A Novel Scheme for Cost Reduction and Data Availability in Heterogeneous Multi-Cloud

Yogitha C
M.Tech, CSE, VTU,
Don Bosco Institute of Technology,
Bengaluru,India.

Yasahaswini B M
Assistant Professor, VTU,
Don Bosco Institute of Technology,
Bengaluru,India

Abstract— **These days, more endeavors and associations are facilitating their information into the cloud, so as to decrease the IT support cost and upgrade the information unwavering quality. Be that as it may, confronting the various cloud merchants and their heterogeneous valuing arrangements, clients may well be astounded with which cloud(s) are suitable for putting away their information and what facilitating system is less expensive. The general business as usual is that clients for the most part put their information into a solitary cloud (which is liable to the seller lock-in danger) and afterward basically trust to good fortune. In view of far reaching investigation of different best in class cloud merchants, this paper proposes a novel information facilitating plan (named CHARM) which coordinates two key capacities wanted. The first is selecting a few suitable mists and a proper excess procedure to store information with minimized fiscal expense and ensured accessibility. The second is setting off a move procedure to re-disperse information as indicated by the varieties of information access example and estimating of mists. We assess the execution of CHARM utilizing both follow driven reproductions and model investigations. The outcomes demonstrate that contrasted and the major existing plans, CHARM spares around 20% of financial expense as well as displays sound versatility to information and value conformities.**

Index Terms—Multi-cloud; data hosting; cloud storage.

I. INTRODUCTION

Late years have seen a “dash for unheard of wealth” of online information facilitating benefits (or say distributed storage administrations, for example, Amazon S3, Windows Azure, Google Cloud Storage, Aliyun OSS, et cetera. These administrations furnish clients with dependable, versatile, and minimal effort information facilitating usefulness. More ventures and associations are facilitating all or a portion of their information into the cloud, to diminish the IT upkeep cost (counting the equipment, programming, and operational cost) and improve the information unwavering quality. For instance, the United States Library of Congress had moved its digitized substance to the cloud, trailed by the New York Public Library and Biodiversity Heritage Library. Presently they just need to pay for precisely the amount they have utilized. Heterogeneous mists. Existing mists show awesome heterogeneities as far as both working exhibitions and estimating strategies. Distinctive cloud sellers manufacture their individual bases and continue updating them with recently rising apparatuses. They likewise plan diverse framework designs and apply different procedures to make their administrations aggressive. Such framework assorted qualities prompts detectable execution varieties crosswise over cloud merchants. In addition, estimating arrangements of existing stockpiling administrations gave by various cloud sellers are unmistakable in both valuing levels and charging things. For example, Rack space does not charge for Web operations (regularly by means of a progression of REST ful APIs), Google Cloud Storage charges more for data transfer capacity utilization, while Amazon S3 charges more for storage room (allude to x II-A). Seller lock-in danger. Confronting various cloud sellers and in addition their heterogeneous exhibitions/arrangements, clients might be confused with which cloud(s) are suitable for putting away their information and what facilitating methodology is less expensive. The general business as usual is that clients as a rule put their information into a solitary cloud and afterward just trust to good fortune. This is liable to the purported "merchant lock-in danger", since clients would be gone up against with a quandary on the off chance that they need to change to other cloud sellers. The seller lock-in danger first lies in that information relocation unavoidably creates extensive cost. For instance, moving 100 TB of information from Amazon S3 (California datacenter) to Aliyun OSS (Beijing datacenter) would devour as much as 12,300 (US) dollars. Also, the merchant lock-in danger makes clients experience the ill effects of value changes of cloud sellers which are not unprecedented. For instance, the variance of power bills in a locale will influence the costs of cloud administrations in this district. We see that goliath cloud sellers like Windows Azure and Google Cloud Storage have been changing their evaluating terms. Surprising liquidation of cloud merchants further irritates the circumstance. Nirvanix, which has a large number of clients including main 500 organizations, all of a sudden close down its distributed storage administration in Sep. 2013 [9]. Ubuntu One, likewise a well known player in the business sector of distributed storage administration, got away in Apr. 2014 [10]. So unmistakably, it is rash for an endeavor or an association to host all information in a solitary cloud — "your most logical option is likely not to put all your investments tied up on one place. Finally, uncontrolled information accessibility is (it could be said) another sort of seller lock-in danger. In spite of the fact that the administration quality is formally ensured by administration level assentions (SLA), disappointments and blackouts do happen. Almost all the major cloud vendors experienced service outages in recent years. Some outages even lasted for several hours. Multi-cloud data hosting. Recently, multi-cloud data hosting has received wide attention from researchers, customers, and startups. The basic principle of multi-cloud (data hosting) is to distribute data across multiple clouds to gain enhanced redundancy and prevent the vendor lock-in risk, as shown

