CHARM: A COST-EFFICIENT MULTI-CLOUD DATA HOSTING SCHEME WITH HIGH AVAILABILITY

Sagar YS[1], Achyutha Prasad N[2]
[1]M.Tech Student, [2]Assistant Professor, Dept of CSE, SSIT, TUMAKURU

Abstract: Nowadays, more and more enterprises and organizations are hosting their data into the cloud, in order to reduce the IT maintenance cost and enhance the data reliability. However, facing the numerous cloud vendors as well as their heterogeneous pricing policies, customers may well be perplexed with which cloud(s) are suitable for storing their data and what hosting strategy is cheaper. The general status quo is that customers usually put their data into a single cloud (which is subject to the vendor lock-in risk) and then simply trust to luck. Based on comprehensive analysis of various state-of-the-art cloud vendors, this paper proposes a novel data hosting scheme (named CHARM) which integrates two key functions desired. The first is selecting several suitable clouds and an appropriate redundancy strategy to store data with minimized monetary cost and guaranteed availability. The second is triggering a transition process to re-distribute data according to the variations of data access pattern and pricing of clouds. We evaluate the performance of CHARM using both trace-driven simulations and prototype experiments. The results show that compared with the major existing schemes, CHARM not only saves around 20% of monetary cost but also exhibits sound adaptability to data and price adjustments.

Keywords: Multi-Cloud, Data Hosting, Cloud Storage.

I. INTRODUCTION

Recent years have witnessed a "gold rush" of online data hosting services (or says cloud storage services) such as Amazon S3, Windows Azure, Google Cloud Storage, Aliyun OSS [1], and so forth. These services provide customers with reliable, scalable, and low-cost data hosting functionality. More and more enterprises and organizations are hosting all or part of their data into the cloud, in order to reduce the IT maintenance cost (including the hardware, software, and operational cost) and enhance the data reliability [2], [3], [4]. For example, the United States Library of Congress had moved its digitized content to the cloud, followed by the New York Public Library and Biodiversity Heritage Library [5]. Now they only have to pay for exactly how much they have used.

Heterogenous clouds: Existing clouds exhibit great heterogeneities in terms of both working performances and pricing policies. Different cloud vendors build their Respective infrastructures and keep upgrading them with newly emerging gears. They also design different system architectures and apply various techniques to make their services competitive. Such system diversity leads to observable performance variations across cloud vendors [6].

Moreover, pricing policies of existing storage services provided by different cloud vendors are distinct in both pricing levels and charging items. For instance, Rack space does not charge for Web operations (typically via a series of Restful APIs), Google Cloud Storage charges more for bandwidth consumption, while Amazon S3 charges more for storage space

Vendor lock-in risk: Facing numerous cloud vendors as well as their heterogeneous performances/policies, customers may be perplexed with which cloud(s) are suitable for storing their data and what hosting strategy is cheaper. The general status quo is that customers usually put their data into a single cloud and then simply trust to luck. This is subject to the so-called "vendor lock-in risk", because customers would be confronted with a dilemma if they want to switch to other cloud venders. The vendor lock-in risk first lies in that data migration inevitably generates considerable expense. For example, moving 100 TB of data from Amazon S3 (California datacenter) to Aliyun OSS (Beijing datacenter) would consume as much as 12,300 (US) dollars. Besides, the vendor lock-in risk makes customers suffer from price adjustments of cloud vendors which are not uncommon. For example, the fluctuation of electricity bills in a region will affect the prices of cloud services in this region. We notice that giant cloud vendors like Windows Azure and Google Cloud Storage have been adjusting their pricing terms [7], [8]. Unexpected bankruptcy of cloud vendors further aggravates the situation. Nirvanix, which has thousands of customers including top 500 companies, suddenly shut down its cloud storage service in Sep. 2013 [9].

