Integrating Caching Techniques on a Content Distribution Network

Konstantinos Stamos, George Pallis, and Athena Vakali

Department of Informatics
Aristotle University of Thessaloniki,
54124, Thessaloniki, Greece
kstamos@csd.auth.gr, gpallis@ccf.auth.gr, avakali@csd.auth.gr

Abstract. Web caching and replication tune capacity with performance and they have become essential components of the Web. In practice, caching and replication techniques have been applied in proxy servers and Content Distribution Networks (CDNs) respectively. In this paper, we investigate the benefits of integrating caching policies on a CDN' s infrastructure. Using a simulation testbed, our results indicate that there is much room for performance improvement in terms of perceived latency, hit ratio and byte hit ratio. Moreover, we show that the combination of caching with replication fortifies CDNs against flash crowd events.

1 Introduction

The rapid evolution of the Internet along with the increasing interest of the end-user for Web services has lead to the development of a wide variety of on-line applications (video-on-demand (VOD), e-commerce, information retrieval (IR), on-line gaming). Web sites (i.e. news sites), offering those services, have to deal with the increasing number of requests, whereas, on the server side, this results in increasing demand for processing time and overloading. On the end-user side, noticeable latency, connection interrupts and Denial of Service (DoS) are perceived due to network traffic and server overloading. For instance, in an application which gets real-time data from the NASDAQ Stock Market and makes buying decisions, delayed (stale) data will lead to misleading actions. In order to deal with such situations caching and replication have been proposed.

The Caching Approach. The key idea behind caching [10,22] is to keep content close to the end-user according to a cache replacement policy. Specifically, the end-user's request for an object is posed to a proxy server, which may contain a cached version of the object. If the proxy server contains a "fresh copy" of the requested object (cache hit) then the end-user receives it directly from the proxy cache, elsewhere (cache miss) the end-user is redirected to the origin server (where the Web site is located). Therefore both the bandwidth consumption and the network traffic are reduced [11,2]. Additionally, network availability is significantly improved since the end-user may receive a copy even if the origin server is unavailable. Another advantage of caching is that fresh content is added into the caches leading to better storage usage.

Y. Manolopoulos, J. Pokorný, and T. Sellis (Eds.): ADBIS 2006, LNCS 4152, pp. 200–215, 2006.

A complementary to caching technique is prefetching [14]. Prefetching is proposed to find meaningful object access patterns in order to predict future requests. Therefore, objects may be transferred to the proxy server a priori (before they are even requested).

The Replication Approach. The main idea is to bring static content replicas close to the end-user. This is currently applied in the Content Distribution Networks [16,23]. A CDN consists of a set of surrogate servers geographically distributed in the Web, which contain copies (replicas) of content belonging to the origin server (according to a specific storage capacity). Therefore, CDNs act as a network layer between the origin server and the end-users, for handling their requests. With this approach, content is located near to the end-user yielding low response times and high content availability since many replicas are distributed. The origin server is "relieved" from requests since the majority of them is handled by the CDN, whereas, Quality of Service (QoS) and efficiency are guaranteed in a scalable way. Finally an important characteristic of the CDNs is the efficiency against flash crowd events [25]. Specifically, a flash crowd event occurs when unpredictably numerous users access a Web site. Events that affect global communities (i.e. Sept. 11th, Tsunamis etc) lead to flash crowd events affecting popular news Web sites. The side effects are significant: DoS, increased network latency and Web servers overloading. Therefore it is important to enhance content delivery management especially in unpredictable crisis situations.

Caching and replication deals with situation as separate approaches. While caching is mainly addressed to proxy servers, replication is the main technology of CDNs. However, implementing caching techniques over a CDN may improve performance by allowing fresh content to be replicated. In this paper, we focus on adapting representative cache replacement policies over a CDN along with replication. We explore the potential performance benefit in terms of perceived latency, hit ratio and byte hit ratio by using surrogate servers both as replicators and proxy caches. Caching and replication may benefit if used together, shown by our extensive experimentation using a detailed simulation model. Moreover, we demonstrate the robustness of the integrated approach in a CDN during a flash crowd event since it is a crucial issue.

The rest of this paper is organized as follows. In Sect. 2 we discuss the motivation of this work and present some previous related work. Section 3 formally presents the problem of content management in CDNs when using replication and caching. In Sect. 4 a brief description of the developed simulation model is given which has been used in order to perform the experiments presented in Sect. 5. Finally, the conclusion of this work and potential future work are given in Sect. 6.

2 Previous Work and Motivation

2.1 Previous Work

The performance of CDNs is affected by three main issues:

- **Surrogate servers placement over the network:** The optimal selection of proper spots over the network [12,19,20] where the surrogate servers should be placed yielding optimized performance. Algorithms that proposed to solve this problem are investigated in [19].

- **Content outsourcing:** The detection of the proper content for outsourcing[6]. Full mirroring is a naive approach because even if disk prices are continuously dropping, the sizes of Web objects increase and updating such a huge amount of Web objects is a cumbersome task.
- **Object replicas' placement:** Placing object replicas on surrogate servers [7,17,18,8] in a way that leads to optimized performance. Benchmarks of algorithms that manage this issue can be found in [7].