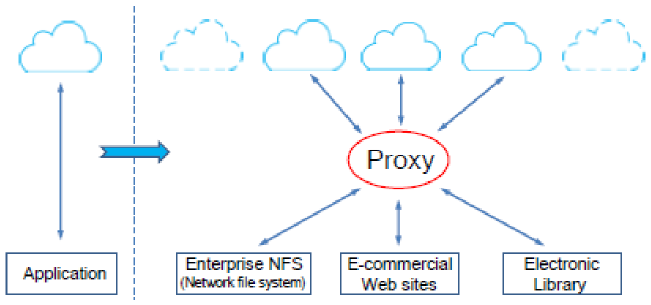

Fig. 1. Basic principle of multi-cloud data hosting.

in Fig. 1. The "proxy" component plays a key role by redirecting requests from client applications and coordinating data distribution among multiple clouds. The potential prevalence of multi-cloud is illustrated in three folds. First, there have been a few researches conducted on multi-cloud. DepSky guarantees data availability and security based on multiple clouds, thus allowing critical data (e.g., medical and financial data) to be trustingly stored. RACS deploys erasure coding among different clouds in order to prevent vender lock-in risk and reduce monetary cost. Second, new types of cloud vendors (e.g., DuraCloud and Cloud Foundry) have emerged and rapidly grown up to provide real services based on multiple clouds. Third, new development tools like Apache libcloud provide a unified interface above different clouds, which facilitates migrating services among clouds. Nevertheless, as for multi-cloud people still encounter the two critical problems: (1) How to choose appropriate clouds to minimize monetary cost in the presence of heterogenous pricing policies? (2) How to meet the different availability requirements of different services? As to monetary cost, it mainly depends on the data-level usage, particularly storage capacity consumption and network bandwidth consumption. As to availability requirement, the major concern lies in which redundancy mechanism (i.e., replication or erasure coding) is more economical based on specific data access patterns. In other words, here the fundamental challenge is: How to combine the two mechanisms elegantly so as to greatly reduce monetary cost and meanwhile guarantee required availability? The proposed CHARM scheme. In this paper, we propose a novel cost-efficient data hosting scheme with high availability in heterogeneous multi-cloud, named "CHARM". It intelligently puts data into multiple clouds with minimized monetary cost and guaranteed availability. Specifically, we combine the two widely used redundancy mechanisms, i.e., replication and erasure coding, into a uniform model to meet the required availability in the presence of different data access patterns. Next, we design an efficient heuristic-based algorithm to choose proper data storage modes (involving both clouds and redundancy mechanisms). Moreover, we implement the necessary procedure for storage mode transition (for efficiently re-distributing data) by monitoring the variations of data access patterns and pricing policies. We evaluate the performance of CHARM using both trace driven simulations and prototype experiments.

Summary of contribution. At last, our contributions in this paper can be briefly summarized as follows:

1) We propose and implement CHARM, a novel, efficient, and heuristic-based data hosting scheme for heterogeneous multi-cloud environments. CHARM accommodates different pricing strategies, availability requirements, and data access patterns. It selects suitable clouds and an appropriate redundancy strategy to store data with minimized monetary cost and guaranteed availability.
2) We design and implement a flexible transition scheme for CHARM. It keeps monitoring the variations of pricing policies and data access patterns, and adaptively triggers the transition process between different data storage modes. It also starts a data migration process among different clouds if necessary.
3) We evaluate the performance of CHARM using two typical real-world traces and prototype experiments. Both trace driven simulation and experiment results confirm the efficacy of CHARM..

