Replication based on Objects Load under a Content Distribution Network

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

Abstract

Users tend to use the Internet for "resource-hungry" applications (which involve content such as video, audio on-demand and distributed data) and at the same time, more and more applications (such as e-commerce, e-learning etc.) are relying on the Web. In this framework, Content Distribution Networks (CDNs) are increasingly being used to disseminate data in today's Internet, providing a delicate balance between costs for Web content providers and quality of services for Web customers. The growing interest in CDNs is motivated by a common problem across disciplines: how does one reduce the load on the origin server and the traffic on the Internet, and ultimately improve 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. Taking into account that this problem is NP complete, an heuristic method should be developed. All the approaches developed so far either take as criterion the network's latency or the workload. This paper develops a novel technique to place the outsourced content to CDN's servers, integrating both the latency and the load. Through a detailed simulation environment, using both real and synthetic data, we show that the proposed method can improve significantly the response time of requests while keeping the CDNs' servers' load at a very low level.

1. Introduction

The explosive growth of the Web has imposed a heavy demand on networking resources and Web services. In this framework, 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.

A traditional method to cure this situation includes caching [7] (temporary storage of objects closer to the consumer). Although, caching offers several benefits (reduced network traffic, shorter response times), it has drawbacks (small hit rates, compulsory misses). To compensate for such problems, traditional caching is coupled with another, complementary technique, the prefetching [11]. Prefetching aims at predicting future requests for Web objects and bringing those objects into the cache in the background, before an explicit request is made for them. The most common prefetching practice is to make predictions based on the recent history of requests of individual clients. Although, these methods offer several benefits (reduced network traffic, shorter response times) the content access is problematic, because they do not improve availability during "flash crowd events"[1] and can not resolve the performance problems related to Web server processing and Internet delays [5].

From the above it is occurred that standalone Web servers are unable to provide reliable and scalable services. In contrast, distributed solutions are a popular way to improve the efficiency, reliability and scalability of Web services. In this framework, the Content Distribution Networks (CDNs), [13, 18] are targeted to resolve such problems, by moving the content to the "edge" of the Internet, closer to the end-user. With the

[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.

"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 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 provides a background of CDNs and Section 3 outlines the motivation and contribution of this work. Section 4 formulates the problem, and the proposed object replication strategy is described. In Sections 5 and 6, the simulation testbed is described and the performance evaluation of the proposed scheme is shown. Finally, Section 7 concludes the paper.

2. Background

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. Typically, a CDN topology involves 1) a set of surrogate servers which cache the origin servers' content, 2) routers and network elements which deliver content requests to the optimal location and the optimal surrogate server, and 3) an accounting mechanism which provides logs and information to the origin servers.

Under a CDN infrastructure, the client-server communication is replaced by two communication flows: one between the client and the surrogate server, and another between the surrogate server and the origin server. This distinction into two communication flows reduces congestions (particularly over popular servers) and increases content distribution and availability.

Typically, 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. In order to exploit the full potential of CDNs crucial data management issues must be addressed. Up to now, various content distribution policies have appeared in the context of the CDNs: Uncooperative pull-based [18, 20], cooperative pull-based [1], cooperative push-based and uncooperative push-based are the basic approaches, as reported in [13].

3. Motivation and Paper's Contribution

The most important problems related to content management on CDNs problems are the replica/surrogate server placement [9, 15, 16], the content selection [2] and the content replication [6], as reported in [13]. In this paper, we study the content replication problem, which refers to the issue 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. A naive and simple solution to this problem is to replicate all the outsourced objects[4] of the Web site (full-mirroring) to all the surrogate servers. 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.

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]. This means that for a large number of outsourced objects and surrogate servers is not feasible to solve this problem optimally. Therefore, an heuristic solution should be found.

In this framework, authors in [6] used four heuristics methods: 1) random, 2) popularity, 3) greedy-single, and finally 4) greedy-global. Apart from the naive, unscalable approaches, where the outsourced objects either are placed randomly to surrogate servers or are placed according to their popularity, the greedy approaches are not feasible to implement on real applications, due to their high complexity. For instance, because of the huge memory requirements, authors in [6] reported that they could not run all the experiments for the greedy heuristic policies.

In [17], the authors studied the content replication problem from another point of view. Specifically, they presented a set of greedy approaches where the placement is occurred by balancing the loads and sizes of the surrogate servers. The drawback of these algorithms is their high complexity, since they require quite a long time to produce a sufficient placement. A quite similar approach has also been presented in [22].

