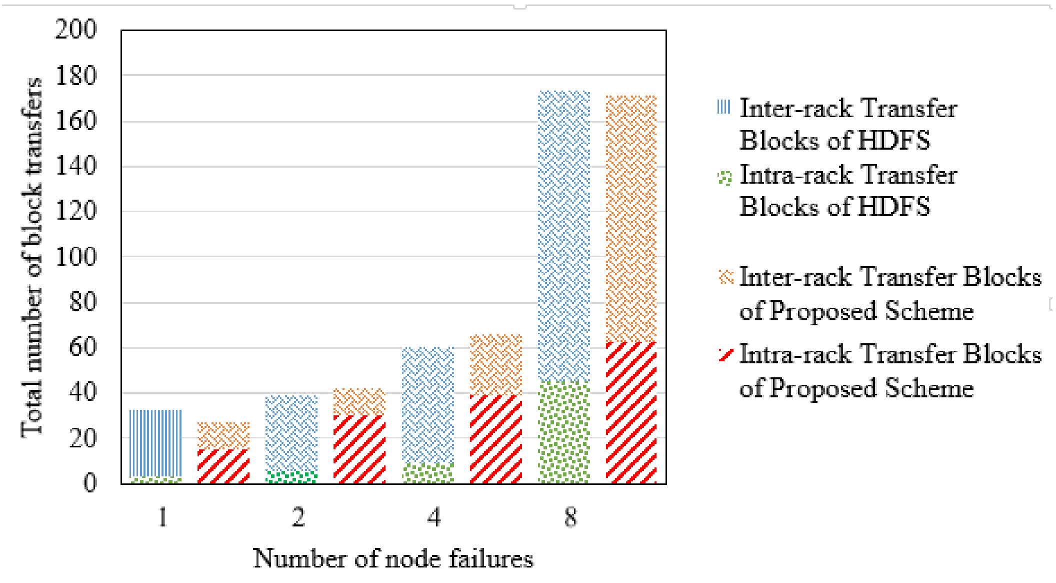

Figure 6. Number of intra-rack and inter-rack block transfers during re-replication

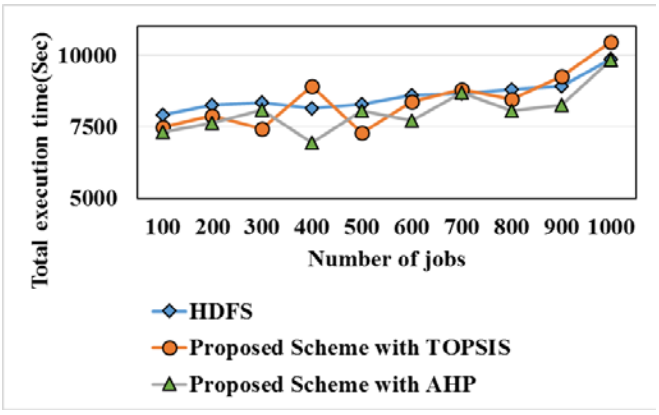

Figure 7. Total job execution time

Although there is a few possibility of data node failures again in a short period of time after the whole rack failure, our proposed scheme takes that kind of condition into consideration to prevent data loss from any kind of situation. Figure 8 compares our proposed scheme and baseline HDFS in terms of vulnerability to data lost in percentage. We injected different number of node failures after the whole rack failure and before the re-replication process finishes completely. Figure 8 demonstrates that vulnerability to data lost in baseline HDFS is higher than our proposed scheme because our proposed scheme performs the re-

replication by grouping the replicas that depend on the number of remaining replicas and popularity, and gives the priority to the replicas which remains only one replica in the system.

