Journal of Network and Computer Applications ■ (■■■■) ■■■–■■■

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Optimization on content service with local search in cloud of clouds

Lingfang Zeng [a], Yang Wang [b,*]

[a] Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore
[b] IBM CAS Atlantic, Faculty of Computer Science, University of New Brunswick, Canada E3B 5A3

ARTICLE INFO

Article history:
Received 16 November 2012
Received in revised form
5 August 2013
Accepted 18 September 2013

Keywords:
Cloud of Clouds
Mobile service
Content distribution
Data movement
Local search

ABSTRACT

This paper studies the problem of distributing a content service in Cloud of Clouds to satisfy a sequence of mobile request demands with minimum monetary costs. A content may have single or multiple replicas, each being stored in one or more virtual machines (VMs) to facilitate the accesses via downloading to or replicating at (i.e., migrating) the VM sites where the mobile requests are made. As the origins of the mobile access patterns are frequently changed over time, this problem is particularly important for the users to achieve improved QoS for those time-bounded services and also beneficial to the service providers to minimize the expense on using the cloud infrastructure as well. However, these benefits do not come without compromise. The content distribution comes at cost of bulk-data transfer and service disrupts, which may increase the costs of the services. In this paper, to reap the content migration benefits while minimizing the service costs, we propose an online heuristic algorithm based on the local-search techniques to migrate the content replicas in adaptation of the mobile access patterns. For comparison purposes, we also study this problem in its off-line form, and propose a heuristic to evaluate the algorithm via a YouTube trace-based simulation. The simulation results show that our algorithms can effectively adapt to the changes of mobile access patterns to efficiently satisfy the user requests in a cost effective way.

1. Introduction

A Cloud of Clouds is generally viewed as the next revolution in the Cloud computing paradigm wherein the computational and storage infrastructure for handling scientific, business and enterprise applications could span across multiple Clouds and data-centers. However, as the complexity, heterogeneity and scale of applications grow, it will be increasingly important to be able to compose federated "Cloud of Clouds" that can address requirements for heterogeneous capabilities and large scales. For example, with an anticipated growth of mobile users of cloud services in the near future, issues related to the interoperability between cloud service providers (CSPs) and users are becoming challenging.

A content provider may reduce its expense by taking advantage of the geographically distributed datacenters and the competitive prices offered by different CSPs. Also, the content provider may use the on-demand scaling feature of the Clouds and easily adjust its requirement for the cost-effective computation powers and storage capacities. Since the costs of computation, storage and bandwidth resources are highly sensitive to the varying access models, to efficiently distribute the contents, optimizations on the content placement strategies are usually required. Although the content distribution network (CDN) problem has been widely studied in the literature (DiPalantino and Johari, 2009; Li et al., 2010; Borst et al., 2010; Jiang et al., 2009), there are some unique challenges when considering this problem in a Cloud of Clouds environment. On one hand, a content provider can construct an arbitrary network based on the demands to facilitate the accesses. This overlay network may have different topologies with respect to the underlying physical networks provisioned from infrastructure service provider (ISP). As a consequence, the content service is becoming a joint problem, requesting for both access routing and content distributing in a Cloud of Clouds. On the other hand, in Clouds the charge models for uploading and downloading the content replicas are often asymmetric with different prices, which implies that the content replication directions are usually needed to take into account in the distribution decisions (Chen et al., 2012).

To maximize the benefits of the content migration while minimizing the monetary cost of the computation, storage and transferring, in this paper, we study this content service problem by proposing an online heuristic algorithm to effectively distribute the shared content replicas in a Cloud of Clouds environment for mobile applications via content service migration and replication to adapt to the mobile access patterns. Our primary objective is to reduce the monetary cost which could exemplify the features of

* Corresponding author.
E-mail addresses: elezengl@nus.edu.sg (L. Zeng), lfzeng@hust.edu.cn, ywang8@unb.ca (Y. Wang).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.09.006

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

system qualities such as bandwidth, virtual machine (VM) and storage utilization. A content provider always considers the budget constraints since the resource provisioning of cloud systems is typically based on *pay-as-you-go* model.

Although the benefits are obvious, content service migration and replication could incure high cost of bulk-data transfer and service disruption. However, several researchers have demonstrated that it is feasible to migrate virtual services in a wide-area network (Hirofuchi et al., 2009; Voorsluys et al., 2009; Wood et al., 2011). Furthermore, regarding service migration, some preliminary results on single server migration have already been achieved with respect to virtual networks (Bienkowski et al., 2010; Oikonomou and Stavrakakis, 2010) and autonomic networks (Mortier and Kiciman, 2006; Dobson et al., 2006). On the other hand, the time delay of accessing the contents during service migration can also be hidden if we properly implement the algorithms in reality,[1] rendering our proposed algorithms to be practical.

This paper makes the following main contributions:

- We study the problem of content service in a Cloud of Clouds for the ease of mobile accesses with minimum costs. We model the content placements in the Cloud of Clouds as an optimization problem.
- By exploiting the local-search techniques, we develop an efficient heuristic solution to this problem.
- We evaluate the heuristic via YouTube trace-based simulations, and show that our algorithms can adapt to the changes of mobile access patterns to efficiently satisfy the user request sequences in a cost effective way.

The remainder of the paper is organized as follows. In Section 2, we discuss the motivation with comparisons to related work. In Section 3, we describe the system model which forms the basis for our discussion. The two local-search based online algorithms are proposed in Section 4 while a heuristic for the off-line setting is presented in Section 5. All algorithms are then evaluated in Section 6, and finally, Section 7 concludes the paper.

2. Related work and motivations

2.1. Related work

To mitigate above mentioned problem, cooperation among computations, storage and networks to adapt to the access patterns could be an effective way to minimize the user access latency, and reduce the operation cost of content providers. This adaptation may include dynamically shipping the requests as well as the related content replicas to some vantage locations in the clouds that are close to the users. Since fast provisioning of contents and VM instances has significant influence on the overall system performance, a considerable amount of research has been conducted for content placements with minimum costs in cloud-based CDNs.

The monetary cost models have evolved to include network (download/upload), storage, computation and power costs to facilitate the data and service managements in Clouds (Jiang et al., 2009; Dán, 2011). Wang et al. (2011, 2013) studied the data staging problem based on a homogeneous cost model in cloud-based CDNs for efficient mobile accesses. In their work, a variety of practical constraints with respects to the storage and communication overhead are investigated in the migration of the requested content replicas with minimum costs, which bears some similarities to our case. MetaCDN by Broberg et al. (2009) is a low cost CDN model using cloud-based storage resources. The system provides mechanisms to place contents in different cloud-based storage networks and route user requests to appropriate replicas. Peng et al. (2012) took advantage of the hierarchical network topologies of data centers to reduce VM instance provisioning time and in the meantime minimized the overhead of maintaining chunk location information. Zheng et al. (2011) presented storage migration scheduling algorithm that can greatly improve storage I/O performance during wide-area migrations. Most recently, SpotCloud by Wang et al. (2012) is a customer-provided cloud platform which enables the customers to find the best trade-off between the benefits and the costs. By exploiting social influences among users, Wu et al. (2012) proposed efficient proactive algorithms for dynamic, optimal scaling of a social media application in a geo-distributed cloud. Dai et al. (2012) discussed the collaborative hierarchical caching with dynamic request routing for massive content distributions. Chen et al. (2012) presented and evaluated a suite of online heuristic algorithms based on Integer Programming (IP). Essentially, their scheme focuses on "Cloud CDN" based on multiple storage clouds.

In most of existing work, requests are simply forwarded to the upper-layer parent servers when the content is not locally available. Or request-redirection occurs over distributed sets of servers, to minimize redirection latency (Pathan, 2009). Our work is similar to Dai et al. (2012) and Chen et al. (2012) – dynamic request routing is designed jointly with content placement strategies – and focuses on evaluating the new features of adaptation to the changes of mobile access patterns to efficiently satisfy the user requests in a cost effective way.

