PPHA-Popularity Prediction based High Data Availability for Multimedia Data Center

Kuo-Chi Fang, Husnu S. Narman, Ibrahim Hussein Mwinyi, and Wook-Sung Yoo

Computer Science, Marshall University, Huntington, WV 25755

{*fang5, narman, mwinyi, yoow*}*@marshall.edu*

Abstract

Due to the growth of Internet-connected devices and extensive data analysis applications in recent years, cloud computing systems are largely utilized. Because of high utilization of cloud storage systems, the demand for data center management has been increased. There are several crucial requirements of data center management, such as increase data availability, enhance durability, and decrease latency. In previous works, replication technique is mostly used to answer those needs according to consistency requirements. However, the most of the works consider full data, popular data, and geo-distance based replications by considering storage and replication cost. Moreover, the previous data popularity based-techniques rely on the historical and current data access frequencies for replication. In this paper, we approach this problem from a distinct aspect while developing replication techniques for a multimedia data center management system which can dynamically adapt servers of data center by considering popularity prediction in each data access location. Therefore, we first label data objects from one to ten to track access frequencies of data objects. Then, we use those data access frequencies from each location to predict the future access frequencies of data objects to determine the replication levels and locations to replicate the data objects, and store the related data objects to close storage servers. To show the efficiency of our proposed methods, we conduct an extensive simulation by using real data. The results show that our proposed method has an advantage over the previous works in terms of data availability and increases the data availability upto 50%. Our proposed method and related analysis can assist Multimedia service providers to enhance their service qualities.

Keywords: Multimedia, Data center management, popularity, prediction, data access frequencies, latency

1 Introduction

Due to the growth of Internet-connected devices and extensive data access applications in recent years, cloud computing systems are largely utilized. Because of high utilization of

cloud storage systems, the demand for data center management has been increased. One of the challenges of data center management is the low performance because of the data unavailability when the scale of data and data center increase. To enhance system performance, several methods have been proposed (Vdovin & Kostenko, 2014; Nagendram, Lakshmi, Rao, & Jyothihi, 2011; Arzuaga & Kaeli, 2010; Plakunov & Kostenko, 2014; Hieu, Di Francesco, & Jääski, 2014). Authors in (Vdovin & Kostenko, 2014; Nagendram et al., 2011) proposed a resource scheduling method to increase the utilization rate of each server in the system by considering the multi-dimensional resource requirements (e.g., CPU, Memory and Storage) of applications, and then schedule these applications to different servers. It is expected that the performance of the system would increase if the scheduling method does not waste any resource. However, the performance of system may decrease if many queries were scheduled to the same servers, or some applications spend more time in allocated servers. Authors in (Arzuaga & Kaeli, 2010; Plakunov & Kostenko, 2014; Hieu et al., 2014) considered the above limitations and balance server loads in the data center. Distinct loading metrics have been used to analyze the loading of each server node for rearrangement in the system. For example, authors in (Arzuaga & Kaeli, 2010) calculated each physical server load and dynamically move the assigned tasks from high loaded servers to lower loaded ones. Authors in (Plakunov & Kostenko, 2014; Hieu et al., 2014) have a different approach to solve the same balancing problem. Instead of moving tasks inside the data center, initially, the tasks are assigned to servers which utilize the servers by balancing loads.

All of the similar aforementioned approaches can solve unbalanced load problem in the data center, but they neglect network-bottlenecked problem, which may be the critical limitation for systems performance in data centers. Although some researchers consider the bandwidth utilization in the data center network by balancing link utilization of network system, some traffic issues still cannot be avoided (Botero, Hesselbach, Fischer, & De Meer, 2012; Mustafa & Nadeem, 2015). For example, the network traffic may still appear if some server nodes in the network are essential or have popular data objects, which are accessed by many users. Therefore, the network-bottleneck will still exist even though system tries to make an appropriate utilization of bandwidth.