In order to reduce the impact on cluster normal jobs due to re-replication, our proposed scheme divides the replicas on failed nodes into the four different groups and replicates the high priority group replicas as quickly as possible and schedules low priority group replicas depending on the workload of the current system. Table 3 describes average re-replication time in case of whole rack failure with and without grouping and delay scheduling of replicas. We observe that average re-replication time of grouping and delay scheduling of replicas is slightly slower than average re-replication time of without grouping and delay scheduling of replicas because in case of whole rack failure, there are massive amount of data blocks that need to be re-replicated. If we re-replicate all data blocks at once, re-replication jobs will compete with normal cluster jobs for resource utilization, resulting in decreasing performance of normal cluster jobs. Thus, we delayed some portions of re-replication jobs which are in lower priority groups to later time, resulting in increasing repair time but reducing competition with normal cluster jobs, which consequently can reduce performance impacts on normal cluster jobs.

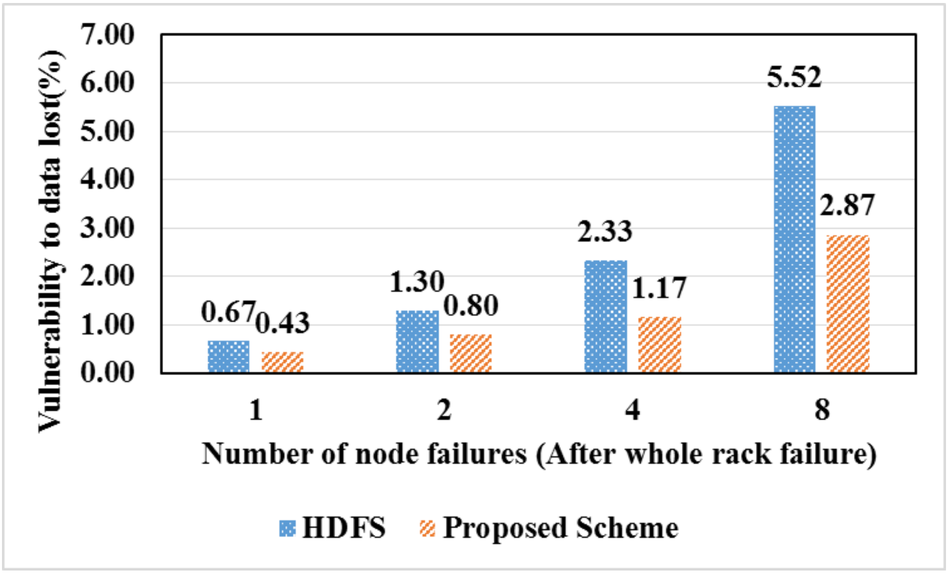

Figure 8. Vulnerability to data lost

Table 3. Re-replication Time with/ without Grouping and Delay Scheduling of Replicas

Proposed Re-replication Type	**Average Re-replication Time (Sec)**
Re-replication without grouping and delay scheduling of replicas	162.3225
Re-replication with grouping and delay scheduling of replicas	174.9225

Results for proactive re-replication

In order to compare the feasibility of linear regression and local regression prediction methods, we use three metrics: prediction accuracy, relative error and execution time. We employ random workload in which load may change abruptly in a random manner and real workload traces that are provided as a part of CoMon project, a monitoring infrastructure for PlanetLab [26] [10]. This real workload traces consist of CPU utilization of more than a thousand VMs from servers located at more than 500 places around the world.

In Figure 9, we compare prediction results with actual CPU utilization using random workload for both prediction methods. As observed from the results, the accuracy of the prediction results for random workload is not good and both prediction methods cannot predict good enough for random workload. This is because of random workload characteristics which can change abruptly through time and difficult to predict resource utilization at next time interval.

Figure 10 compares the actual CPU utilization with the predicted values using linear regression and local regression for real workload. We observe that prediction with locally weighted regression is closer with actual value than linear regression.

