Greedy Heuristic for Replica Server Placement in Cloud based Content Delivery Networks

Jagruti Sahoo, Roch Glitho
CIISE, Concordia University, Montreal (QC), Canada
Email: jsahoo@encs.concordia.ca, glitho@ece.concordia.ca

Abstract- Recently, Cloud based Content Delivery Network (CCDN) has emerged as efficient content delivery architecture to provide content delivery services with improved Quality of Service (QoS), scalability and resource efficiency. Replica server placement is a key design issue in CCDNs and involves deciding the placement of replica servers on geographically dispersed cloud sites that minimizes the operational cost and satisfies QoS of the end-users. Since replica server placement problem is NP-hard, it is necessary to design an efficient heuristic for CCDNs. In this paper, we propose an efficient greedy heuristic for the replica server placement problem. The heuristic consists of two main procedures: placement and refinement. The placement procedure obtains an initial placement of replica servers on cloud sites with low operational cost. The refinement procedure removes the redundant cloud sites to reduce the operational cost further. The simulation results demonstrate that the proposed greedy heuristic outperforms the existing greedy heuristics in terms of computation time and the operational cost.

Keywords: **Cloud Computing; Content Delivery Networks (CDNs); Replica Server Placement**

I. INTRODUCTION

Content Delivery Networks (CDNs) are large distributed infrastructures of replica servers placed in strategic locations around the world [1] [2]. By replicating content of origin servers on replica servers, the content is delivered to end-users with reduced latency and the load on the origin servers is reduced significantly. By avoiding long distance transmissions of voluminous traffic such as videos, the network congestion is alleviated. CDNs also improve the content availability by providing multiple delivery points to the end-users. CDNs are used to deliver a variety of content including static/dynamic web pages, streaming audio/videos, news and software patches. Both small and large enterprises are highly dependent on CDNs to maintain and grow their business. Akamai [3] and Limelight [4] are the earliest commercial CDNs that still continue to deliver a large fraction of internet traffic today.

Despite their success, traditional CDNs have difficulty to cope with the recent exponential growth in rich and premium content and its consumers. According to a recent report by Cisco, the global IP traffic has increased fivefold over the past five years, and will increase threefold over the next five years [5]. The unprecedented content growth limits the scalability of traditional CDNs. In traditional CDNs, resources are over-provisioned in order to handle peak load. As a result, they lack resource efficiency. Moreover, resource provisioning requires significant amount of time because of the need to install dedicated hardware/software.

To overcome these limitations, CDNs start to rely more and more on emerging paradigms such as cloud computing [6] [7] resulting in a new architecture generally referred to as Cloud based CDN (CCDN). In a CCDN, the CDN provider also referred to as CCDN provider builds its replica servers by leasing resources (storage, compute and bandwidth) from one or more cloud providers. However, the CCDN provider may also play the role of cloud provider by using its own private cloud for providing CDN services. Cloud as virtualized pool of resources enables CCDNs to achieve the desired scalability. Moreover, due to elasticity and on-demand resource provisioning features of cloud computing paradigm, CCDNs can manage resources in an efficient way, yet able to react to fluctuating end-user demand. The pay-as-you-go pricing allows CCDNs to deploy replica servers with minimal costs. Commercial cloud based CDNS include Rackspace [8] , Amazon CloudFront [9], and CloudFlare [10]. MetaCDN [11], ActiveCDN [12] and CoDaaS [13] are examples of CCDN architectures proposed in the literature.

The overall efficiency of a CCDN is achieved when it is able to deliver content to end-users with QoS satisfactions while minimizing its operational cost. Providing QoS guarantee with a lower operational cost however depends on a meticulous placement of replica servers across cloud sites and connecting end-users to one of the replica servers, a key design problem referred to as replica server placement. It is modelled as an optimization problem to find the optimal placement (i.e. number and location) of replica servers. In the literature, replica server placement has been modelled after several well-known graph theoretic problems such as facility location problem [14], K-median problem [14] [15] and their variants. For general graphs, facility location and K-median have been shown as NP-hard [16]. As a result, efficient heuristic is needed to find placement of replica servers. To design heuristic, greedy approach [17] [15] [18] [19] [20] has gained its wide-spread use in traditional CDNs as well as CCDNs due to its simplicity and ability to produce good enough solutions.

In this paper, we formulate the replica server placement problem in CCDNs as an Integer Linear Program (ILP). We propose a new greedy heuristic for solving the problem. Our heuristic includes two procedures. First, we decide the placement and then refine it by removing cloud sites and re-assigning end-users to the already chosen cloud sites. We design our greedy heuristic in a way that it is superior to the existing greedy heuristics in CCDNs in terms of computation time and cost of the placement.

978-1-5090-0679-3/16/$31.00 ©2016 IEEE

The reminder of the paper is organized as follows. In Section 2, we provide a brief overview of the related works. Section 3 presents the proposed heuristic. Section 4 depicts the performance evaluations. Finally, in Section 5, we conclude the paper and outline some future works.