Because the core of cloud application is virtualization, our work in content service is necessarily built upon work in a number of other related areas: Many VM placement (suk Kee and Kesselman, 2008; Celesti et al., 2010) and data placement algorithms address distinct problems, such as replica placements and server consolidations (Vogels, 2008). Users' requests are pre-known in terms of their positions and access frequencies in the networks, while the location and the minimum number of the VMs to serve the requests with minimum total access costs are to be determined (Benoit et al., 2008; Gupta and Tang, 2006; Kalpakis et al., 2001). Liu and Datta (2011) discussed the data placements given the intermediate data for workflow applications. Mills et al. (2011) proposed an objective method for comparing VM placement algorithms in large clouds. Recently, Calcavecchia et al. (2012) proposed and evaluated a novel technique called *Backward Speculative Placement* (BSP), which analyzes the past demand behavior of a VM to a candidate target host. Biran et al. (2012) presented a VM placement algorithm, namely *Min-Cut Ratio-aware VM Placement* (MCRVMP), which satisfies the predicted communication demands and time-variations. They claimed that the general MCRVMP problem is NP-Hard and introduced several heuristics to solve it in reasonable time.

To offer both cloud service infrastructure and content delivery, a content provider is faced with the coupled application server (e.g., VM) selection, content placement and request routing problems. These problems interact with each other because the server selection and the content placement affect the operation costs while the request routing influences the offered load visible to the servers. Actually, all these factors, the server selections, content placements as well as the request routing, have great impact on our algorithm design since they not only influence the operation costs of CSPs but also affect their profitability.

[1] Given the read-only access patterns in CDN, there could be two basic strategies to achieve this goal. First, we can simply allow the source service replica continue to serve the incoming requests while migrating to the target site. When the target is ready, the source replica can be informed of being closed or removed. After the migration, the subsequent requests will be served by the target replica. Second, we can redirect the incoming requests to other service replicas when the selected content is replicated/migrated to other sites.

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

2.2. Motivations

In general, content distribution algorithm is not only needed to figure out the user's access patterns (i.e, a sequence of requests, each being made by certain application server at a predicted time instant) but also required to replicate or cache the involved contents across the network at particular time instance to optimize some performance metric for future accesses.

To better understand the impact of content distribution and request pattern on the cost efficiency, we conduct experiments to evaluate the costs by measuring the different types of costs for a content service provider (CoSP). The information on different CSP's sites is extracted from the iPlane Internet PoP topology (Madhyastha et al., 2006). We assume CoSP provides streaming video service (10 million users/day). The resource configuration of the provider is shown in Table 1. Fig. 1 shows the costs (in a month) with respect to the different content distribution schemes, *Greedy Request* (GR) and *Greedy Site* (GS). Both algorithms are proposed by Chen et al. (2012) to minimize the cost incurred on the CDN providers while satisfying QoS requirements of user requests. In GR, site opening operation is always triggered by the first request of a user. Only after a site has been opened and provisioned with replicas, requests can be redirected to the site. GS algorithm iteratively decides to open a closed site which has the highest utility and assigns all its potential users to it, until all the users have been assigned. Here, utility refers to the ratio of the total request volume (bytes) handled by the site to the total cost (dollars) to serve them while the potential users refer to those users who are within the QoS requirement distance, but have not yet allocated sites.

By comparing with the costs between the two algorithms, we found that the communication costs by the GS dominate the total costs (e.g., accounting for 62% in Fig. 1(b), the readjusted contents is about 1% of the total contents) whereas for GR, the computation and storage have the larger fraction (51% and 29% respectively, as shown in Fig. 1(a)). These observations are not surprising as the service costs are linearly increased with respect to the number of communications. The rationale behind these observations is that the request assignments by GS *bias* toward a large set of users, incurring relatively large number of communications especially at the initial several time stages. In contrast, for the GR case, the requests are reasonably assigned among a small set of providers, resulting in the reductions of the inter-node communications and the increases of the caching (storage) costs.

Table 1
An example of content service.

Resource type	Reserved/original value
Computation (# of VMs)[a]	25,000
Storage capacity	10 PB
Communication (upload)[b]	6.5 PB
Number of CSPs	40

[a] To simplify the evaluation, for different cost types, the price by each CSP is randomly generated in the range of [0.10, 0.20]: computation ($/h), storage ($/GB/month), and communication ($/GB).

[b] Levin (2006) shows that the percentage of storage managed by SANs is nearly the same for UNIX (65%) and Windows environments (63.6%), so the transfer data size (also the capacity occupied by the system) is set as 65% of total storage capacity.

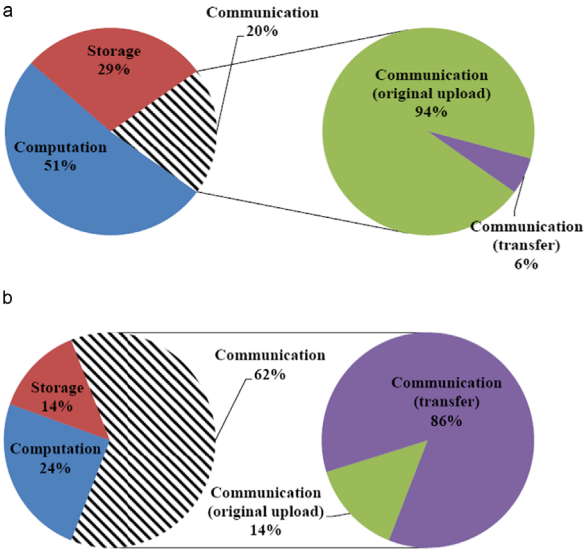

Fig. 1. Impact of content service on service cost. (a) Greedy request and (b) Greedy site.

3. System model and problem formulation

The notations used in this paper are summarized in Table 2.

We consider a substrate n-provider network $\mathcal{G}=(\mathcal{P}\cup\mathcal{A},\mathcal{E})$ with an arbitrary shape rented by a CoSP from n (CSPs). Let $\mathcal{P}=\{P_1,P_2,\ldots,P_j,\ldots,P_n\}$ denote the set of CSPs. Let P_j denote the j-th class of cloud service provider. Each CSP supplies a pool of resources (e.g., computing or storage resources) to customers. In this work, the CSP supplies computing power (in a unit of CPU-hours), storage capacities (in a unit of gigabytes or GBs), and network bandwidth to facilitate the data transfer (in a unit of GBs/day).

Edge $(P_b,P_j)\in\mathcal{E}$ indicates a feasible provisioning or replication path from P_b to P_j (see Fig. 2). Edge $(P_j,A_k)\in\mathcal{E}$ indicates a potential assignment of access point A_k (defined later) to P_j. Each VM $vm_x\in V_j, \mathcal{V}=\{V_1,V_2,\ldots,V_j,\ldots,V_n\}$ has a computational capacity associated with it represented by its CPU power and memory size.

There is a set of external mobile users that access $\mathcal{G}$ by issuing a sequence of batch requests $\sigma=\sigma_1\sigma_2\ldots\sigma_m$ to the content service, which has $\kappa\geq 1$ locations distributed in *configuration* H, each reserving a certain amount of compute and storage resources. The distribution of users and therefore the request patterns change over time. The requests arrive in an online fashion and are served by triggering the replication of the stored contents without a prior knowledge on the future requests. Consequently, the subset of the VMs and storage servers are frequently changed with the time going on. We denote the subset H at time t_i as H_i.

In order to allow the external requests to access the VMs in $\mathcal{G}$, a fixed subset of nodes $\mathcal{A}$ serve as the *Access Points*. For example, the mobile users can connect to the network via wireless links, each with a constant latency ι which is charged in a rate of W_ι. Therefore, a batch request σ_i at time t_i can be represented by a set $\sigma_i=\{(a_{ij},R_{ij})|a_{ij}\in A_i,R_{ij}=\cup_l r_{ij}^l\}$ where $1\leq j\leq|A_i|, l\in I$, I is integer set, $A_i\subseteq A$ is the subset of access points at t_i, and $r_{ij}^l=\{D_{ij}^l\}$, D_{ij}^l is the requested data subset of the l-th request r_{ij}^l in R_{ij}. Therefore, we have $R_i=\cup_j R_{ij}$, which represents the set of the total requested contents of σ_i. In the sequel, to simplify the presentation, we sometimes write $r_{ij}^l\in R_i$ as $r\in R_i$ and its requested data set D_{ij}^l as D_r if there are no confusions.

A CSP charges for the computation, storage, incoming and outgoing traffics. The monetary cost rates for the VM v_j to compute the service are determined by the CSP P_j, denoted as c_j^{vm}. For storing a replica, the provider P_j will charge per-unit cost of c_j^{sto}/GB; for uploading a replica onto P_j, P_j will charge a unit price of c_j^{in}/GB for incoming traffics; for distributing a replica to other providers or delivering contents to the end users, P_j will charge per-unit cost of c_j^{out}/GB for outgoing traffics. Clearly, under these

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

conditions, the cheapest path (CP) between any pair of providers in $\mathcal{G}$ can be measured by the total cost along the path from the source to the destination.