In addition, data replication (Zeng et al., 2017) is used to decrease the data loss because of network-bottleneck to increase the data availability rate. In the replication, data popularity is critical because of storage costs and system efficiency. The popular data has higher load because of the excessive access requests (Ananthanarayanan et al., 2011) which can result in data unavailability. However, the less popular data has only a few requests for data access. Therefore, while replicating data, data diversity must be considered with load balancing to utilize the resources efficiently (Keller, Szefer, Rexford, & Lee, 2010). However, unpredictability of future access can lead unbalancing in long-term. Therefore, the system which predicts the future popularity of data can be more efficient. Therefore, the *aim* of this paper is to increase data availability in multimedia data centers by considering not only the current data popularity but also possibilities for future data popularity in addition to bandwidth limitation and load balancing. Unlike (Arzuaga & Kaeli, 2010), our system can dynamically adapt the structure of data center with future prediction. In other words, the system can pre-manage the data center so that the performance of the system will be improved.

The *objective* of this paper is to develop a self-learning management system for a data center which replicates data to different server nodes according to popularity predictions of data objects. We consider both utilization of local servers and bandwidth limitations so that

system can have better performance. We adapt tree-like network architecture (Plakunov & Kostenko, 2014; Goswami, Pattanaik, Bharadwaj, & Bharti, 2014). Each server node has appropriate data based on the data access location and their access frequencies. Then, the system keeps track the popularity of each data object and decides their status (hot, warm, or cold according to data access frequencies) (Zeng et al., 2017). After that, the proposed method associates a rank table with prediction information about the future data access probabilities and establishes the prediction table for future data accesses. Based on the prediction table, the data center can replicate hot data to more server nodes to increase scalability and reduce network traffic. Moreover, for the data which access frequencies are decreased, the proposed system holds only three copies as similar to Three Random Replication in different types of storages (e.g., HDD and Flash) to decrease the storage cost. Inspired by (Kim, Paek, & Bahk, 2015; Shi, Shi, Wei, Ding, & Wei, 2017), the proposed method dynamically activates or inactivate nodes according to the data access frequencies so that system could save energy (Shi et al., 2017; Lin, Wierman, Andrew, & Thereska, 2013; DeCandia et al., 2007; Brandon et al., 2010). It is important to note that our proposed method is only for the multimedia data center. The performance of the proposed method may not be efficient for the other type of data center management because of task size and task scheduling features. The results show that our proposed method can dynamically adapt server nodes for future input and has an advantage over other method strategies by increasing data availability upto 50% although the prediction can have errors.

The key *contributions* of this paper are as follows:

- A multimedia data center management system has been proposed according to popularity prediction by considering load balancing and bandwidth limitations.
- We develop an extensive simulation by using real-data to test the performance of the proposed method and compare the proposed method with popularity, location, and classification based data center management techniques.
- The proposed method also tested by considering 0% (perfect without error), 10%, 20%, 30%, 40% and 50% prediction error rates to show the effectiveness of the proposed system.

The rest of the paper is organized as follows: Section 2 describes a general model of data center. In Section 3, our proposed data management system and the details of the proposed method are explained. Section 4 describes the simulation design and the used test cases to analyze the effectiveness of our proposed method. Section 5 consists of the results, and Section 6 includes final discussion and our plan to extend this work.

2 Data Center Model

In this section, we describe the general model of data center (Figure 1). A data center is a cluster of servers with multiple tiers. It contains routers, computers, and storages. The multiple servers connect to each other by a number of routers. Based on different approaches, managers can choose different network architectures (Plakunov & Kostenko, 2014; Meng, Pappas, & Zhang, 2010; Goswami et al., 2014; Ito, Mohri, Shiraishi, & Morii, 2016; Bharti & Pattanaik, 2013).

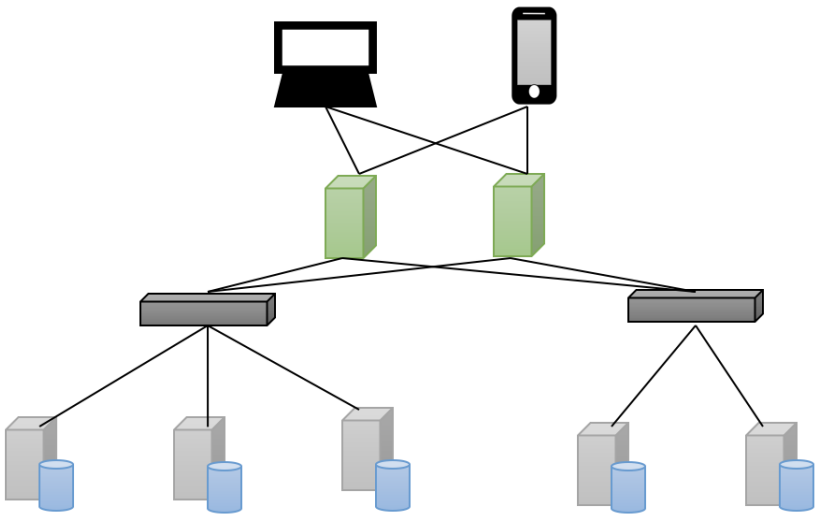

Figure 1: Illustration of a data center.