II. RELATED WORKS

Rappaport and Raz [21] address replica server placement in CCDNs by modeling it as soft capacitated connected facility location problem. As evident from the model, the storage cost, delivery cost and update cost, which can be collectively known as operational cost is minimized. Apart from replica server locations, they also find an optimal tree that interconnects the replica servers to ensure consistency and synchronization. Minimization of operational cost ensures cost efficiency in CCDNs. The authors in [21] also define a λ-loaded facility location problem by introducing a constraint that the load served by each replica server must be above a predefined threshold. They present a constant factor approximation algorithm for the soft capacitated facility location problem by using approximation algorithm for the uncapacitated facility location problem and approximation algorithm for minimum Steiner tree problem. The main limitation of this work is the lack of QoS guarantee to the end-users. As a result, it cannot achieve the overall efficiency of CCDN.

Chen *et al.* [20] address replica server placement wherein the objective is to minimize operational cost while satisfying QoS of end-users and connecting every end-user to at least one replica server. The operational cost includes the storage cost and update cost (incurred by replicating content from origin server). They present various offline (Greedy Site(GS) and Greedy User(GU)) and online algorithms (e.g., Greedy Request Only and Greedy Request with Pre-allocation) The GS heuristic is based on set covering algorithm and places replica servers incrementally that accommodates maximum number of end-users and satisfies QoS. On the other hand, in GU heuristic, for each end-user the replica server that incurs the lowest cost and satisfies QoS is placed. Replica server placement finishes once all end-users are connected to their replica servers. Note that offline algorithms pre-deploy replica servers before the end-user request is received. The online algorithm Greedy Request Only executes the GU heuristic to deploy a suitable replica server based on the first request received from an end-user. However, the time needed to deploy a replica server may violate QoS. Hence, once the first request is received from end-user, if the best replica server for the end-user is not yet placed, the request is served from the origin server.

In [22], replica server placement for CCDN includes finding the number and location of replica servers such that the operational cost (storage cost, delivery cost, update cost) is minimized and QoS requirements of end-users are satisfied. The replica server placement is formulated using MIP (Mixed Integer Programming). The SNA-GVSP (Social Network Analysis Inspired Greedy Virtual Surrogate Placement) heuristic is designed based on centrality metric motivated by SNA. In particular, the replica servers are prioritized based on SPBC (Shortest Path Betweeness Centarlity) metric which is the ratio of the number of shortest paths that pass through a replica server to the total number of shortest paths. In the SNA-GVSP heuristic, end-users are assigned to replica servers that maximize the average SPBC of the set of already selected replica servers. The SNA-GVSP heuristic does not consider cost. Hence, it cannot find a cost-effective placement of replica servers in CCDNs.

In [23], the optimal CDN provisioning is presented for finding optimal configuration for video CDN that uses layered cache servers. The optimal configuration involves number of cache servers, size of the caches and amount of peering bandwidth needed to serve end-users. The objective is to provision cache servers in a way that minimizes the storage cost and bandwidth cost. However, the authors do not incorporate QoS parameter in their replica server placement model. As a result, the model is less suitable for CCDNs.

Optimal deployment of video servers is addressed in He *et al.* [24]. The authors jointly minimize the operational cost and latency. Since the cost of using other computing resources (e.g. CPU, memory) is negligible in comparison to bandwidth cost only the latter is considered as the operational cost. The joint minimization problem is tackled by an offline algorithm designed based on Nash bargaining solution. The algorithm considers the predicted end-user demand in future time slots. An online algorithm is also designed to minimize operational cost and probability of under-provisioning of resources. At each time slot, the predicted end-user demand is used to trigger the adjustment or redeployment of resources in the data centers.

In [25], the replica server placement problem is solved by considering dynamic pricing of resources and fluctuating end-user demands. Dynamic pricing may occur as a result of varying electricity costs in regional power systems and the cloud providers thus vary the resource leasing prices to obtain stability in their profit margins. The authors aim to find the optimal number of replica servers in data centers to meet QoS while minimizing the operational cost. The problem is formulated as a joint optimization problem consisting of finding the optimal replica server placement and assignment of end-users to suitable replica servers in order to satisfy Service Level Agreements (SLAs). Model predictive control approach is used to find the optimal placement of replica servers. Wang *et al.* [26] address the placement of replica servers by finding an optimal lease schedule in order to accommodate spatial and temporal variations in end-user demand. The authors incorporate locality awareness i.e. regional (i.e. ISPs) distribution of end-users in to the placement decisions. The latency is reduced by leasing replica servers placed on local cloud sites to serve as many end-users as possible. The algorithm minimizes the operational cost and also the cross-region traffic. The authors use an enhanced DFS (Depth First Search) algorithm to solve the optimization problem.

Table. I shows the evaluation of the related works with respect to two requirements:1) Operational Cost minimization, and 2)QoS satisfaction.