In our model, the number of distinct contents g is bounded by a constant, and each distinct content has at most k ($k<\kappa$) replicas. The number of the providers κ is upper bounded by $O(\log n/\log(1+\Delta))$ where Δ is the maximum node degree of the n-node network $\mathcal{G}$. Therefore, $|\cup_{r\in R_i} D_r| \le gkD_{ini}$, where, D_{ini} denotes the original requested data size. We assume that all the g distinct contents are initially located at CSP P_{ini} (an example P_1 as shown in Fig. 2), each having a single replica.

Table 2
Major notations used in this paper.

Symbol	Definition
$\mathcal{G}$	A directed graph for a substrate n-provider network
$\mathcal{E}$	A set of paths
$\mathcal{A}$	A set of access points
$\mathcal{P}$	A set of CSPs
$\mathcal{V}$	A set of VMs
P_j	The j-th class of CSP
V_j	The set of VMs in P_j
H	A system configuration
H_i	A subset H at time t_i
ε_i	An epoch for H_i
R_{ij}	A set of requests for j-th access point at time t_i
A_i	The subset of access points at time t_i
R_i	Request set at time t_i
σ_i	Batch request sequence at time t_i
r_{ij}^l	The l-th request in R_{ij}
D_{ij}^l	The requested data subset of the l-th request r_{ij}^l in R_{ij}
D_{ini}	The original requested data size
c_j^{vm}	The monetary cost rates for the VM vm_j to compute the service is determined by the CSP P_j
c_j^{out}	Outgoing traffics cost (Download cost from P_j)
c_j^{in}	Incoming traffics cost (Upload cost to P_j)
c_j^{sto}	Storage cost determined by the CSP P_j
CP	The cheapest path between any pair of providers in $\mathcal{G}$
g	# of distinct contents
k	# of content replicas
κ	# of locations
w_k	Request size of r
$\phi(r)$	Redirecting function
ζ	Content distribution function
W_ι	The charged rate for a constant latency ι
$\mathcal{N}(H_i)$	A neighborhood of H_i (*local space*)
Z	One-hop neighbor set
Y	Storage server set
ϱ'	The node with the cheapest replica
Set_{rep}	The node set that host the replicas of the request item at time t_i
T_ϱ	A cumulative mileage count for a replica ϱ
$\mathbb{H}_i$	The set of all the configurations at time t_i

For example, in Fig. 2 batch request σ_i contains four request subset $\sigma_i=\{(a_{i1},R_{i1}),\ldots,(a_{i4},R_{i4})\}$ and $A_i=\{a_{i1},\ldots,a_{i4}\}$. For each $R_{ij}, j=1,2,3,4$, associated with access point a_{ij} represented by rectangle in the figure, its constituent requests $\cup_l r_{ij}^l$ are also shown in the figure (see the circles in each rectangle). In Fig. 2 (a) potential access paths are drawn as dashed line and potential content distribution paths are drawn as bold lines among an origin data server S_0, and the potential replica providers $P_1\sim P_{11}$. We assume each provider has a routing path to each access point a_{ij}.

An example solution of the request routing and content placement problem is shown in Fig. 2(b), where P_5 is chosen to serve requests from R_{i1} and R_{i2}, and P_4 is chosen to serve requests from R_{i3}. P_6 is chosen to serve the requests from R_{i4}. P_2 relays the replica from S_0 in P_1 to $P_3\sim P_6$. The total cost for this solution includes the computation, storage, upload and download costs in $P_1\sim P_6$. For a replica that serves batch requests (e.g., P_5 or P_6), its upload cost is incurred by the incoming traffics to provision contents at the provider, and its download cost is incurred by the outgoing traffics to serve batch requests.

Clearly, to satisfy σ_i, each request $r\in R_i$ will be eventually routed to a certain server $h\in H_i$ via the closest $a_r\in A_i$ by some *routing function* $\phi(r)$ (i.e., $\phi(r)=h$, implying that a request is only processed by one VM). To minimize the total access cost, one may observe that the *cheapest routing strategy* that simply routes each request in R_i to its cheapest server in H_i along the cheapest path could be an optimal way. However, this observation is only true if there is no incurred data replication to service the requests. Otherwise, the observation might not true again when considering its requested data subset D_r.

To illustrate these concepts, we still use the example in Fig. 2, where for σ_i, a provider is selected (e.g., P_5), and serves the requests $r_{i1}^1, r_{i1}^2\in R_{i1}, r_{i2}^1, r_{i2}^2\in R_{i2}$, which are routed by ϕ according to some criteria. As a consequence, the corresponding requested data subsets D_{i1}^1, D_{i1}^2 and D_{i2}^1, D_{i2}^2 are also needed to replicate at P_5 to serve σ_i.

Therefore, the service cost of request $r\in R_i$ in our model includes three parts. One is the cost of routing request r to the sever at $\phi(r)\in H_i$ which is equal to $W_\iota + w_r CP(a_r,\phi(r))$, where $a_r\in A_i$ is r's nearest access point, w_r is the request size of r, $CP(u,v)$ is the cheapest path between location u and v in the network $\mathcal{G}$.

The second is the cost of content distribution $Cost_{dis}$ (including the caching, transferring or replicating costs) in our model: the data set D_r at t_{i-1} of the request from locations $\zeta(D_r)$ to $\phi(r)$ at t_i,

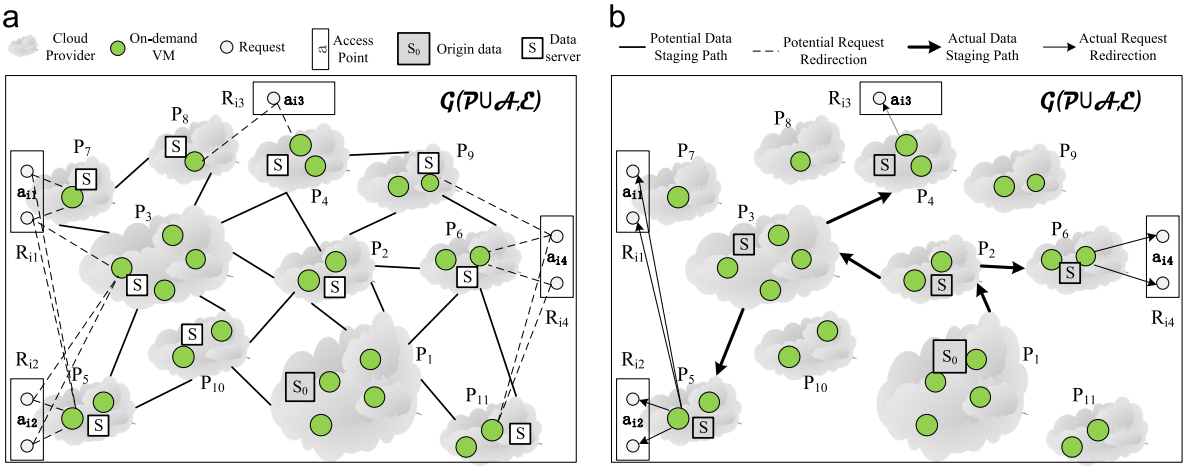

Fig. 2. An example of the model. (a) A Problem Instance and (b) a Solution.

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

which can be written as $\sum_{u \in \zeta(D_r)}(c_u^{sto}+c_{\phi(r)}^{sto}+c_{\phi(r)}^{in}+c_u^{out})D_r$. This cost is clearly determined by the content distribution function ζ;

The third is the cost $Cost_{run}$ for running the VMs in the physical machines, which reflects the cost due to satisfying the request sequence σ. Since the machines in a datacenter as assumed are typically heterogeneous, each with its own running cost rate, $Cost_{run}(H_i)$ can be given as $\sum_{h \in H_i} c_h^{vm}\Delta_t$ for a particular set of H_i machines to satisfy σ_i, each running a virtual server for a period of time Δ_t (defined later).

So, given an online request sequence $\sigma=\sigma_1\sigma_2,\ldots,\sigma_m$ and sever configuration of H_i at t_i, our main objective is by content distribution and request routing to shed light onto the trade-off among

- the access cost $Cost_{acc}$ which can be written by

$$Cost_{acc}(H_i,R_i,\phi)=\sum_{r \in R_i}(W_\iota+w_r CP(a_r,\phi(r))) \quad (1)$$

where a_r is the access point of r.