Some server nodes of the network contain storage for storing data. There are several types of storages, such as Solid State Disk (SSD) and Hard Disk Drive (HDD). The system can get faster I/O reading with SSD, but the life cycle of SSD is usually lower than HDD, and data might be lost forever when storages are broken. Some server nodes are external tiers to communicate to outside devices and allocate tasks to server nodes. For example, in searching systems, an application sends a searching query to the external interface or tier 1. Then, the manager node allocates searching tasks to internal tier nodes. After the internal nodes get requests, they search the requested information from their storages and send back searching results to manager nodes.

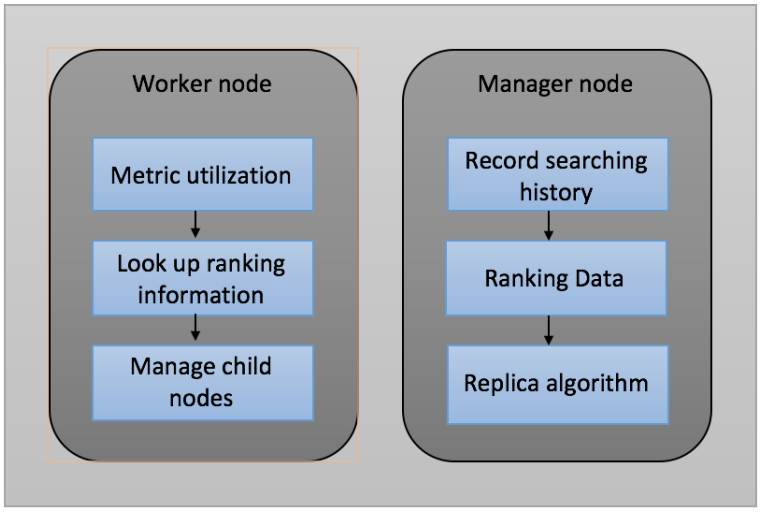

Figure 2: System flowchart for worker and master nodes.

3 Proposed System

In this section, we discuss the proposed method by giving the details under Management, Network, Ranking, Metrics and Replication subsections.

3.1 System Management

In this section, we present a system management strategy which considers the resource utilization and network-bottlenecked issues. We assume each server node in data center network is located in different locations (see Section 3.2). Initially, data is allocated to nodes according to their location information. For example; data object A would be stored in node 1 which is located at location K if it is uploaded from location K or has high searching frequency at location K. Each server node tracks its own data access and manages its child server nodes and data replication. Besides, manager server nodes calculate the status of data based on ranking algorithm (see Section 3.4) and dynamically active or deactivate server nodes according to ranking information if the server nodes do not serve for other data objects. As shown in Figure 2, worker and manager nodes would dynamically adapt server nodes based on the utilization and data ranking information. Therefore, the system can balance the loading of each server node and increase the utilization.

3.2 Network Architecture

In our system, we use the tree-like structure of a data center as shown in Figure 3. Server controller connects to each working server node and allocates queries to each node according to the bandwidth of each link. There are three types of server nodes. The first type is the server node which stores original data. Other is the replica server node, contain some popular data replication to decrease network traffic by spreading out the popular data to different server nodes. The final type is the child server node which replicates data from parent's node to reduce the loading of parent's nodes.

