A Latency-based Object Placement Approach in Content Distribution Networks*

George Pallis, Athena Vakali, Konstantinos Stamos, Antonis Sidiropoulos, Dimitrios Katsaros, Yannis Manolopoulos
Department of Informatics
Aristotle University of Thessaloniki,
54124, Thessaloniki, Greece
gpallis@ccf.auth.gr, { avakali, kstamos, asidirop, dkatsaro, manolopo}@csd.auth.gr

Abstract

Content Distribution Networks (CDNs) are increasingly being used to disseminate data in today's Internet aiming at reducing the load on the origin server and the traffic on the Internet, and ultimately improving response time to users. In this direction, crucial data management issues should be addressed. A very important issue is the optimal placement of the outsourced content to CDN's servers. All the approaches developed so far assume the existence of adequate popularity statistics. Such information though, is not always available, or it is extremely volatile, turning such methods problematic. This paper develops a network-adaptive, non-parameterized technique to place the outsourced content to CDN's servers, which requires no a-priori knowledge of request statistics. We place the outsourced objects to these servers with respect to the network latency that each object produces. Through a detailed simulation environment, using both real and synthetic data, we show that the proposed technique can yield up to 25% reduction in user-perceived latency, compared with other heuristic schemes which have knowledge of the content popularity.

1. Introduction

The Web has evolved rapidly from a simple information-sharing mechanism offering only static text and images to a rich assortment of dynamic and interactive services, such as video/audio conferencing, e-commerce, and distance learning. The explosive growth of the Web has imposed a heavy demand on networking resources and Web servers. Users often experience long and unpredictable delays when retrieving Web pages from remote sites. For instance, in networked online games a game player's gaming experience is negatively affected by large propagation delays. Hence, an obvious solution in order to improve the quality of Web services would be the increase of the bandwidth, but such a choice involves increasing economic cost. However, the higher bandwidth would solve temporarily the problems since it would ease the users to create more and more resource-hungry applications, bunching again the network. Therefore, the network limitations will remain or worsen unless effective software solutions are also provided.

Traditional methods to cure this situation include caching [7] (temporary storage of objects closer to the consumer) and prefetching [11] (the process of predicting future requests for Web objects and bringing those objects into the cache in the background, before an explicit request is made for them). Although, these methods offer several benefits (reduced network traffic, shorter response times) the content access is problematic, because it does not improve availability during "flash crowd events"[1] and can not resolve the performance problems related to Web server processing and Internet delays [5].

In this framework, the Content Distribution Networks (CDNs), [12, 15] are targeted to resolve such problems, by moving the content to the "edge" of the Internet, closer to the end-user. With the "key" content outsourced as well as the "key" content placement, the load on the origin server is reduced, the connection from a local content delivery server is shorter than between the origin Web server and the user, thus

* This research is supported by the ΗΡΑΚΛΕΙΤΟΣ and ΠΥΘΑΓΟΡΑΣ ΙΙ national programs funded by the ΕΠΕΑΕΚ.

[1] The flash crowd event occurs when numerous users access a Web site simultaneously, such as the one occurred in September 11th 2001 when users flooded popular news sites (with requests about the terrorist attack in the US), and results in serious caching problems.

reducing latency, and since many users share the CDN's servers, this service greatly increases the hit ratio. In this paper, we focus on finding an effective policy for placing the outsourced content to a CDN infrastructure.

The rest of the paper is organized as follows: Section 2 reviews the related work and Section 3 outlines the motivation and contribution of this work. Section 4 formulates the problem, whereas the proposed object replication strategy is described in Section 5. In Sections 6 and 7, the simulation testbed is described and the performance evaluation of the proposed scheme is shown. Finally, Section 8 concludes the paper.