Table. I. Evaluation of Replica Placement Heuristics

Algorithm	Operational Cost Minimization	QoS Satisfaction
Chen *et al.* [20]	Yes	Yes
Rappaport and Raz [21]	Yes	No
Papagianni et al. [22]	No	Yes
Mokhtarian et al. [23]	Yes	No
He et al. [24]	Yes	No
Zhang et al. [25]	Yes	Yes
Wang et la. [26]	Yes	Yes

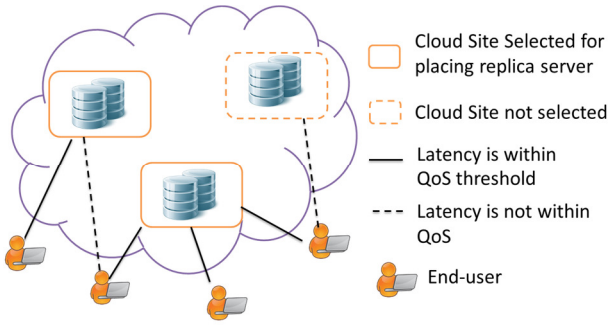

Fig. 1. Replica Server Placement in CCDN

III. REPLICA SERVER PLACEMENT HEURISTIC

A. Replica Placement Model

We consider a scenario where CCDN provider decides to deploy replica servers by leasing resources on geographically dispersed cloud sites. Each cloud site represents a potential location of replica server. The CCDN provider aims to deploy the replica servers with lower operational cost while providing content to end-users with QoS satisfactions. Replica server placement is modelled as an optimization problem wherein the objective is to find optimal location and number of cloud sites and assignment of end-users to one of the cloud sites such that the operational cost is minimized and the QoS thresholds of all end-users are satisfied. Fig. 1 shows replica server placement scenario in CCDN.

We define the operational cost in CCDN as sum of storage cost and delivery cost. The storage cost is incurred by leasing storage resources from the cloud provider. The delivery cost is the cost of delivering content from the deployed replica servers to the end-users. In our replica server placement model, the delivery cost is incurred by leasing bandwidth resources from the cloud provider.

Our replica server placement model is close to Uncapacitated Facility Location (UFL) model [14]. In UFL model, each facility has an opening cost and a city is served by a facility by incurring a delivery cost. The storage cost in our replica server placement model is the equivalent of opening cost in UFL besides the fact that the opening cost is one time cost. A city is equivalent to an end-user in our model. Our replica server placement model incorporates QoS constraints to meet the viewing expectations of the end-users. Relaxing the QoS constraint makes our replica server placement model an instance of the UFL. Since UFL is NP-hard, our replica server placement model is also NP-hard.

B. Problem Formulation

The inputs to solve the above replica server placement model include a network topology of cloud sites and end-users, demand of end-users, QoS threshold of end-users, size of replica to be hosted on the replica servers and the leasing costs of cloud sites. Formally, let $G=(V, E)$ be a graph, where V is a set of nodes and E is the set of edges. We define two set of nodes: set of end-users H and set of cloud sites (i.e. potential location of replica servers) F, where $V = H \bigcup F$, $|F|=M$, $|H|=N$. Each end-user node $j \in H$ contains a demand w_j that denotes the total load (i.e. number of requests generated per a time unit multiplied by size of replica requested) originated at that node. Each edge $e \in E$ represents the logical communication link between two nodes. Each edge between a cloud site and end-user has a QoS metric. We consider latency as the QoS metric. Latency can be approximated by round-trip times (RTT), hop-count or geographical distance. L_{ij} denotes the QoS metric of the edges (i, j) between any cloud site $i \in F$ and any end-user $j \in H$. Each end-user j is associated with an acceptable QoS threshold denoted as Q_j. We assume a replica of size W. Let α_i^S and α_i^B denote the leasing cost of unit storage and leasing cost of unit download bandwidth of cloud site $i \in F$.

We formulate the replica server placement as an ILP having the following binary decision variables.

$$y_i = \begin{cases} 1 & \text{if cloud site } i \text{ is selected for placing replica server} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$x_{ij} = \begin{cases} 1 & \text{if end-user } j \text{ is assigned to cloud site } i \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The ILP is stated as follows.

Min

$$C = \sum_{i \in F} W \alpha_i^S y_i + \sum_{i \in F} \sum_{j \in H} w_j \alpha_i^B x_{ij} \quad (3)$$

Subject to:

$$\sum_i x_{ij} = 1 \ \forall j \in H \quad (4)$$

$$x_{ij} \le y_i \ \forall \ i \in F, j \in H \quad (5)$$

$$\sum_i x_{ij} L_{ij} \le Q_j \ \ j \in H \quad (6)$$

In the objective function (3), the first term denotes the storage cost and the second term denotes the delivery cost. Constraint (4) ensures that an end-user must be assigned to exactly one of the cloud sites. Constraint (5) indicates that an end-user j is assigned to a cloud site i only if the cloud site has been selected. Constraint (6) ensures that for each end-users j, the QoS metric of the edge between end-user j and cloud site i

to which it is assigned must satisfy the QoS threshold of end-user *j*.