3.3 Prediction

In this section, we inspired by the prediction method which was proposed for a recommendation system (Mwinyi, Narman, Fang, & Yoo, 2017). However, we have modify the prediction method (Mwinyi et al., 2017) according to our needs and requirements of the media data center. In order to have a prediction methods, we have to identify and define some parameters to find the future access rates for each data object. The parameters are listed in Table 1. By using the parameters, Equation (1) is developed to find future access rates of each data object.

$$R = \alpha DA + \beta IA + (\gamma \frac{AD}{10} + \eta \frac{DL}{10}) + \kappa TU + \varsigma AR \tag{1}$$

where $\alpha, \beta, \gamma, \eta, \kappa$, and ς are constant; and DA, IA, AD, DL, TU, and AR are normalized by obtaining the maximum value of the P-prediction as 10 for each access location of the data object. According to this prediction, we create the prediction ranking table for each data object as shown in Figure 4. Then, by considering AL rates for each data object, the

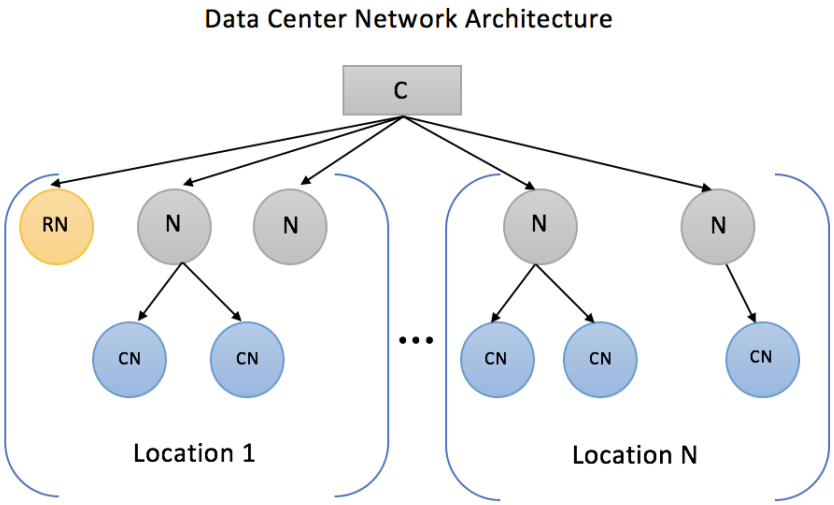

Figure 3: The network architecture: C as Server controller, N as Server Node which store data, RN as replica Node, CN as child Node of Server Node.

data object is replicated on the nodes which are close to the those locations and determined according to Equation (2).

Table 1: Parameters which are used for prediction

DA	The direct access amount of the data object
UL	Upload location of the data object
AL	Access locations with their access amounts to the data object
IA	Indirect access amount of the data object
AD	Application or user diversities of accesses to the data object
DL	Location diversities of accesses to the data object
TU	Duration of the data object existence
AR	The access rates of the related data objects

3.4 Ranking Strategy

The ranking strategy in our system is inspired by (Zeng et al., 2017). Authors in (Zeng et al., 2017) calculate the weight of data objects in terms of the importance level of data according to data access. In our system, we consider the popularity of data as the feature to determine the popularity level of data. Each server node records data access history and creates the frequency table of data accesses. After that, the system generates a ranking table by combining the frequency table and input of prediction to determine the possible data access frequencies in the future. Then, the system marks the levels of data objects, such as hot, warm or cold, to decide whether the data should be replicated or not. The highly accessed data objects are replicated to more server nodes in case of high network traffic. On the contrary, cold data is moved to low-performance nodes and not replicated more than three times because of common Random Three Nodes replication. Figure 4 shows the schematic

diagram of the ranking algorithm. The ranking table has the information about future data access so that the data center can adapt data of each node better to improve the performance.

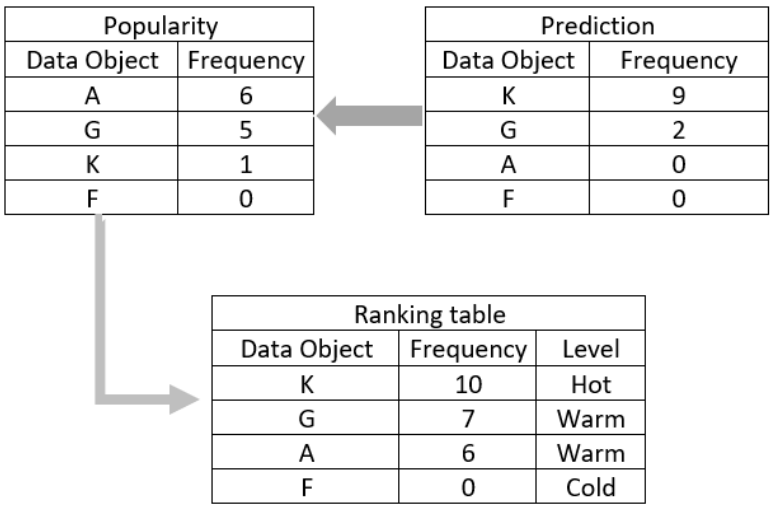

Figure 4: Schematic diagram of the ranking algorithm.

