A Novel Resource Provisioning Model for DHT-Based Cloud Storage Systems

Jingya Zhou, Wen He

A Novel Resource Provisioning Model for DHT-based Cloud Storage Systems

Jingya Zhou[1,2] and Wen He[2]

[1] School of Computer Science and Technology, Soochow University, 215006 Suzhou, P.R. China
jyz@seu.edu.cn
[2] School of Computer Science and Engineering, Southeast University, 211189 Nanjing, P.R. China
wenhe@seu.edu.cn

Abstract. Cloud storage providers build a distributed storage system by utilizing cloud resources located in data centers. The interactions among servers in a DHT (Distributed Hash Table)-based cloud storage system depend on the routing process, and its execution logic is more complicated. Hence, how to allocate resources to not only guarantee service performance (*e.g.*, data availability, response delay), but also help service providers to reduce cost became a challenge. To address this challenge, this paper presents a novel resource provisioning model for cloud storage systems. The model utilizes queuing network for analysis of both service performance level and cost calculation. Then the problem is defined as a cost optimization with performance constrains, and a novel algorithm is proposed. Furthermore, we implemented a DHT-based storage system on top of an infrastructure platform built with OpenStack. Based on real-world traces collected from our system, we show that our model could effectively guarantee the target data availability and response delay with lower cost.

1 Introduction

Cloud storage utilizes cloud technologies to build storage systems based on IT resources located in datacenters, and provides customers with data storage, data sharing, data access and management, and so on. Recently cloud storage services have attracted more and more attention from both academia and industry [1][2]. One of the most attractive features of cloud storage is the ability to provide customers with convenient data access services without worrying about data loss. When customers use services they are mainly concern about data availability and response delay. The former represents the probability that customers can successfully access the target data, and the latter refers to the time required for the system to respond to requests. Both of them directly affect the service level that customers experienced, and become the preferred performance metrics discussed in this paper.

High amount of concurrent access requests is another feature of cloud storage, which makes storage providers choose to build distributed storage systems

based on P2P structure (*e.g.*, Dynamo [1], Cassandra [2]). It is a type of shared nothing architecture (SNA) [3], in which each server has its own disk for storage. DHT mechanism is responsible for both storing data to all servers and requests routing. It can provide an "always-on" experience as the continuous growth of system scale. Due to the distributed nature of systems, customers' requests need to be matched and forwarded among many servers after they arrive at systems. There are many interactions among servers during requests being processed. Different from multi-tier web applications, servers interact sequentially layer by layer according to the hierarchy, while the interactions occurred in cloud storage systems depend on the routing process and are more complicated. Hence how to model the relationship between service performance and resource provisioning becomes a challenge. In addition, cloud storage systems are based on the infrastructure services offered by IaaS providers, for example, Dropbox chooses IT resources that come from Amazon as its servers to store data and deal with requests [4]. Storage providers only pay for resources that are needed according to the current amount of access requests, which will reduce cost.

We explore the problem from the cloud storage provider's point of view. The overall cost paid by a storage provider mainly includes server cost, storage cost and traffic cost. As mentioned above, this paper mainly concerns access performance, so we make assumptions that all data have been stored in the system, and then the storage cost has been fixed. Most traffic is generated by both retrieving data from datacenter and geo-replication across multi-datacenter. Traffic cost caused by retrieving data can be reduced by data compression techniques, while traffic cost caused by geo-replication depends on the specific replication scheme. Both of them are outside the scope of this paper. Therefore, the cost discussed here mainly refers to server cost, and the final objective of resource provisioning is to generate the server level resource demands to minimize cost while satisfying performance requirements. In this paper we propose a novel model to achieve server level resource provisioning with optimal cost-performance trade-off. Our proposed model strives to rent just enough resources for systems to minimize resource waste, while avoiding performance degradation.

2 Related Work

Jing *et al.* [5] proposes a novel resource auto-scaling scheme that try to find the optimal number of VMs by modeling system as a M/M/m queue so as to achieve cost-latency trade off. However, It assume a simple scenario that VMs run independently, and the interactions among VMs are not considered completely. Ferretti *et al.* [6] designs a middleware architecture for resource management that aims to satisfying quality of service (QoS) requirements as well as optimizing resource utilization. It only provides a common framework for analysis, and does not optimize for addressing a specific execution logic.