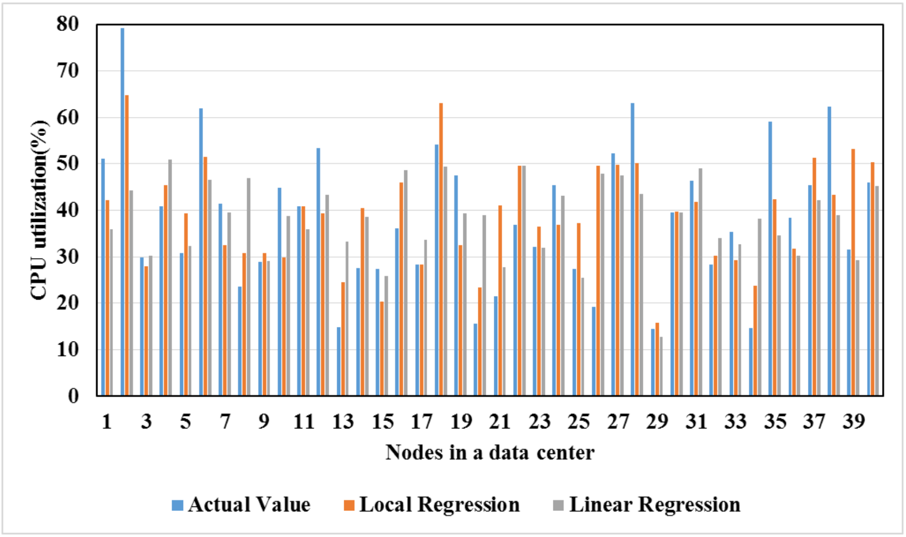

Figure 9. Actual vs predicted CPU utilization using random workload

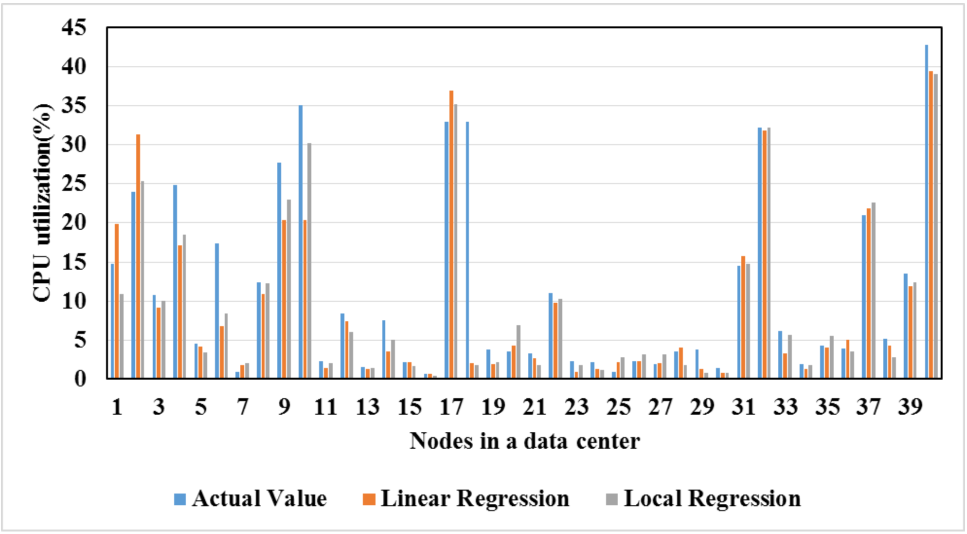

Figure 10. Actual vs predicted CPU utilization using real workload

In order to see the efficiency of the two prediction methods clearly, we also use relative error, $\text{Utilization}_{\text{error}}$ as an evaluation matric to evaluate the accuracy of the regression model, and its computation is in the following equation:

$$Utilization_{\,error} = \frac{Utilization_{prediction} - Utilization_{actual}}{Utilization_{actual}}$$

where, Utilization $_{prediction}$ is the result that we get using regression model and Utilization$_{actual}$ is the actual CPU utilization that we measure during the simulation. From Figure 11 and Figure 12, we can clearly see that relative error of local regression is closer to the origin line than liner regression results.