3.5 Metrics

The performance of the system can decrease if many users request data from the same server node at the same time. Our goal is to dynamically forward the requests to the nodes which have the same data. If such nodes have not been created, the data is replicated to an adjacent-child node and forward requests to the recently created node to balance the high loading. We adapt the loading metric from (Arzuaga & Kaeli, 2010). We consider the multiple resources of each server node, such as CPU utilization, memory utilization, and disk I/O performance. Each node creates child nodes and replicates high popular data objects if the node load is over the threshold. We define our loading metric as:

$$N(t)_i = C_{Avg}(t)_i / C_{Max}(t)_i + M_{Avg}(t)_i / M_{Max}(t)_i + IO(t)_i / T(t)_i \tag{2}$$

where $N(t)_i$ is the resource utilization of server node i in t period time. $C_{Avg}(t)$ is the average utilization of CPU of server node i in t period time. $M_{Avg}(t)$ is the average of memory utilization of server node i in period t. $IO(t)$ is the reading and writing time of server node i in time t period time. Each node monitors their own utilization periodically and decides if they need to balance the loading or not.

3.6 Replication

The purpose of data replication in distributed systems is to provide a robust and consistent data access rate, and improve application performance. According to (Zeng et al., 2017), there are several algorithm strategies for data replication, such as SB-MFA, the larger data set has more replicas, or IB-MFA, more important data set has more replicas. In our system, we adjust the IB-MFA strategy and consider the ranking of data as the importance of data object to make decisions to replicate data to different server nodes. For example, data which

is highly accessed is replicated to more server nodes in the location which data is highly accessed. On the contrary, data which accessed less frequently is not be replicated and moved to the server node which has low system performance (see Algorithm 1). Moreover, the replication algorithm measures the data relation in order to help system to decrease latency and network traffic when many users or applications request the same data or similar types of data at the same time. The reason we consider the relation of data objects is that the same application can access the multiple data objects at the same time or in order. Therefore, the relation of data objects are keep tracked, and if a data object is replicated, the related data is also replicated to the same location or adjacent nodes. Because of replication, the system can easily recover data from other server nodes if correlated or non-correlated machine failures happen. In our system, data replication process would execute in the background during low loading time of system. Therefore, it will not occupy usage of bandwidth when users search data from system.

Algorithm 1 Data-Replication Algorithm.

Require:

$R(x), x \in$ data in Server Node

$N(y), y \in$ [hot, warm, cold]

```
1:  procedure
2:      for each Node_i i ∈ [1, n] do
3:          check rank of data in each server node
4:          for each x_j j ∈ [1, K] do
5:              if R(x_j) > Threshold_hot then
6:                  replicate data to N(hot) nodes
7:              else if R(x_j) > Threshold_warm then
8:                  replicate data to N(warm) nodes
9:              else
10:                 move data to low-performance node
11:                 deactivate nodes which have replicated data
12:             end if
13:         end for;
14:     end for
15: end procedure
```

4 Analysis

An extensive simulation has been designed according to previous descriptions to evaluate the performance of our system. We have used media data center searching scenario in order to present the effectiveness of the proposed prediction-based model by comparing with other based strategies such as popularity, location (geo-distance), and classification. We simulate the media data searching scenario with media files. For testing media searching, we download raw metadata from YouTube and obtain the data set based on our needs which include upload location, tag information, ranks, upload time and so forth. Then, we also generate servers with their locations. The location information is used to calculate the distance between

servers from the location of data access requests. Each URL would be related a least one data tag for searching to simulate the data relation between multiple tags. For the simulation, all server nodes can have different CPU, Memory, and Storage capacities. However, for simplicity, we assume that the servers have enough capacity for replications. After the system stores the uploaded data to server nodes according to different strategies, we evaluate the cost time by random searching sets. We consider the response time as a measurement to test the data availability because of bandwidth limitation and the server reply time for each request (Botero et al., 2012; Kim et al., 2015). We calculate the response time by considering the number of hops which is required to send the requested data (according to delay and the distance between the node locations of the data object in inner tiers). There are a number of different data objects. The number of server nodes is changing between 100 and 1000 to test our proposed method under small and large data center. We assume that the servers are located in 30 different areas, but data objects can be accessed from a number of locations.