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

[4] We consider that the outsourced objects (the objects that will be replicated to CDN's surrogate servers) are known [2].

In [14], we presented a self-tuning, parameterless algorithm, called lat-cdn, for optimally placing outsourced objects in CDN's surrogate servers, which is based on network latency (an object's latency is defined as the delay between a request for a Web object and receiving that object in its entirety). The main advantage of this algorithm is that it does not require popularity statistics, since the use of them has often several drawbacks (e.g. quite a long time to collect reliable request statistics, the popularity of each object varies considerably etc.). However, this approach does not take into consideration the load of the objects (the load of an object is defined as the product of its access rate and size). Therefore, in this approach, it is possible to replicate in the same surrogate server objects with high loads and, thus, during a flash crowd event the server will be overloaded.

Therefore, we need an heuristic approach which will consider both the object's latency and the object's load in order to decide in which surrogate server to place the outsourced objects. In this framework, at first, we introduce a self-tunable strategy, which does not exploit popularity statistics and does not use any administratively set parameters in order to determine for each outsourced object which is the best surrogate server to place its replica. Then, the algorithm determines which one of the outsourced objects will be replicated with respect to their loads.

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. Most works [6], which have studied this problem, do 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 their works 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. 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).

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, dividing it into two sub-problems.
- We provide a novel strategy for optimally placing outsourced objects in CDN's surrogate servers, integrating both the network's latency and the objects' load.
- We develop an analytic simulation environment to test the efficiency of the proposed 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.

4. Problem Formulation

Here, we formulate the content replication problem for cooperative-push based over CDNs, since it has been proved to have the best results [6]. In this scheme, the content is pushed (proactively) from the origin Web server to CDNs' surrogate servers and then, the surrogate servers cooperate in order to reduce the replication and update cost. Specifically, 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 case that the requested object has not been replicated by anyone surrogate server (the requested object has not been outsourced), the request is served by the origin server. Although this practice requires cooperation among the surrogate servers incurring some communication and management cost to implement the cooperation, the key advantages of this scheme is that the surrogate servers can efficiently share the bandwidth and reduce the replication redundancy (number of replicas deployed), and consequently reduces the replication and update cost.

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.

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

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 $i \in \{1,...,N\}$. Furthermore, each surrogate server can hold at most one replica of the object.

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 may be separated into two sub-problems:

1. **Choice of the best surrogate server to replicate an outsourced object**

Specifically, for each outsourced object, we select its optimal surrogate server such that it minimizes:

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

where $D_{ik}(x)$ is the distance to a replica of object *k* from surrogate server *i* under the placement *x* (defines the placement of outsourced objects to CDN's surrogate servers), λ_i is the request rate for surrogate server *i*, and p_k is the probability that a user will request the object k. 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. The distance may reflect several metrics such as the number of traversed nodes (hops), the latency, servers' load etc. Here the distance reflects the latency.

2. **Arrange priorities for outsourced objects replication**

So far, we have made the optimal assignment of the outsourced objects to the surrogate servers. From the objects assigned to a single server we replicate the one which has the maximum utility value. Here, the utility value of each object is given by the following equation:

$$\text{Utility\_Value}_k = \text{load}_k * \text{latency}_k, \text{ where}$$
$$\text{load}_k = \text{access\_rate}_k * s_k \quad (3)$$

In the following equation, latency_k is the latency that the object k produces if it is replicated to the surrogate server which has been determined by the previous step, load_k is the total load due to object k and access_rate_k is defined as the number of accesses of object k per unit time. We name this algorithm il2p, which stands for the ***i***ntegration of ***l***atency and ***l***oad object ***p***lacement in CDNs. The following section describes this algorithm.