In this paper, we address the problems of object replicas selection and placement by applying caching policies integrated with the existing replication scheme. The problem of optimal content placement is proved to be NP-complete and therefore only heuristic approaches are feasible [7] such as Greedy Global algorithm. Greedy global recursively, for all objects and surrogate servers, detects the object that if placed at a specific surrogate server leads to optimized performance. Although Greedy Global seems to be the choice, its complexity is too high for applying a per-object placement on a large set of surrogate servers and objects. An alternative self-adaptive algorithm (lat-cdn) has been proposed in [17] that requires no other knowledge (such as recorded access logs) besides the network topology. The il2p algorithm [18] is proposed which takes into account the servers' load. Specifically, il2p using recursively two phases selects which object should be placed and where. During the first phase for each object the appropriate surrogate is selected minimizing network latency. Given the candidate pairs of (object, surrogate server), at the second phase, the one that yields the maximum utility value (depended on server's load) is selected.

Since CDNs have to deal with large amounts of data it is crucial to apply several data and communication management policies. Up to now, the uncooperative pull-based [23,26], cooperative pull-based [1], cooperative push-based and uncooperative push-based are the basic approaches, as reported in [16].

2.2 Motivation

The motivation of this work originates from the idea of improving a cooperative push based CDN by solving problems arising from pure replication. More specifically:

- Due to replication and distribution cost, a replicas' placement should be static for a large amount of time. This leads to unoptimized storage capacity usage since the surrogate servers would contain redundant content. If the end-users' access patterns change the replicas will no longer cover a large percentage of the requests. Besides replication no other action, such as replacement of unpopular objects by other currently popular, is performed.
- The placement of surrogate servers on the network is static reducing the flexibility of the CDN.

A possible brute-force solution to fight these drawbacks is to upgrade the Web servers and the network infrastructure. Faster Web servers and increased bandwidth solves the problem of fast data transfer and handling large amount of requests. However, this is a temporary solution. It includes increasing economic cost, since more and more resource-demanding services would emerge flooding again the network. Furthermore,

it is not scalable since upgrading the hardware infrastructure is not always practically and economically feasible.

In order to deal with the static nature of the information stored on the surrogate servers we propose the integration of caching and replication. If replication and caching cooperate they may be beneficial since both deal with the same problem but from a different approach. Although caching may suffer from low hit ratio and byte hit ratio [11] (typically below 50%) the performance gain from static replication along with caching in terms of response time (as we will show in the experiments) is significant. An evaluation of caching and replication as seperate approaches in CDNs is covered in [9], where caching outperforms but replication is still preferred for content availability and reliability of service. In [3] authors proved that integrating a simple LRU with replication, on a CDN, via a hybrid greedy algorithm yields performance outperforming a pure caching or replication scheme. Specifically, in each iteration a benefit value for every server-object pair is assigned and the one that produces the best benefit is selected for replication. At the end of the algorithm a percentage of the available storage capacity is reserved by static content and the rest is available for LRU. However, the possibility of using various representative cache replacement policies is not examined and the proposed approach is not tested during flash crowd events.

To the best of our knowledge, in the past the possibility of using caching along with replication on CDNs has not been studied in more extend. Therefore, the challenge is to improve the performance by using caching and replication together. In the context of integrating caching policies on a CDN's infrastructure our primary contributions are:

- Extend the policies of content selection and placement on CDNs by adopting representative cache replacement techniques. We select the LRU, LFU and SIZE as representatives of the main categories of cache replacement algorithms namely *Recency, Frequency* and *Size based* [24].
- Develop a detailed trace-driven simulation environment to test the efficiency of the proposed integrated scheme. The development of such an environment is crucial since we can capture the behavior of a realistic CDN infrastructure. Moreover, we avoid the oversimplified approach of a hop-based implementation that may give misleading results.
- Provide extensive experimentation covering all the possible combinations with real and artificial datasets, using representative cache replacement policies and replication at different levels of integration showing that pure caching or replication cannot meet the performance benefit of the integrated method.
- Demonstrate results proving that the integration has superior performance during flash crowd events and address several considerations and future road maps for such an integrated approach.

3 Integrating Caching in a Cooperative Push-Based CDN

Here we formally propose the problem of content management on cooperative push-based CDNs using replication integrated with caching policies. We choose the cooperative push-based scheme since it has been proved in [7] that is optimal. According to this approach, replicas are selected and placed at the surrogates servers a priori. Then

the surrogate servers cooperate with each other in order to reduce the response times and replication cost. Specifically, the end-users' requests are directed to the closest surrogate server. If the surrogate server contains the requested object then it is satisfied without causing traffic to the network backbone. Otherwise, the request is redirected to another server. The CDN may redirect the request to a surrogate server which contains the requested object or to the origin server, if the object is not outsourced at all. The bandwidth is shared among the surrogate servers and the objects replication redundancy is reduced.