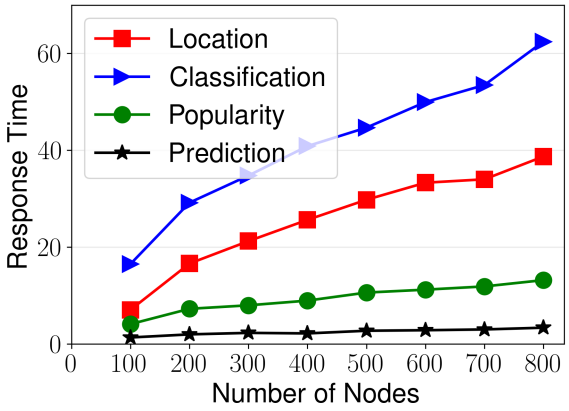

Figure 5: The response times of the methods while randomly requesting 200 times 50 different data objects from servers.

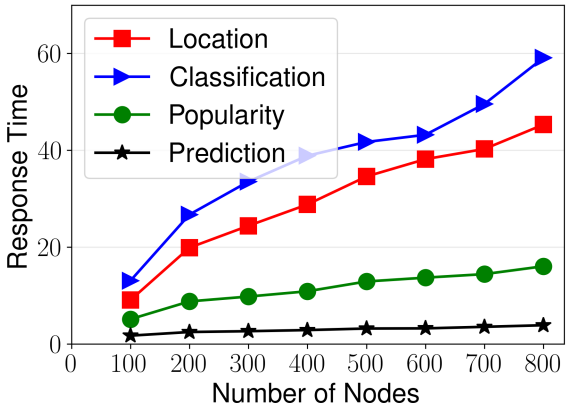

Figure 6: The response times of the methods while randomly requesting 200 times 100 different data objects from servers.

In order to analyze the performance of our system, we compare our prediction-based method with other strategies. We designed four different management approaches with the same network architecture. The first is the classification data management (Ito et al., 2016). The classification type methods allocate data to different servers based on their relations. The similar types of data objects are located to close nodes. The second is the location-based data management. This type of methods allocates the data based on the distance between data and servers. For example, data object A can be allocated in server node K if the location of the data object is close to server node K. In the third version, the data objects are managed according to the popularity (Ananthanarayanan et al., 2011) (see Section 3.4). For example, highly accessed data objects in location A can be placed in the server node, which is close to location A. The final is our proposed method which combines popularity and prediction to manage a data center. We simulate a different number of server nodes of four cases and generate random and expected data access request sets to evaluate the effectiveness of our proposed method.

5 Results

In this section, we show the experiment results under different cases based on response times of the methods. The results which are obtained here are as a result of 200 realizations.

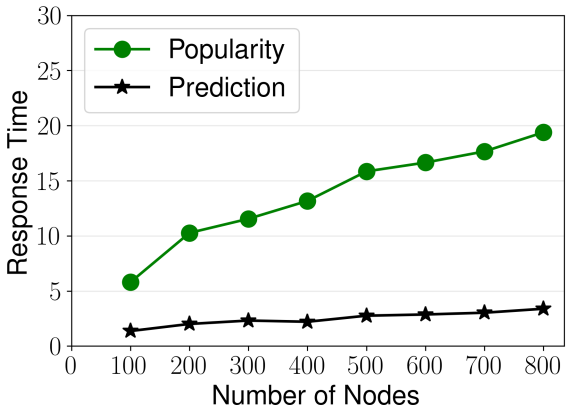

Figure 7: Popularity and prediction response times while requesting 200 times 50 different data objects from servers.