C. Proposed Heuristics

We propose a heuristic called Least Usage Greedy (LUG) for the above replica server placement model. The heuristic is based on greedy approach. Unlike existing greedy heuristics (GS [20], GU [20]), the proposed greedy heuristic has high computational efficiency and high cost-efficiency. GS and GU heuristics are both computationally inefficient. This is because GS always finds highest utility cloud site in each iteration. Similarly, GU finds minimum cost cloud site for each end-user. Among GS and GU, the GU heuristic lacks cost-efficiency. The reason is that in search of best cloud-sites for each end-user, it ends up placing more cloud sites.

Algorithm 1 Placement

1. Sort F in ascending order of storage cost
2. Sort F again in ascending order of unit delivery cost.
3. for each cloud site $i \in F$
4. for each end-user $j \in U_i$ //potential end-users of cloud site i
5. if end-user j is not yet assigned
6. $A_i \leftarrow A_i \cup \{ j \}$ //Assign end-user j to cloud site i
7. end if
8. end for
9. if $|A_i| \neq \phi$ then
10. $S \leftarrow S \cup \{ i \}$ //Select cloud site i
11. end if
12. end for
13. Construct solution X from sets S and A_i for each chosen cloud site $i \in S$
14. Return Solution X

For each cloud site, we define its potential end-users as the end-users for which the latency between the cloud site and the end-users remains below the QoS thresholds of the end-users [20]. Similarly, for each end-user, its potential cloud sites are defined as the ones which satisfy the QoS threshold of the end-user. The LUG heuristic consists of two procedures: placement and refinement. The placement procedure involves selecting cloud sites in the ascending order of their operational cost and assigning all potential end-users that are not yet assigned to the cloud sites. The refinement procedure is intended to improve the cost of the solution returned by the placement procedure. Thus, this phase aims at removing cloud sites whose removal results in cost improvement and re-assigning end-users to other cloud sites already selected. The sites are removed until no cost improvement is observed. For re-assignment, end-users are always re-assigned to their next best site in terms of unit delivery cost (i.e. unit leasing bandwidth cost). The solution obtained by the refinement procedure is considered as the solution of LUG. The LUG heuristic is shown in Algorithm 3 and the details are as follows.

The placement procedure of LUG heuristic starts with sorting all potential cloud sites in the ascending order of their operational cost. First, the cloud sites are sorted in the ascending order of their storage cost and then sorted again in the ascending order of their unit delivery cost. This two-level sorting operation ensures the differentiation among two cloud sites with same unit delivery cost but with different storage costs. Moreover, due to sorting operation, among a group of cloud sites having same unit delivery cost, the cloud site having the lowest storage cost is considered first. Once, the cloud sites are sorted, we start to assign end-users starting with the lowest cost cloud site. For each cloud site, all its potential end-users that are not yet assigned are assigned to it. A cloud site is chosen for placing replica servers if at least one end-user is assigned to it. The selection of cloud sites and assignment of end-users to one of their potential cloud sites continues until all end-users are assigned. The placement procedure is illustrated in Algorithm 1.

Algorithm 2 Refinement (X)

1. $C^{cur} \leftarrow$ Cost of solution X
2. for each end-user $j \in H$
3. Form set N_j of potential cloud sites in S excluding the current cloud site of j
4. $s_j^{\min} \leftarrow$ current best site in N_j
5. end for
6. for each cloud site $i \in$ S
7. $Cnt \leftarrow NULL$
8. for each end-user $j \in A_i$
9. if $s_j^{\min} \neq NULL$ then
10. $Cnt \leftarrow Cnt + 1$
11. end if
12. end for
13. if $Cnt = |A_i|$ then
14. $C^{new} \leftarrow$ Cost after removing i and re-assigning j to $s_j^{\min}$
15. if $C^{new} \leq C^{cur}$ then
16. for each end-user $j \in A_i$
17. $A_{s_j^{\min}} = A_{s_j^{\min}} \cup \{ j \}$ //Re-assign end-user j //to cloud site i
18. Update $s_j^{\min}$ //Next best cloud site for //end-user j
19. end for
20. $S = S - \{ i \}$ //Remove cloud site i
21. $C^{cur} \leftarrow C^{new}$
22. end if
23. end if
24. end for
25. Construct solution X^* from sets S and A_i for each chosen cloud site $i \in S$
26. Return X^*

Algorithm 3 Least Usage Greedy ()

1. $X \leftarrow$ Placement ()
2. $X^* \leftarrow$ Refinement (X)
3. Return Solution X^*

The solution obtained by the placement procedure is passed as input to the refinement procedure. We term this solution as the initial solution. The refinement procedure checks for cost improvement by removing cloud sites and re-assigning end-users to other cloud sites in the initial solution. Before refining the solution i.e. removing sites and re-assigning end-users, the set of potential cloud sites to which an end-user can be re-assigned is determined for all end-users. The set is also sorted in terms of unit delivery cost and always provides the next best cloud site for re-assigning an end-user. Note that, the set of cloud sites in the initial solution is already sorted in terms of unit delivery cost. Starting with the lowest unit delivery cost cloud site, for each cloud site in the initial solution, the refinement procedure checks the possibility of removing site and re-assigning its end-users to other sites.