4.1. The il2p Algorithm

The main idea is to place the outsourced objects to surrogate servers with respect to the latency and their load. Initially all the outsourced objects are stored in the origin server and all the CDN's surrogate servers are empty. Specifically, the proposed algorithm is implemented into two phases:

- **Phase 1**: For each outsourced object, we find its best surrogate server to place it, without taking into account its popularity (produces the minimum network latency).
- **Phase 2**: We select from all the pairs of *outsourced object – surrogate server* that have been occurred in the previous phase, the one which has the largest *utility value*, and thus place this object to that surrogate server.

```
il2p
{
   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)
   {
//Phase 1
     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];
                evaluate  the latency(obj[k],ss[n]);
                if (latency (obj[k],ss[n]) < min[obj[k]])
//find the minimum cost
                    min[obj[k]]=cost(obj[k],ss[n]);
               }
       }
//Phase 2
    for (k=1; k<=K; k++)
       evaluate the Utility_Value[k] for obj[k];
//Utility_Value[k]=load[k]*latency[k]
   select the object y with the maximum Utility_Value;
    placement (object y, surrogate server z); //place the
object y to surrogate server z.
   }
}
```

Figure 1. The il2p Algorithm

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. The detailed algorithm is described in pseudo-code in Figure 1. Concerning the complexity of the il2p is polynomial, since each phase requires polynomial time. In order to by-pass this problem, we may use clusters of objects [2].

5. 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, since the real traces of CDNs' providers are not available

5.1. System Model

We have implemented a simulation model for CDNs using the ParaSol library[5], which is a parallel discrete event simulation system (called *CDNsim*). Figure 2 depicts the basic screen shots of the CDNsim. We do not provide any details for CDNsim tool (e.g. architecture etc.) since it is beyond the scope of this paper.

In this work, 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].

As we referred above, we formulate the content replication problem for cooperative-push based over CDNs. In this context, CDNsim 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). Furthermore, similar to previous work [2, 6, 19], we consider that the Web objects fetched upon a cache miss are not inserted into the surrogate's cache, but simply forwarded to the requesting client.

Another important issue is to consider how the proposed CDN environment tackles the problem of staleness of cached objects, which is also known as cache consistency problem. In general, cache consistency may affect significantly the performance of a CDN scheme. In general, consistency mechanisms fall in two categories: strong consistency (accessed copies are always up to date) and weak consistency (accessed copies might be stale). However, the stability of the CDN architecture (fixed number of surrogate servers) makes us to enforce a strong consistency mechanism. This assumption is strengthened by the fact that the probability of requesting a stale object is very small [12].

5.2. Network Topology

Using the GT-ITM internetwork topology generator [21], 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.