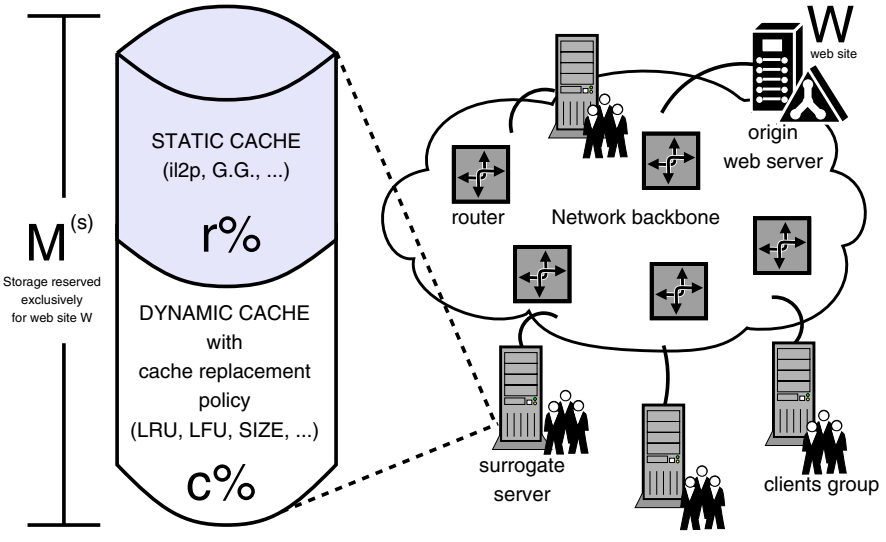

Fig. 1. CDN infrastructure

Here, we propose a modification of the cooperative push based scheme. Specifically, we consider the surrogate servers to operate both as static caches and proxy caches by partitioning the available storage capacity into two parts. The first one is used for replicating statically content and the second one for running a caching policy replicating content dynamically (Fig. 1). Assigning such a "dual" role to surrogate servers is feasible due to their increasing capacities and capabilities. When a surrogate server receives a request for an object a check to the static cache is performed. If it is a hit the request is served, else another check to the dynamic cache is performed. In case the object is in the dynamic cache, it is served and the cache is updated according to the cache replacement policy. If the requested object is not outsourced either in the dynamic cache, it is pulled from another server (selected based on proximity measures) and stored into the dynamic cache according to the current cache replacement policy and then the end-user receives the cached object. Therefore the end-user deals only with the nearest surrogate server and is not redirected elsewhere. Cached objects will be available in cache for future requests as long as they are allowed by the current cache replacement policy. The content of a surrogate server adapts to the current needs for objects and the static

nature of replication is overcome. The surrogate server plays a more active role and performs content management deciding which object should remain or not. Furthermore, besides the static cache, the content selection and placement is automated and it fits to the current objects' access pattern. The reason to keep the static part of the cache, as we will prove in the experiments, is to maintain content availability by distributing a large number of replicas to the network.

Table 1. Variables description

Variable	*Description*
W	The Web site
N	Number of objects of W
$W^{(\mathrm{s})}$	Web site's size
$U_k^{(\mathrm{s})}$	Size of k^{th} object
M	Number of surrogate servers
M_i	The i^{th} surrogate server
$M_i^{(\mathrm{s})}$	Storage capacity of the i^{th} surrogate server
$M^{(\mathrm{s})}$	Storage capacity of each surrogate server
f_{ik}	Function indicating whether the k^{th} object is placed at the i^{th} surrogate server or not
r	Percentage of the $M^{(\mathrm{s})}$ for replication
c	Percentage of the $M^{(\mathrm{s})}$ for caching

Therefore, consider a Web server representative who has signed a contract with the described CDN for outsourcing content of a Web site W. The Web site contains N objects initially located only at the origin Web server (outside of the CDN). The total size of W is $W^{(\mathrm{s})}$ and is given by the following equation:

$$W^{(\mathrm{s})} = \sum_{k=1}^{N} U_k^{(\mathrm{s})} \tag{1}$$

where $U_k^{(\mathrm{s})}$ is the size of the k^{th} $(1 \leq k \leq N)$ object.

Let M be the number of surrogate servers consisting the CDN. Each surrogate server $M_i (1 \leq i \leq M)$ has a total cache size $M_i^{(\mathrm{s})}$ dedicated (hired) for W. However, the surrogate servers may contain content from other Web sites without interfering with $M_i^{(\mathrm{s})}$. The $M_i^{(\mathrm{s})}$ is exclusively reserved for replicating content of W, of which the original copies are located in the origin Web server. For simplicity, we consider that the surrogate servers are homogeneous (same storage capacity $M_i^{(\mathrm{s})} = M^{(\mathrm{s})} (1 \leq i \leq M)$).

In order to apply replication and caching techniques the available storage capacity is split into two parts (Fig. 1):

– **Static cache:** Dedicated for replicating content statically. Its size is a percentage $r, (r \in [0..1])$ of $M^{(\mathrm{s})}$. Therefore, the replicated objects, in static cache, obey the following constrain:

$$\sum_{k=1}^{N}(f_{ik}U_k^{(\mathrm{s})}) \leq rM^{(\mathrm{s})} \tag{2}$$

where f_{ik} is a function denoting if an object exists (outsourced) in cache or not. Specifically, $f_{ik} = 1$ if the k^{th} object is placed at the i^{th} surrogate server and $f_{ik} = 0$ otherwise. The content of the static cache is defined by applying a replication algorithm like il2p.