For each cloud site *i*, the operation is described as follows. First, we check if next best cloud site exists for all its end-users. The next best cloud site of an end-user is its next lowest unit delivery cost cloud site. Note that, for an end-user, the cloud site to which it is currently assigned is the lowest unit delivery cost site among the remaining cloud sites. If next best cloud site exist for all end-users of *i*, then the cost that will incur after removing *i* and re-assigning all its end-users to their next best cloud sites is computed. If the cost is improved, then site *i* is removed and its end-users are re-assigned to their next best sites. Once end-users are re-assigned, their next best site is updated. If cost does not improve, the next cloud site is checked for possibility of refining the solution further. The refinement procedure is shown in Algorithm 2. Table. II shows some of the notations used in the algorithms. The complexity of LUG heuristic is $O(M*N)$.

Table.II. Summary of Notations

Notations	Meaning
U_i	Set of potential end-users of cloud site *i*
S	Current set of cloud sites chosen for placing the replica servers.
A_i	Current set of end-users assigned to site $i \in S$
X	Solution returned by Algorithm 1. It consists of set S and sets A_i for each chosen cloud site $i \in S$
X^*	Solution returned by Algorithm 2 and is also the final solution. It has same structure as *X*.

IV. PERFORMANCE EVALUATION

In this section, we study the performance evaluation of the proposed LUG heuristic and assess its effectiveness under various scenarios. We used real leasing prices of commercial storage cloud providers in the evaluation. We first introduce the experiment set up followed by the metrics for evaluation. Then, we present the simulation results.

A. Experiment Set up

We compare our heuristic with two well-known heuristics for replica server placement in CCDN. They are Greedy-Site (GS) [20] and Greedy-User (GU) [20]. We developed a simulator using C programming language to implement LUG, GS and GU heuristics.

The parameters in our evaluation are the number of cloud sites, number of end-users, replica size, end-user load and QoS threshold of end-users. Among these parameters, we vary the number of cloud sites, number of end-users and the QoS threshold while keeping other parameters at their default values.

The cloud sites and end-users are randomly generated in a geographic space. We generate different scenarios by varying number of cloud sites from 30 to 70 and by varying number of end-users from 100 to 700. For each end-user, we set the default load generated to 1.5 GB per month. We consider a default replica size of 5 GB.

Since, latency information is subject to heavy fluctuations over time, we use the geographic distance between a cloud site location and an end-user location to compute the latency. As stated in [27] round-trip delays (RTT) between a cloud site *i* and end-user *j* can be approximated by using the following expression.

$$RTT\ (i, j) = 0.02 * Dist\ (i, j) + 5 \quad (7)$$

where, $Dist(i, j)$ represents the geographic distance between location of cloud site *i* and location of end-user *j*. Note that our heuristic is an offline heuristic and is executed at certain time intervals (e.g. one month). Hence, distance based latency estimation is suitable for solving our optimization model. Each end-user is assigned a QoS threshold in the default range of 45ms-60ms. We vary the QoS threshold from 45ms to 75ms in steps of 5 ms. We run 20 experiments and compute the average to obtain the performance graphs.

To place a replica server on a cloud site, the CCDN provider incurs the operational cost (i.e. storage cost and delivery cost).The storage cost is based on unit storage price of the cloud provider while delivery cost is calculated based on unit download price of the cloud provider. In our evaluation, we use the leasing prices of commercial storage cloud providers. The leasing prices of Amazon S3, Windows Azure and Rackspace in 2015 are shown in Table-III. In case of Amazon S3, the bandwidth prices are applicable up to 10 TB of data downloaded per month, whereas in case of Windows Azure, the bandwidth prices are applicable for 5GB-10TB data downloaded per month. To ensure maximum diversity in the price offerings, we consider all distinct prices of the cloud providers. We assign random combination of the prices shown in Table-III to the cloud sites in all scenarios.

We consider the time unit of one month to compute the operational cost. Please also note that all the storage cloud providers shown in Table-III allow leasing of resources (storage as well as download bandwidth) on a monthly basis.

B. Performance Metrics

1) Computation Time: It is defined as the time needed to execute the heuristics. This metric indicates the computational efficiency of a heuristic.

2) Cost: It is the value of the objective function in (3) which is the total operational cost incurred by the CCDN provider in deploying replica servers on cloud sites and serving end-users. It is defined in the unit of dollar/month.

The computation time is measured with respect to number of cloud sites and number of end-users whereas the cost is measured with respect to the number of cloud sites, number of end-users, and the maximum QoS threshold. When one parameter is considered, we set all other parameters to their default values as provided in the experiment set up.