Ubuntu One, also a famous player in the market of cloud storage service, escaped in Apr. 2014 [10]. So clearly, it is unwise for an enterprise or an organization to host all data in a single cloud — "your best bet is probably not to put all your eggs in one basket." [11] Finally, uncontrolled data availability is (in a sense) another type of vendor lock-in risk. Though the service quality is formally guaranteed by service level agreements (SLA), failures and outages do occur. Almost all the major cloud vendors experienced service outages in recent years [12], [13], [14]. 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 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 [15]. 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 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

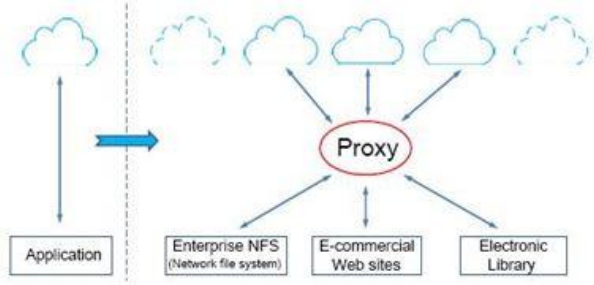

Fig. 1. Basic principle of multi-cloud data hosting.

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 heterogeneous 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 andmeanwhile guarantee required availability?

The proposed CHARM scheme: In this paper, we propose a novel cost-efficient data hosting scheme with high availability in heterogenous 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 redistributing data) by monitoring the variations of data access patterns and pricingpolicies. We evaluate the performance of CHARM using both trace- driven simulations and prototype experiments. The traces are collected from two online storage systems: Amazing Store and Corsair, both of Which possess hundreds of thousands of users. In the prototype experiments, we replay samples from the two traces for a whole month on top of four mainstream commercial clouds: Amazon S3, Windows Azure, Google Cloud Storage, and Aliyun OSS. Evaluation results show that compared with the major existing schemes.

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

• 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.

• 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.

• We evaluate the performance of CHARM using two typical real-world traces and prototype experiments. Both trace-driven simulation and experiment results confirmthe 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, Rack space, and Aliyun OSS (deployed in China). Their latest pricing models (in 2014) are presented Storage and bandwidth pricing and Operation pricing. Basically for these clouds, customers are charged in terms of storage, out-going (i.e., from cloud to client) bandwidth, 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. 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.

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.

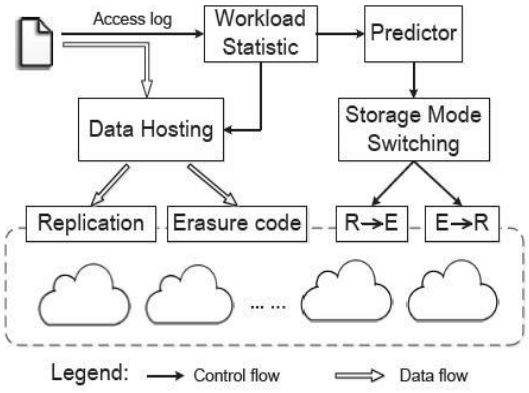

Fig. 3. The architecture of CHARM. "R" represents replication and "E" represents erasure coding.

B. Comparison of Data Hosting Modes

The traditional view of replication and erasure coding 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 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. space. For the same reason, in the multi-cloud scenario.

IV. DATA HOSTING SCHEME

A. CHARM Overview

In this section, we elaborate a **c**ost-efficient data **h**osting model with high **a**vailability in hete**r**ogenous **m**ulti-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 V, that is, when and how many times should the transition be implemented.

B. Formal Definition of Data Hosting M o d e l

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 ($n \leq N$), 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.

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 , 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. T

Algorithm 1: Heuristic algorithm of data placement

Input: file size S, read frequency c_r, n's upper limit ξ
Output: minimal cost C_{sm}, the set ψ of the selected clouds

```
1  C_sm ← inf; ψ ← {}
2  L_s ← sort clouds by normalized αa_i + β/P_i from high to slow
3  for n = 2 to ξ do
4      G_s ← the first n clouds of L_s
5      G_c ← L_s − G_s
6      for m = 1 to n do
7          A_cur ← calculate the availability of G_s
8          if A_cur ≥ A then
9              C_cur ← calculate the minimal cost
10             if C_cur < C_sm then
11                 C_sm ← C_cur
12                 ψ ← G_s
13             end
14         else
15             /*heuristically search better solution*/
16             G_s ← sort G_s by a_i from low to high
17             G_c ← sort G_c by P_i from low to high
18             for i = 1 to n do
19                 flag ← 0
20                 for j = 1 to N − n do
21                     if a_Gc[j] > a_Gs[i] then
22                         swap G_s[i] and G_c[j]
23                         flag ← 1
24                         break
25                     end
26                 end
27                 if flag = 0 then
28                     break
29                 end
30                 A_cur ← calculate the availability of G_s
31                 if A_cur ≥ A then
32                     C_cur ← calculate the minimal cost
33                     if C_cur < C_sm then
34                         C_sm ← C_cur
35                         ψ ← G_s
36                     end
37                     break
38                 end
39             end
40         end
41     end
42 end
43 return C_sm, ψ
```

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 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 opera tions (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. The storage mode 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.