– **Dynamic cache:** Reserved for applying cache replacement policies. The size reserved for dynamic caching is a percentage $c, (c \in [0..1])$ of $M^{(\mathrm{s})}$. More specifically, the stored objects respect the following storage capacity constrain:

$$\sum_{k=1}^{N}(f_{ik}U_k^{(\mathrm{s})}) \leq cM^{(\mathrm{s})} \tag{3}$$

Initially, the dynamic cache is empty since it is filled with content at run-time according to the selected cache replacement policy (upon misses).

Given the above cache segmentation scheme, the percentages (r, c) and must obey the following:

$$r + c = 1 \tag{4}$$

If $c = 0$ the cooperative push-based scheme is applied where the pulled objects are not stored for future use (pure replication). If we set $r = 0$ the surrogate servers turns into cooperative proxy caches (dynamic caching only). For $c > 0$ and $r > 0$ we get the integrated approach where replication is used along with caching. Here the problem addressed is to select the optimal values for r and c, given a replica placement and caching algorithm, which improves the performance of the CDN.

4 CDNsim: The Simulation Testbed

For the experimentation needs, we have implemented a complete simulation environment, called CDNsim. CDNsim simulates a main CDN infrastructure and is implemented in the C programming language. It is based on the ParaSol library[1] which provides a parallel and discrete event simulation environment. Further details about CDNsim along with the source code can be found at http://oswinds.csd.auth.gr/~cdnsim/. Due to space limitations, only the basic characteristics of the simulator are presented here.

CDNsim uses a network graph generated by the GT-ITM internetwork topology generator [27] in order to build the network backbone with a realistic TCP/IP protocol implementation. This includes packets routing, retransmissions on errors or DoS, finite bandwidth links, bottlenecks, etc. Packets routing is performed by following the shortest paths generated by the Dijkstra algorithm. The nodes of the generated network topology are assigned to specific network elements which include the following a) routers, b) surrogate servers, c) origins servers and d) client groups (clients grouped according to their domains). Communication via routers causes the main network traffic and

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

perceived delays. Therefore, requests that lead to cache misses and must be pulled are "expensive" in order to be satisfied. We avoid the oversimplified approach of network latency depended only by the number of network-hops since we simulate a realistic network.

There are several clients' log files on the Web[2] but we do not have the respective Web sites' structure, and vice versa. Moreover, the CDN providers do not offer their log files. Therefore, we use articial workloads and Web sites. For that reason, we used the R-MAT Web site generator [5] and we assigned sizes to the objects according to the log-t distribution as described in [13]. For each of the generated sites we have produced a set of object requests using the generator presented in [14].

An issue that may affect the performance of a simulation is cache consistency. Since there is a certain amount of literature that deals with the problem of cache consistency we assume that there is implemented an appropriate mechanism like Web server invalidation [4] that ensures the freshness of the objects. Moreover the probability of requesting for a stale object is low because according to [15] the duration between two modifications in the same object is up to 24 hours.

5 Performance Evaluation and Experimentation

In this section we present results demonstrating the behavior of the integrated scheme in terms of mean response time, hit ratio and byte hit ratio. Section 5.1 summarizes the performance parameters evaluated in the experiments. In Sect. 5.2 the simulations' setup and the used datasets are described while Sect. 5.3 presents the experimentation.

5.1 Parameters

Here we briefly present the performance criteria used in the experiments, namely the a) mean response time, b) response time CDF, c) hit ratio and d) byte hit ratio. These criteria have been used since they are the most indicative ones for performance evaluation.

- **Mean response time.** This is the expected time for a request to be satisfied. It is the summation of all request times divided by their quantity. Low values denote that content is close to the end-user.
- **Response time CDF.** The Cumulative Distribution Function (CDF) in our experiments denotes the probability of having a response times lower or equal to a given response time. The goal of a CDN is to increase the probability of having response times around the lower bound of response times.
- **Hit ratio.** It is defined as the fraction of cache hits to the total number of requests. A high hit ratio indicates an effective cache replacement policy and defines an increased user servicing, reducing the average latency.
- **Byte hit ratio.** It is the hit ratio expressed in bytes. It is defined as the fraction of the total number of bytes that were requested and existed in cache to the number of bytes that were requested. A high byte hit ratio improves the network performance (i.e. bandwidth savings, low congestion etc.).

[2] Traces available in the Internet Traffic Archive: http://ita.ee.lbl.gov/html/traces.html

5.2 Simulation Configuration

Network and CDN Topology. Using the GT-ITM we have indicatively created an AS network topology with a total of 3037 nodes. Given a standard link speed of 1MB per second we have generated the shortest paths of all nodes to all nodes for optimal packets routing. A set of 20 surrogate servers is randomly attached in the existing network backbone.

For the experimentation needs we express $M^{(\mathrm{s})}$ as a percentage p of the origin server's Web site size $W^{(\mathrm{s})}$ (i.e. $M^{(\mathrm{s})} = pW^{(\mathrm{s})}$). For the static cache, we have used the il2p algorithm since its complexity is acceptable on a per-object replication. Specifically, we follow the following steps to initialize and run the surrogate servers caches:

1. Initially the surrogate servers are empty. We set the (r, c).
2. We fill the static cache specified by the r by running the il2p algorithm.
3. We set the cache replacement policy for the dynamic cache specified by c. In our experiments the caching policy may be LRU, LFU or SIZE.