For multi-tier applications, Jing *et al.* [7] focuses on how to minimize cost while satisfying response delay constraint. It employs a flexible hybrid queuing model that consists of one M/M/c queue and multiple M/M/1 queues to de-

termine the number of VMs at each tier. Different from layer-by-layer research ideas, Lama *et al.* [8] suggests employing fuzzy theory to guide server provisioning and designs a model-independent fuzzy controller, so as to minimize VMs while guarantee end-to-end response delay. The works in [7][8] are based on the assumption that VMs are identical, but usually IaaS providers provide various types of VMs. Furthermore, the assumption of single VM type results in coarse-grained resource provisioning and limited cost optimization.

Zhu *et al.* [9] creates a resource provisioning model by employing M/G/1 queuing system, and develop meta-heuristic solutions based on the mixed tabu-search optimization algorithm to solve the provisioning problem. It only take response delay into consideration, and focus on the maximization of IaaS provider's profit which is different from the goal of this paper. By considering budget constraint as well as response delay, Zhu *et al.* [10] presents a feedback control based dynamic resource provisioning algorithm for maximizing application QoS.

For cloud storage services, customers' requests usually should be matched and forwarded among many servers after they arrive at systems. The interactions among servers depend on the routing process, and do not be executed in accordance with the fixed order. Hence the interactions occurred in cloud storage systems are complicated and lack of an effective resource provisioning model for characterization. Zhang *et al.* [11] presents a resource management algorithm for cloud storage systems. The proposed algorithm aims to achieve load balancing by using two types of operation, i.e., merge operation and split operation. However, such an algorithm does not consider server interactions during the execution of services, and only consider load balancing as performance metrics. This paper explores the resource provisioning model based on the execution logic of cloud storage services. We consider two performance metrics in the model, *i.e.*, data availability and response delay, and strive to optimize cost as well as guarantee performance.

3 Resource Provisioning Model

3.1 System Model

Cloud storage systems run on the infrastructure of datacenter, and the system overview is described by Figure 1. Cloud storage providers rent servers from IaaS providers and build the system by organizing servers into a distributed network, so that the system can store massive data in a distributed manner. When customers' requests arrive at the system, they are dispatched to servers and will be processed according to DHT mechanism. It is assumed that the data have been stored in the system, and then the primary performance metrics concerned by customers should be data availability, denoted by P_{suc} which represents the probability that customers can successfully access the target data, and response delay, denoted by R which represents the time required for the system to respond to requests. This paper strives to research on resource provisioning from the cloud storage provider's point of view. Our problem is how to generate a resource provisioning demand according to the current customers' requests, so that

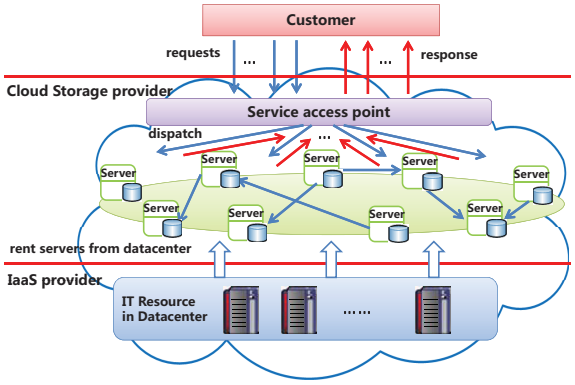

Fig. 1. Cloud storage system overview

it can meet performance metrics while optimize economic metrics. The resource provisioning demand consists of three parameters, *i.e.*, the number of servers, the processing capacity of each server, and cost.