Algorithm 2: Storage mode transition process

Input: the generated table Γ, the ith file's current storage mode $M[i]$, current read frequency $R[i]$, file size $S[i]$
Output: void

```
1  dSize ← the size dimension of Γ
2  dRead ← the read frequency dimension of Γ
3  for each file i do
4      for j in len(dSize) do
5          if S[i] ≥ dSize[j] then
6              dS ← j
7          else
8              break
9          end
10     end
11     for j in len(dRead) do
12         if R[i] ≥ dRead[j] then
13             dR ← j
14         else
15             break
16         end
17     end
18     if M[i] ≠ Γ[dS][dR] then
19         T ← monetary cost of transiting from M[i]
20         if M[i] > Γ[dS][dR] + T then
21             transit from M[i] to Γ[dS][dR]
22         end
23     end
24 end
```


VI. DISCUSSION

A. Performance of Multi-cloud

Lots of data centers are distributed around the world, and one region such as America, Asia, usually has several data centers belonging to the same or different cloud providers. So technically all the data centers can be access by a user in a certain region, but the user would experience different performance. The latency of some data centers is very low while that of some ones may be intolerable high. CHARM chooses clouds for storing data from all the available clouds which meet the performance requirement, that is, they can offer acceptable throughput and latency when they are not in outage. The storage mode transition does not impact the perfor- mance of the service. Since it is not a latency-sensitive process, we can decrease the priority of transition operations, and implement the transition in batch when the proxy has low workload.

B. Service Level Agreement (SLA) and Auditing

CHARM uses the availabilities declared in the SLAs of cloud services. However, SLA does not represent the real Mean Time Between Failures (MTBF) of the cloud service, that is, it does not represent the availability of the system directly. Violating SLA is allowed, and cloud vendors only need to pay "service credits" for the violation. So, for cloud vendors, there is a tradeoff between SLA and payment. In order to test the real availability, some works Propose approaches to audit the real performance of clouds. Some of them require the coordination of cloud vendors, such as providing specific service API, to verify the real availability. CHARM can also rely on the performance data from thirdparty auditing to make storage decision. A work in takes the first step to systematically detect correlations of clouds. We can take the output of their model into consideration to select the available clouds, for example, discarding the clouds which have high correlations.

C. Concern of Erasure Coding

The computational complexity of erasure coding is one of the most significant concerns, because it needs to implement lots of multiplication operations in Galois Field. The CPU resource may become the bottleneck of the applications and services which apply erasure coding. Recently, however, this is not the case for erasure coding scenarios any more, since we can leverage Intel SIMD Instructions to greatly increase the coding speed (the multiplication speed is as high as 8 GB/s using commodity CPU such as Intel Core i7-3770) . Thus, coding complexity can be addressed easily using commodity CPUs, making erasure coding more popular in storage systems Moreover, we give n an upper limit to guarantee high performance as described in § IV-C, which reduces the computational overhead further. How to set the upper limit is the problem that the real system developers have to deal with through real world measurements.

D. Other System Concerns

As a holistic storage system, there are several other factors to be considered, such as cache strategies, geographical data consistency, etc. However, we only focus on the data hosting strategy to minimize monetary cost while meeting flexible availability requirements. Though we have considered the complexity and feasibility when designing this strategy, the system design is out of the scope of this paper, and we put the detailed system design of multi-cloud data hosting into future work.

VII. 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 Amazing Store and Corsair. AmazingStore is a popular file storing and shar- ing 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 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. Methodology

We split the traces into pieces with the same time interval which is 30 days in our experiments. The pieces are put into CHARM one by one. CHARM reads the piece of the trace to get the files' size and current read count. Then it decides the storage mode, and calculates monetary cost for each file. We set the upper limit ξ to be 9 in CHARM.

C. 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 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.

D. 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. Since the read count of files in AmazingStore trace is high (i.e., 39.9 on average 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, 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.

E. Tolerating Price Adjustment

We adjust the price of clouds based on the fact that cloud providers have adjusted the price for several times [7], [8]. So when the simulation runs half of the traces (i.e., 9 months for AmazingStore, and 2 months for Corsair), we decrease and increase the price by 50% respectively to simulate the situation. Moreover, the prices that are modified include storage, bandwidth, and operation.