C. Results and Discussions

Fig. 2 shows the computation time of LUG, GS and GU heuristic with respect to number of cloud sites. We observe that for all three heuristics, the computation time increases with increase in number of cloud sites. Among the three heuristics, GS heuristic has the highest computation time. The proposed heuristic LUG, however, requires the lowest amount of time to execute and shows a very slow increase as number of cloud sites increases. The computation time of GU heuristic lies between that of GS and LUG. We notice that both GS and GU require significantly higher computation time than LUG. For the scenario of 70 cloud sites, the computation time of GS and GU are respectively 11 times and 5 times higher than the computation time of LUG.

Fig. 3 shows the computation time of LUG, GS and GU heuristics with respect to number of end-users. We observe that both GS and GU are very sensitive to the number of end-users. Their computation time increases almost linearly with respect to the number of end-users. On the other hand, LUG is least affected by increase in number of end-users. Once again, the computation time of GS and GU is significantly higher than that of LUG. For the scenario having largest number of end-users (i.e. 700), the computation time of GS and GU are respectively 12 times and 9 times higher than that of LUG. Fig. 2 and Fig. 3 justifies that the proposed heuristic is computationally inexpensive.

Fig. 4 shows the cost for all three heuristics: LUG, GS and GU for different number of cloud sites. We notice that with increase in number of cloud sites, all three heuristics achieve improvement in their respective cost values. Note that, during generation of scenarios, the leasing prices remain same although we increase the number of cloud sites. Thus, more the number cloud sites available, the higher the number of low cost sites and this is the reason behind cost reduction in case of all heuristics.

We observe that GU experiences the highest cost of replica server placement. GU selects the minimum cost site for each end-user. For each end-user, the candidate cloud sites are evaluated based on different costs. If a cloud site has been already chosen, only delivery cost is considered; otherwise both storage and delivery cost are considered. Due to different cost parameters, the minimum cost site may not be the best site among the candidate sites. As a result, higher cost is incurred in case of GU. On the other hand, the proposed heuristic LUG outperforms GU. The rationale behind lower cost in LUG is that it always selects the cloud site having lower cost using the same cost parameter. Moreover, once the cloud sites are chosen, a refinement procedure removes the sites as long as cost is improved, yet all end-users are accommodated. We also observe that LUG and GS obtain same cost, while slightly outperforming each other in some scenarios. The similar behavior of LUG and GS is the fact that both of them avoid selecting cloud sites from an end-user perspective. However, as GS is computationally expensive than LUG, similar cost performance of LUG ad GS is justified. On average, LUG and GS achieves 16% and 17% improvement in cost over GU respectively.

Fig. 5 shows cost of replica server placement with respect to number of end-users. We observe that for all three heuristics LUG, GS and GU, cost increases as number of end-user increases. This is due to increase in delivery cost as more bandwidth resources are required to deliver content. LUG and GS outperforms GU for all end-user scenarios. We observe that cost improvement of LUG and GS over GU gets higher as number of end-users increases. In case of LUG and GS, maximum cost improvement over GU is around 17% which corresponds to the scenario of 700 end-users. The reason solely lies on the method used by GU which repeats the cloud site selection for each end-user instead of considering one cloud site at each step. The larger the number of end-users, higher the extra cost added to the solution returned by GU.

Table. III. Storage Cloud Provider Prices in 2015

Storage Cloud Provider	Amazon S3						Windows Azure			Rackspace
	Regions						Regions			Regions
Type of Resource	US (West)	Sao Paulo	Frankfurt	Singapore	Tokyo	Sydney	US (West)	Brazil South	Australia South East	All Regions
Storage (dollar/GB/ Month)	0.033	0.0408	0.0324	0.03	0.033	0.0330	0.07	0.1	0.08	0.12
Out-going Bandwidth (dollar/GB/ Month)	0.09	0.09	0.09	0.12	0.14	0.140	0.087	0.138	0.181	0.12

