Content Classification for Caching under CDNs

George Pallis[1,2], Charilaos Thomos[2], Konstantinos Stamos[2], Athena Vakali[2], George Andreadis[3]
[1]*Department of Computer Science, University of Cyprus*
[2]*Department of Informatics, Aristotle University of Thessaloniki*
[3]*School of Engineering, Aristotle University of Thessaloniki*
gpallis@cs.ucy.ac.cy, {chthomos, kstamos, avakali}@csd.auth.gr, andreadi@eng.auth.gr

Abstract

Content Delivery Networks (CDNs) provide an efficient support for serving "resource-hungry" applications while minimizing the network impact of content delivery as well as shifting the traffic away from overloaded origin servers. However, their performance gain is limited since the storage space in CDN's servers is not used optimally. In order to manage their storage capacity in an efficient way, we integrate caching techniques in CDNs. The challenge is to decide which objects would be devoted to caching so as the CDN's server may be used both as a replicator and as a proxy server. In this paper we propose a nonlinear non-parametric model which classifies the CDN's server cache into two parts. Through a detailed simulation environment, we show that the proposed technique can yield significant reduction in user-perceived latency as compared with other heuristic schemes.

1. Introduction

CDNs (Content Delivery Networks) have been proposed to accelerate the delivery of the Web content. On a daily basis, users use the Internet for "resource-hungry" applications which involve content such as video, audio on-demand and distributed data. For instance, the Internet video site YouTube hits more than 100 million videos per day. Estimations of Youtube's bandwidth go from 25TB/day to 200TB/day. At the same time, more and more Web content servers are delivering greater volumes of content but with high sensitivity to delays. For instance, a delay on financial data-feed Web site (e.g., USD to EUR currency stock markets) may cause serious problems to the end-users.

A CDN is an overlay network across Internet, which consists of a set of surrogate servers (distributed around the world), routers and network elements. Surrogate servers are the key elements in a CDN, acting as replicators. They store copies (also called replicas) of identical content, such that clients' requests are satisfied by the most appropriate site. Once a client requests for content on an origin server (managed by a CDN), client's request is directed to the appropriate CDN's surrogate server.

1.1. Paper's Motivation & Contribution

While the CDNs result in significant benefits, in terms of availability, stability and Web transfer speed, their performance increase is limited due to the facts that: the storage space in surrogate servers is not used optimally [1], the content which is replicated in the surrogate servers remains static for a considerable period of time.

In order to alleviate the above problems we deploy caching in conjunction with replication. Specifically, we consider a CDN whose surrogate servers act simultaneously both as proxy servers and content replicators. According to the literature, the resulting problem of finding which objects' replicas should be created where, given that any free space will be used for caching, is NP-complete [1]. In this context, two heuristic approaches have been proposed [1, 6] towards managing the capacity of surrogate servers. Specifically, the key issue of Hybrid [1] and SRC [6] is to determine the percentage of storage space of CDN's surrogate servers that would be devoted in caching. However, these approaches are offline and, consequently, are unable to handle efficiently the sudden changes in the interest of the end users. This is a crucial issue if we consider that the most popular objects remain popular for a short time period [3]. Furthermore, the Hybrid algorithm suffers from "administratively" tunable parameters which determine the percentage of storage space for caching [1].

In this context, the ideal content management policy of surrogate servers should: (a) Handle the sudden changes of Web users' request streams; (b) Lack of any administratively tunable parameters; (c) Achieve a delicate balance between replication and caching towards improving the CDN's performance. The major contributions of this work are:

- Proposing a CDN framework where the surrogate servers act both as proxy caches and static content replicators under a cooperative environment.
- Presenting a method, the so-called R-P (Reward-Penalty), which partitions the surrogate servers'

978-1-4244-1841-1/08/$25.00 ©2008 IEEE

cache into two parts: The first part is devoted to caching and the second one is devoted to replication. The replicas are classified to one of the above categories by using a nonlinear model. The nonlinear model is preferred since it classifies better the replicas than any linear model [4].

- Providing an experimentation showing that our method performs better than the examined algorithms. We evaluate the performance of the proposed method using a dataset which captures the workloads of a streaming media Web site.

The rest of this paper is organized as follows. In Section 2 we present the proposed R-P method. Section 3 presents the simulation testbed, and section 4 evaluates the experiment results. Finally, the conclusion of our work is given in section 5.