- the content distribution cost $Cost_{dis}$:

$$Cost_{dis}(H_i)=Cost_{sto}(H_i,R_i,\phi,\zeta)+Cost_{tra}(H_i,R_i,\phi,\zeta)$$
$$=\sum_{u \in \zeta(D_r)}(c_u^{sto}+c_{\phi(r)}^{sto}+c_{\phi(r)}^{in}+c_u^{out})D_r \quad (2)$$

where $Cost_{sto}$ is the storage cost and $Cost_{tra}$ is the data transferring cost between CSPs.

Note that the costs of content distribution operations are assumed to be the same and shown in the equation and are determined by the storage constraints, the concrete content distribution schema computed by ζ is not shown and will be discussed later.

- the service running cost $Cost_{run}$, which is defined as the total cost of running the VMs from t_i to t_{i+1} (i.e., Δ_t) on the selected set H of the computational nodes, each with cost $C_h, h \in H$, i.e.,

$$Cost_{run}(H_i)=\sum_{h \in H_i} c_h^{vm}\Delta_t \quad (3)$$

with the identified ϕ and ζ to minimize the total cost for serving σ.

4. Online algorithm

In this section, we present our online algorithm which dynamically service the incoming requests in a time sequence order in absence of any knowledge of the complete access patterns. As such the decisions made in the algorithm have to be online and merely based on the historical request data. To ease the presentation and understanding of the algorithm, we overview the algorithm from a high-level perspective by first defining some used concepts and data structures and then introducing the basic idea of the algorithm.

4.1. Definition

Our algorithm operates on per-*epoch* basis which is defined as a period of time that no content-replication is incurred to serve a sub-sequence of requests. Each *epoch* is signified by a service configuration, called *pivot configuration* which is created at the beginning of the epoch. Given a batch-request σ_i (also R_i) and service configuration H_i at time t_i when epoch ε_i starts, we first construct a neighborhood of H_i (*local space*), denoted by $\mathcal{N}(H_i)=\{H_i^0,H_i^1,\ldots,H_i^j,\ldots\}$, where $H_i^j=H_i-Y^l+Z^l$ [2] and $Z^l=N_1(Y^l)$ is one-hop neighbor set of $Y^l \in 2^{H_i}$ with size of l (i.e., $|Z^l| \le |Y^l|=l$), $0 \le l \le k$. Clearly, each H_i^j signifies a distinct neighbor configuration of H_i. Depending on the definition of $\mathcal{N}(H_i)$, we can design different online algorithms, each with different properties and time complexities. In our studies, we select $H_i^{j^*}$ that has the minimum service cost for R_i.

In the algorithm, we distinguish two cases: (**1**) when $Y^{j^*}=\varnothing$ (i.e., $j^*=0$), there is no content-replication to serve R_i, and the algorithm operates in the same *epoch* i to continue to serve the next batch request at $i+1$. Otherwise, (**2**) $Y^{j^*} \neq \varnothing$ (i.e, $j^*>0$), which implies that there exists content-replication from H_i (also H_i^0) to $H_i^{j^*}$ to minimize the service costs. In this case, the new *epoch* ε_{i+1} is created by $H_{i+1}=H_i^{j^*}$, and the follow-up procedure is repeated as discussed above.

Our algorithm typically starts from case one, serves incoming batch requests one by one using each configuration in $\mathcal{N}(H_i)$, and cumulatively collects their respective service costs. Therefore, unless the access pattern is not significantly changed, the algorithm will eventually cause the case two by replicating the requested contents to adapt to the access patterns. Note that to enable the algorithm to be deployed in reality, we sacrifice some efficiency by resetting the cumulative service cost of each configuration in a newly created epoch.

The algorithm maintain two data structures for each configuration in $\mathcal{N}(H_i)$ on per-epoch basis. One is a Cost Recorder (CR) that is used to monitor and accumulate the access costs in the epoch. The other is an Access Logger (AL) used to record the history of the corresponding access requests in the same epoch. During the service, the algorithm progressively accumulates and records for the pivot configuration H_i the access costs and requests in the pivot recorder *CR* and pivot logger *AL*, respectively.

4.2. Basic idea

The essence of our algorithm is to leverage the rent-or-buy paradigm in the Ski rental problem to determine the content replication and take advantage of a short historical access information to prune the local space for efficient finding of the distribution target with the service cost reduction. The distribution cost (i.e., the buy cost) is defined as the maximum service cost from the source pivot configuration H_i to its target neighbors in $\mathcal{N}(H_i)$. The historical access information is gathered by the current epoch during the service and used in the subsequent one or more epochs to facilitate the content distribution with the total service cost reduction as the goal. In each epoch the algorithm is composed of multiple iterations. In the beginning of each iteration, the algorithm first leverages the rent-or-buy paradigm to determine the distribution by comparing the access cost in pivot counter and the distribution cost. If the value of the pivot counter is less than the distribution cost, the pivot configuration will be fixed to continually provide services to outside till this value is greater than the distribution cost. Otherwise, the algorithm first overwrites each neighbor's *AL* with the pivot logger *AL* (obtained from previous pivot recorder by copying), and then mimics the service of each neighbor to the requests in its *AL* via accumulating each service cost in its *CR*. After that, the algorithm compares the value of each neighbor's *CR* with that of the pivot recorder, and then selects the one with the minimum CR as the next target node to conduct the content replications.

4.3. Formal algorithm

As discussed, our algorithm consists of epochs, each being signified by a pivot configuration. In exhaustive search, the elements in $\mathcal{N}(H_i)$ are defined based on each $Y^l \in 2^{\mathcal{H}_i}$, and $Z^l=N_1(Y^l)$ for all $0 \le l \le \kappa$. We use $\Gamma_i(H_i^{lpq},Y_p^l,Z_q^l)$ where $l \in [0,\ldots,\kappa]$, $p \in [1,\ldots,\binom{\kappa}{l}]$ and $q \in [1,\ldots,\Delta^l]$ to represent the total

[2] We use $X-Y+Z$ to denote $X \backslash Y \cup Z$.

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

service cost in epoch ε_i to a dynamically incoming request sequence until σ_i where $H_i^{lpq} = H_i - Y_p^l + Z_q^l$ who replicates exactly l data sets from $Y_p{}^l$ to $Z_q{}^l$ to reach H_i^{lpq} while minimizing the overall service cost up to σ_i. Then $\mathcal{N}(H_i)$ can be written as

$$\mathcal{N}(H_i) = \{H_i^{lpq} | H_i^{lpq} = H_i - Y_p^l + Z_q^l\}$$

Note that the subscript p of $Y_p{}^l$ specifies a particular configuration of the selected replicated data sets in H_i whereas the target locations are specified in the subscript q of $Z_q{}^l$.

Given that κ locations and g distinct contents are initially co-located at H_0, which signifies epoch ε_0 and can be determined by the static algorithm without data transferring (Section 5), we have $\Gamma_0(H_0^{0_{11}}, Y_1^0, Z_1^0) = \Gamma_0(H_0, \varnothing, \varnothing) = 0$, and for $l \geq 1, p, q > 1$, $\Gamma_0(H_0^{lpq}, Y_p^l, Z_q^l) = Cost_{dis}(H_0, H_0^{lpq})$. Suppose that at time t the algorithm is in epoch ε_i signified by H_i. To serve the request σ_i, the algorithm works as follows:

*Step*1: Initialize ε_i: If the pivot configuration of ε_i is changed from H_i to H_{i+1}, then $i = i+1$ and a new epoch ε_i starts, which is signified by H_i. We then initialize ε_i by computing

$$\Gamma_i(H_i^{lpq}, Y_p^l, Z_q^l) = Cost_{dis}(H_i, H_i^{lpq}) \tag{4}$$

for all elements in $\mathcal{N}(H_i)$. Note that when $l = 0, p, q = 1$, we have $H_i^{0_{11}} = H_i$ and $Cost_{dis}(H_i, H_i^{0_{11}}) = 0$. Otherwise, go to Step 2;