2. Related Work

2.1. Content Distribution Network

A CDN (such as Akamai[2], Mirror Image[3] etc.) is a network of cache servers, called *surrogate servers*, owned by the same Internet Service Provider (ISP) that delivers content to users on behalf of content providers. Surrogate servers are typically shared, delivering content belonging to multiple Web sites, though all servers may not be used for all sites. The networking functional components of a CDN include user redirection services for directing user to the closest or best cache server, distribution services for intelligently distributing content to users or cache, and accounting and billing system for measuring, logging and billing customers based on usage.

2.2. CDNs Schemes

Each end-user sends requests for Web objects to its nearest surrogate server in the CDN. The specific details of how to handle a cache miss (i.e., the policy that determines whether to fetch the object from another surrogate server or the origin server) and the meta-data information required at the surrogate server to make such decisions are CDN-dependent. Similarly, issues such as organization of the CDN into a hierarchy or surrogate server groups, the degree of cooperation among surrogate servers to service user requests, the policies used to determine a suitable surrogate server to serve a particular end-user are also CDN-specific. Up to now, three distinct content distribution policies have appeared in the context of the CDNs:

- **Uncooperative pull-based**: The clients' requests are directed (by using either DNS redirection or URL rewriting mechanisms) to their closest surrogate server. If there is a cache miss (i.e, the requested content is not found), the request is directed either to a peering surrogate server of the underlying CDN or to the origin server. A problem in this practice is that CDNs do not always choose the optimal server from which to serve the content (as pointed out in [15]). Moreover, there is excessive replication cost, in terms of number of replicas [16]. However, many popular CDN's providers use uncooperative pulling (e.g. Akamai, Digital Island etc.).
- **Cooperative pull-based**: As previous, the clients' requests are directed to their closest surrogate server. The key in the cooperative pull-based schemes is that the surrogate servers are cooperating with each other in case of cache misses. Specifically, using a *distributed index*, the surrogate servers find nearby copies of requested objects, and store them in their caches [1].
- **Cooperative push-based**: The content is pushed (proactively) from the origin Web server to CDNs' surrogate servers. Initially, the content is prefetched to the surrogate servers and then, the surrogate servers cooperate in order to reduce the replication and update cost. In this scheme, the CDN maintains a mapping between content and surrogate servers, and each request is directed to the closest surrogate server. This server may or may not have a replica of the requested object. If it has, the request is served locally, incurring no traffic over the network backbone. Otherwise, it forwards the request to the closest server that has the object replica and relays the response to the client. In this case, the indirect request service generates traffic over the network backbone between the two servers involved in the operation. A key advantage of this scheme is that the surrogate servers can efficiently share the bandwidth. On the other hand, an over-aggressive cooperative push-based scheme may cause excessive network traffic.

2.3. CDNs Challenges

The most important problems related to content management on CDNs and the solutions which have been proposed can be summarized as follows:

- **Replica/Surrogate server placement problem**: In order to deliver content to end users with quality of service (QoS) guarantees, CDN administrators must ensure that surrogate servers are strategically placed across the Web. Generally,

[2] http://www.akamai.org
[3] http://www.mirror-image.com

the problem is to place N surrogate servers among M different sites (M > N) in a way that yields the lowest cost (widely known as the Minimum K-Median problem). A number of previous works have studied how to efficiently place the surrogate servers on the network. In this context, several placement algorithms have been proposed (such as Greedy - incrementally places replicas, Hot Spot - places replicas near the clients generating the greatest load [13], Tree-based replicas- based on the assumption that the underlying topologies are trees [9], HotZone- a latency-driven replica placement [14] etc.). These algorithms specify the locations of the surrogate servers, in order to achieve improved performance (with low infrastructure cost) and earlier experimentation [13] has shown that the Greedy placement strategy can yield close to optimal performance.