5.3. 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

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

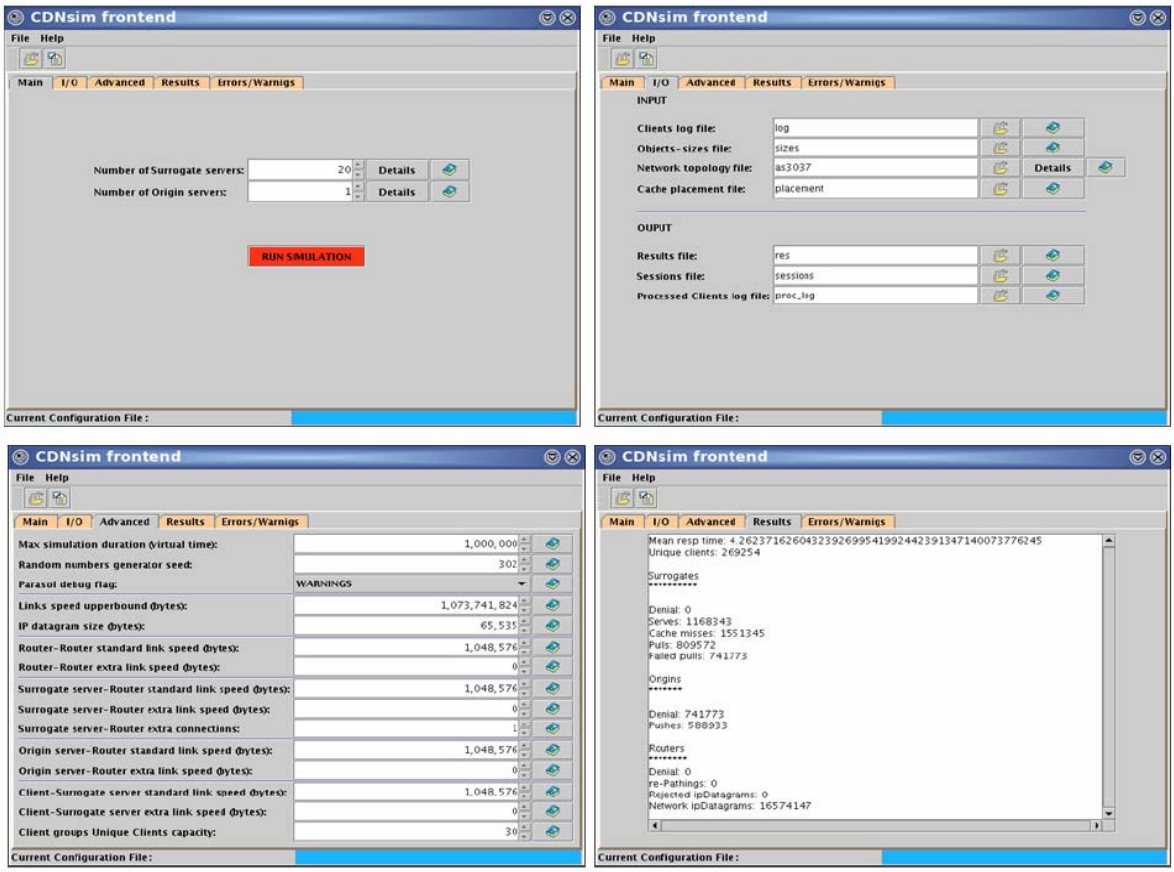

Figure 2. CDNsim: A Simulation Tool for Content Distribution Networks

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.

5.4. Request Streams Generation

The next is to generate the workloads of the above graphs. Specifically, these workloads are streams of Web users' requests, called *client transactions*. To generate these 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.

6. 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.

6.1. 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.
- **Lat-cdn**: The outsourced objects are placed to surrogate servers with respect to the total network's latency, without taking into account the objects' popularity. Specifically, each surrogate server stores the outsourced objects which produce the maximum latency.

6.2. Synthetic Data Experimentation

Based on our testbed, we performed an analytic investigation of the performance of the proposed object replication method, il2p, 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.

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 3. 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 il2p approach, gives the best response times for both network topologies. The second best is the lat-cdn, which is followed by Popularity and 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.

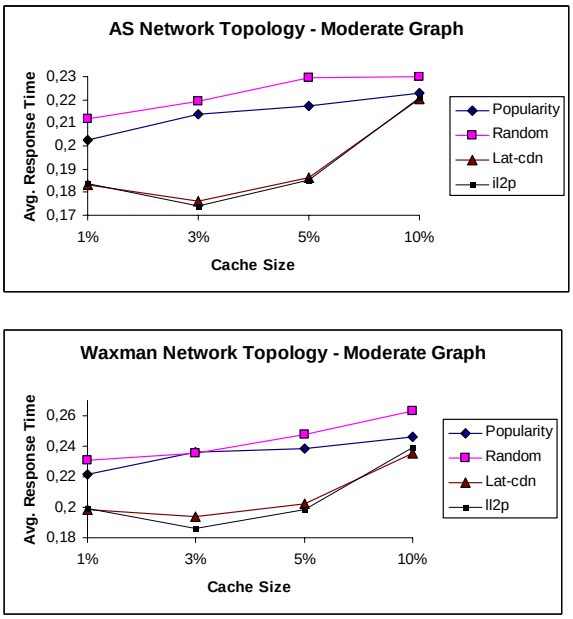

Figure 3. Average Response Time for Moderate-density Web Graphs (3000 objects)

In Figure 4, 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 il2p approach outperforms all the other heuristics.

6.3. Real Data Experimentation

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[6] 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 both network topologies. The results are reported in Figure 5. 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 il2p has quite similar performance with lat-cdn and Random. On the other hand, the il2p outperforms the Popularity approach. 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

[6]It is available at http://www.stanford.edu/ ~sdkamvar/ research.html

majority of objects of Stanford Web site have very small sizes.

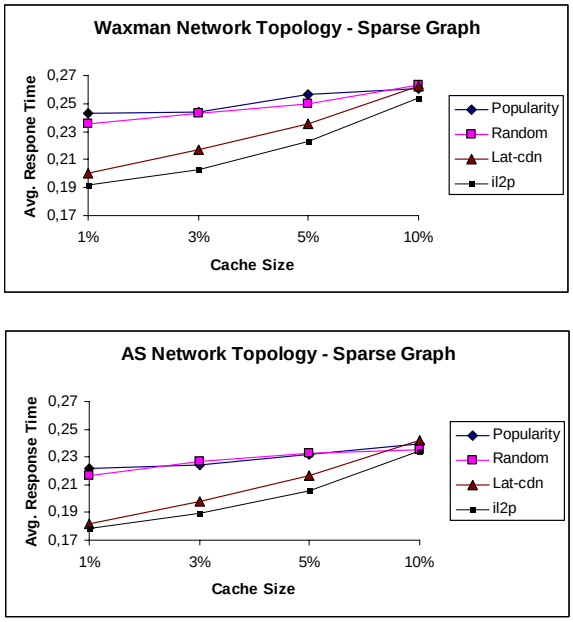

Figure 4. Average Response Time for Sparse-density Web Graphs (4000 objects)