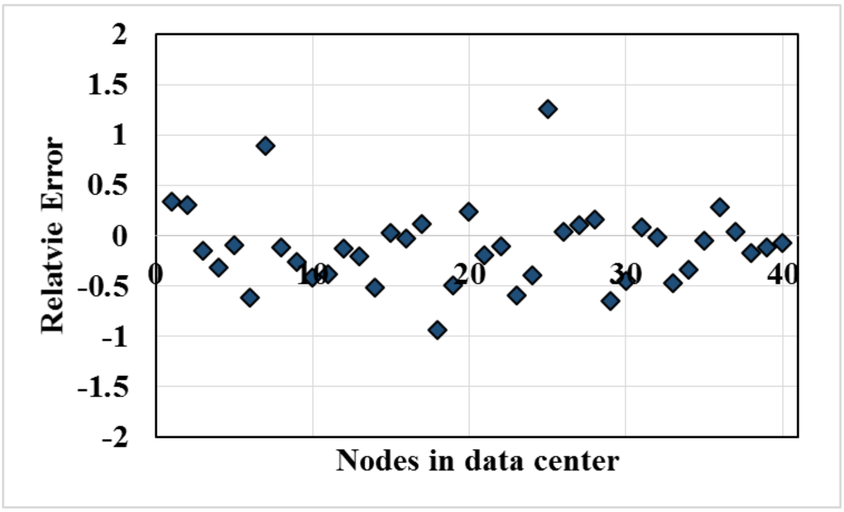

Figure 11. Relative error for linear regression

Table 4 describes execution times of both prediction methods. Our execution environment was Intel (R) Core i7-4770 CPU 3.40GHz and memory was 8GB. In the experiment, we run simulations ten times for real workload data and calculated average execution time of prediction methods. From the table, we observe that average execution time of linear regression was slightly faster than local regression method. We used real workload because real workload has fixed pattern of utilization values that came from real workload traces [26]. As we discussed in the section of linear regression and local regression, we used utilization values in recent one hour. As workload trace records utilization at every five minute, both prediction methods predict future resource utilization using 12 data values. Thus, the amount of utilization values both methods take for all simulation runs was 12 data points. The reason why local regression took more time is that local regression has to calculate weight function as extra step compared with linear regression. By seeing the results of each simulation run, we did not see big difference in each simulation run. Thus, we believe that mean value of 10 simulation runs gave us enough information to compare the execution time of both prediction methods. As data size we used for prediction is not big, the execution time difference between the two methods is not significant. But we find that at every simulation run, linear regression is faster than local regression.

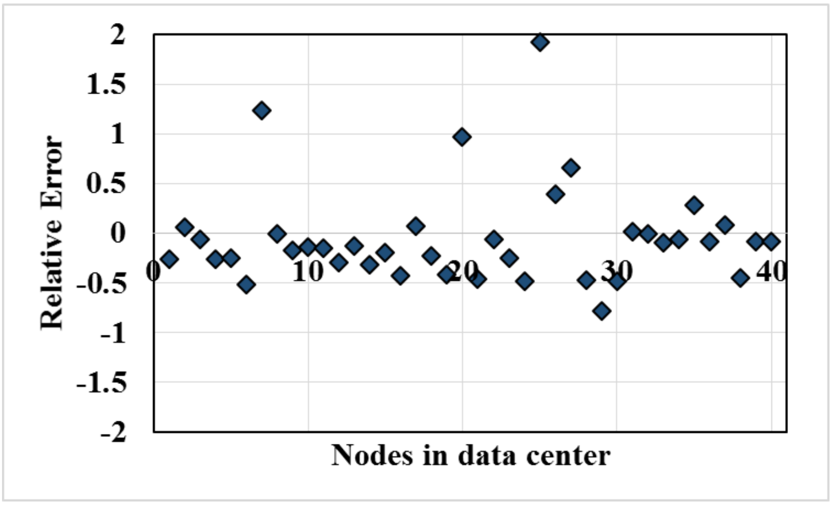

Figure 12. Relative error for local regression

From the experiments, we can see that local regression can predict more accurately than linear regression but in terms of execution time, linear regression is faster than local regression.

