A similarity based approach for integrated Web caching and content replication in CDNs

Konstantinos Stamos, George Pallis, Charilaos Thomos and Athena Vakali
Aristotle University of Thessaloniki
54124, Thessaloniki, Greece,
kstamos@csd.auth.gr, gpallis@ccf.auth.gr, chthomos@csd.auth.gr, avakali@csd.auth.gr

Abstract

Web caching and content replication techniques emerged to solve performance problems related to the Web. We propose a generic non-parametric heuristic method that integrates both techniques under a CDN. We provide experimentation showing that our method outperforms the so far separate implementations of Web caching and content replication. Moreover, we show that the performance improvement compared with an existing algorithm is significant. We test all these techniques in a simulation environment under a flash crowd event and a workload of a typical lightweighted CDN operation.

1. Introduction

The current network technology is challenged by the increasing growth of Web usage. A massive wave of simultaneous clients, might decrease the performance or even halt a state-of-the-art web server, the so-called "bottleneck" phenomenon [5]. Therefore, clients may experience latency or even denial of service (DoS). Two solutions have been proposed: *Web caching and prefetching* mainly applied in Web proxy servers and *content replication* currently implemented in the CDNs.

Web caching temporarily stores the recently requested objects in cache to become available for future requests [6]. Web caching may be combined with *prefetching* which detects meaningful object access patterns predicting future requests [7] . Therefore, objects may be transferred to the proxy server a priori.

Content replication creates replicas of objects as close to the clients as possible. Its current implementation is the Content Distribution Network (CDN), a trusted high-speed overlay network which is used to deliver Web objects, static data and streaming multimedia content [9]. Using points-of-presence, the so called surrogate servers, CDNs manage to serve data faster and in a more reliable way. An important advantage of CDNs is that origin servers are protected from flash crowd events [10]. A flash crowd event occurs when numerous users access a Web site simultaneously, such as the one occurred in September 11^{th} 2001 when users flooded popular news sites.

Web caching is characterized by a dynamic nature in terms of its content update opposite to the static nature of replicated content in CDNs. A scheme that implements both under the same technology could demonstrate performance improvement. Therefore, we consider a CDN that its surrogate servers act both as cooperative replicators and proxy caches. We propose a method that shares the storage capacity of each surrogate server of a CDN for Web caching and content replication. Our method, called *Similarity Replication Caching* (*SRC*), uses the heuristic criterion of *placement similarity*.

This paper is organized as follows. In Section 2 we discuss the motivation of this paper and present the related work. Section 3 propose the Web caching and static content replication integration problem along with our developed method. In Section 4 the experimentation is presented. Finally, the conclusion of our work are given in Section 5.

2. Motivation and Paper's Contribution

The content delivery service has a crucial impact in terms of CDN pricing and performance. In this work we focus on the placement of object replicas which is proved in [4] to be NP-complete. In general, an efficient replica placement reduces the cost of replica's placement and improves the overall network performance [4, 8]. However, replicas' placement is forced to be static for a considerable amount of time due to replication and distribution cost. This leads to inefficient storage capacity usage since the surrogate servers' cache after a period of time would contain unwanted objects.

The key issue is to reduce the impact of the disadvan-

15th International Database Engineering and Applications Symposium (IDEAS'14)
0-7695-2577-6/06 $20.00 © 2014 IEEE

tages of static replication by enabling partially dynamic replication. Towards this direction, a greedy *Hybrid* algorithm that combines an LRU cache replacement policy with static content replication on a CDN was proposed in [1]. The experimental results showed that the performance of the *Hybrid* algorithm outperformed the Web caching and static content replication as separate implementations.

Motivated by this work, we further study this approach by presenting an alternative method which would detect automatically the appropriate percentage that splits the available storage capacity of each surrogate server for Web caching and static content replication. The challenge for us is to achieve a delicate balance between replication and caching towards improving the CDN's performance under flash crowd events. Our contributions can be summarized as follows:

- We propose a CDN where the surrogate servers act both as proxy caches and static replicators under a cooperative environment.
- Develop a placement similarity approach (the so called *SRC*) a non parametric method, for evaluating the level of integration of Web caching with content replication.
- Provide experimentation showing that our method performs better than the *Hybrid* algorithm [1] during flash crowd events and typical CDN load.