*Step*2: Serve σ_i: The algorithm serves σ_i under each H_i^{lpq} in $\mathcal{N}(H_i)$ according to Eq. (5):

$$\Gamma_i(H_i^{lpq}, Y_p^l, Z_q^l) = \Gamma_i(H_i^{lpq}, Y_p^l, Z_q^l) + Cost_{run}(H_i^{lpq}) + Cost_{acc}(H_i^{lpq}, R_i, \phi, \zeta) \tag{5}$$

The service cost for each other configuration (when $l \neq 0, p, q \neq 1$) in the neighborhood is the potential cost if some contents were replicated from the pivot configuration to that configuration and use that configuration to serve the request sequence up to σ_i. Go to Step 3;

*Step*3: Identify new epoch: We obtain the minimal configuration by local search

$$(l^*, p^*, q^*) = \underset{l,p,q}{\operatorname{argmin}}\{\Gamma_i(H_i^{lpq}, Y_p^l, Z_q^l)\} \tag{6}$$

If $(l^*, p^*, q^*) = (0, 1, 1)$, then $Y_{p^*}^l = Z_{q^*}^l = \varnothing$ and $H_i^{lpq} = H_i$, indicating that there is no data sets replications and the pivot configuration H_i is still the minimal configuration to serve σ_i. In this case, the algorithm does not create new epoch ε_{i+1} and continue service the following request σ_{i+1} within ε_i by going to Step 4. Otherwise, $|Z_{q^*}^{l^*}| \leq |Y_{p^*}^{l^*}| = l^* > 0$, and $Y_{p^*}^{l^*}$ is the selected replicas in H_i to satisfy the request R_i by moving it to $Z_{q^*}^{l^*}$, the representative set of R_i. Then let $H_{i+1} = H_i^{l^*p^*q^*}$, replicate contents located at $Y_{p^*}^{l^*}$ to $Z_{q^*}^{l^*}$ and finally go to Step 1 for the next request σ_{i+1}.

*Step*4: Avoid overflow: To prevent Γ_i from overflowing, we obtain the service cost for maximal configuration from $\mathcal{N}(H_i)$, i.e., $\Gamma_{max}^i = \max\{\Gamma_i(H_i^{lpq}, Y_p^l, Z_q^l)\}$. If Γ_{max}^i is greater than a pre-defined threshold value τ, then all the service costs of the elements in $\mathcal{N}(H_i)$ are divided by 2, and go to Step 1 to serve the next request σ_{i+1}.

As discussed previously, for the general case when optimal ϕ and ζ are left to be determined given the large space of the configurations, the content distribution problem is unduly complex to be solved within a reasonable time. Therefore, instead of finding the optimal strategies to ϕ and ζ, we adopt an alternative way to treat them as the second-class citizens in the algorithm by pre-defining some heuristics for them while focusing on the optimization of the content-distribution cost by allowing the content-distribution decision to invoke those second-class algorithms. However, this treatment by no means indicates that the selections of ϕ and ζ are not as important as the content distribution. Rather, our empirical studies showed that they have big impact on the overall performance. The rationale behind this treatment is primarily for simplicity in the sense that the different strategies for request routing and content distribution can be studied separately without intermingling with each other.

Ideally, the selections of the representative location set Z for the current request R_i and the content-distribution set $Y \subset H$ (to Z) are independent with each other. However, the independent selections of these two sets will dramatically increase the complexity of the algorithm. Therefore to address this problem, we deliberately restrict the relationship between Z and Y in the algorithm.

- *Request routing function* ϕ: Due to the large number of possibilities to route a batch request R_i to a particular node set Z, (i.e., $O(R_i^{|Z|})$), our algorithm simplifies the routing function ϕ by a heuristic strategy. We try to minimize $Cost_{dis}$ by finding the median for the data locations which is determined by $\zeta(D_r)$. To this end, for $\forall r \in R_i$, we define $\phi(r)$ as

$$\phi(r) = \underset{v \in Z}{\operatorname{argmin}} \sum_{u \in \zeta(D_r)} (c_u^{sto} + c_{\phi(r)}^{sto} + c_{\phi(r)}^{in} + c_u^{out}) D_r \tag{7}$$

By following the same arguments, the time complexity of ϕ is $O(\kappa|\zeta(D_r)| + n)$ where $|\zeta(D_r)|$ represents the size of $\zeta(D_r)$ (discussed later) which is less than or equal to g since the time complexity of data staging schema is not a concern in computing $Cost_{tra}$. Therefore, ϕ is also $O(\kappa + n)$ time complex.

Obviously, this heuristic strategy is greedy in nature and each correspondingly defined ϕ is also a *general function*, which dramatically reduces the complexity to compute the routing function. Again, for each $Z \subset H$, $\phi(R_i)(\subseteq Z)$ *uniquely* determines the representative locations for R_i in range of Z to which the data set will replicate.

- *Content distribution function* ζ: We first computes the center of Z, denoted as $v^* \in Z$,

$$v^* = \underset{u \in Z}{\operatorname{argmin}} \max_{v \in Z} \{CP(u, v)\} \tag{8}$$

and then investigate the following strategies to select the candidate replicas to serve the request r, which is pretendedly made at $v^* \in Z$.

(a) *Greedy strategy* (*GS*) serves each request by the cheapest replica ϱ'

$$\varrho' = \underset{\varrho \in Set_{rep}}{\operatorname{argmin}}\{CP(\varrho, v^*)\} \tag{9}$$

where Set_{rep} is the node set that host the replicas of the request item at time t_i. As this strategy does not resort to the historical data, it is memory-less.

(b) *Balance strategy* (*BS*) works by trying to minimize the maximum transferring cost by any replica. We associate with each replica ϱ a cumulative mileage count T_ϱ. Upon a request at location v^*, we select replica ϱ' according to the following criterion:

$$\varrho' = \underset{\varrho \in Set_{rep}}{\operatorname{argmin}}\{T_\varrho + CP(\varrho, v^*)\} \tag{10}$$

where T_ϱ is updated by $T_\varrho = T_\varrho + CP(\varrho, v^*)$ after the service.

Now, we achieve two online heuristics: One is local search with greedy strategy (referred to as *LSGS*) and the other is local search with balance strategy (referred to as *LSBS*). Roughly speaking, the design of the two heuristics can be summarized as follows. First, we determine the candidate replica for each content requested by $r \in R_i$. To this end, we first compute the center v^* of Z (used to represent Z) and then select the replica with minimum

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

transferring cost to v^* as the candidate, which is only relied on the network transferring cost. Second, for each requested content, after determining its candidate replica, we obtain the representative set for R_i within the range of Z by using ϕ and then leverage the caching costs of corresponding nodes to determine the distribution operations to satisfy the requests of D_r. Finally, we process the constraints based on the LRU paradigm.

4.4. The time-complexity analysis

Given the definitions of ϕ and ζ in Section 3, and $\kappa \le \log\, n/\log(1+\Delta)$, if all pairs of the cheapest paths in advance, the online algorithm can serve batch request R_i within the time complexity of $O(n \log\, n/\log(1+\Delta)|R_i|)$.

Proof. In each epoch, according to Eq. (4), since there is at most l replicated data sets in ${X_p}^l$ with known targets in ${Z_q}^l$, the time complexity of epoch initialization can be written as $O(\sum_{l=0}^{\kappa} l\binom{\kappa}{l}\Delta^l)$ and furthermore time complexity of serving σ_i, according to (5), can be upper bounded by,

$$O\left(\sum_{l=0}^{\kappa}\binom{\kappa}{l}\Delta^l(\kappa+\kappa^2+|R_i|(\kappa+k))\right) \tag{11}$$

Similar to $Cost_{dis}(H_i, H_i^{lpq})$, the time to compute $Cost_{run}(H_i^{lpq})$ is also easy to know as $O(\kappa)$. As for the time complexity of $Cost_{acc}(H_t^l, R_i, \phi, \zeta)$, according to (1) it can also be broken into two parts with respective complexities of $O(|R_i|\kappa)$ and $O(\kappa^2+|R_i|k)$.

Since $l \le \kappa$ and $|R_i|$ is usually greater than κ, given $k < \kappa$ in the model, we can then upper bound the time complexity for serving R_i by

$$O\left(\kappa|R_i|\sum_{l=0}^{\kappa}\left(\binom{\kappa}{l}\Delta^l\right)\right) = O(\kappa(1+\Delta)^{\kappa}|R_i|) \tag{12}$$

On the other hand, the time complexities of Steps 3 and 4 are not greater than that of Step 1. Consequently, the time complexity of the algorithm is determined by Step 2. Therefore, given $\kappa \le \log\, n/\log(1+\Delta)$, the algorithm can serve the batch request R_i within $O(n \log\, n/\log(1+\Delta)|R_i|)$.