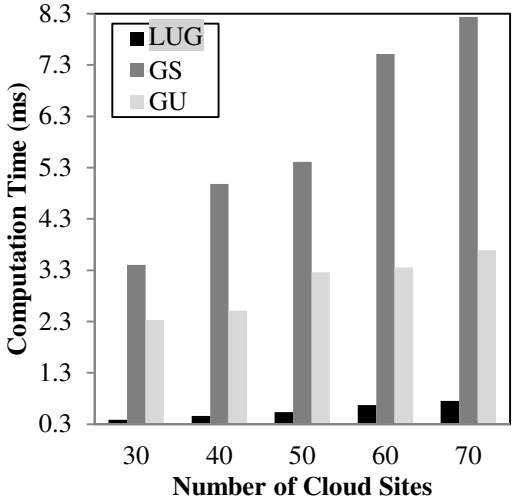

Fig. 2. Computation Time vs Number of Cloud sites

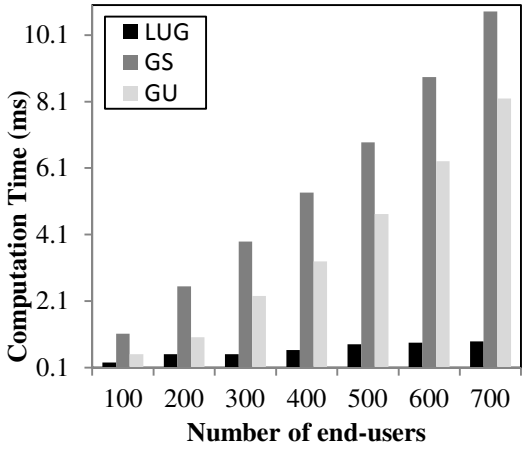

Fig. 3. Computation Time vs Number of end-users

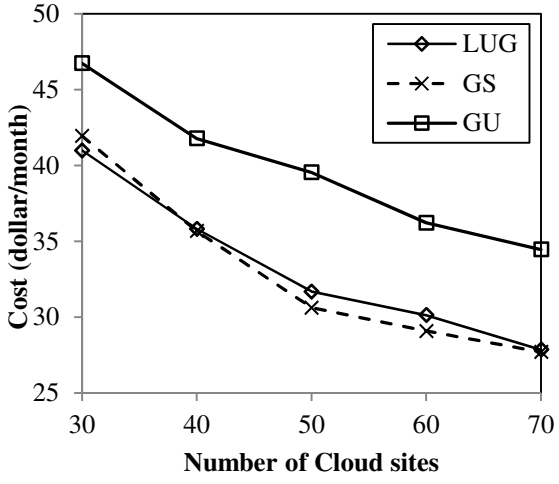

Fig. 4. Cost vs Number of Cloud Sites

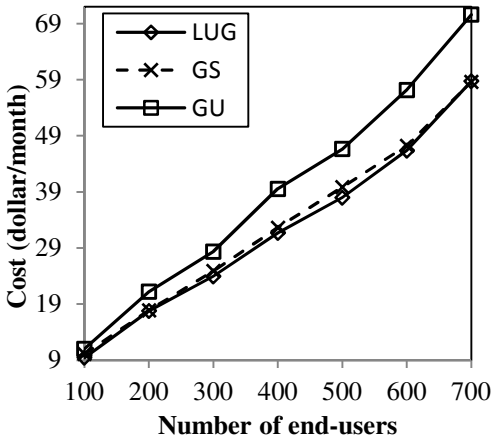

Fig. 5. Cost vs Number of End-users

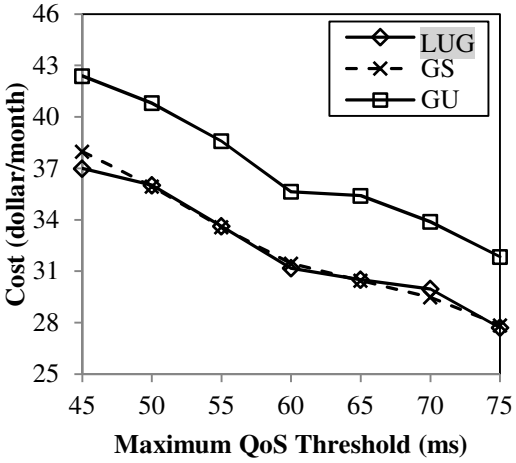

Fig. 6. Cost vs Maximum QoS Threshold

Fig. 6 shows cost with respect to maximum QoS threshold. We notice that as the QoS threshold increases, all three heuristics experience a reduction in cost. GU achieves a reduction of 24.93% at the highest value of maximum QoS threshold (i.e. 75ms), whereas LUG and GS achieve a reduction of 26.69 % and 25.07 % respectively at the same scenario. The rationale behind cost reduction is that by increasing the maximum QoS threshold, more cloud sites can satisfy the QoS requirements of end-users. In other words, end-users have more potential sites than before. As a consequence, it is possible to find cloud site with lower cost compared to when end-users have strict QoS requirement. As shown in Fig. 6, both LUG and GS outperform GU, where the cost difference of LUG and GS from GU remains same for all QoS scenarios.

V. CONCLUSION AND FUTURE WORKS

In CCDNs, the replica servers must be placed in a way that minimizes the operational cost of the CCDN provider and satisfies QoS of end-users. In this paper, we propose a new greedy heuristic to solve the replica server placement problem in CCDN. Our simulations show that the proposed heuristic achieves lower operational cost than the existing greedy based heuristics. Moreover, it has very low computation time, hence

can be applied to large-scale CCDNs involving large number of end-users and cloud sites.

In future, we plan to extend the greedy strategy to design an on-line heuristic for replica server placement in cloud based CDN. Moreover, we will also design a load balanced assignment of end-users to the replica servers to ensure effective resource utilization in CCDNs.

REFERENCES

[1] M. Pathan, R. Buyya and A. Vakali, "Content delivery networks: State of the art, insights, and imperatives," in *Content Delivery Networks*, Springer Berlin Heidelberg, 2008, pp. 3-32.