II. BACKGROUND

A. Pricing Models of Mainstream Clouds

In order to understand the pricing models of mainstream cloud vendors, we select to study five most popular cloud storage services across the world: Amazon S3, Windows Azure, Google Cloud Storage, Rackspace, and Aliyun OSS (deployed in China). Their latest pricing models (in 2014) are presented in Table I (Storage and bandwidth pricing) and Table II (Operation pricing). Basically for these clouds, customers are charged in terms of storage, out-going (i.e., from cloud to client) bandwidth 1, and operations (such as PUT, GET, and LIST). However, each vendor's pricing model has some difference from the others. For instance, in Asia Amazon S3 has lower bandwidth price and higher storage price than Google Cloud Storage. Aliyun OSS provides the lowest bandwidth price, but its storage price is still higher than Google Cloud Storage. Besides, prices of operations are also different across different clouds.

B. *Erasure Coding*

Erasure coding has been widely applied in storage systems in order to provide high availability and reliability while introducing low storage overhead [4]. As we all know, the storage mode of "three replicas" is putting replicas into three different storage nodes. Then the data is lost only when the three nodes all crash. However, it occupies 2x more storage space. Erasure coding is proposed to reduce storage consumption greatly while guaranteeing the same or higher level of data reliability. A representative erasure-coding scheme is the so-called "Reed-Solomon code", which is a type of Maximum Distance Separable erasure coding. Considering a storage system with M nodes, we divide data into blocks of equal size, and each block is further divided into m equal-sized data chunks. After that, we encode the m chunks into n m parity chunks and put the total n chunks into different nodes (n _ M). We use RS(m, n) to denote this coding scheme and we call the n chunks a "segment".

III. A NEW OPPORTUNITY IN MULTI-CLOUD STORAGE

In this section, from a quantitative perspective, we demonstrate that there is still plenty of space for optimizing the multi-cloud data hosting by combining the two widely used redundancy mechanisms, i.e., replication and erasure coding.

A. *Combining Replication and Erasure Coding*

In existing industrial data hosting systems, data availability (and reliability) are usually guaranteed by replication or erasure coding. In the multi-cloud scenario, we also use them to meet different availability requirements, but the implementation is different. For replication, replicas are put into several clouds, and a read access is only served (unless this cloud is unavailable then) by the "cheapest" cloud that charges minimal for out-going bandwidth and GET operation. For erasure coding, data is encoded into n blocks including m data blocks and n□m coding blocks, and these blocks are put into n different clouds. In this case, though data availability can be guaranteed with lower storage space (compared with replication), a read access has to be served by multiple clouds that store the corresponding data blocks. Consequently, erasure coding cannot make full use of the cheapest cloud as what replication does. Still worse, this shortcoming will be amplified in the multi-cloud scenario where bandwidth is generally (much) more expensive than storage space.

B.Comparison of Data Hosting Modes

The traditional view of replication and erasure coding [5], [6] does not hold in the multi-cloud scenario. For example, the biggest preponderance of erasure coding lies in much less storage space for guaranteed high availability. However, this preponderance shrinks because of the clouds' pricing policies — bandwidth is (much) more expensive than storage space. For the same reason, in the multi-cloud scenario replication regains its competitiveness, though it is traditionally regarded as inferior to erasure coding in terms of storage saving. Therefore, it is difficult now to determine which mechanism is better in the presence of complex workload patterns and various pricing policies. Below we compare the two mechanisms quantitatively to shed light on this problem.