2. The R-P Method

We consider a CDN framework, where the surrogate servers act both as dynamic content replicators (proxy caches) and static ones under a cooperative environment. During a training time period the Web objects of each surrogate server are classified into two categories by using a nonlinear model: volatile and static. The volatile objects are replicated to the dynamic part of cache, whereas, the static ones are replicated to the static part of cache.

In the proposed framework, the available storage capacity of each surrogate server *i*, which is denoted by $K_i^{(s)}$, is partitioned into two parts: The first one ($S_i^{(s)}$) is used for replicating content statically and the second one ($D_i^{(s)}$) is used for replicating content dynamically (running a cache replacement policy):

$$D_i^{(s)} + S_i^{(s)} = K_i^{(s)} \quad (1)$$

From the above equation it holds that if $D_i^{(s)} = 0$ the cooperative push-based scheme is applied where, the surrogate servers cooperate upon cache misses and the content of the caches remains unchanged. On the other hand, if $S_i^{(s)} = 0$ the surrogate servers turn into cooperative proxy caches (dynamic caching only).

Our proposed method assigns a quality value "q" for each object which has been replicated in surrogate servers. In particular, the quality value of replicas is expressed by the users' interest (increasing its value by using equation 2) or the lack of users' interest (decreasing its value by using equation 3) for the underlying replica. The intuition behind is that each time *t* an object is requested, it is rewarded by the function *R(t)*. At the same time, all the other replicas receive a penalty by the function *P(t)*, since they have not been requested. Specifically, the following functions have been defined:

$$R(t) = 1 - \frac{1}{T}(t - t_i^{'}) \quad (2)$$

$$P(t) = -\frac{1}{T}(t - t_i^{'}) \quad (3)$$

The $t_i^{'}$ expresses the time of the previous user's request for the specific object *i* and *T* denotes the training time period. Regarding the relation between t and $t_i^{'}$, it holds $t - t_i^{'} \le T$. Therefore, it is obviously occurred that *R(t)*$\in [0,1]$ and *P(t)*$\in [-1,0]$. The quality value of each object *i* for a specific time *T'* is calculated by the sum of the total reward and penalty values as follows:

$$q_i^{T'} = \sum_{t=0}^{T'} (P(t) + R(t)) \quad (4)$$

Taking into account the quality value of each object (equation 4), we classify them into two categories by using a classification model. Considering that linear models do not classify efficiently the Web objects due to their inefficiency to find correlations among Web objects' features [4], we make use of the logistic sigmoid function. Specifically, the logistic sigmoid function has been widely used by neural networks [2] to introduce nonlinearity in the model. Thus, it has been proven useful in case of two-class classification [4]. Here, the following model splits the objects' population into two groups (dynamic and static) with respect to the equation 4:

$$N(q_i^T) = \frac{1}{1 - e^{-q_i^T}} \quad (5)$$

Eventually, the above nonlinear model (equation 5) will classify each object i into one of the two desired categories: volatile ($N(q_i^T) \cong 0$) or static ($N(q_i^T) \cong 1$). A similar model has also been used in [4] in order to predict the cache utility value of each cached object by using features from Web users' traces.

In this paragraph, a description of the R-P method is given. Initially, we consider that a warm-up phase for the surrogate servers' caches has been preceded where the replicas of each surrogate server have been classified into volatile and static with respect to the equation 5. Furthermore, it is critical to consider a time period *T* for resetting the quality values of replicas. The functionality of the R-P method is depicted by the flowchart in Figure 1. When a surrogate server receives a request for an object, the quality values of replicas are updated with respect to equation 4. Then, 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 requested object is in the cache, it is served and the cache's content is updated with respect to the quality values of objects. In case of a cache miss, the requested object is pulled from another server (selected based on proximity measures) and stored into the dynamic cache. Then, the end-user receives the cached object. The objects of the dynamic part of cache will be available in surrogate server's cache for future requests as long as they are allowed by the cache replacement policy. According to this policy, if there is no space to store this object, it is removed from the dynamic part of cache the object which has the lowest quality value. The quality value of each object is calculated by the equation 4. The above procedure is repeated until the time threshold (*T*) is exceeded. In such a case, all the objects, which are stored in surrogate server, are re-classified according to the equation 5 and their quality values are reset. Thus, for small values of *T*, R-P captures more efficiently the sudden changes of Web users' request streams.