3.2 Resource Provisioning Problem

To tackle the above problem, we need to establish a resource provisioning model. As we know, cloud storage system is distributed, when it receives requests it will dispatch them to servers randomly. The server matches the received requests with the data stored upon it. If match success, server will return results directly, otherwise server will forward requests to the next one according to DHT rules till finding the target data. Servers interact with each other through forwarding requests. For better describing this kind of interactions, we propose a resource provisioning model based on queuing network. As shown in Figure 2, the system consists of N servers, and each one is modeled as an M/G/1/k queue with independent general execution time distribution. The request arrivals are poisson with rate λ_1. Servers are classified and charged by processing capacity, *e.g.*, the processing capacity of server i is represented by μ_i ($\mu_{\min} \leq \mu_i \leq \mu_{\max}$), where $\mu_{\max}$, $\mu_{\min}$ are the upper bound and lower bound respectively. The cost of server is represented by the function of processing capacity $f(\mu_i)$. Due to the limit of processing capacity, server cannot simultaneously receive and process an unlimited number of requests. As requests increase, the length of queue becomes larger, which results in a higher response delay. To avoid high response delay, we set up a size limit k for each queue. When the length of queue reaches k, the workload of server will be saturated, and then the new arrived requests will be denied. Once a request is denied, the customer's access will fail, and as a consequence the data availability will decrease. The formal definition of resource provisioning problem is described as follows:

Given that the thresholds of performance metrics (*i.e.*, data availability and response delay) are P^*_{suc} and R^*, and the threshold of server rejection rate is

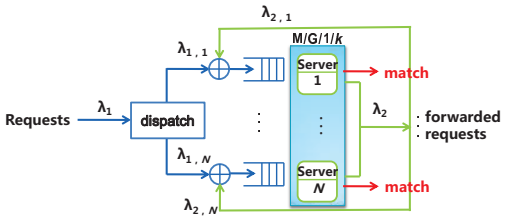

Fig. 2. Queuing network model for resource provisioning

P^*_{rej}. The server cost is $f(\mu_i)$ that is a non-decreasing function of μ_i, and the request arrival rate is λ_1. We need to generate the optimal resource provisioning demand (N, μ, Cost(μ)) that meets performance metrics while optimize server cost. In the demand, N represents the number of servers, μ represents the vector of server's processing capacities, and Cost(μ) is the sum of server cost. *i.e.*,

$$\begin{aligned} \text{Min} \quad & Cost(\mu) = \sum_{i=1}^{N} f(\mu_i) \\ \text{s.t.} \quad & (1)\ P_{suc} \geq P^*_{suc} \\ & (2)\ R \leq R^* \\ & (3)\ P_{rej} \leq P^*_{rej} \\ & (4)\ \mu_{\min} \leq \mu_i \leq \mu_{\max} \end{aligned} \tag{1}$$

In cloud storage systems, customers' requests can be satisfied within O($\log N$) hops forwarding according to DHT rules, so that the mean match rate at each hop is at least $1/(\log N + 1)$. Assume that the mean rejection rate of server is P_{rej}, then

$$P_{suc} = \sum_{j=0}^{\log N} A(j)B(j)\frac{j+1}{\log N + 1} \tag{2}$$

where $A(j) = (1 - P_{rej})^{j+1}$ represents the probability that the request arrives at the $j+1$th server after it finish j hops forwarding without being denied, while $B(j) = \prod_{m=0}^{j}(1 - \frac{m}{\log N+1})$ represents the probability that the request has

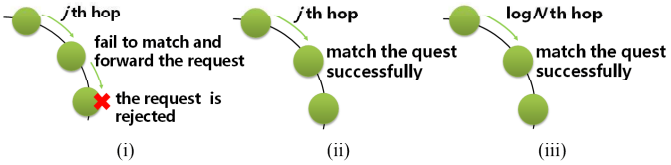

Fig. 3. An example of request forwarding

not been matched at previous j servers. The probability of being matched successfully at the $j+1$th server is $\frac{j+1}{\log N+1}$. Assume that the request stops at the