As shown in Figure 3, replication almost always outperforms erasure coding in the multi-cloud scenario. When the read count is 30, replication can save 32% monetary cost compared with erasure coding. The advantage of replication originates from the cloud with the lowest bandwidth price. In general, when the read frequency is high, the bigger the gap between the lowest bandwidth price and the average one is, the greater the superiority of replication is

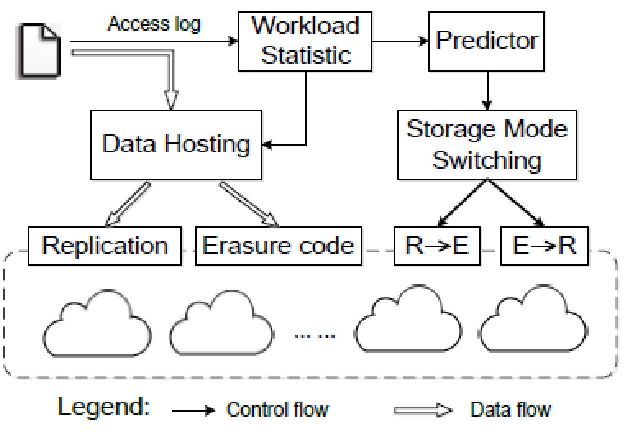

Fig.2. The architecture of CHARM. "R" represents replication and "E" represents erasure coding.

effect of erasure coding to a similar level as that of replication. On the other hand, significant monetary saving can be achieved if we combine their advantages elegantly. This is why we propose the novel data hosting scheme CHARM which will be elaborated in the following sections.

IV. PROPOSED METHODOLOGY

A. *CHARM Overview*

In this section, we elaborate a cost-efficient data hosting model with high availability in heterogeneous multi-cloud, named "CHARM". The architecture of CHARM is shown in Figure 3. The whole model is located in the proxy in Figure 1. There are four main components in CHARM: Data Hosting, Storage Mode Switching (SMS), Workload Statistic, and Predictor.

Workload Statistic: keeps collecting and tackling access logs to guide the placement of data. It also sends statistic information to Predictor which guides the action of SMS.

Data Hosting: stores data using replication or erasure coding, according to the size and access frequency of the data.

SMS decides whether the storage mode of certain data should be changed from replication to erasure coding or in reverse, according to the output of Predictor. The implementation of changing storage mode runs in the background, in order not to impact online service.

Predictor

Is used to predict the future access frequency of files. The time interval for prediction is one month, that is, we use the former months to predict access frequency of files in the next month.

However, we do not put emphasis on the design of predictor, because there have been lots of good algorithms for prediction. Moreover, a very simple predictor, which uses the weighted moving average approach, works well in our data hosting model. Data Hosting and SMS are two important modules in CHARM. Data Hosting decides storage mode and the clouds that the data should be stored in. This is a complex integer programming problem demonstrated in the following subsections. Then we illustrate how SMS works in detail in x

V, that is, when and how many times should the transition be implemented.

B. *Formal Definition of Data Hosting Model*

We first formally define the mathematical model applied in Data Hosting. When talking about erasure coding, we usually mean m > 1 (not replication). However, replication is a special case of erasure coding (i.e., m = 1). So we combine the two storage mechanisms and define a unified model. Assuming we have N clouds that meet performance requirements. We choose n cloud to store a file, the file should be encoded into n blocks of equal size (n _ N), including m data blocks and n□m coding blocks. If m = 1, the n□m coding blocks are the same with the data block, i.e., replication. Then the n blocks are distributed into the n clouds. We call a (m; n) pair with its corresponding clouds a storage mode. We first formally define the availability of a (m; n) pair. For the n clouds, which one stores data block, and which one stores coding block do not impact the availability. It is only impacted by the value of m.

C. *Heuristic Solution*

The key idea of this heuristic algorithm can be described as follows: We first assign each cloud a value _i which is calculated based on four factors (i.e., availability, storage, bandwidth, and operation prices) to indicate the preference of a cloud. We choose the most preferred n clouds, and then heuristically exchange the cloud in the preferred set with the cloud in the complementary set to search better solution. This is similar to the idea of Kernighan-Lin heuristic algorithm [30], which is applied to effectively partition graphs to minimize the sum of the costs on all edges cut. The preference of a cloud is impacted by the four factors, and they have different weights. The availability is the higher the better, and the price is the lower the better. So we use _i = _ai + _ Pi as the preference of the ith cloud, where Pi is the synthetical price of storage, bandwidth, and operation. Intuitively, if a file has much read access, the cloud with lower bandwidth price is more preferred. If a file is very small, operation price occupies a big proportion. So we let Pi = SPsi +crSPbi +crPoi. Specifically, ai and Pi are both normalized into (0; 1).