F. Supporting Varying Availability

Different types of data may require different availabilities. For example, backup data usually requires relatively low availability, while documents in work folders demand high availability. The experiments in § VII-D prove the effectiveness of CHARM in this scenario since it performs best for various availabilities. However, a more complicated use case is that the availability is varying with the access frequency of data. For example, "hot" data may demand high availability while "cold" data does not have that strict requirement. In order to show that CHARM can also naturally adapt to this scenario, we runthe two traces and assign different availabilities to the files according to their access frequency. More specifically, in our experiments 3 read requests a month is the boundary between high (99.99999%) and low (99.99%) availabilities. In order to avoid switching back and forth frequently, when the frequency drops below 1 request a month we change the availability from high to low, and when the frequency rises above 5 requests a month, the availability is changed from low to high. Setting these threshold values is reasonable since there is no strict boundary between different availabilities. The other schemes also use the same way to switch the availability.

VIII. RELATED WORK

With the blossom of cloud services , there is a recent interest in addressing how to migrate data and applications into clouds seamlessly . The system designed in migrates Network File System (NFS) into the cloud, and meanwhile makes it feel like working locally. A similar work in proposes a hybrid cloud-based deployment, where enterprise operations are partly hosted on-premise and partly in the cloud. Lots of works optimize the performance of the services from diverse aspects. The SCADS Director reconfigures the storage system on-the-fly, while guaranteeing strict per- formance Service-Level Objectives (SLOs) expressed using upper percentiles of request latency. addresses how to select the best combination of diverse storage devices (e.g., disks, SSDs, DRAM) to minimize cluster storage cost. Some

IX. 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.

X. REFERENCES

[1]"AliyunOSS(OpenStorageService)," http://www.aliyun.com/product/ oss.
[2]"Gartner:Top10cloudstorageproviders," http://www.networkworld.com/news/2013/010313-gartner-cloud-storage- 265459.html? page=1.
[3] Z. Li, C. Jin, T. Xu, C.Wilson,Y. Liu, L.Cheng, Y. Liu, Y. Dai, and Z.-L. Zhang, "Towards Network-level Efficiency for Cloud Storage Services," in IMC. ACM, 2014.
[4] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Y. Zhao, C. Jin, Z.-L. Zhang, and Y. Dai, "Efficient Batched Synchronization in Dropbox-like Cloud Storage Services," in Middleware. ACM/IFIP/USENIX, 2013.
[5] C. M. M. Erin Allen, "Library of Congress and DuraCloud Launch Pilot Program Using Cloud Technologies to Test Perpetual Access to Digital Content," The Library of Congress,News Releases,http://www.loc.gov/ today/pr/2009/09-140.html.
[6] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing Public Cloud Providers," in IMC. ACM, 2010.
[7]"Windows Azure pricing updates," http://azure.microsoft.com/en-us/ updates/azure-pricing-updates/.
[8]"Google Cloud Platform pricing updates," http:// google cloudplatform.blogspot.com/201403/google-cloud-platform-live-blending-iaas-and-paas-moores-law-for-the-cloud.html.
[9]"Its Official, The Nirvanix Cloud Storage Service Is Shut- ting Down," http://techcrunch.com/2013/09/27/its-official-the-nirvanix- cloud-storage-service-is-shutting-down/.
[10]"Shutting down Ubuntu One file services," http://blog.canonical.com/2014/04/02/ shutting-down- ubuntu-one-file-services/.
[11]"Nirvanix Provides Cautionary Tale For Cloud Storage," http://www.forbes.com/sites/tomcoughlin/2013/09/30/nirvanix-provides- cautionary- tail- for -cloud-storage/.

[12]"GoogleOutagesDamageCloudCredibility,"https://www.networkworld.com/news/2009/092409-google-outages-damage- cloud.html.
[13] "Rackspace to issue as much as $3.5M in customer credits after outage," http://www.networkworld.com/ news/ 2009/ 070609-rackspace- outage.html.
[14]"Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US EastRegion," http://aws. amazon. com/ cn/message/65648/.
[15] A. Bessani, M. Correia, B. Quaresma, F. Andre´, and P. Sousa, "DepSky: Dependable and Secure Storage in a Cloud-of-Clouds," in EuroSys. ACM, 2013.