The (r, c) pairs that we used for the simulator are i) (1,0) for pure replication, ii) (0.8,0.2), iii) (0.5, 0.5), iv) (0.2, 0.8) and v) (0, 1) for pure caching. Additionally for setting the upper and lower bound of performance we have configured CDNsim for full mirroring of the Web sites and then to have empty disks (caches) without possible addition of objects into the cache.

Datasets. With the above described configuration we present three experiments using three datasets. The first dataset concerns an artificial Web site of 2994 objects, size 746.86 MBs and set of 1969114 requests. We set $p = 0.15$ leading to a cache size of 112.03 MBs. As mentioned before, we had to use synthetic datasets due to the lack of real CDN traces. The second dataset includes the same Web site but it runs under a flash crowd event. We have shrunk the time window of the requests in order to increase their density and rate. This leads to greater network traffic since more packets travel simultaneously and the surrogate servers' load is increased because of the greater number of simultaneously active sessions. The final dataset is a real Web site. We have used the Standford's Web site [3] which contains 281904 objects and its size is 8.66 GBs. The number of requests is 3744460 and the $p = 0.015$, since it is much larger than the artificial Web sites, leading to storage capacity of 133,15 MBs. Table 2 summarizes the overall experimentation configuration.

5.3 Simulation Results

In this section we present the results and we compare the performance parameters of pure replication, pure caching and integration of replication with LRU, LFU and SIZE (representatives of *Recency, Frequency* and *Size based* policies [24]).

Experiments Without Flash Crowd Event

Artificial Data. The resulting mean response times are depicted in Fig. 2. The x axis represents the integration level of replication and caching (the (r, c) values) while the

[3] http://www.stanford.edu/~sdkamvar/research.html

Table 2. Experimentation configuration summary

Network topology	AS 3037 nodes
Link speed	1MB/s
N	20
(r, c)	(1, 0), (0.8, 0.2), (0.5, 0.5), (0.2, 0.8), (0, 1), full mirroring, empty disks
Rep. alg.	il2p
Caching policies	LRU, LFU, SIZE
Dataset 1	Artificial, 2994 objects, 746.86 MBs, 1969114 reqs., $p = 0.15$
Dataset 2	Artificial, 2994 objects, 746.86 MBs, 1969114 reqs., $p = 0.15$, flash c.e.
Dataset 3	Real, 281904 objects, 8.66 GB, 3744460 reqs., $p = 0.015$

y axis is the resulting mean response times of the requests. The performance limits are bounded by the cases of full mirroring and empty disks. By using pure replication the mean response time is reduced significantly denoting that the existence of replicas on the network participates to the improvement of performance setting the limits of pure replication. However, there is still room for optimization. For the integration level where $(r, c) = (0.5, 0.5)$ the mean response times are reduced up to 40% compared to pure replication and 15% compared to pure caching, for all caching schemes. Reducing c the results are gradually worsen for values of $c > 0.8$ meaning that there is still need for static replicas to exist. This behavior is presented also in Fig. 3 which shows the hit ratio at the y axis and the integration level at x axis. The combination of replication with SIZE outperforms all the other combinations in terms of hit ratio. The hit ratio gain may reach 70% compared to pure replication and around 10% compared to pure caching. Examining the byte hit ratio (y axis) at Fig. 4 at different pairs of (r, c) (x axis), we can conclude that the performance gain in terms of byte hit ratio is not significant, however a gain around 10% is still possible with all cache replacement algorithms demonstrating the same behavior. Another illustration of the situaton is shown in Fig. 5. The x axis contains the response times of all requests ascending while the y represents the portion of requests that their response time is lower than a given value. The CDFs of the intergrated approach tend to fit the ideal situation of full mirroring with SIZE as the leading algorithm.

In this experiment we can conclude that LRU, LFU and SIZE has similar behavior in terms of mean response time, byte hit ratio and CDFs. The leading algorithm, in the hit ratio case, is SIZE which can be explained by the fact that SIZE favors smaller objects leading to increased number of objects in cache.

Real Data. The second experiment, given the same execution environment, is run with the Standfors's Web site. Figure 6 illustrates the mean response time. As we can see the pure replication is unable to offer considerable performance benefit. This can be explained by the fact that $W^{(\mathrm{s})} \gg M^{(\mathrm{s})}$, therefore a relatively small set of replicas cannot cover a large enough percentage of the requests. However, for $(r, c) = (0.8, 0.2)$ we get a 70% which is peak for all caching policies. Pure caching seems to be the choice here because the average object size is too small (around 33 Kb). Although the pure caching outperforms the use of integration is preferred because it has comparable results to the pure caching and distributes a number of replicas over the network increasing