Table 4. Execution Time of Prediction Methods in Milliseconds

No of Simulations	Local Regression	Linear Regression
1	5.4530	5.0220
2	5.4899	5.1075
3	5.7073	5.2222
4	5.4550	5.1355
5	5.6462	5.1031
6	5.9219	5.2114
7	5.5049	5.1105
8	5.9940	5.1811
9	5.6705	5.0215
10	5.7739	5.0156
Average	5.6616	5.1130

Discussion

As the proposed system considers the popularity of the replicas, the name node has to keep access count of every replica in the system to determine whether a replica is popular or not. The extra space for managing these metadata is needed for the proposed system. We also argue the fact that AHP made the pairwise comparison among the criteria to select the appropriate node. If the number of data nodes in the system increases, penalty for calculation of AHP can be a concern for the proposed system.

In this paper, we start our work towards proactive re-replication approach by predicting CPU utilization and disk utilization on historical usage data. We will use predicted resource utilization in order to determine overloaded node and schedule re-replication in a proactive manner.

Conclusions

In this paper, we presented a re-replication approach that takes into consideration both performance and reliability perspectives. In order to balance the workload among the nodes during re-replication phase and reduce impact on cluster normal jobs performance, appropriate nodes for re-replication were selected based on AHP by considering the current utilization of resources. Our proposed system divided the data blocks that need to be re-replicated into the four different priority groups to balance the system from both reliability and performance perspectives and automatically scheduled re-replication. In order to perform re-replication effectively, we investigated the feasibility of using linear regression and local regression methods. The proposed system was studied and evaluated through a simulation in CloudSim framework. The results demonstrated that re-replication execution time and total job execution time were less than baseline HDFS and the system reduced bandwidth consumption in top-of-rack switch. Based on the study and results of prediction methods, we will use predicted resource utilization information to perform re-replication in a proactive manner.

Acknowledgement

We would like to thank an anonymous reviewer for kind suggestion to improve our manuscript especially the suggestion to add execution time of both prediction methods for all simulation runs.

References

[1] T. Shwe, and M. Aritsugi, "A re-replication approach in HDFS based cloud data center," *The AUN/SEED-NET Regional Conference on Computer and Information Engineering (RCCIE)*, Yangon, Myanmar, pp.181-186, 2016.

[2] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," Paper presented at *IEEE 26th Symposium on Mass Storage Systems and Technologies*, Nevada, USA, 2010, doi: 10.1109/MSST.2010.5496972

[3] T. White, *Hadoop: The Definitive Guide*, 3rd Edition, O'Reilly Media, Inc, 2012.

[4] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F.D. Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software-Practice and Experience,* Vol. 41, Issue 1, pp.23-50, 2011. doi: 10.1002/spe.995

[5] A. Higai, A. Takefusa , H. Nakada, and M. Oguchi , "A study of effective replica reconstruction schemes at node deletion for HDFS," Paper presented at *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Chicago,* United States of America, 2014, doi: 10.1109/CCGrid.2014.31

[6] A. Higai, A. Takefusa, H. Nakada, and M. Oguchi, "A study of replica reconstruction schemes for multi-rack HDFS clusters," Paper presented at *IEEE/ACM 7th International Conference on Utility and Cloud Computing,* London, United Kingdom, 2014, doi: 10.1109/UCC. 2014.28

[7] G.A.F. Seber, and A.J. Lee, *Linear Regression Analysis*, 2nd Edition, John Wiley and Sons, Inc, 2003.

[8] W.S. Cleveland, "Robust locally weighted regression and smoothing scatter plots," *Journal of the American Statistical Association,* Vol. 74, No. 368, pp.829-836, 1979. doi: 10.1080/01621459.1979.10481038

[9] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," Paper presented at the *39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA),* Santander, Spain, 2013, doi: 10.1109/SEAA.2013.23

[10] A. Beloglazov, and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience (CCPE),* Vol. 24, Issue 13, pp. 1397-1420, 2012. doi: 10.1002/cpe.1867

[11] T.L. Satty, "Decision making with the analytic hierarchy process," *International Journal of Services Sciences,* Vol. 1, No. 1, pp. 83-98, 2008. doi: 10.1504/IJSSci.2008.