$j+1$th server, and then there exists three cases, as shown in Figure 3: (i) The request is not matched at the $j+1$th server. Then the request is forwarded to the $j+2$th server, and is denied by the server. The probability of such case should be $B(j+1)P_{rej}$. (ii) The request is matched successfully at the $j+1$th server. The probability of such case should be $B(j)\frac{j+1}{\log N+1}$. (iii) The request has arrived at the last hop, *i.e.*, $j = \log N$. The probability of such case should be $A(\log N)B(\log N)$. Combining the above cases, we conclude that the mean hop counts of request can be represented by

$$\begin{aligned} H = & \sum_{j=0}^{\log N-1} \left(A(j) \cdot B(j)((1-\tfrac{j+1}{\log N+1})P_{rej} + \tfrac{j+1}{\log N+1})j \right) \\ & + A(\log N)\mathrm{B}(\log N) \cdot \log N \end{aligned} \tag{3}$$

We can deduce the mean number of forwarded messages in the same way:

$$\begin{aligned} M = & \sum_{j=0}^{\log N-1} \left(A(j) \cdot B(j) \left((1-\tfrac{j+1}{\log N+1})P_{rej} \cdot (j+1) + \tfrac{j+1}{\log N+1} \cdot j \right) \right) \\ & + A(\log N)\mathrm{B}(\log N) \cdot \log N \end{aligned} \tag{4}$$

As described by Figure 2, the arrival rate of servers consists of both the requests λ_1 issued from customers and the forwarded requests λ_2, and $\lambda_2 = \lambda_1 M$. So the mean arrival rate can be calculated by λ_1/N. It is noted that the probability of receiving requests depends on the access frequency of data stored on server. $Q_i (0 < Q_i < 1)$ is used to represent the access frequency of server i, and then the arrival rate of forwarded requests at server i should be $\lambda_{2,i} = Q_i \lambda_2$. For server i, the arrival rate can be represented by

$$\lambda(i) = \lambda_{1,i} + \lambda_{2,i} \tag{5}$$

Response delay consists of two parts, *i.e.*, the mean time required to forward the request, denoted by T, and the mean sojourn time at a server, denoted by W. Thus the mean response delay should be

$$R = T \cdot H + W \cdot (H+1) \tag{6}$$

We should deduce the sojourn time by analyzing M/G/1/k queuing system. This paper chooses to use two-moment approximation approach [13] that is based on diffusion theory [14]. The key idea of approach is concluded that the discrete queuing process is approximated to a continuous diffusion process. The rejection rate of server i equals the probability of having k requests in the queue, *i.e.*,

$$\begin{aligned} P_{k,i} &= \frac{\rho_i^{(\Phi_i+2k)/(2+\Phi_i)}(\rho_i-1)}{\rho_i^{2(\Phi_i+k+1)/(2+\Phi_i)}-1} \\ \text{where } \Phi_i &= \sqrt{\rho_i e^{-s_i^2}}\, s_i^2 - \sqrt{\rho_i e^{-s_i^2}} \end{aligned} \tag{7}$$

$\rho_i = \lambda(i)/\mu_i$ represents the service intensity of server i, and s_i represents the coefficient of variation of the service process. The mean rejection rate is calculated by

$$P_{rej} = \frac{1}{N}\sum_{i=1}^{N} P_{k,i} \tag{8}$$

Because server may deny the incoming requests, the effective arrival rate at server i should be less than $\lambda(i)$, and can be represented by $\lambda_e(i) = \lambda(i)(1-P_{k,i})$. Thus the probability of empty workload at server i is given by

$$P_{0,i} = 1 - \frac{\lambda_e(i)}{\mu_i} = \frac{(\rho_i - 1)}{\rho_i^{2(\Phi_i+k+1)/(2+\Phi_i)} - 1} \tag{9}$$