- **Content selection problem**: is to determine which content should be outsourced. A "naïve" idea would be to replicate the entire content of a Web site on surrogate servers. However, such a solution is not feasible/practical because, although disk prices are continuously dropping, the sizes of Web objects increase as well (e.g., Video On Demand, Audio). Moreover, the problem of updating such a huge collection of Web objects is unmanageable. In this framework, the practice of replicating the Web content in units of content clusters is mainly used [2].
- **Content replication problem**: It refers to the problem of optimally replicating the outsourced content in surrogate servers of a CDN. Under a CDN's infrastructure (with a given set of surrogate servers) and a chosen content for delivery it is crucial to determine in which surrogate servers the outsourced content should be replicated. Authors in [6] conclude that Greedy-Global heuristic algorithms are the best choice in making the replication decisions between cooperating surrogate servers.

3. Motivation and Paper's Contribution

In this paper, we study the content replication problem. Authors in [6] have shown that this problem is NP complete. In particular, they have proved that it is identical to the well-known NP-complete knapsack problem [4]. In this framework, the authors used four heuristics methods: 1) random, 2) popularity, 3) greedy-single, and finally 4) greedy-global.

Apart from the naive, unscalable approach, where the outsourced objects are placed randomly to surrogate servers, the other approaches require popularity statistics. However, the use of those statistics has several drawbacks. Firstly, it requires quite a long time to collect reliable request statistics for each object. Such a long interval though may not be available, when a new site is published to the Internet and should be protected from flash crowds. Moreover, as authors in [2] report, the popularity of each object varies considerably; for the WorldCup'98 trace, only 40% of the "popular" objects of the one day remain "popular" and the next day. Furthermore, the use of administratively tuned parameters to select the hot outsourced objects, or decide the number of clusters causes additional headaches, since there is no a-priori knowledge about where to set the popularity threshold or how many clusters of objects exist. In addition, the greedy approaches are not feasible to implement on real applications, due to their high complexity[4].

Another motivation of this work is to study the content replication problem under an analytic CDN simulation model which considers both the network traffic and the server load. Until now, the most noteworthy work [6], which has studied this problem, does not take into account several critical factors, such as the bottlenecks that are likely to occur in the network, the number of sessions that can serve each network element (e.g. router, surrogate server) etc. Thus, the results that the authors presented in [6] may be misleading (they measure the number of traversed nodes (hops) without considering the TCP/IP network infrastructure). Therefore, the motivation for us is to develop a flexible simulation model that simulates in great detail the TCP/IP protocol as well as the main characteristics of a cooperative push-based CDN infrastructure model[5]. Specifically, the main benefit of a detailed CDN simulation model is that it gives a (closely) realistic view to the CDNs' developers about which will be the profits for both the CDNs' providers and CDNs' customers if the proposed approach adapts to a real CDN's provider (e.g. Akamai).

Thus, we introduce a self-tunable strategy, which will not exploit popularity statistics and will not use any administratively set parameters to optimally replicate the outsourced objects to surrogate servers. In the context of this problem, the present paper makes the following contributions:

- We formulate the content replication problem for a cooperative push-based scheme.
- We provide a novel, self-tuning, parameterless strategy for optimally placing outsourced objects

[4] Because of the huge memory requirements, authors in [6] reported that they could not run all the experiments for the greedy heuristic.

[5] We use the cooperative-push based policy since it has been proved to have the best results [6].

in CDN's surrogate servers, which is based on network latency.

- We develop an analytic simulation environment to test the efficiency of the proposed latency-based scheme. Using real and synthetically generated test data, we show the robustness and efficiency of the proposed method which can reap performance benefits better than an analogous heuristic method which has a priori knowledge of the object popularity statistics.

4. Problem Formulation

Here, we formulate the content replication problem for cooperative-push based over CDNs. Therefore, we consider a popular Web site that signs a contract with a CDN's provider with *N* surrogate servers, each of which acts as an intermediary between the servers and the end-users. We further assume that the surrogate server *i* has S_i bytes of storage capacity, where $i \in \{1,...,N\}$.