3. Similarity based caching-replication integration

3.1. Problem formulation

Consider an origin server which contains a Web site O. The Web site includes N objects with total size $O^{(s)}$. The objects may refer to simple Web pages or groups of pages. Let K be the number of surrogate servers consisting the CDN. Each surrogate server $K_i (1 \leq i \leq K)$ has a total cache size $K_i^{(s)}$ reserved for storing objects from O. However, the surrogate servers may contain content from other Web sites without interfering with $K_i^{(s)}$. The $K_i^{(s)}$ is exclusively reserved for replicating content of O. Under our approach, each $K_i^{(s)}$ is split into two parts, namely the:

- **Static cache:** Dedicated for static content replication referred as s. Its size $s^{(s)}$ is less or equal to $K_i^{(s)}$. The content of the static cache is identified by applying a content replica placement algorithm (such as il2p [8]).
- **Dynamic cache:** Reserved for Web caching using cache replacement policies. It is refered as d with size $d^{(s)}$ less or equal to $K_i^{(s)}$. Initially, the dynamic cache is empty and it gradually fills with objects upon cache misses, according to the cache replacement policy.

From the above it occurs that $s^{(s)} + d^{(s)} = K_i^{(s)}$. Therefore, if $d^{(s)} = 0$ the cooperative push-based scheme [4] is applied where, the surrogate servers cooperate upon case misses and the content of the caches remains unchanged. On the other hand, if $s^{(s)} = 0$ the surrogate servers turn into cooperative proxy caches (dynamic caching only). For $s^{(s)} > 0$ and $d^{(s)} > 0$ we get the integrated approach where replication is used along with caching. 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 replace ment policy. If the requested object is not outsourced either in the dynamic cache, it is pulled from another server and stored into the dynamic cache according to the current cache replacement policy. Then the end-user receives the cached object

The problem addressed is to select the optimal values for $s^{(s)}$ and $d^{(s)}$, given a replica placement and cache replacement algorithm, which improves the performance of the CDN.

3.2. Proposed method

Given a replication algorithm, a cache replacement policy, a clients trace file and each $K_i^{(s)}$, as input parameters, the storage capacity of each surrogate server is shared to Web caching and static replication without further tuning. It should be mentioned that *SRC* is a generic method and it is not focused on a specific replication or cache replacement algorithm. Therefore, it can be employed under any possible combination of cache replacement and replication algorithms.

In order to assess the importance of an initial objects placement through time, at each time slot t_x the similarity of the current placement C with the initial placement R should be evaluated. The placement varies through time as new objects are stored replacing existing ones. In Figure 1 this concept is depicted. Placement similarity is a measure that quantifies the quality of a placement through time which is the heuristic criterion used by *SRC*, defined as follows:

Definition (Placement similarity). The similarity degree of two objects placement sets X_{t_1} and X_{t_2} placed at a surrogate server at different time slots t_1, t_2 is defined as

$$|S_{t_{1,2}}|_{norm} = \frac{|X_{t_1} \cap X_{t_2}|}{|X_{t_1}|} \quad (1)$$

where $S_{t_{1,2}}$ is the intersection of X_{t_1} with X_{t_2}, $|S_{t_{1,2}}|$ the cardinality of this intersection and $|S_{t_{1,2}}|_{norm}$ the normalized cardinality (similarity). More specifically, the X_{t_1} and

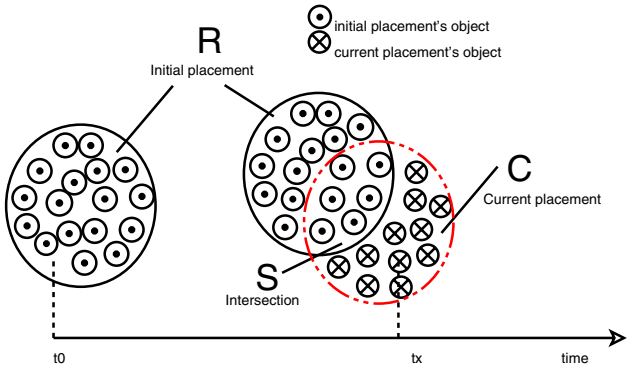

Figure 1. Evolution of the initial placement through time

X_{t_2} sets contain the actual objects placed at a surrogate server's cache at the time slots t_1 and t_2 respectively.