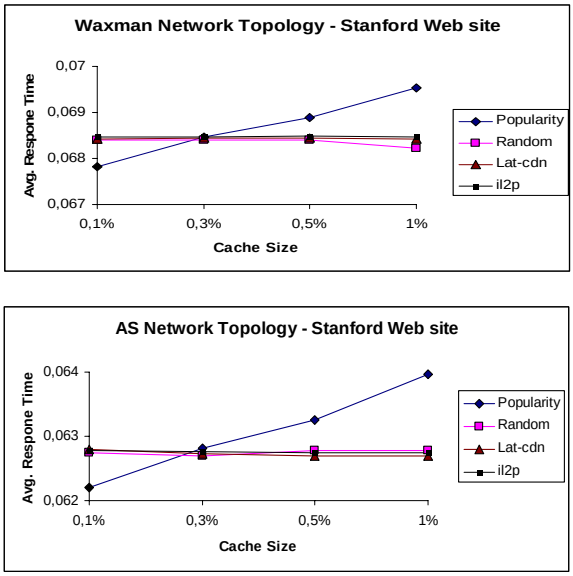

Figure 5. Average Response Time for Real Web Site

In general, from our results, we can conclude that the best performance is obtained by the il2p heuristic, taking into account the surrogate servers are cooperated with each other. The difference in performance between il2p and the other three heuristics is quite significant especially for artificial Web sites (in most cases around 5% absolute improvement with respect to lat-cdn[7], and consistently around 25% absolute improvement with respect to other two heuristics), which have on average larger objects in size than the Stanford Web site. Despite the low improvement rates on Stanford Web site, the il2p 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 il2p context.

7. Conclusions

In this paper, we addressed the content replication problem for CDNs. Differently from all other relevant heuristics approaches, we partition this problem into two sub-problems. The first one defines the pairs of outsourced object - surrogate server which achieve the lowest latency, refrained from using any request statistics. The second one determines which objects to replicate. 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 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.

8. 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

[7]The low gains of il2p compared to lat-cdn can be explained by the fact that it is used the same process in order to choose the best surrogate servers to replicate the outsourced objects.

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] V.N. Padmanabhan, and L. Qiu, "The Content and Access Dynamics of a Busy Web Site: findings and implications", In Proceedings of ACM SIGCOM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Stockholm, Sweden, Aug. 2000, 111-123.
[13] G. Pallis and A. Vakali, "Insight and Perspectives for Content Delivery Networks", *Communications of the ACM* (CACM), 49(1), Jan. 2006, pp. 101-106.
[14] G. Pallis, A. Vakali, K. Stamos, A. Sidiropoulos, D. Katsaros, Y. Manolopoulos, "A Latency-based Object Placement Approach in Content Distribution Networks", Proceedings of the 3rd Latin American Web Congress (La-Web 2005), IEEE Press, Buenos Aires, Argentina, Oct. 2005, pp. 140-147.
[15] 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.
[16] 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.
[17] S.S.H. Tse: "*Approximate Algorithms for Document Placement in Distributed Web Servers*", Parallel and Distributed Systems, IEEE Transactions on 16(6), Jun. 2005, pp. 489 – 496.
[18] A. Vakali and G. Pallis, "Content Delivery Networks: Status and Trends", *IEEE Internet Computing*, 7(6), 2003, pp. 68-74.
[19] B. Wu and A.D. Kshemkalyani, "Objective-Greedy Algorithms for Long-term Web Prefetching", Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications (NCA 2004), Cambridge, MA, USA, Aug.-Sep. 2004, pp. 61-68.
[20] 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.
[21] 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.
[22] L. Zhuo, C-L Wing, F. C. M. Lau, "Load Balancing in Distributed Web Server Systems with Partial Document Replication", Proceedings of the 2002 International Conference on Parallel Processing (ICPP'02), Vancouver, BC, Canada, Aug. 2002, pp. 305-313.