In order to formulate the placement's cost function, we assume that we have *K* outsourced objects. Each object *k* has a size of s_k, where $k \in \{1,...,K\}$. In this context, we define a variable which determines if an object *k* is stored to surrogate server *k*.

$$f_{ik} = \begin{cases} 1 & \text{if object } k \text{ is stored at surrogate } i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The storage is subject to the constraint that the space available at surrogate server *i* is bounded by

$\sum_{k=1}^{K} s_k f_{ik} \le S_i$, where $i \in \{1,...,N\}$.

Considering that all the outsourced objects are initially placed on an origin server (the initial placement is denoted by x_o), the content replication problem is to select the *optimal placement x* (defines the placement of outsourced objects to CDN's surrogate servers) such that it minimizes:

$$\cos t(x) = \sum_{i=1}^{N}\sum_{k=1}^{K} \frac{p_k \lambda_i}{\sum_{j=1}^{N} \lambda_j} \left(D_{ik}(x)\right) \quad \textbf{(2)},$$

where $D_{ik}(x)$ is the "distance"[6] to a replica of object *k* from surrogate server *i* under the placement *x*, λ_i is the request rate for surrogate server *i*, and p_k is the probability that a client will request the object k[7].

However, as it has been proved in [6], this problem is NP complete (it is similar to the NP-complete knapsack problem), which means that for a large number of outsourced objects and surrogate servers is not feasible to solve this problem optimally. In this context, we propose a new heuristic strategy where its criterion is the overall latency of the network. We name this algorithm latency-based object placement in CDNs, in short *Lat-cdn*.

5. The Lat-cdn Algorithm

The main idea is to place the outsourced objects to surrogate servers with respect to the total network's latency, which is produced by these objects, without taking into account the objects' popularity. Therefore, the distance *D(x)* in equation 2 reflects the latency.

In this framework, each surrogate server maintains a cache that is typically stored on disk. Upon receiving a request, the surrogate server services the request from the local cache (in the event of a cache hit) or by fetching the requested object from another surrogate server or the origin server (in the event of a cache miss). Here, we make the assumption that the surrogate servers are collaborating and each one knows a priori what content is cached to all the other surrogate servers that belong to the same CDN (via the CDN's distribution system[8]). In addition, we consider that the Web objects fetched upon a cache miss are not inserted into the cache for servicing future requests.

Initially all the outsourced objects are stored in the origin server and all the CDN's surrogate servers are empty. For each outsourced object, we find which is the best surrogate server in order to place it (produces the minimum network latency). Then, we select from all the pairs of *outsoursed object – surrogate server* that have been occurred in the previous step, the one which produces the largest network latency, and thus place this object to that surrogate server. The above process is iterated until all the surrogate servers become full. As a result, an outsourced object may be assigned to several surrogate servers, but a surrogate server will have at maximum one copy of an outsourced object. Concerning the complexity of the *Lat-cdn* is polynomial. In order to by-pass this problem, we may use clusters of objects [2]. The

[6] The distance may reflect several metrics such as the number of traversed nodes (hops), the latency, servers' load etc.

[7] For simplicity, we assume that the client request patterns are homogenous. Therefore, the values of p_k are the same for all the surrogate servers.

[8] It is a collection of network elements that support distribution for a CDN [15].

detailed algorithm is described in pseudo-code in Figure 1.

```
Lat-cdn
{
   Input:
   obj[1…K] //outsourced objects
   ss[1…N]  //surrogate servers
   Output:
   a placement x of outsourced objects to surrogate servers

while (there is free cache space on surrogate servers)
   {
    for (k=1; k<=K; k++){
        min[obj[k]]= ∞ ;
        for (n=1; n<=N; n++)
           if (free cache size of ss[n]  <= size obj[k] && obj[k] does not exist in ss[n])
              {
               place obj[k] to ss[n];
                find  the cost(obj[k],ss[n]);
                if (cost(obj[k],ss[n])<min[obj[k]]) //find the minimum cost
                    min[obj[k]]=cost(obj[k],ss[n]);
              }
       }
    for (k=1; k<=K; k++)
       find the maximum of min[obj[k]];
    placement (object y, surrogate server z); //place the object y to surrogate server z which has the maximum value of minimum costs.
   }
}
```