Figures 5 and 6 show the average response times of four different methods as a result of 200 times random 50 and 100 data object requests, respectively. The results in Figure 5 show the response time order as Classification > Location > Popularity > Prediction. The prediction has lower latency comparing to other algorithms because Prediction method estimates the future data request before the requests are made. Moreover, the gap between methods is getting larger while the number of nodes is increased. Figure 6 shows that response time of Prediction method is still lower than others even though the data object requests are doubled. Again, the gap between Prediction and other methods is getting larger.

Figures 7 and 8 show the average response times of Popularity and Prediction methods as a result of 200 times 50 and 100 data object requests, respectively. However, the requests

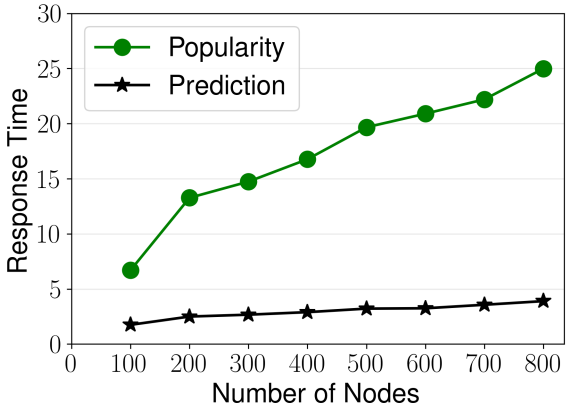

Figure 8: Popularity and prediction response times while requesting 200 times 100 different data objects from servers.

are not made randomly but according to Popularity and Prediction assumptions. In both Figures 7 and 8, Prediction method has lower response time than Popularity. Comparing to Figures 5 and 6 with Figures 7 and 8, the response time is decreased because the requests are made accordingly.

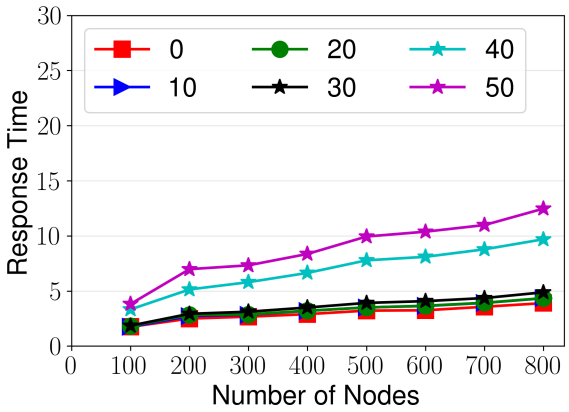

Figure 9: Prediction response times while requesting 200 times 100 different data objects from servers based on different error rates.

In addition to above results, we evaluate our proposed method based on prediction errors to show the reliability. Figure 9 shows Prediction response times while requesting 200 times 50 and 100 different data objects, respectively from servers based on different error rates. As shown in Figure 9, the response time of Prediction maintain in a small range even though the input prediction is not perfect. It indicates that the performance of the system is not affected much when the prediction has an error upto 30%. However, while the number of the requests are increased, 40% and 50% prediction error rates are getting higher.

6 Conclusion and Future Works

In this paper, we propose a media data center management method to increase data availability. To increase the data availability by using a replication technique for a media data center, the proposed method can dynamically adapt servers of data center by considering popularity prediction in each data-access location. We first label data objects from one to ten to track access frequencies of data objects. Then, we use those data access frequencies from each location to predict the future access frequencies of data objects to determine the replication levels and locations to replicate the data objects, and store the related data objects to close servers which are determined according to their CPUs, Memories, and storages. In order to show the efficiency of our proposed methods, we conduct an extensive simulation by using real data. We compare our proposed method with Popularity, Location (geo-distance), and Classification based strategies. The results show that our proposed method has an advantage over the other strategies in terms of data availability and increases data availability upto 50%. Our proposed method and related analysis can assist multimedia service providers to enhance their service qualities.

Our future plan is to extend this work by investigating the relation of data further to improve the performance. Although storage is not expensive, we also want show the cost of storage because of replications. Therefore, in our future work, the storage cost analysis with data relation will be investigated with our proposed prediction method.

References

Ananthanarayanan, G., Agarwal, S., Kandula, S., Greenberg, A., Stoica, I., Harlan, D., & Harris, E. (2011). Scarlett: Coping with skewed content popularity inmapreduce clusters. In *Proc. of eurosys.* Salzburg.