We use the mean placement similarity $\bar{\mu}_i$ for each surrogate server as a criterion for selecting the values of $s^{(s)}$ and $d^{(s)}$. The $\bar{\mu}_i$ is an indication of the amount of the static content that remains popular through time. Then, for each surrogate server i the storage capacity is arranged as:

$$s_i^{(s)} = \bar{\mu}_i K_i^{(s)} \quad (2)$$

$$d_i^{(s)} = (1 - \bar{\mu}_i) K_i^{(s)} \quad (3)$$

where the $s_i^{(s)}$ and $d_i^{(s)}$ are the respective values of $s^{(s)}$ and $d^{(s)}$ for the i^{th} surrogate server. The *SRC*'s operation can be summarized by the following steps taking place at each surrogate server i of the CDN:

1. Define the replica placement algorithm, the cache replacement policy, the trace file and the $K_i^{(s)}$
2. Create an initial objects placement R_i using the replica placement algorithm and the $K_i^{(s)}$ surrogates' capacity constrains.
3. Execute the cache replacement policy by using the trace file and record the similarity of current placement C_i with R_i. The trace file is a training set of requests (history log) which provides knowledge for the future clients behavior.
4. Calculate the mean similarity $\bar{\mu}_i$.
5. Set $s_i^{(s)} = \bar{\mu}_i K_i^{(s)}$ and $d_i^{(s)} = (1 - \bar{\mu}_i) K_i^{(s)}$.
6. Run the replica placement algorithm using the $s_i^{(s)}$ constrains.

It can be concluded that *SRC* acts as a "thin wrapper" of the existing techniques suitable for a generic integration of static replication with Web caching. This versatile behavior is the key advantage of *SRC* since it may exploit existing, well documented and established algorithms.

4. Performance evaluation and experimentation

4.1. The simulation testbed

In order to meet the experimentation needs, we have developed a full trace driven simulation environment called CDNsim. Further details about the simulation setup can be found in Table 1. We assume the existance of a strong cache-consistency technique, such as invalidation reports [2].

Piece of Software	Description	Reference
CDNsim	CDN simulator	http://oswinds.csd.auth.gr/~cdnsim/
R-MAT	Web site generator	[3]
	Requests' stream generator	[7]
Parameter	**Description**	**Value**
Number of surrogate servers	Randomly placed	100
Cache size		20% of the Website's size
Number of requests	70% training set, 30% test set	~7million
Network topology	AS internet topology	3037 nodes

Table 1. Simulation testbed

4.2. Performance parameters

The criteria used in the experiments are considered to be the most indicative ones for performance evaluation. These criteria are: a) mean response time, b) hit ratio and c) byte hit ratio.

4.3. Content Management Policies

We experiment with the following content management policies:

- **Caching:** All the storage capacity of the surrogate servers is allocated to dynamic caching ($d_i^{(s)} = K_i^{(s)}$). The selected cache replacement policy is LRU since we would like to compare with the existing *Hybrid* approach [1] where the LRU is used.
- **Replication:** Using the il2p algorithm [8] we define which objects will be replicated statically in each server using all the available storage capacity ($s_i^{(s)} = K_i^{(s)}$). We choose the il2p since due to its complexity can handle large object sets and perform a per-object replication.

- **Hybrid:** This is the existing algorithm [1] that combines static replication and Web caching using an analytic model of LRU. The Hybrid gradually fills the surrogate servers caches with static content at each iteration, as long as it contributes to the optimization of response times.

- **SRC:** Our proposed method, which uses the LRU (for comparison with Hybrid) as cache replacement policy and il2p for static replication. We run il2p with the total storage capacity of the surrogate servers as constrain. Then we calculated the mean similarity by executing LRU. Specifically, the proposed $\bar{\mu}_i$ values where around 0.26 for all surrogate servers. Finally, we executed il2p, again, with the new constrains posed by $s^{(s)}$ at each surrogate server.