Figure 1. The Lat-cdn Algorithm

6. Simulation Testbed

To evaluate the proposed methods we use trace-driven simulations developing an analytic simulation environment, which includes the following: a) a system model simulating the CDN infrastructure, b) a network topology generator, c) a Web site generator, modeling file sizes, linkage, etc., and d) a client request stream generator capturing the main characteristics of Web users' behavior.

6.1. System Model

We have implemented a simulation model for CDNs using the ParaSol library[9], which is a parallel discrete event simulation system. We consider a CDN infrastructure consisting of N=20 surrogate servers. We assume the case of homogeneous servers (all the servers have the same storage capacity). Then, we group the users based on their domains. The number of client groups is equal to the number of surrogate servers. Thus, each client group is connected with only one surrogate server and contains a few thousands clients. All CDN networking issues, like surrogate server selection, propagation, queuing, bottlenecks and processing delays are computed dynamically via the simulation model, which provides an implementation as close as possible to the working TCP/IP protocol, implementing packet switching, packet retransmission upon misses, etc. Finally, in order to efficiently manage the outsourced objects stored in surrogate servers, we modeled their disks using the Bloom filters, as in [8].

6.2. Network Topology

Using the GT-ITM internetwork topology generator [17], we generated a random network topology, called Waxman, with a total of 1008 nodes. Specifically, in Waxman model, the nodes are randomly assigned to locations on a plane, but an edge is created between a pair of node u and v with probability $P(u,v) = \alpha e^{\frac{-d}{\beta L}}$, where $d = |\vec{u} - \vec{v}|$, L is the maximum Euclidean distance between any two vertices, $\alpha > 0$ and $\beta \leq 1$. Furthermore, we constructed an AS-level Internet topology with a total of 3037 nodes, using BGP routing data collected from a set of 7 geographically-dispersed BGP peers in April 2000.

6.3. Web Site Generation

In order to generate the outsourced objects, we used artificially generated Web graphs, constructed by the R-MAT tool [3]. The R-MAT produces realistic Web graphs capturing the essence of each graph in only a few parameters. In this framework, we create two graphs with varying number of nodes (objects). Specifically, the sparse-density graph has 4000 nodes, and a moderate-density graph consists of 3000 nodes. Finally, we should also assign a size for each node (a node represents a Web object), since the R-MAT model gives us only the nodes which are inter-communicated with each other. For this task we have used the *log-t* distribution as described in [10]. The total objects' sizes for sparse graph and the moderate graph are 746 MB and 1022 MB respectively.

[9] http://www.cs.purdue.edu/research/PaCS/parasol.html

6.4. Request Streams Generation

The workloads to the above Web graphs are streams of requests, called client transactions. To generate transactions, we used the generator described in [11], which given a Web site graph, generates transactions as sequences of page traversals (random walks) upon the site graph. After producing the transactions, we follow three steps in order to convert them to a log file.

- **Step 1.** We define the number of clients and distribute the transactions to the clients, so that each client will make at least one transaction).
- **Step 2.** We define the time window that the transactions will be spread out; the length of the window determines how "heavy" or "light" the system load is. The default value that we used is one week.
- **Step 3.** For each transaction, repeat the following:
    - **Step 3a.** Assign a client who has made no transactions yet to the current transaction. If such a client does not exist, we select a client at random.
    - **Step 3b.** A random timestamp is selected uniformly within the time window. This timestamp determines the starting time of the transaction. The time interval between two successive requests of the same transaction is selected uniformly with an average of 2 minutes.

7. Performance Evaluation

In our experiments, we use the *average response time* measure in order to evaluate our proposed scheme. In practice, we compute the elapsed time between when a user issues a request and when it receives the response; it measures the user satisfaction and it should be as small as possible.