To find out optimal n and m, we first traverse n from 2 to _, where _ is the upper limit of n and _ < N. We do not set _ to N for two reasons: reality and complexity. For reality, n tends to be small in practice, usually less than 10. It has much higher probability for large n to induce degraded performance. More specifically, if a cloud becomes unavailable, the proxy has to get corresponding data from other m clouds (m is usually close to n), which determines that n cannot be very large in order to achieve good performance. For complexity, calculating the availability of erasure coding (m; n) has very high complexity. We have to check the availability for every possible solution that is traversed. If we give an upper limit to n, the availability can be calculated in polynomial running time. Then we traverse m from 1 to n for each n. The availability is calculated using Eq. 2. If the availability meets the required value and the monetary cost is lower, we update Csm and (i.e., the set of the selected clouds). If the availability does not meet the required value, we exchange the cloud in the current set Gs with the one in the complementary set Gc, using a greedy method: Firstly, Gs is sorted by ai, and Gc is sorted by Pi. Then we try to exchange the cloud in Gs from the lowest ai, one by one, with the cloud which has the lowest Pi in Gc but higher availability than that cloud in Gs, until the availability meets the required value.

V. TRANSITION SCHEME

A. *Transition of Storage Modes*

Intuitively, when a file changes from “hot” to “cold”, we should change its storage mode. More specifically, when the read frequency of the file drops below or increases above a certain value, changing storage mode can save more money. The value is determined by the prices of clouds. Given the available clouds including their prices and availability, we can figure out the storage mode and the selected clouds with the input of file’s size and read count, using Algorithm 1. We calculate the storage modes for different file sizes and read counts, in order to get a storage mode table (see Figure 4 in x VII for an example). The table has two dimensions: file size and read count. There is one corresponding storage mode for each pair of file size and read count, but the storage modes are the same for many different pairs. There are explicit boundaries between different storage modes in the table. However, it does not mean we should change the storage mode once a file’s storage mode crosses the boundary, because the transition of storage mode also generates cost, which is definitely not negligible. Bandwidth is (much) more expensive than storage space for online storage services. The cost of one read access for a file can afford this file to be stored for around 4 months with no read access. Thus, we should be prudent to deal with storage mode transition. A good transition scheme can actually save large amount of money.

We first demonstrate the implementation of storage mode transition: the proxy gets the data from the clouds where the data is originally stored, and puts it into the newly selected clouds using new storage mode. The implementation consumes out-going bandwidth, in-going bandwidth, and several operations (i.e., GET, DELETE, and PUT). Since DELETE and ingoing bandwidth are free, the transition cost T is composed of out-going bandwidth, GET, and PUT. Out-going bandwidth is more expensive than storage, so we have to make sure that the cost of transition can be earned back by the new storage mode. That is, the following inequality has to be met:

$$Mf > Mp + T \qquad (11)$$

where Mf and Mp are the monetary cost of the previous storage mode and new storage mode respectively. They are both calculated using the read frequency provided by Predictor. Eq. 11 is impacted by the time period t. Since the storage cost is storing a file of size S for a time period t and cr is the read count during t, we should set t first in order to calculate Mf and Mp. So, Eq. 11 means the new storage mode will earn back the transition cost within the time period t (t equals 30 days in our experiments). We implement the transition for each one month, which also equals to the time period t.