4.5. Remarks

As discussed, the algorithm operates on per-epoch basis where the contents are replicated and downloaded to the requesting sites to facilitate the mobile accesses when new epochs are established. Therefore the frequency of the transition between epochs is essential to the effectiveness of the algorithm. To strike a good balance between the overhead of the content movements and the benefits of the adaptation in the demand services, we deliberately leverage the rent-or-buy paradigm in the ski rental problem to control the epoch transition, which is achieved with the designed migration strategy based on the CR and AL data structures. The rationale behind this design is that the rent-or-buy paradigm can provably achieve the best competitive ratio to such an online problem.

5. Off-line algorithm

In this section, we study the optimal off-line algorithm, denoted as *Opt*, to this problem based on the dynamic programming (DP) techniques. Given the complete knowledge of the request sequence σ in advance, the off-line algorithm can not only identify the optimal distribution pattern at hindsight but also form the basis to evaluate our online algorithm. *Opt* exploits the optimal sub-structure property of this content service problem to construct the DP recurrence in Eq. (13), where a minimum-cost solution $\Phi_i(H_i^j)$ to the problem at t_i can be obtained efficiently from the minimum-cost solutions to all its sub-problems of $H' \in \mathbb{H}_{i-1}$ at t_{i-1} given σ and sever configuration H_i^j (there could be $\binom{n}{\kappa}$ possible configurations) for $1 \le j \le \binom{n}{\kappa}$.

$$\begin{aligned}\Phi_i(H_i^j) = \min_{H' \in \mathbb{H}_{i-1}} \{&\Phi_{i-1}(H') + Cost_{acc}(H_i^j, \sigma_i, \phi) + Cost_{sto}(H', H_i^j) \\ &+ Cost_{tra}(H', H_i^j) + Cost_{run}(H_i^j)\}\end{aligned} \tag{13}$$

where $\mathbb{H}_{i-1} = \{H_{i-1}^1, \ldots, H_{i-1}^{\binom{n}{\kappa}}\}$ is the set of all the configurations at time t_{i-1} with a size of $\binom{n}{\kappa}$ and R_i is the batch request at t_i. Initially, H_0^1 is resource set from κ providers at t_0 and $\Phi_0(H_0^1) = Cost_{sto}(H_0^1) + Cost_{run}(H_0^1)$. For any other configuration H_0^j, $\Phi_0(H_0^1, H_j^1) = Cost_{dis}(H_0^1, H_j^1) + Cost_{run}(H_j^1)$, $2 \le j \le \binom{n}{\kappa}$, and the optimal solution is then given by

$$\Phi_n^*(H_n) = \min_j\{\Phi_n(H_n^j)\} \tag{14}$$

the optimal content distribution strategy can be reconstructed from H_n by recursively finding the optimal configuration H' at time t_{i-1} which led to the optimal configuration H_i^j at time t_i.

It is easy to see that handling the large configuration space and minimize $\Phi_n(H_n)$ in the optimal content distribution is an overly complex problem. In the sequel, we propose our heuristics to overcome the configuration complexity and solve this problem within a reasonable time. We present one heuristic greedy request algorithm for the off-line setting (referred to as *Offline-GR*).

In Offline-GR (Algorithm 1), we assign requests to providers with the lowest cost and open new providers if necessary. We consider requests with the least number of potential providers first. Because our scheme tends to assign a request to a running resources and avoids opening unnecessary resources. Note that the cost of reserved VMs/storage is non-zero only when P_j is closed.

Algorithm 1. Offline-GR algorithm.

```
1 P_i is the set of providers that R_i can be assigned to;
2 Sort R in increasing order of |P_i|;
3 for each R_i ∈ R do
4   k* = argmin_{k ∈ {k|R_k ∈ P_i}} {Φ_k^i}
5   Assign R_k to P_k*;
6   if V_k* and S_k* is reserved in P_k* then
7     Active V_k* and S_k*;
8   end
9 end
```

6. Performance evaluations

In this section, we evaluate our algorithm using system settings referred to real-world Clouds, conduct simulation studies driven by real trace.

6.1. Simulation setup

YouTube is the most popular video serving website. We extract end user request patterns from YouTube trace (Zink et al., 2009). The traces used in our experiments are collected from a campus network with a total of six different traces. Since our main design goal is to significantly benefit mobile applications in content distribution and performance, we select one of the six traces, which have the larger number of overall requests. We use them in the main portion of our experiments to show how different design parameters may impact our strategies. The traces chosen for our experimental studies have different lengths, number of unique clients, videos and requests, to represent multiple types of

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

Table 3
YouTube trace characteristics.

Trace	Length (h)	# of clients	# of videos	# of req.
T5	336	16336	303,331	611,968

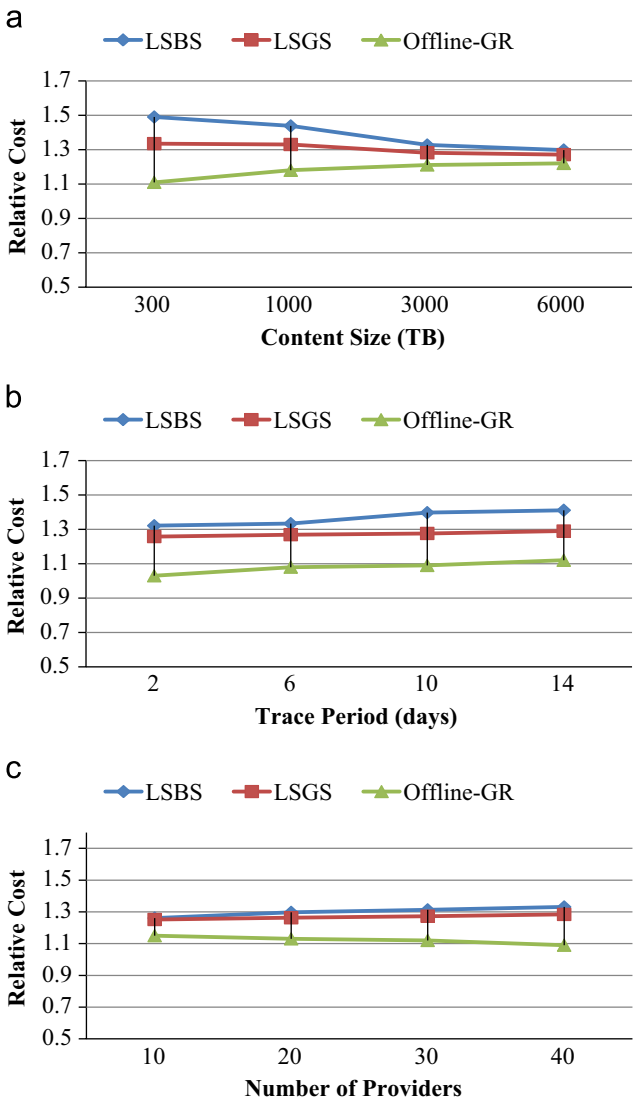

Fig. 3. Relative cost for proposed algorithms. (a) Varying content size, (b) Varying trace period and (c) Varying number of providers.