4.4. Experimentation

The results of the expirements are summarized in Table 2 where the last column holds the performance gain of *SRC* compared to (*Caching*, *Replication*, *Hybrid*). For the "no flash crowd event" the *SRC* is the leading algorithm. The close performance of *SRC*, *Hybrid* and *Caching* in terms of mean response time can be explained by the fact that a large percentage of storage space is allocated to Web caching in all three cases (~70%). The major advantage of *SRC* in this case is the hit ratio, while clearly static replication suffers from bad performance due to its static nature.

Performance parameter	Caching	Replication	Hybrid	SRC	SRC gain %
Mean response time (no flash)	0.767250	1.202716	0.776148	0.760608	(1%, 36%, 2%)
Hit ratio (no flash)	0.260668	0.178074	0.247060	0.303923	(15%, 41%, 19%)
Byte hit ratio (no flash)	0.244855	0.241380	0.231352	0.253078	(3%, 5%, 9%)
Mean response time (flash)	2.924987	-	1.192838	1.532845	(47%, -, 29%)
Hit ratio (flash)	0.260281	-	0.247091	0.326257	(20%, -, 24%)
Byte hit ratio (flash)	0.244124	-	0.231478	0.266260	(8%, -, 13%)

Table 2. Results

At the case of "flash crowd event" we monitor the behavior of the CDN using the same logs (we have shrunk the time window of the requests). The CDN operation is intensive as a large amount of requests is served simultaneously. The impact of the flash crowd event was significant that in the *Replication* experiment only the 50% of the requests have been satisfied due to DoS. For this reason it has been excluded from Table 2. Moreover, the response times of *Replication* where extremely high (up to 200 times higher than the "no flash crowd event" case. The *Hybrid* and *SRC* approaches present the best mean response times compared to *Caching* and *Replication* ones. It is important to note that the mean response times in the flash crowd event for the *Caching*, *Hybrid* and *SRC* policies are comparable to the non flash crowd values which means that the CDN copes with the flash crowd event efficiently. *SRC* not even is the leading algorithm but it demonstrates major perfomance improvement in comparison to the "no flash crowd event" case.

5. Conclusion

This paper proposes a novel methodology for integrating Web caching and static content replication over CDNs. The notion of placement similarity is introduced and used as a heuristic criterion. Its architecture is open and ready for various cache replacement policies and content replication algorithms. We can conclude that using both caching and replication policies, in the context of the *SRC* method, leads to improved performance in mean response time, hit ratio and byte hit ratio. Moreover, the CDN can successfully withstand the pressure imposed by a flash crowd event.

References

[1] S. Bakiras and T. Loukopoulos. Increasing the performance of CDNs using replication and caching: A hybrid approach. In *19th International Parallel and Distributed Processing Symposium*. IEEE Computer Society, April 2005.

[2] G. Cao. A scalable low-latency cache invalidation strategy for mobile environments. *IEEE Transactions Knowledge and Data Engineering*, 15(5):1251–1265, May 2003.

[3] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A recursive model for graph mining. In *4th SIAM International Conference on Data Mining*. SIAM, April 2004.

[4] S. Jin and L. Wang. 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*, pages 79–86. ACM Press, October 2005.

[5] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *11th International World Wide Web Conference*, pages 293–304. WWW, May 2002.

[6] D. Katsaros and Y. Manolopoulos. Caching in web memory hierarchies. In *19th Annual ACM Symposium on Applied Computing*, pages 1109–1113. ACM Press, March 2004.

[7] A. Nanopoulos, D. Katsaros, and Y. Manolopoulos. A data mining algorithm for generalized web prefetching. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1155–1169, September 2003.

[8] G. Pallis, K. Stamos, A. Vakali, D. Katsaros, A. Sidiropoulos, and Y. Manolopoulos. Replication based on objects load under a content distribution network. In *22nd International Conference on Data Engineering Workshops*. IEEE Computer Society, April 2006.

[9] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1):101–106, January 2006.

[10] L. Wang, V. Pai, and L. Peterson. The effectiveness of request redirection on cdn robustness. In *5th Symposium on Operating System Design and Implementation*, pages 345–360. USENIX, December 2002.