We calculate the storage mode for each file using its predicted read frequency in the time interval t. If the storage mode is different from the previous one and it meets Eq. 11, we change the storage mode of this file. The storage mode table can be calculated in advance because it is only affected by the available clouds, their pricing policies, and availabilities. When deciding the storage mode for each file, we use the read frequency and the size of the file to look up the table for the corresponding storage mode. This table is re-calculated through Algorithm 1, only when availabilities and prices are modified, some clouds are kicked out due to performance issue, or new available clouds emerge. And the new table will be input into Algorithm 2 to accommodate these situations. Algorithm 2 shows the detailed transition process.

B. *Complexity*

Here we analyze the computational complexity of this algorithm. The two loops in line 4 and 11 are used to look up the table, the complexity of which can be approximately considered constant, since the table is small and has only limited number of values in each dimension. Specifically, since the table is split into several pieces, we only need to find out which piece the file belongs to. Transition cost in line 19 can also be calculated in constant time. Thus, the complexity of this algorithm is mainly the first loop, and the worst case complexity is O(Fn), where Fn is the number of files. In order to reduce the complexity further, we can classify files with similar access patterns into groups, and implement transition in the unit of group. This is out of the scope of this paper.

VI. EVALUATION

We conduct extensive simulations to evaluate the performance of our scheme. The simulations are driven by two typical real-world traces. We first briefly introduce the two collected traces and present the evaluating methodology, then show the performance of our scheme. At last, to make the results more convincing, we also implement the prototype experiments on top of four mainstream commercial clouds, the results of which prove the correctness of the simulations and the efficacy of CHARM.

A. *Datasets*

The two traces are collected from AmazingStore [20] and Corsair [21]. AmazingStore is a popular file storing and sharing platform in China. It has been deployed and maintained since April 2009, and has 10K log-in users everyday. The files in this system are mainly music and video. Corsair is a cloud storage system deployed at Tsinghua University, China. There had been already 19,892 registered users and 17.5 TB of data by September 2010. The files stored in this system have diverse types. We collected the trace of Amazing Store from January 1, 2012 to July 15, 2013 from four main servers. For Corsair the trace is collected from March to July 2010. Each line of the traces is a file access record which includes timestamp, file name, file size, and operation type (e.g., GET, PUT). The detailed properties of the two traces are shown in Table IV. We use 15 clouds in the experiments, and they all meet the requirement of performance. The prices of these clouds are configured referring to the prices of current famous clouds (e.g., Amazon S3, Windows Azure) and their data centers. We set the clouds' availability in the interval of [99:5%; 99:95%]

B. *Storage Mode Table*

We generate the storage mode table based on the 15 clouds guaranteeing 99.9999% availability. We use different file sizes varying from 1KB to 1GB and different read counts varying from 0 to 100 with the step of 0.1 to calculate their corresponding storage modes (using Algorithm 1). We get four different storage modes as shown in Figure 4 with gray levels from 1 to 4. We only plot the read count from 0 to 3, because the storage modes are the same (i.e., gray level 4) for the read count larger than 3 no matter how much the file's size is. When the file's size is larger than 1MB, the storage modes have explicit vertical boundaries with different read counts. That means, for large files, read count is the key to impact the storage mode. When the file's size drops below 1MB, the operation cost has more and more impact on the total cost. High read frequency (generating high bandwidth cost) gives advantages to replication mechanism (i.e., m = 1). So, similarly, high operation cost also gives advantages to replication mechanism when the file's size is small. That is why gray level 4 puts its feet into the region of lower read count and smaller file size. This storage mode table only depends on prices of the available clouds and required availability. If the prices change, the table will change accordingly, becoming a different one.

C. *Monetary Cost*

We set different availability levels from 99.99% to 99.99999%, and run the two traces applying the five schemes respectively. The total cost of CHARM includes storage/bandwidth/operation costs and transition cost. The results of AmazingStore trace are shown in Table V. Since the read count of files in AmazingStore trace is high (i.e., 39.9 onaverage in 575 days), RepGr is better than EraGr except the highest availability case. In order to guarantee high availability, RepGr has to store more replicas whose storage cost exceeds the saving on bandwidth. The cost of EraGr for 99.99% is higher than that in higher availability, because EraGr has to reduce m to get higher availability, and it happens to exclude the cloud with higher bandwidth cost. CHARM has the lowest cost, it reduces about 9.3%-23.1% compared to RepGr, and reduces about 19.3%-24.3% compared to EraGr. From the detailed monetary cost as shown in Table VII, we can see that CHARM spends a little more storage cost to achieve much lower bandwidth cost. The detailed monetary cost of other availability levels shows similar results. RepRa and EraRa select clouds randomly, so the cost does not show strictly increase with the increase of availability.