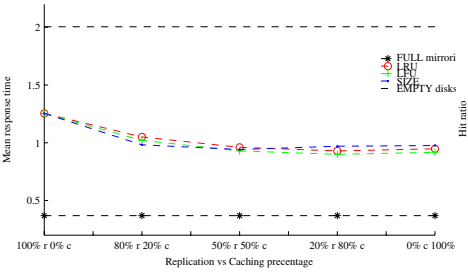

Fig. 2. Artificial Web site - Mean response time

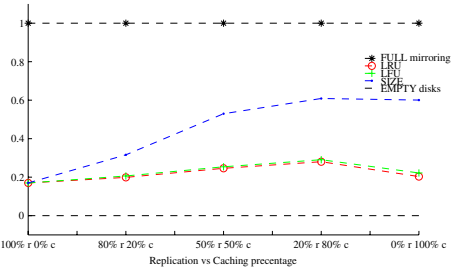

Fig. 3. Artificial Web site - Hit ratio

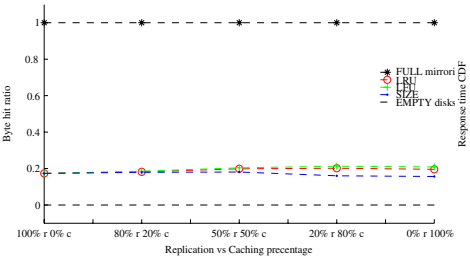

Fig. 4. Artificial Web site - Byte hit ratio

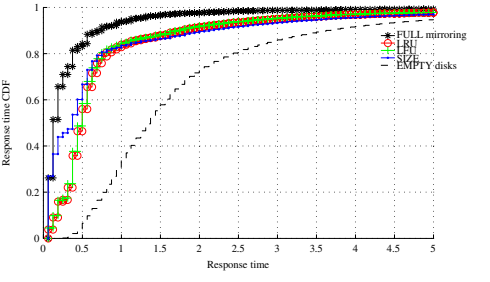

Fig. 5. Artificial Web site - CDF (r, c) = (0.5, 0.5)

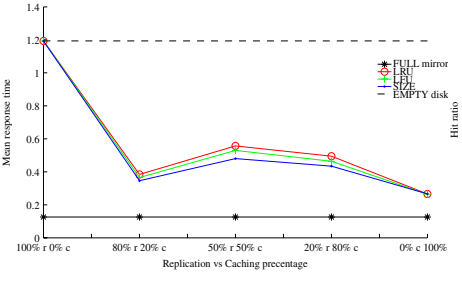

Fig. 6. Real Web site - Mean response time

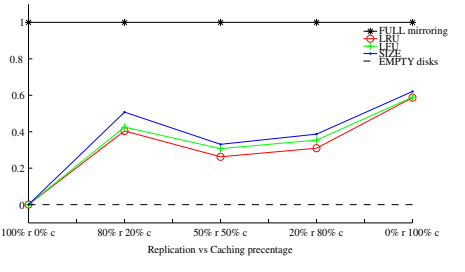

Fig. 7. Real Web site - Hit ratio

content availability. Both in hit ratio Fig. 7 and byte hit ratio Fig. 8 the peak occurs at the same integration level ($(r, c) = (0.8, 0.2)$) with SIZE slightly better in terms of hit ratio and worse in terms of byte hit ratio. At $(r, c) = (1.0, 0.0)$ the estimated values are quite low since the available storage capacity is too limiting and therefore the replica placement algorithm does not perform well. The reasons we have selected this storage capacity constrain ($p = 0.015$) are: a) the il2p execution time for the real dataset was restricting and b) we would like to monitor the system performance with low cache sizes. It is clear that in low cache sizes caching is preferred since it updates the content

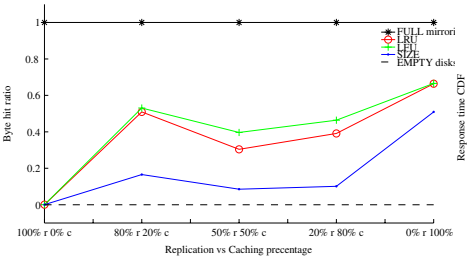

Fig. 8. Real Web site - Byte hit ratio

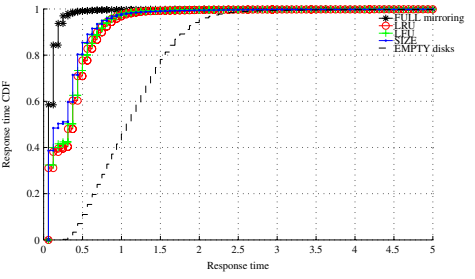

Fig. 9. Real Web site - CDF (r, c) = (0.8, 0.2)

while replication is not recommended since the static replicas cannot cover a satsifying portion of the requests. In Fig. 9 for $(r, c) = (0.8, 0.2)$ can the intergrated approach fits to the upper bound performance limit.

As a conclusion for this experiment, pure caching outperforms in case the objects have small sizes. However, at the integration level where $(r, c) = (0.8, 0.2)$, the results are comparable. As expected, LRU and LFU has similar behavior while SIZE is leading in terms of hit ratio and it is not recommended for optimizing byte hit ratio.

Experiments with Flash Crowd Event

As mentioned in the Introduction, it is crucial to enhance content delivery during flash crowd events. Therefore the integration of caching with replication should be tested appropriately.