7.1. Examined Methods

In order to evaluate the proposed algorithm, we examine also the following heuristics:

- **Random**: Assigns the outsourced objects to CDN's surrogate servers randomly subjected to the storage constraints. Both the outsourced object and the surrogate server are selected by uniform probability. If the surrogate server already stores that object, a new object and a new surrogate server are selected. This heuristic plays the role of the baseline for our experiments.
- **Popularity**: Each surrogate server stores the most popular outsourced objects among its clients. The node sorts the objects in decreasing order of popularity and stores as many outsourced objects in this order as the storage constraint allows. The surrogate server estimates the popularities by observing the requests it receives from its clients.

7.2. Lat-cdn for Typical Object Sizes

Based on our testbed, we performed an analytic investigation of the performance of the proposed object replication method, *Lat-cdn*, with the aforementioned methods. We performed extensive experiments with various graph sizes (in terms of number of vertices and edges), with various client populations and request patterns, etc. Due to the interest of space, in this paper we present only a small selection of the result obtained.

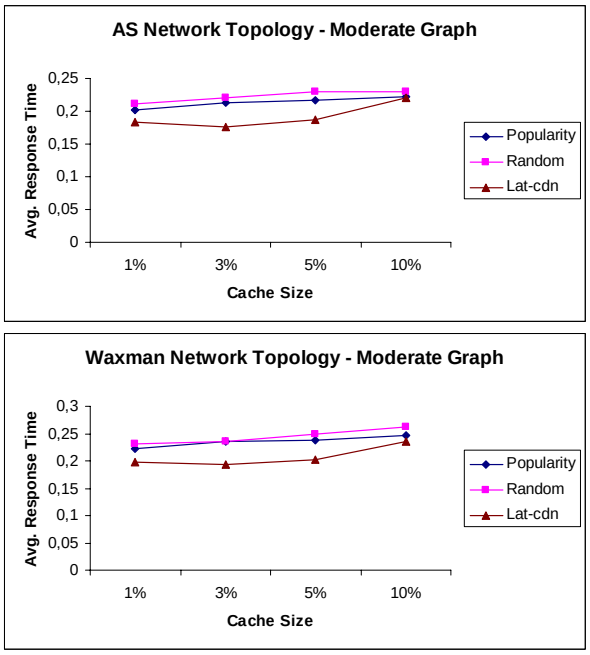

Figure 2. Average Response Time for Moderate-density Web Graphs (3000 objects)

Our first experiment demonstrates the average response time for the moderate-density Web graph (3000 outsourced objects) on both network topologies with respect to surrogate servers' cache size. Specifically, the size of the cache is expressed in terms of the percentage of the total number of bytes of the Web site. The results of this set of experiments are reported in Figure 2. The *x-axis* represents the cache size of CDN's surrogate servers, while the *y-axis* represents the average response time. From this Figure, it can be seen that the *Lat-cdn* approach, gives the best response times for both network topologies. The second best is the Popularity, which is followed closely

by Random. Furthermore, we observe that as the cache size increases, the average response time also increases. Although it looks quite strange at first sight (one may expect to have lower times), it is explained by the fact that the larger in size caches may satisfy more requests. Thus, the average response time is increased, as the size of surrogate servers' caches increases.