[2] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM,* vol. 49, no. 1, pp. 101-106, 2006.

[3] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman and B. Weihl, "Globally distributed content delivery," *Internet Computing, IEEE,* vol. 6, no. 5, pp. 50-58, 2002.

[4] "Limelight Networks," [Online]. Available: limelight.com.

[5] "The Zettabyte Era—Trends and Analysis," Cisco Visual Networking Index white paper, 2013.

[6] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *SIGCOMM Comput Commun Rev,* vol. 39, no. 1, pp. 50-55, 2008.

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A view of cloud computing," *Communications of the ACM,* vol. 53, no. 4, pp. 50-58, April 2010.

[8] "Accelerate your websites, web applications, and media delivery," Rackspace US, Inc.,, 2015. [Online]. Available: http://www.rackspace.com/cloud/cdn-content-delivery-network. [Accessed 30 August 2015].

[9] "AWS | Amazon CloudFront CDN - Content Delivery Network & Streaming," Amazon Web Services, Inc. , [Online]. Available: //aws.amazon.com/cloudfront/. . [Accessed 17 Aug 2015].

[10] "Content delivery network," CloudFlare, Inc., [Online]. Available: https://www.cloudflare.com/features-cdn. [Accessed 30 Sept 2015].

[11] J. Broberg, R. Buyya and Z. Tari, "MetaCDN: Harnessing 'Storage Clouds' for high performance content delivery," *J. Netw. Comput. Appl.,* vol. 32, no. 5, pp. 1012-1022, 2009.

[12] S. R. Srinivasan, J. W. Lee, D. L. Batni and H. G. Schulzrinne, "ActiveCDN: Cloud Computing Meets Content Delivery Networks," Columbia University, 2011.

[13] Y. Jin, Y. Wen, G. Shi, G. Wang and A. V. Vasilakos, "CoDaaS: An experimental cloud-centric content delivery platform for user-generated contents," in *International Conference on Computing, Networking and Communications (ICNC)*, 2012.

[14] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis and A. Bestavros, "Distributed Placement of Service Facilities in Large-Scale Networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM)*, pp.2144-2152, 6-12 May 2007.

[15] L. Qiu, V. Padmanabhan and G. Voelker, "On the placement of Web server replicas," in *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Anchorage, AK, 2001.

[16] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems, part II: p-medians," *SIAM Journal on Applied Mathematics,* vol. 37, p. 539–560, 1979.

[17] S. Choi and Y. Shavitt, "Proxy location problems and their generalizations," in *Proceedings of the Distributed Computing Systems Workshops 2003 23rd International Conference on*, pp.898-904, 19-22 May 2003.

[18] T. V. Nguyen, C. T. Chou and P. Boustead, "Provisioning content distribution networks over shared infrastructure," in *2003. ICON2003. The 11th IEEE International Conference on Networks*, pp.119-124, 28 Sept.-1 Oct. 2003.

[19] Y. Chen, R. H. Katz and J. D. Kubiatowicz, "Dynamic Replica Placement for Scalable Content Delivery," in *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS 02), LNCS 2429, Springer-Verlag*, 2002, pp. 306–318.

[20] F. Chen, K. Guo, J. Lin and T. La Porta, "Intra-cloud Lightning: Building CDNs in the Cloud," in *Proceedings of IEEE INFOCOM*, Orlando, Forida, 2012.

[21] A. Rappaport and D. Raz, "Update aware replica placement," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Zurich, 2013.

[22] C. Papagianni, A. Leivadeas and S. Papavassiliou, "A Cloud-Oriented Content Delivery Network Paradigm: Modeling and Assessment," *IEEE Transactions on Dependable and Secure Computing,* vol. 10, no. 5, pp. 287-300, 2013.

[23] K. Mokhtarian and H.-A. Jacobsen, "Server Provisioning in Content Delivery Clouds," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pp.65-72, June 27 2015-July 2 2015.

[24] J. He, D. Wu, Y. Zeng, X. Hei and Y. Wen, "Toward Optimal Deployment of Cloud-Assisted Video Distribution Services," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 23, no. 10, pp. 1717-1728, , Oct. 2013.

[25] Q. Zhang, Q. Zhu, M. Zhani, R. Boutaba and J. Hellerstein, "Dynamic Service Placement in Geographically Distributed Clouds," *Selected Areas in Communications, IEEE Journal on,* vol. 31, no. 12, pp. 762-772, December 2013.

[26] F. Wang, J. Liu, M. Chen and H. Wang, "Migration Towards Cloud-Assisted Live Media Streaming," *IEEE/ACM Transactions on Networking,* 2014.

[27] M. Hu, J. Luo, Y. Wang and B. Veeravalli, "Practical Resource Provisioning and Caching with Dynamic Resilience for Cloud-Based Content Distribution Networks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 8, pp. 2169-2179, 2014.