Artificial Data. In this experiment we record the behavior of the CDN during a flash crowd event in the considered logs. The CDN's operation is intensive since a large amount of requests is served simultaneously. Figure 10 depicts the mean response times. The situation where the disks are empty leads to an unstable state where, as expected, increased response times are observed. In the case of full mirroring the performance is similar to the no flash crowd event case (Fig. 2) since the entire network backbone is skipped. Using pure replication an important performance benefit exists, however, now the response times are 100% larger than the no flash crowd event operation. For $(r, c) = (0.8, 0.2)$ the percieved mean response times are comparable to the ones depicted in Fig. 2 during no flash crowd event, meaning that the CDN copes with the flash crowd event efficiently. For LRU and LFU the performance of the model is the same as the no flash crowd event but for SIZE at $c > 0.2$ it is worse but still better than pure replication or caching. In terms of hit ratio (Fig. 11) the combination of SIZE with replication for $(r, c) = (0.2, 0.8)$ yields performance 60% greater than replication and around 8% better than caching. For this integration level, all caching policies reach peak in their performance. The expected byte hit ratio Fig. 12 seems to follow the same behavior just like the no flash crowd event case. Looking the model from the perspective of CDF Fig. 13 for $(r, c) = (0.8, 0.2)$ we notice that in the case of empty disks the distribution is uniform explained by the flash crowd event. For the inergated approach the choice algorithm is the SIZE.

Summarizing this experiment, during a flash crowd event the absence of a caching or replication mechanism leads to unacceptable response times. However, pure replication, as applied currently in CDNs, improves the performance. The performance may be significantly ameliorated using replication with caching showing the robustness of the integrated approach.

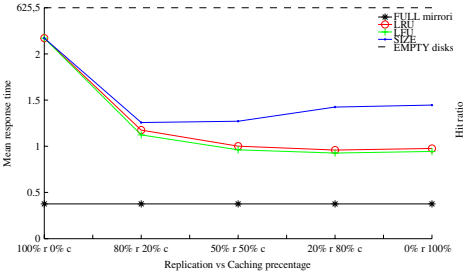

Fig. 10. Artificial Web site - Mean response time at flash crowd event

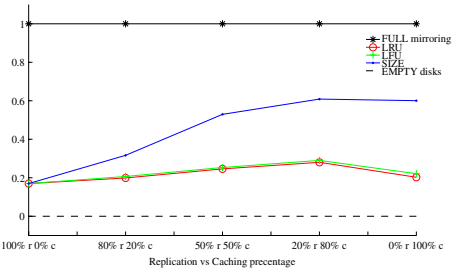

Fig. 11. Artificial Web site - Hit ratio at flash crowd event

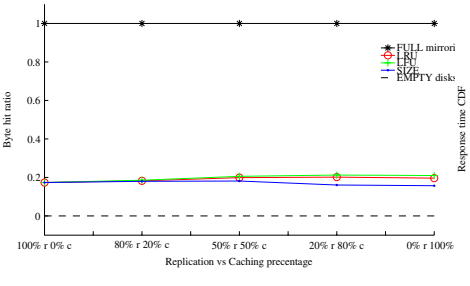

Fig. 12. Artificial Web site - Byte hit ratio at flash crowd event

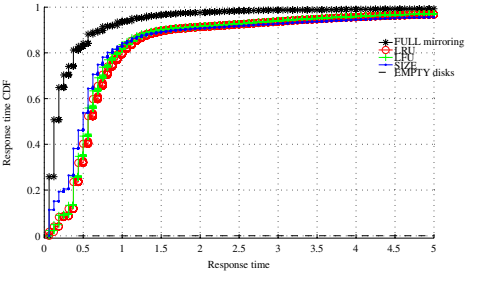

Fig. 13. Artificial Web site - CDF (r, c) = (0.8, 0.2) at flash crow event

Summary of Experiments

To summarize the experiments, we can conclude that the integration of replication with caching leads to improved performance in terms of perceived network latency, hit ratio and byte hit ratio. The results reinforce the initial intuition that replicating replicas statically for content availability along with caching policies improves the performance. Our experimentation has shown that:

– The integrated approach demonstrates mean response times up to 40% better than pure replication.

- A performance benefit of 15% may be achieved when compared with pure caching in terms of mean response time.
- Pure replication yields poor performance, 70% worse than the integrated approach, in terms of hit ratio.
- Pure caching demonstrates performance in hit ratio which may be 10% worse than the caching-replication combination.
- It can be observed in the experiments that there is not a fixed pair of (r, s) that gives us the peak of performance.
- As presented in the experimentation , CDNs using the integrated approach, demonstrate improved performance during a flash crowd event, comparable to the case of a no flash crowd event.
- The performance peak appears to be independent from the selected cache replacement policy.

6 Conclusion and Future Work

This paper investigates the potential performance gain occurring by replication and caching if used together in a CDN. We offered an extensive set of experiments exploring the performance limitations. For the purposes of the experiments a detailed simulation environment has been developed. It has been shown that caching outperforms static content replication. Moreover, a possible integrated scheme outperforms the pure replication or caching scheme as separate implementations. CDNs may take advantage of the dynamic nature of cache replacement policies while maintaining static object replicas for availability, reliability and bounded update propagation cost. Finally our experiments shown that CDNs are effectively fortified against flash crowd events.

The integrated approach should be tested and applied on several network topologies such as ad-hoc mobile wireless networks [21]. Currently we are working on the extension of the replicas placement in terms of dynamic data and various dynamic parameters of QoS, since it is an open issue in this work. Moreover, the development of an automated mechanism for detecting the appropriate level of integration (i.e. (r, c) pair) which leads to performance peak is crucial. Finally, another consideration is the implementation of a mechanism that dynamically recalculates the (r, c) at run-time adapting to the varying needs.