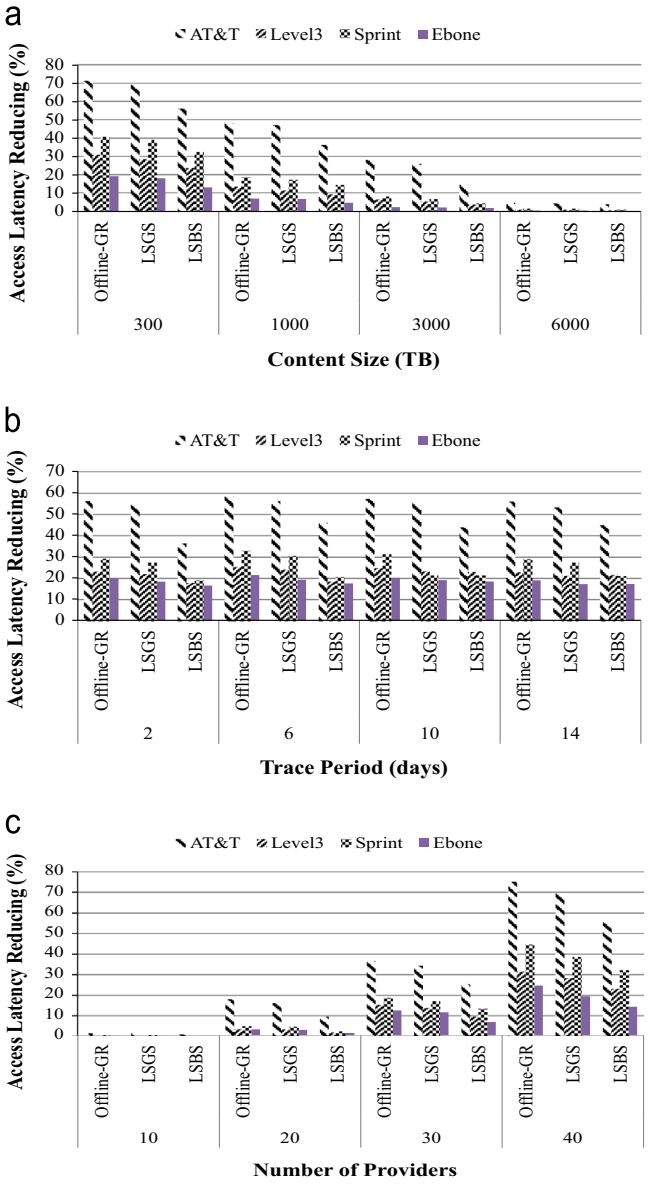

Fig. 4. Access latency evaluation over different network topologies. (a) Varying content size, (b) Varying trace period and (c) Varying number of providers.

workloads in real enterprise-level data centers. The summary of the characteristics of the trace (T5) is listed in Table 3.

In Fig. 3, we randomly construct an arbitrary network topology with 40 providers. In Figs. 4 and 5, we use Java to implement a simulator that constructs the system topology under several network topologies in north America: *AT&T*, *Level3*, *Sprint* and *Ebone* (Spring et al., 2004). For each problem instance we evaluate, we randomly pick one site which has the origin dataset and a subset of sites as cloud provides. We then place the group of chosen sites including the origin dataset and the cloud provides onto a metric field using the geographical information of each site. To place a server, we first look up its IP address in the GeoLite city database Geolite city database by maxmind and convert the IP address to its corresponding (latitude, longitude) pair. The cost function can be the performance cost like latency, dollar cost charged by cloud providers. We consider the case where latency, cost and resource utilization are the primary performance metrics. We choose a small number of providers and vary the number from 10 to 40. We show such a small number because it reflects the fact that there are few CSPs today.

6.2. Performance

In Fig. 3, we determine the cost effect of three heuristic algorithms under several scenarios by reporting the relative costs, defined as a ratio of total cost over the optimal cost. In Fig. 3(a) we observed that Offline-GR performs best and incurs the near-optimal cost with respects to the content sizes. LSGS and LSBS incur a little more service costs than Offline-GR. From Fig. 3(a), we find LSGS has the similar relative costs both in small content sizes and in large content sizes and is not sensitive to content size as much as LSBS and Offline-GR. This behavior of LSGS is expectable because LSGS serves each request by the cheapest replica even the content size varies. The LSBS is decreasing to lower relative cost with the content size larger because LSBS works by trying to

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

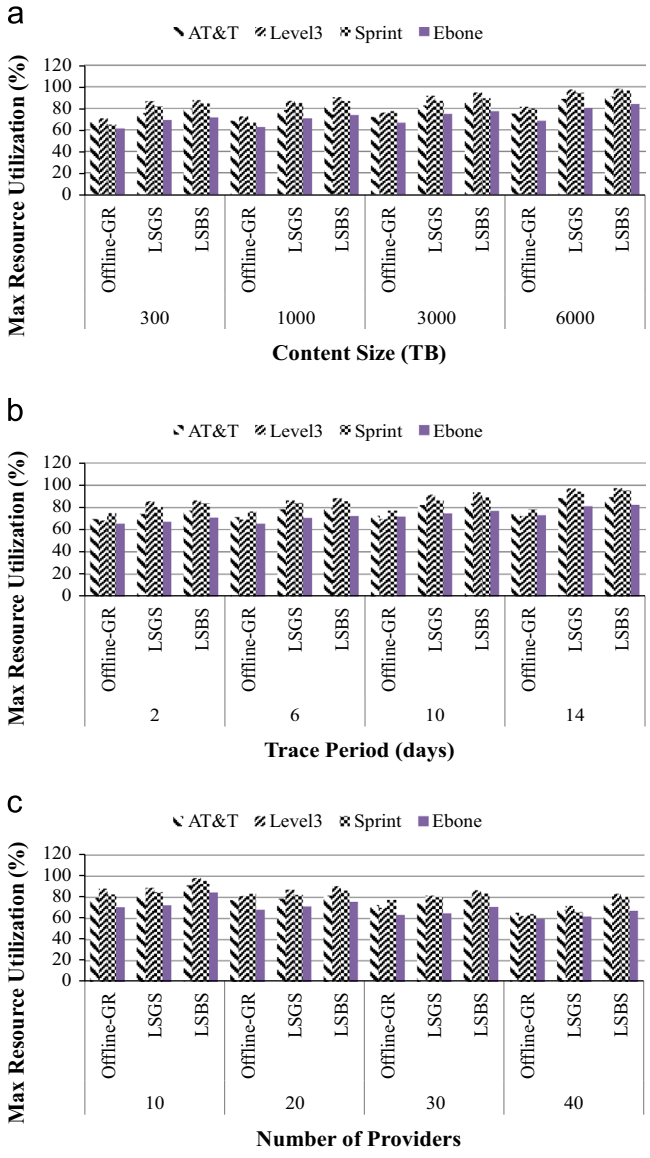

Fig. 5. Resource utilization evaluation over different network topologies. (a) Varying content size, (b) Varying trace period and (c) Varying number of providers.

minimize the maximum transferring cost by any replica: the more content the greater the room for improvement.

We then increase the trace period from 2 to 14 days, which would have different effects compared with the increasing content sizes. As shown in Fig. 3(b), the relative costs for LSBS and LSGS increase slightly while the relative costs of Offline-GR (still the best) does not show much variation. In the third case shown in Fig. 3(c), we vary the number of providers from 10 to 40. The relative costs of LSGS and LSBS are varied only slightly within a reasonable range of parameter values. The performance of Offline-GR shows best and its relative costs decrease slightly. In Fig. 3 we can observe that, by varying the content sizes, the trace period, and the number of providers, our heuristics behave very close to the optimal, moreover, for Offline-GR, the value of relative costs largely remains the same in all cases.

In the previous evaluations, the proposed three algorithms are able to obtain a cost effective content service. However, these evaluations are conducted in arbitrary network topologies. To determine the performance of the three heuristic algorithms, we evaluate our algorithms on different real-world network topologies which have the topological properties of different graphs. For instance, Ebone is the simplest graph. The backbones of AT&T has a hub-and-spoke structure with some shortcuts between nodes pairs. The topology of Level3 is almost a complete mesh, while Sprint is in between these two kinds.

In Fig. 4(a), we observe that the access latency improvements of AT&T over other three network topologies. And with the content sizes increasing, the performance improvements are not substantial. This is because when the total storage capacities are constrained, the larger content sizes mean the less storage capacities for the content replications thus leaving a little space for improvements. We then increase the trace period from 2 to 14 days, which has the same effect as increasing the content sizes (shown in Fig. 4(b)). In the third scenario shown in Fig. 4(c), we vary the number of providers from 10 to 40. These test illustrates an important factor that affects performance: the number of the CSPs. When we increase the total number of providers, contents have more replications and a request has more choices for provider selection. Moreover, the performance has different improvements: Our evaluation shows that Offline-GR achieves up to 71.3% of the maximum improvement of average access latency.

To better understand the results, we plot the maximum resource utilization to illustrate the level of utilization in the Cloud of Clouds. The resource utilization varies with respects to the service loads. Our findings are depicted in Fig. 5. It is interesting that the performance improvements are more relatively significant in more complex topologies. Higher resource capacity and more providers show more space for potential improvements. For example, by Offline-GR, Exodus is improved with this metric generally, the maximize resource utilization (59.27–70.36%) with the varying number of providers is less than Level3 (61.85–87.98%), which might be another important consideration for CoSPs. Note that Fig. 5 is also consistent with the claim that our heuristics show considerable improvements and close to the optimal under various resource conditions because the smaller utilization implies more requests would be serviced.