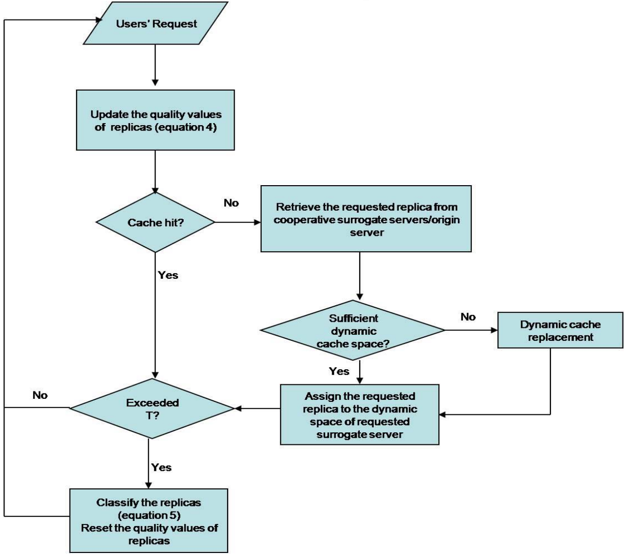
Users' Request
Update the quality values of replicas (equation 4)
Cache hit?
No
Retrieve the requested replica from cooperative surrogate servers/origin server
Yes
Sufficient dynamic cache space?
No
Dynamic cache replacement
Yes
No
Exceeded T?
Assign the requested replica to the dynamic space of requested surrogate server
Yes
Classify the replicas (equation 5)
Reset the quality values of replicas

Figure 1. An Outline of R-P Approach

3. Simulation Testbed

CDNs host real time applications and they are not used for research purposes. Therefore, for the experimentation needs it is crucial to implement a simulation testbed.

In this work, we use the CDNsim – a tool that simulates a main CDN infrastructure. A demo of our tool can be found at http://oswinds.csd.auth.gr/~cdnsim/. It is based on the OMNeT++ library which provides a public-source, component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. All CDN networking issues, like surrogate server selection, propagation, queueing, bottlenecks and processing delays are computed dynamically via CDNsim, which provides a detailed implementation of the TCP/IP protocol, implementing packet switching, packet re-transmission upon misses, objects' freshness etc. Here, the CDNsim simulates a CDN with 20 homogeneous surrogate servers which have been located all over the world. The size of each surrogate server has been defined as the percentage of the total bytes of the Web server content. Finally, the outsourced content has been replicated to surrogate servers using the il2p algorithm [5]. According to the il2p, the outsourced objects are placed to the surrogate servers with respect to the total network's latency and the objects' load. This policy is preferred since it achieved the highest performance.

Considering that the role of CDNs is primarily focused on improving the QoS of the "resource-hungry" applications in Web sites, such as Digital Television, Interactive TV, Video On Demand (VOD), etc., streaming media services are of interest in CDNs. In this context, we used the MediSyn workload generator described in [7], which generates realistic streaming media server workloads. Specifically, this generator reflects the dynamics and evolution of content at media sites and the change of access rate to this content over time. Furthermore, the MediSyn changes the popularity of objects over a daily time scale within a certain period of time.

In this work, we have generated a data set, which represents the HP Corporate Media Solutions Server (HPC) Web site. Table 1 presents the characteristics of the examined data sets. Finally, concerning the network topology, we used an AS-level Internet topology with a total of 3037 nodes. This topology captures a realistic Internet topology by using BGP routing data collected from a set of 7 geographically-dispersed BGP peers.

4. Experimentation

4.1. Examined Policies

The proposed approach (R-P) integrates both caching and replication in CDNs. Thus, we evaluate the R-P's performance with respect to the above stand-alone approaches. Furthermore, we compare R-P with SRC. Previous results [6] have shown that SRC is the leading algorithm in the literature for integrating caching and replication over CDNs. Specifically, the following approaches are examined:

- **SRC**: A placement similarity measure is used in order to evaluate the level of integration of Web caching with content replication.

- **Caching**: All the storage capacity of the surrogate servers is allocated to caching. The selected cache replacement policy is LRU since it is used by the most proxy cache servers (e.g., Squid).
- **Replication**: All the objects are replicated statically in each surrogate server using all the available storage capacity.