01759

[12] R.T. Kaushik, and M. Bhandarkar, “GreenHDFS: Towards an energy-conserving, storage-efficient, hybrid Hadoop compute cluster,” In: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, USENIX Association, Berkeley, United States of America, pp.1-9, 2010.

[13] R.T. Kaushik, and K. Nahrstedt, “T*: A data-centric cooling energy costs reduction approach for big data analytics cloud,” Paper presented at *International Conference on High Performance Computing, Networking, Storage and Analysis,* Salt Lake City, Utah, United States of America, 2012, doi: 10.1109/SC.2012.103

[14] S. Long, Y. Zhao, and W. Chen, “A three-phase energy-saving strategy for cloud storage systems,” *Journal of Systems and Software*, Vol. 87, pp. 38-47, 2014. doi; 10.1016/j.jss. 2013.08.018

[15] B. Liao, J. Yu, T. Zhang, G. Binglei, S. Hua, and C. Ying, “Energy-efficient algorithms for distributed storage system based on block storage structure reconfiguration,” *Journal of Network and Computer Applications*, Vol.48, Issue C, pp. 71-86, 2015. doi: 10.1016/ j.jnca.2014. 10.008

[16] L. Zhang, Y. Deng, W. Zhu, J. Zhou, and F. Wang, “Skewly replicating hot data to construct a power-efficient storage cluster,” *Journal of Network and Computer Applications*, Vol.50, Issue C, pp. 168-179, 2015. doi: 10.1016/j.jnca.2014.06.005

[17] A. Cidon, R. Escriva, S. Katti, M. Rosenblum, and E.G. Sirer, “Tiered replication: A cost-effective alternative to full cluster geo-replication,” In: *the Proceedings of the 2015 USENIX Annual Technical Conference (USENIC ATC’15)*, USENIX Association, Santa Clara, California, United States of America, pp.31-43, 2015.

[18] A. Cidon, S.M. Rumble, R. Stutsman, S. Katti, J. Ousterhout, and M. Rosenblum, “Copysets: Reducing the frequency of data loss in cloud storage,” In: *2013 USENIX Annual Technical Conference (USENIX ATC’13)*, USENIX Association, pp. 37–48, 2013.

[19] J. Wang, H. Wub, and R. Wand, “A new reliability model in replication based big data storage systems,” *Parallel and Distributed Computing*, Vol. 108, pp.14-27, 2017. doi: 10.1016/j.jpdc.2017.02.001

[20] W. Li, Y. Yang, J. Chen, and D. Yuan, “A cost-effective mechanism for cloud data reliability management based on proactive replica checking,” Paper presented at the *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Ottawa, ON, Canada, 2012, doi:10.1109/CCGrid.2012.33

[21] M. Tang, B.-S. Lee, X. Tang, and C.-K. Yeo, “The impact of data replication on job scheduling performance in the data grid,” *Future Generation Computer Systems,* Vol. 22 Issue 3, pp.254-268, 2006. doi:10.1016/j.future.2005.08.004

[22] “Linear Regression”. [Online] Available: https://en.wikipedia.org/wiki/Linear_regression. [Accessed: December, 2016]

[23] “Local Regression”. [Online] Available: https://en.wikipedia.org/wiki/Local_regression. [Accessed: December, 2016]

[24] M.C. Nita, F. Pop, M. Mocanu, and V. Cristea, “FIM-SIM: Fault injection module for CloudSim based on statistical distributions,” *Journal of Telecommunications and Information Technology,* Vol. 4, pp. 14-23, 2014.

[25] M. Velasquez, and P.T. Hester, “An analysis of multi-criteria decision making methods,” *International Journal of Operations Research,* Vol.10, No.2, pp.56-66, 2013.

[26] K.S. Park, and V.S. Pai, “CoMon: A mostly-scalable monitoring system for PlanetLab,” *ACM SIGOPS Operating Systems Review,* Vol.40, Issue 1, pp.65-74, 2006. doi:10.1145/ 1113361.1113374