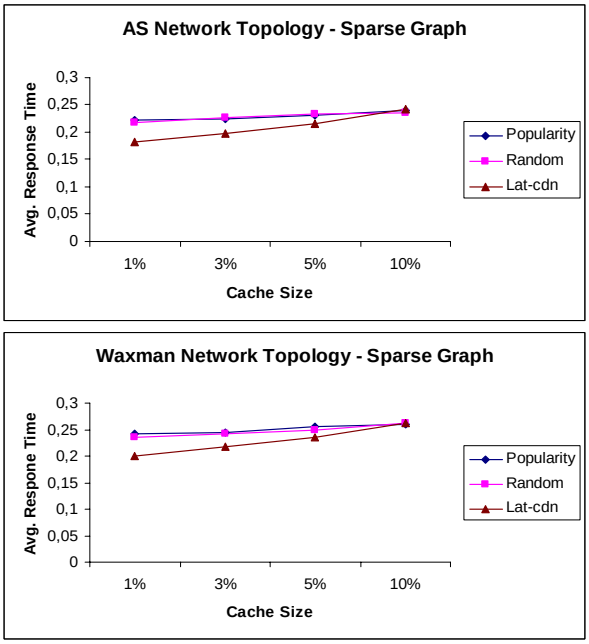

Figure 3. Average Response Time for Sparse-density Web Graphs (4000 objects)

In Figure 3, we plot the results from experiments with 4000 outsourced objects (sparse-density Web graph). The results are very similar to the results from the previous experiment. In general, for both network topologies, the *Lat-cdn* outperforms all the other heuristics, whereas Popularity and Random are very close.

7.3. Lat-cdn Limitations

We further conclude the evaluation by reporting on some experiments conducted using outsourced objects from a real Web site. The real Web site we used is the Stanford Web site from a September 2002 crawl[10] that consists of 281903 Web objects. Note, that the network topologies, client populations and request stream generation are the same as with synthetic data.

Our experiment demonstrates the average response time for AS network topology. The results are reported in Figure 4. As previous, the *x-axis* represents the cache size of CDN's surrogate servers, while the *y-axis* represents the average response time. Notice that in this experiment we use a different scale for the cache sizes (compared with the previous ones) due to the large amount of objects of the Stanford Web site. From this Figure, it can be seen that the *Lat-cdn* outperforms all the other approaches. The only exception is when the surrogate servers have very small cache sizes, where the Popularity has the best performance. Another observation that we make is that the response times are too small. The reason is that the majority of objects of Stanford Web site have very small sizes.

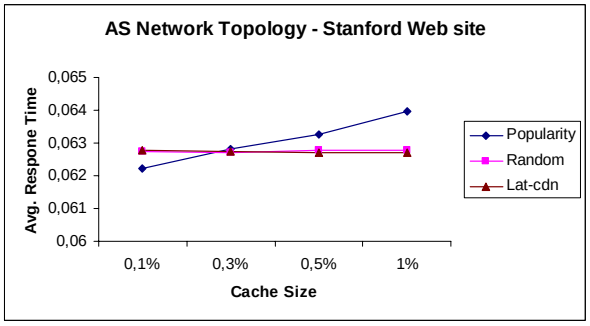

Figure 4. Average Response Time for Real Web Site

In general, from our results, we can conclude that the best performance is obtained by the *Lat-cdn* heuristic, taking into account the surrogate servers are cooperated with each other. The difference in performance between *Lat-cdn* and the other two heuristics is quite significant especially for artificial Web sites (ranges from 6% to 25%), which have on average larger objects in size than the Stanford Web site. Despite the low improvement rates on Stanford Web site, the *Lat-cdn* is still in most cases beneficial. In this context, it should be noticed that the role of CDNs is focused on improving the QoS of the explosive growth of resource-hungry applications in Web sites, such as Digital Television, Interactive TV, Video On Demand (VOD), etc. Therefore, the medium to large size objects are of interest in the *Lat-cdn* context.

8. Conclusions

In this paper, we addressed the content replication problem for CDNs. Differently from all other relevant heuristics approaches, we refrained from using any request statistics in determining in which surrogate servers to place the outsourced objects. Our goal is to find an efficient placement so that when clients fetch objects from the nearest surrogate server, the average response time is minimized. Implementing a detailed

[10] Available at http://www.stanford.edu/ ~sdkamvar/ research.html

simulation environment, the CDNs' developers may have a (closely) realistic view about which will be the profits for both the CDNs' providers and CDNs' customers if the proposed approach adapts to a real CDN's provider (e.g. Akamai). The results have shown that the proposed algorithm outperforms the other examined heuristic methods in a cooperative push-based scheme. For the future we plan to investigate the content replication problem in CDNs for uncooperative pull-based schemes as well as for cooperative pull-based schemes.