Arzuaga, E., & Kaeli, D. R. (2010). Quantifying load imbalance on virtualized enterprise servers. In *Proceedings of the first joint wosp/sipew international conference on performance engineering* (pp. 235–242).

Bharti, S., & Pattanaik, K. K. (2013). Dynamic distributed flow scheduling with load balancing for data center networks. *Procedia Computer Science*, *19*, 124–130.

Botero, J. F., Hesselbach, X., Fischer, A., & De Meer, H. (2012). Optimal mapping of virtual networks with hidden hops. *Telecommunication Systems*, *51*(4), 273–282.

Brandon, H., Srinivasan, S., Priya, M., Yiannis, Y., Puneet, S., Sujata, B., & Nick, M. (2010). Elastictree: Saving energy in data center networks. In *Proceedings of the 7th usenix conference on networked systems design and implementation* (pp. 249–264).

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., ... Vogels, W. (2007). Dynamo: amazon's highly available key-value store. *ACM SIGOPS operating systems review*, *41*(6), 205–220.

Goswami, A., Pattanaik, K. K., Bharadwaj, A., & Bharti, S. (2014). Loss rate control mechanism for fan-in-burst traffic in data center network. *Procedia Computer Science*, *32*, 125–132.

Hieu, N. T., Di Francesco, M., & Jääski, A. Y. (2014). A virtual machine placement algorithm for balanced resource utilization in cloud data centers. In *Cloud computing (cloud), 2014 ieee 7th international conference on* (pp. 474–481).

(n.d.). Retrieved from https://www.youtube.com/

Ito, D., Mohri, M., Shiraishi, Y., & Morii, M. (2016). Cloud storage with key-value stores over content-centric networking architecture. In *Ubiquitous computing, electronics & mobile communication conference (uemcon), ieee annual* (pp. 1–6).

Keller, E., Szefer, J., Rexford, J., & Lee, R. B. (2010). Nohype: Virtualized cloud infrastructure without the virtualization. In *Proc. of isca.*

Kim, H.-S., Paek, J., & Bahk, S. (2015). Qu-rpl: Queue utilization based rpl for load balancing in large scale industrial applications. In *Sensing, communication, and networking (secon), 2015 12th annual ieee international conference on* (pp. 265–273).

Lin, M., Wierman, A., Andrew, L. L., & Thereska, E. (2013). Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking (TON)*, *21*(5), 1378–1391.

Meng, X., Pappas, V., & Zhang, L. (2010). Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Infocom, 2010 proceedings ieee* (pp. 1–9).

Mustafa, I. B., & Nadeem, T. (2015). Dynamic traffic shaping technique for http adaptive video streaming using software defined networks. In *Sensing, communication, and networking (secon), 2015 12th annual ieee international conference on* (pp. 178–180).

Mwinyi, I. H., Narman, H. S., Fang, K.-C., & Yoo, W.-S. (2017, October). *Recommendation system based on predictive approach* (Tech. Rep.). Retrieved from http://hsnarman.oucreate.com/TR/17-Predictive-TR-MU-CITE-17-100.pdf

Nagendram, S., Lakshmi, J. V., Rao, D., & Jyothihi, C. (2011). Efficient resource scheduling in data centers using mris. *Indian Journal of Computer Science and Engineering*, *2*(5), 764–769.

Plakunov, A., & Kostenko, V. (2014). Data center resource mapping algorithm based on the ant colony optimization. In *Science and technology conference (modern networking technologies)(monetec), 2014 first international* (pp. 1–6).

Shi, L., Shi, Y., Wei, X., Ding, X., & Wei, Z. (2017). Cost minimization algorithms for data center management. *IEEE Transactions on Parallel and Distributed Systems*, *28*(1), 60–71.

Vdovin, P., & Kostenko, V. (2014). Algorithm for resource allocation in data centers with independent schedulers for different types of resources. *Journal of Computer and Systems Sciences International*, *53*(6), 854–866.

Zeng, L., Xu, S., Wang, Y., Kent, K. B., Bremner, D., & Xu, C. (2017). Toward cost-effective replica placements in cloud storage systems with qos-awareness. *Software: Practice and Experience*, *47*(6), 813–829.