The probability that there are j requests waiting in the queue of server i is $\rho_i^j P_{0,i}$, and then the mean number of requests waiting in the queue of server i should be

$$L_i = \sum_{j=0}^{k-1} j\rho_i^j P_{0,i} + kP_{k,i} \tag{10}$$

Based on Little's Formula [12], the sojourn time at server i is represented by $W_i = L_i/\lambda_e(i)$. Therefore, the mean sojourn time is given by

$$W = \frac{1}{N}\sum_{i=1}^{N} W_i \tag{11}$$

3.3 Solution

Recall that the cloud storage provider's greatest concern is to maximize profit (*e.g.*, by minimizing cost) while providing high quality service (*e.g.*, by guaranteeing data availability and response delay). The resource provisioning is defined as a non-linear cost optimization problem with performance constraints from the cloud storage provider's point of view. By solving the problem, we can achieve the optimal resource demands for system resource provisioning. The previous works only focus on the optimization of number of servers. However, in practice the minimal number of servers does not reflect the lowest cost. In our solution, we are not only trying to answer how many servers need to rent, but also answer what the capacities vector of these servers is. μ is used to represent the vector of server capacities, while N is the number of rented servers, and is also the dimension of capacities vector. We should determine the feasible range of N at the first step.

Substitute P_{suc}^* and P_{rej}^* in constraints (1) and (3) into equation (2), then we can obtain the value of N that satisfies constraints (1) and (3), denoted by N'. In the same way we can obtain the value of N, denoted by N'' that satisfies constraints (2) and (3) by substituting R^* and P_{rej}^* in constraints (2) and (3) into equation (6). It is noted that the server rejection rate $P_{k,i}$ is the non-increasing function of μ_i, then substitute $\mu_{\max}(\mu_{\min})$ in constraint (4) and $P_{k,i} = P_{rej}^*$ into equation (7), we can deduce the maximal (minimal) arrival rate $\lambda_{\max}(\lambda_{\min})$. Combine equations (4) and (5) together, we find that $\lambda(i)$ is related

to total customers' requests arrival rate λ_1, rejection rate P_{rej} and the number of servers N. By substituting λ_1, P^*_{rej} and $\lambda_{\max}(\lambda_{\min})$ into equation (5), we can achieve the feasible range of N , denoted by $[N_1, N_2]$ that satisfies the threshold of rejection rate. In order to satisfy all constraints, the feasible range should be trimmed by N' and N''. Proposition 1 shows us the proof of feasible range of N.

Algorithm Resource_provisioning

Input

λ_1: the total customers' requests arrival rate
$\mu_{\max}$: the upper bound of processing capacity
$\mu_{\min}$: the lower bound of processing capacity
Q_i: the access frequency of server i
T: the mean time required to forward the request

Output

Opt_solution$(N, \mu, Cost(\mu))$: the optimal resource demands

1. Calculate N' by subjecting P^*_{suc} and P^*_{rej} to equation (2);
2. Calculate N'' by subjecting R^* and P^*_{rej} to equation (6);
3. Calculate $\lambda_{\max}(\lambda_{\min})$ by subjecting $\mu_{\max}(\mu_{\min})$ and $P_{k,i} = P^*_{rej}$ to equation (7)
4. Calculate N_1 (N_2) by subjecting $\mu_{\max}(\mu_{\min})$, P^*_{rej} and λ_1 to equation (5)
5. $N_{\min} = \max(N', N_1, N'')$, $N_{\max} = \max(N', N_2)$;
6. Opt_solution $(N, \mu, Cost(\mu))$ = **NLP_OPT** $(N_{\min})$;
7. **if** $N_{\min} \neq N_{\max}$
8. **for** $N = N_{\min} + 1$to$N_{\max}$
9. solution $(N, \mu, Cost(\mu))$ = **NLP_OPT** (N);
10. **if** solution$(N, \mu, Cost(\mu))$ is better than Opt_solution$(N, \mu, Cost(\mu))$
11. Opt_solution $(N, \mu, Cost(\mu))$ = solution $(N, \mu, Cost(\mu))$;
12. **end if**
13. **end for**
14. **end if**
15. **return** Opt_solution $(N, \mu, Cost(\mu))$;

Proportion 1 *In the server provisioning problem, the feasible range of number of servers that satisfies all constraints is* $[\max(N', N_1, N''), \max(N', N_2)]$.