7. Conclusions

In this paper, we first exploit the local-search techniques to develop a suit of off-line and online heuristic algorithms to migrate the content services to facilitate the mobile accesses. The algorithms can take as inputs the content services and access information such as the batch requests and resource locations to make decisions on the content replications and placements. We then evaluate our algorithms via YouTube trace-based simulation, and show that our heuristics behave very close to the optimal under various cloud conditions in a cost effective way. Our solutions provide the cloud-based service providers with the flexibility that they can not only decide the duration of each resource to be rented at some vantage sites but also make a trade-off between the network cost and the caching cost to meet the constraints imposed either by the CSPs or by the CoSPs. As Cloud computing is gaining its prominence, we truly believe these benefits are more important than ever before to the success of the traditional network-based services migrating to Clouds.

References

Benoit A, Rehn-Sonigo V, Robert Y. Replica placement and access policies in tree network. IEEE Transactions on Parallel and Distributed Systems 2008;19 (2):1614–27.

Bienkowski M, Feldmann A, Jurca D, Kellerer W, Schaffrath G, Schmid S, Widmer J. Competitive analysis for service migration in VNets. In: Proceedings of the second ACM SIGCOMM workshop on virtualized infrastructure system and architectures (VISA); 2010. p. 17–24.

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006

Biran O, Corradi A, Fanelli M, Foschini L, Nus A, Raz D, Silvera E. A stable network-aware VM placement for Cloud systems. In: Proceedings of 12th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid); 2012. p. 498–506.

Borst S, Gupta V, Walid A. Distributed caching algorithms for content distribution networks. In: Proceedings of IEEE INFOCOM; 2010.

Broberg J, Buyya R, Tari Z. MetaCDN: Harnessing storage clouds' for high performance content delivery. Journal of Network and Computer Applications 2009;32(5):1012–22.

Calcavecchia NM, Biran O, Hadad E, Moatti Y. VM placement strategies for cloud scenarios. In: Proceedings of the fifth international conference on cloud computing (CLOUD); 2012. p. 852–9.

Celesti A, Tusa F, Villari M, Puliafito A. Improving virtual machine migration in federated cloud environments. In: Proceedings of the second international conference on evolving internet; 2010. p. 61–7.

Chen F, Guo K, Lin J, Porta TL. Intra-cloud lightning: building CDNs in the cloud. In: Proceedings of IEEE INFOCOM; 2012. p. 433–41.

Dai J, Hu Z, Li B, Liu J, Li B. Collaborative hierarchical caching with dynamic request routing for massive content distribution. In: Proceedings of IEEE INFOCOM; 2012. p. 2444–452.

Dán G. Cache-to-Cache: Could ISPs cooperate to decrease Peer-to-Peer content distribution costs?. IEEE Transactions on Parallel and Distributed Systems 2011;22(9):1469–82.

DiPalantino D, Johari R. Traffic engineering versus content distribution: a game theoretic perspective. In: Proceedings of IEEE INFOCOM; 2009. p. 540–8.

Dobson S, Denazis S, Fernandez A, Gaiti D, Gelenbe E, Massacci F, Nixon P, Saffre F, Schmidt N, Zambonelli F. Survey of autonomic communications. ACM Transactions on Autonomous and Adaptive Systems 2006;1(2):223–59.

Geolite city database by maxmind. URL ⟨http://www.maxmind.com/⟩.

Gupta H, Tang B. Data caching under number constraint. In: Proceedings of IEEE international conference on communications; 2006. p. 435–40.

Hirofuchi T, Ogawa H, Nakada H, Itoh S, Sekiguchi S. A live storage migration mechanism over wan for relocatable virtual machine services on clouds. In: Proceedings of the 9th IEEE/ACM international symposium on cluster computing and the grid (CCGRID); 2009. p. 460–5.

Jiang W, Zhang-Shen R, Rexford J, Chiang M. Cooperative content distribution and traffic engineering in an ISP network. In: Proceedings of SIGMETRICS/performance; 2009. p. 239–50.

Kalpakis K, Dasgupta K, Wolfson O. Optimal placement of replicas in trees with read, write, and storage costs. IEEE Transactions on Parallel and Distributed Systems 2001;12(6):628–37.

Levin M. Storage management disciplines are declining; June 2006. URL ⟨http://www.computereconomics.com/article.cfm?id=1129⟩.

Li W, Chan E, Feng G, Chen D, Lu S. Analysis and performance study for coordinated hierarchical cache placement strategies. Computer Communications 2010;33(15):1834–42.

Liu X, Datta A. Towards intelligent data placement for scientific workflows in collaborative cloud environment. In: Proceedings of the IEEE international symposium on parallel and distributed processing workshops and Phd forum (IPDPSW); 2011. p. 1052–61.

Madhyastha HV, Isdal T, Piatek M, Dixon C, Anderson T, Krishnamurthy A, Venkataramani A. iPlane: an information plane for distributed services. In: Proceedings of the seventh symposium on operating systems design and implementation (OSDI), USENIX Association; 2006. p. 367–80.

Mills K, Filliben J, Dabrowski C. Comparing VM-placement algorithms for on-demand clouds. In: Proceedings of the third international conference on cloud computing technology and science (CloudCom); 2011. p. 91–8.

Mortier R, Kiciman E. Autonomic network management: some pragmatic considerations. in: Proceedings of the SIGCOMM workshop on internet network management (INM), ACM, New York, USA; 2006. p. 89–93.

Oikonomou K, Stavrakakis I. Scalable service migration in autonomic network environments. IEEE Journal on Selected Areas in Communications 2010;28(1):84–94.

Pathan R Buyya. Resource discovery and request-redirection for dynamic load sharing in multi-provider peering content delivery networks. Journal of Network and Computer Applications 2009;32(5):976–90.

Peng C, Kim M, Zhang Z, Lei H. VDN: Virtual machine image distribution network for cloud data centers. In: Proceedings of IEEE INFOCOM; 2012. p. 181–9.

Ski rental problem. URL ⟨https://en.wikipedia.org/wiki/Ski_rental_problem⟩.

Spring N, Mahajan R, Wetherall D, Anderson T. Measuring ISP topologies with Rocketfuel. IEEE/ACM Transactions on Networking 2004;12(1):2–6.

suk Kee Y, Kesselman C. Grid resource abstraction, virtualization, and provisioning for time-targeted applications. In: Proceedings of the eighth IEEE international symposium on cluster computing and the grid (CCGrid); 2008. p. 324–31.

Vogels W. Beyond server consolidation. ACM Queue 2008;6(1):20–6.

Voorsluys W, Broberg J, Venugopal S, Buyya R. Cost of virtual machine live migration in clouds: a performance evaluation. In: Proceedings of the 1st international conference on cloud computing (CloudCom); 2009. p. 254–65.

Wang Y, Veeravalli B, Tham C-K. On data staging strategies for mobile accesses to cloud services. In: 4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC); 2011. p. 245–52.

Wang H, Wang F, Liu J, Groen J. Measurement and utilization of customer-provided resources for cloud computing. In: Proceedings of IEEE INFOCOM; 2012. p. 442–50.

Wang Y, Veeravalli B, Tham C-K. On data staging algorithms for shared data accesses in clouds. IEEE Transactions on Parallel and Distributed Systems 2013;24(4):825–38.

Wood T, Ramakrishnan KK, Shenoy P, van der Merwe J. Cloudnet: dynamic pooling of cloud resources by live wan migration of virtual machines. In: Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments (VEE); 2011. p. 121–32.

Wu Y, Wu C, Li B, Zhang L, Li Z, Lau FC. Scaling social media applications into geo-distributed clouds. In: Proceedings of IEEE INFOCOM; 2012. p. 684–92.

Zheng J, Ng TSE, Sripanidkulchai K. Workload-aware live storage migration for clouds. In: Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on virtual execution environments; 2011. p. 133–44.

Zink M, Suh K, Gua Y, Kurose J. Characteristics of YouTube network traffic at a campus network c measurements, and implications. International Journal of Computer and Telecommunications Networking 2009;53(4):501–14.

Please cite this article as: Zeng L, Wang Y. Optimization on content service with local search in cloud of clouds. Journal of Network and Computer Applications (2013), http://dx.doi.org/10.1016/j.jnca.2013.09.006