9. References

[1] S. Annapureddy, M. J. Freedman, and D. Mazières, "Shark: Scaling File Servers via Cooperative Caching", Proceedings of the 2nd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI), Boston, USA, May 2005.

[2] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. H. Katz, "Efficient and Adaptive Web Replication using Content Clustering", *IEEE Journal on Selected Areas in Communications*, 21(6), Aug. 2003, pp. 979-994.

[3] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A Recursive Model for Graph Mining", Proceedings of the 4th SIAM International Conference on Data Mining, Orlando, Florida, USA, 2004.

[4] M. R. Garey and D. S. Johnson, "*Computers and Intractability: A Guide to the Theory of NP-Completeness*", Freeman, New York, 1979.

[5] Y. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites", Proceedings of the 11th International World Wide Web Conference (WWW), Honolulu, Hawaii, USA, May 2002, pp. 293–304.

[6] J. Kangasharju, J. Roberts, and K. W. Ross, "Object Replication Strategies in Content Distribution Networks", *Computer Communications*, 25(4), Apr. 2002, 367-383.

[7] D. Katsaros and Y. Manolopoulos, "Caching in Web Memory Hierarchies", Proceedings of the ACM Symposium on Applied Computing, Nicosia, Cyprus, Mar. 2004, pp. 1109-1113.

[8] P. Kulkarni and P. Shenoy, "Scalable Techniques for Memory-efficient CDN Simulations", Proceedings of the 12th International World Wide Web Conference (WWW), Hungary, May 2003, pp. 609-618.

[9] B. Li, M. J. Golin, G. F. Ialiano, and X. Deng, "On the Optimal Placement of Web Proxies in the Internet", Proceedings of the Conference on Computer Communications, 18th Annual Joint Conference of the IEEE Computer and Communications Societies, Networking the Next Generation (IEEE INFOCOM), New York, USA, Mar. 1999, pp.1282-1290.

[10] M. Mitzenmacher and B. Tworetzky, "New Models and Methods for File Size Distributions", Proceedings of the 41th Annual Allerton Conference on Communication, Control, and Computing, Illinois, USA, Oct. 2003, pp. 603-612.

[11] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos, "A Data Mining Algorithm for Generalized Web Prefetching", I*EEE Transactions on Knowledge Data Engineering*, 15(5), May 2003, pp. 1155-1169.

[12] G. Pallis and A. Vakali, "Insight and Perspectives for Content Delivery Networks", *Communications of the ACM* (CACM), to appear.

[13] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the Placement of Web Server Replicas", Proceedings of the Conference on Computer Communications, 20th Annual Joint Conference of the IEEE Computer and Communications Societies, Networking the Next Generation (IEEE INFOCOM), Anchorage, Alaska, USA, Apr. 2001, pp. 1587-1596.

[14] M. Szymaniak, G. Pierre, and M. Van Steen, "Latency-Driven Replica Placement", Proceedings of the International Symposium on Applications and the Internet (SAINT), Trento, Italy, Feb. 2005, pp. 399-405.

[15] A. Vakali and G. Pallis, "Content Delivery Networks: Status and Trends", *IEEE Internet Computing*, 7(6), 2003, pp. 68-74.

[16] H. Yu and A. Vahdat, "Minimal Replication Cost for Availability", Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC), Monterey, California, USA, Jul. 2002, pp. 98-107.

[17] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork", Proceedings of the Conference on Computer Communications, 15th Annual Joint Conference of the IEEE Computer and Communications Societies, Networking the Next Generation (IEEE INFOCOM), San Francisco, USA, Mar. 1996, pp. 594-602.