Proof: P_{rej} is a non-increasing function of N by analyzing equations (7) and (8). If $N' \leq N_1$, the lower bound of N, denoted by $N_{\min}$, takes the value of N_1 for satisfying constraint (3). Otherwise, $N_{\min}$ takes the value of N' for satisfying constraint (1). In addition, R is a non-increasing function of N by analyzing equation (6) (T is much smaller when compared with W). In order to satisfying constraints (2), $N_{\min}$ takes the value of N'', *i.e.*, $N_{\min} = \max(N', N_1, N'')$. Assume the optimal value $N^* < N_{\min}$, and then it will result in that one constraint or all of constraints cannot be satisfied. Therefore, N_1 should take the value of $\max(N', N_1, N'')$.

In the same way, if $N' \leq N_2$, the upper bound of N, denoted by $N_{\max}$, takes the value of N_2 for satisfying constraint (3). Otherwise, $N_{\max}$ takes the value of N' for satisfying constraint (1), *i.e.*, $N_{\max} = \max(N', N_2)$. Assume

the optimal value $N^* > N_{\max}$, and the corresponding optimal cost is $Cost^* = \sum_{i=1}^{N^*} f(\mu_i), (\mu_{\min} \leq \mu_i \leq \mu_{\max})$. Then all constraints can be satisfied, and N_2 is located in the feasible range. The corresponding cost $Cost\_N_2 = N_2 f(\mu_{\min})$, but $Cost\_N_2 < Cost^*$, which conflicts with the assumption. Therefore, $N_{\max}$ should take the value of $\max(N', N_2)$.

To solve the optimization problem, an novel algorithm is proposed, called **Resource_provisioning**. In the algorithm, lines 1-5 are used to compute the feasible range of N, and then for each N in the feasible range, lines 6-15 use **NLP_OPT(N)** to solve the sub-optimization problem with the fixed value of N.

There are non-linear functions in constraints, so that the sub-optimization problem is a non-linear programming problem which can be formalized as follows:

$$\begin{aligned} \text{Min} \quad & Cost(\mu) = \sum_{i=1}^{N} f(\mu_i) \\ \text{s.t.} \quad & (1)\ \mathrm{g}_1(\mu) = P^*_{suc} - P_{suc} \leq 0 \\ & (2)\ \mathrm{g}_2(\mu) = R - R^* \leq 0 \\ & (3)\ \mathrm{g}_3(\mu) = P_{rej} - P^*_{rej} \leq 0 \\ & (4)\ \mathrm{g}_4(\mu) = \mu_{\min} - \mu_i \leq 0 \\ & (5)\ \mathrm{g}_5(\mu) = \mu_i - \mu_{\max} \leq 0 \end{aligned} \tag{12}$$

In this paper we use augmented lagrangian approach to solve the problem. By introducing slack variable z_j, the inequality constraints become equality constraints, *i.e.*, $g_j(\mu) - z_j^2 = 0, j = 1, 2, 3, 4, 5$. We design the augmented lagrangian function, as follows:

$$F(\mu, \gamma, c) = Cost(\mu) + \frac{1}{2c} \sum_{j=1}^{5} \left\{ [\max\{0, \gamma_j + c g_j(\mu)\}]^2 - \gamma_j^2 \right\} \tag{13}$$

where γ is multiplier vector, c is penalty factor, and $z_j^2 = \frac{1}{c} \max\{0, \gamma_j + c g_j(\mu)\}$. Thus the problem is transformed into a simple unconstrained optimization problem, *i.e.*, Min $F(\mu, \gamma, c)$. The solution of non-linear programming can be obtained by iteratively solving unconstrained optimization problem.

4 Experimental Evaluation

4.1 Experiment setup

We implemented a DHT-based cloud storage system on top of project Voldemort which is an open source implementation of Dynamo. The infrastructure platform is constructed on top of a cluster of 14 IBM HS22 blade servers which are connected in a 1Gbps LAN. These underlying resources are managed by an infrastructure platform built with OpenStack. Servers in our system can be classified as control servers and storage servers. The former is in charge of dispatching requests to storage servers, recording run-time log, and performance statistics. The latter is in charge of processing the incoming requests.