D. *Applying to Complex Request Pattern*

Clearly, RepGr usually performs better for AmazingStore trace while EraGr performs better for Corsair trace. CHARM combines the merits of the two schemes to achieve the best performance, since it picks different storage modes for the files with different access frequency, which determines great adaptation. Cache is a commonly used technique to relieve the burden of back-end storage, shaping the data access pattern that is actually served by the back-end storage. Since Amazing Store trace has higher read frequency, we use a cache to filter the trace to show that CHARM well applies to various access patterns. More specifically, we use LRU for this cache with the cache size varying from 1GB to 2000GB. Figure 6(b) shows the total number of requests received by the back-end storage after filtered by the cache. With the increase of the cache, read count drops quickly. Then we apply the five schemes to the filtered traces with 99.99999% availability.

VII. CONCLUSION

Cloud services are experiencing rapid development and the services based on multi-cloud also become prevailing. One of the most concerns, when moving services into clouds, is capital expenditure. So, in this paper, we design a novel storage scheme CHARM, which guides customers to distribute data among clouds cost-effectively. CHARM makes fine-grained decisions about which storage mode to use and which clouds to place data in. The evaluation proves the efficiency of CHARM.

REFERENCES

[1] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, “Donar: Decentralized Server Selection for Cloud Services,” 2010.
[2] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, “Optimizing Cost and Performance for Content Multihoming,” 2012.
[3] T. G. Papaioannou, N. Bonvin, and K. Aberer, “Scalia: An Adaptive Scheme for Efficient Multi-cloud Storage,” in SC. IEEE, 2012.
[4] J. S. Plank, “Erasure Codes for Storage Systems: A Brief Primer,” The Usenix Magazine, vol. 38, no. 6, pp. 44–50, 2013.
[5] J. S. Plank, K. M. Greenan, and E. L. Miller, “Screaming Fast Galois Field Arithmetic using Intel SIMD Instructions,” in FAST. ACM, 2013.
[6] H. Weatherspoon and J. D. Kubiatowicz, “Erasure Coding vs. Replication: A Quantitative Comparison,” in IPTPS. Springer, 2002.
[7] R. Rodrigues and B. Liskov, “High Availability in DHTs: Erasure Coding vs. Replication,” in IPTPS. Springer, 2005.
[8] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, “The maximum clique problem,” in Handbook of Combinatorial Optimization. Springer, 1999, pp. 1–74.
[9] B. W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs,” Bell System Technical Journal, vol. 49, no. 2, pp. 291–307, 1970.
[10] M. A. Shah, M. Baker, J. C. Mogul, R. Swaminathan et al., “Auditing to Keep Online Storage Services Honest,” in HotOS. ACM, 2007.
[11] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, “Heading Off Correlated Failures Through Independence-as-a-Service,” in OSDI. ACM, 2014.

AUTHORS BIOGRAPHY

Yogitha C received the B.E degree in Computer Science from VTU Karnataka in 2014, and currently she is a post graduate student pursuing M.Tech in Computer Science and Engineering from Don Bosco Institute of Technology kumbalgodu, Bengaluru under Visvesvaraya Technological University Karnataka.

Her main research interests include cloud computing and wireless sensor networks. she is currently doing her project in Cloud Computing.

Yasahaswini B M received the B.E degree in Computer Science from East West Institute of Technology ,VTU Karnataka, and currently she is a working as assistant professor in Don Bosco Institute of Technology Pursed M.Tech in Computer Networks and Engineering from Alpha college of engineering Bengaluru under Visvesvaraya Technological University Karnataka.

Her main research interests include computer networks and wireless sensor network.