References

1. Annapureddy, S., Freedman, M.J., Mazières, D.: Shark: Scaling file servers via cooperative caching. In: 2nd Symposium on Networked Systems Design and Implementation, USENIX, ACM SIGCOMM, ACM SIGOPS (2005)
2. Arlitt, M., Friedrich, R., Jin, T.: Performance evaluation of Web proxy cache replacement policies. Lecture Notes in Computer Science **39** (2000) 149–164
3. Bakiras, S., Loukopoulos, T.: Increasing the performance of CDNs using replication and caching: A hybrid approach. In: 19th International Parallel and Distributed Processing Symposium, IEEE Computer Society (2005)
4. Cao, P., Liu, C.: Maintaining strong cache consistency in the world wide web. IEEE Transactions on Computers **47**(4) (1998) 445–457

5. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-MAT: A recursive model for graph mining. In: 4th SIAM International Conference on Data Mining, SIAM (2004)
6. Chen, Y., Qiu, L., Chen, W., Nguyen, L., Katz, R.H.: Clustering web content for efficient replication. In: 10th IEEE International Conference on Network Protocols, IEEE Computer Society (2002) 165–174
7. Jin, S., Wang, L.: Content and service replication strategies in multi-hop wireless mesh networks. In: 8th ACM International Symposium on Modeling, analysis and simulation of wireless and mobile systems, ACM Press (2005) 79–86
8. Karlsson, M., Karamanolis, C.: Choosing replica placement heuristics for wide-area systems. In: 24th International Conference on Distributed Computing Systems, IEEE Computer Society (2004) 350–359
9. Karlsson, M., Mahalingam, M.: Do we need replica placement algorithms in content delivery networks? In: 7th International Workshop on Web Content Caching and Distribution, IWCW (2002) 117–128
10. Katsaros, D., Manolopoulos, Y.: Caching in web memory hierarchies. In: 19th Annual ACM Symposium on Applied Computing, ACM Press (2004) 1109–1113
11. Kroeger, T.M., Long, D.D.E., Mogul, J.C.: Exploring the bounds of web latency reduction from caching and prefetching. In: USENIX Symposium on Internet Technologies and Systems, USENIX (1997)
12. Li, B., Deng, X., Golin, M.J., Sohraby, K.: On the optimal placement of web proxies in the internet: The linear topology. In: 8th International Conference on High Performance Networking, Kluwer, B.V. (1998) 485–495
13. Mitzenmacher, M., Tworetzky, B.: New models and methods for file size distributions. In: 41st Annual Allerton Conference on Communication, Control, and Computing. (2003) 603–612
14. Nanopoulos, A., Katsaros, D., Manolopoulos, Y.: A data mining algorithm for generalized web prefetching. IEEE Transactions on Knowledge and Data Engineering **15**(5) (2003) 1155–1169
15. Padmanabhan, V.N., Qiu, L.: The content and access dynamics of a busy web site: ndings and implications. In: ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, ACM Press (2000) 111–123
16. Pallis, G., Vakali, A.: Insight and perspectives for content delivery networks. Communications of the ACM **49**(1) (2006) 101–106
17. Pallis, G., Vakali, A., Stamos, K., Sidiropoulos, A., Katsaros, D., Manolopoulos, Y.: A latency-based object placement approach in content distribution networks. In: 3rd Latin American Web Congress, IEEE Computer Society (2005) 140–147
18. Pallis, G., Stamos, K., Vakali, A., Katsaros, D., Sidiropoulos, A., Manolopoulos, Y.: Replication based on objects load under a content distribution network. In: 22nd International Conference on Data Engineering Workshops, IEEE Computer Society (2006)
19. Tang, X., Xu, J.: QoS-aware replica placement for content distribution. IEEE Transactions on Parallel and Distributed Systems. **16**(10) (2005) 921–932
20. Szymaniak, M., Pierre, G., van Steen, M.: Latency-driven replica placement. In: International Symposium on Applications and the Internet, IEEE Computer Society (2005) 399–405
21. Tseng, Y.C., Ni, S.Y., Shih, E.Y.: Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. In: 21st International Conference on Distributed Computing Systems, IEEE Computer Society (2001) 481–488
22. Vakali, A.: LRU-based algorithms for web cache replacement. In: 1st International Conference on Electronic Commerce and Web Technologies, Springer-Verlag (2000) 409–418
23. Vakali, A., Pallis, G.: Content delivery networks: Status and trends. IEEE Internet Computing **7**(6) (2003) 68–74

24. Wang, J.: A survey of web caching schemes for the internet. Computer Communication Review **29**(5) (1999) 36–46
25. Wang, L., Pai, V., Peterson, L.: The effectiveness of request redirection on cdn robustness. In: 5th Symposium on Operating System Design and Implementation, USENIX (2002) 345–360
26. Yu, H., Vahdat, A.: Minimal replication cost for availability. In: 21st Annual Symposium on Principles of Distributed Computing, ACM Press (2002) 98–107
27. Zegura, E.W., Calvert, K.L., Bhattacharjee, S.: How to model an internetwork. In: Conference on Computer Communications, Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Networking the Next Generation, IEEE (1996) 594–602