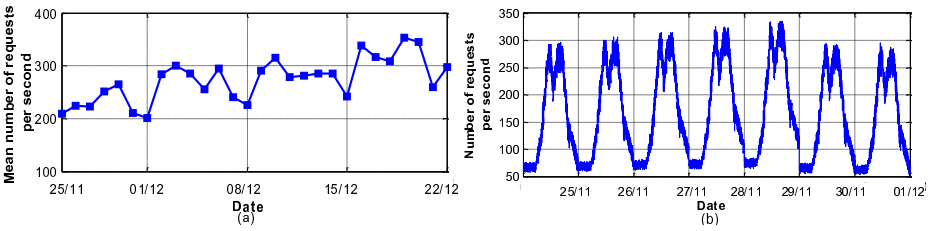

Fig. 4. Distribution of service performance levels

4.2 Trace-driven evaluation

We collected real-world traces from November 25, 2013 to December 22, 2013. Figure 4(a) reports the requests received by the system during the period. The mean number of requests per second became larger as the growth of customers scale from 337 to 512. It reflects a weekly pattern that the amount of concurrent visits is lower on weekends. A daily pattern is reflected by Figure 4(b). There are two peaks appeared in the morning and afternoon separately, and the trough appears at noon and midnight. Note that request for files larger than 4MB will be split into several requests, so the actual arrival rate of requests will be higher.

The amount of concurrent visits was too low to evaluate our provisioning scheme. We reprocessed the traces by adding the last three weeks dataset to the first week. Then we used LoadRunner [15] to test our system. The most common resource provisioning approach is based on Utilization-oriented Principle (UoP) [16]. The UoP approach tries to reduce cost by improving resource utilization (*i.e.*, equals ρ) to a predetermined range. We choose UoP approach to compare with our scheme, and the ranges are set as [60%, 70%] and [80%, 90%]. The thresholds of P^*_{suc}, R^* and P^*_{rej} are set as 99%, 200ms and 0.3%. The processing capacity depends on the type of VM. We measured the capacities through deploying each type of VM in our system, and the values are 224, 535 and 1372.

Figure 5 shows separately the CDF of P_{suc}/P^*_{suc}, R/R^* and P_{rej}/P^*_{rej}. It is concluded from equations (7) and (8) that the rejection rate has a positive correlation with utilization rate. For UoP [60%, 70%] approach concerned, the mean utilization rate are restricted in a low level without large variations, which results in a low level of rejection rate without large variations. When the rejection rate is low, the data availability is so high that we can neglect the influence of other factors on data availability. For UoP [80%, 90%] approach concerned, if the arrival rate is low, system could maintain a high level of both utilization rate and rejection rate. As the arrival rate increases, in order to satisfy performance constraints, both utilization rate and rejection intend to decrease. Note that compared with the threshold, our scheme can achieve much closer data availability and response delay.

Figure 6 describes the comparison of hourly server cost by using different provisioning approaches. UoP approach is sensitive to ρ^*. It appears to be relatively conservative when ρ^* is in the interval [60%, 70%]. Then excessive provisioning

of resources results in a much higher performance level than the threshold level. Furthermore, UoP approach pays 72.9% higher cost than our scheme, *i.e.*, the higher cost in exchange of the higher performance level. When ρ^* is in the interval [80%, 90%], it pays 31.8% higher cost than our scheme. Hence the cost can be reduced by increasing ρ^*, however, excessive increase in ρ^* will greatly increase rejection rate. As a consequence, the data availability decreases and becomes lower than the threshold.

The fixed number of types of VMs indicates the number of processing capacities available for selection is small. Therefore, the practical efficiency of our algorithm became high, and the mean execution time was 2.37 seconds.

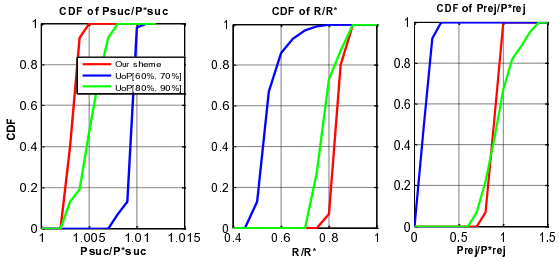

Fig. 5. Comparison of distribution of service performance levels

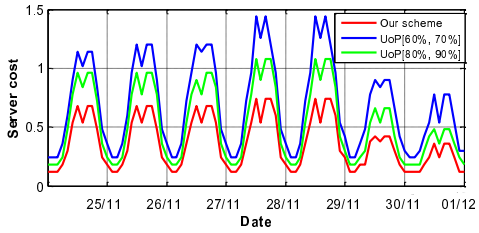

Fig. 6. Comparison of server cost

5 Conclusions

In this paper, we explore the resource provisioning from cloud storage provider's point of view, and propose a novel resource provisioning model. The model considers the complex interactions among servers during system running by using queuing network, and captures the relationship between performance metrics and the allocated resources. Then based on the model, the resource provisioning

problem is defined as a cost optimization with performance constraints. We put forward solution algorithms for solving the optimization problem. We have built a DHT-based storage system in our campus network. Based on real-world traces collected from system, the experimental results demonstrate that the proposed scheme can reduce cost while guaranteeing both data availability and response delay.

References

1. DeCandiaG ., Hastorun D., Jampani M., Kakulapati G., Lakshman A., Pilchin A., *et al.*:Dynamo: Amazon's Highly Available Key-value Store. ACM Symp. Operating Systems Principles (SOSP 07), ACM Press (2007) 205-220.
2. Lakshman A., Malik P.:Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, vol. 44 (2010) 35-40.
3. Stonebraker M.: The Case for Shared Nothing. IEEE Database Engineering Bulletin, vol. 9 (1986) 4-9.
4. Idilio D., Marco M., Maurizio M-M., Anna S., Ramin S., Aiko P. :Inside Dropbox: Understanding Personal Cloud Storage Services. ACM Conf. Internet Measurement Conference (IMC 12), ACM Press (2012) 481-494.
5. Jing J., Jie L., Quan Z-G., Dong L-G.:Optimal Cloud Resource Auto-Scaling for Web Applications. IEEE/ACM Symp. Cluster, Cloud and Grid Computing (CCGrid 13), IEEE CS Press (2013) 58-65.
6. Ferretti S., Ghini V., Panzieri F., Pellegrini M., Turrini E.: QoS-Aware Clouds. IEEE Conf. Cloud Computing (CLOUD 10), IEEE CS Press (2010) 321-328.
7. Jing B., Liang Z-Z., Xiong T-R., Bo W-Q.:Dynamic Provisioning Modeling for Virtualized Multi-tier Applications in Cloud Data Center. IEEE Conf. Cloud Computing (CLOUD 10), IEEE CS Press (2010) 370-377.
8. Lama P., Xiao B-Z.:Efficient Server Provisioning with Control for End-to-End Response Time Guarantee on Multitier Clusters. IEEE Trans. Parallel and Distributed Systems, vol. 23 (2012) 78-86.
9. Zhu Z., Bi J., Yuan H., Chen Y.:SLA Based Dynamic Virtualized Resources Provisioning for Shared Cloud Data Centers. IEEE Conf. Cloud Computing (CLOUD 11), IEEE CS Press (2011) 630-637.
10. Zhu Q., Agrawal G.:Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments. ACM Symp. High Performance Distributed Computing (HPDC 10), ACM Press (2010) 304-307.
11. Zhang C., Chen H-P., Gao S-T.:ALARM: Autonomic Load-Aware Resource Management for P2P Key-Value Stores in Cloud. IEEE Conf. Dependable, Autonomic and Secure Computing, IEEE CS Press. (2011) 404-410.
12. Gross D., Shortle J-F., Thompson J-M., Harris C-M.:Fundamentals of queueing theory. 4th ed., John Wiley & Sons (2008)
13. MacGregor S-J.:Properties and performance modelling of finite buffer M/G/1/K networks. Computers & Operations Research, vol. 38 (2011) 740-754.
14. TijmsH.:Heuristics for finite-buffer queues. Probability in the Engineering and Informational Sciences, vol. 6 (1992) 277-285.
15. HP LoadRunner Tutorial (2010)
16. AWS Elastic Beanstalk. http://aws.amazon.com/elasticbeanstalk/.