Table 1. Parameters for Generated Data Set

Characteristic	**HPC**
Log duration	91 days
Number of requests	1000000
Number of Web objects	1434
Size of Web site	3.8Gbytes

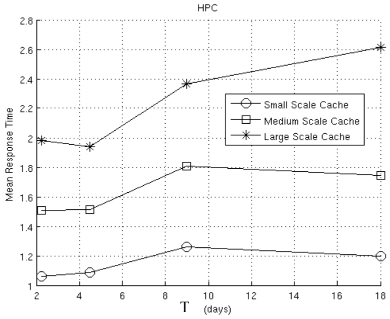

Figure 2. Mean Response Time vs. T

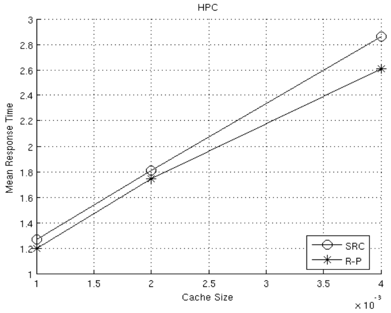

Figure 3. Mean Response Time vs. Cache size

4.2. Evaluation Measures

The measures used in the experiments are considered to be the most indicative ones for performance evaluation. Specifically, the following measures are used:

- **Mean Response Time (MRT)**: the expected time for a request to be satisfied. It is the summation of all requests' times divided by their quantity. This measure expresses the users' waiting time in order to serve their requests and it should be as small as possible.
- **Byte Hit Ratio (BHR)**: it is defined as the fraction of the total number of bytes that were requested and existed in the cache of the closest to the clients surrogate server to the number of bytes that were requested. A high byte hit ratio improves the network performance (i.e., bandwidth savings, low congestion).
- **Hit Ratio (HR)**: 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.

4.3. Evaluation

Firstly, we investigate the proposed approach R-P with respect to the mean response time, with varying *T*. The results are reported in Figure 2. The x-axis represents the *T* which is expressed in days (24 hours reflect to 1 day), whereas, the y-axis represents time units according to CDNsim's internal clock and not some physical time quantity, like seconds, minutes. So the results should be interpreted by comparing the relative performance of the algorithms. This means that if one technique gets a response time 0.5 and some other gets 1.0, then in the real world the second one would be twice as slow as the first technique.

In general, we observe that as the time period (*T*) for taking place a reclassification of replicas increases, the performance of R-P is deteriorated. In other words, this means that the performance of the R-P captures better the sudden changes of Web users' request streams when the replicas are reclassified in small time periods. For instance, the lowest mean response time has been observed when the classification takes place every two days. This observation is common independent of the surrogate servers' cache sizes. Regarding the cache size of surrogate servers the R-P presents lower mean response times for small-scale cache sizes. The cache sizes of surrogate servers are expressed in terms of the percentage of the total size of the examined Web site.

Secondly, we test the performance of the examined integrated approaches (SRC and R-P) with respect to the cache size. Due to lack of space, we have considered that the time period *T* takes the value of two days. The results are depicted in Figure 3. The x-axis represents the cache size of each surrogate server. We observe that the R-P outperforms the SRC approach achieving a delicate balance between caching and replication. Furthermore, we observe that as the cache size of surrogate servers grows, the mean response time of the examined policies increases in a linear way.

Figure 4 presents the BHR of the examined policies. The x-axis represents the cache size, whereas, the y-axis represents the BHR. The R-P is the leading algorithm, achieving the highest BHR. As we expected, the BHR of the examined algorithms increases with respect to the surrogate server's cache size. In particular, the larger the cache size is, the higher the BHR is. As far as the caching approach is concerned, it presents higher BHR than the replication one. The close performance of SRC and caching is explained by the fact that a large percentage of storage space in the SRC is allocated to Web caching (about 85%). On the other hand, the pure replication yields poor performance and it seems that it is not affected by the cache size. This behavior was expected since the pure replication does not manage in an efficient way the storage space. Quite similar results are also obtained when we evaluate the HR of the examined algorithms. The results are reported in Figure 5. As previous, the R-P is the leading algorithm, indicating the highest HR comparing with the examined approaches.

To summarize the experiments, we can conclude that the integration of replication with caching leads to improved performance in terms of perceived network latency, byte hit ratio and hit ratio. The results reinforce the initial intuition that replicating replicas statically for content availability along with caching policies improves the Web performance. The proposed method outperforms the existing integrating approach, achieving a delicate balance between replication and caching. Furthermore, the performance of R-P seems to handle efficiently the sudden changes of Web users' request streams.

5. Conclusion

In this paper, we dealt with the potential performance benefits that can be reaped by combining both caching and replication in CDNs. The challenge for such an approach is to determine which objects would be devoted in caching so as the CDN's server may be used both as a replicator and as a proxy server. In this paper, we propose a nonlinear non-parametric model which classifies the CDN's server cache into two parts. Inspired by the neural networks, we make use of the logistic sigmoid function in order to split the outsourced objects into two groups (volatile and static). The experimentations' results show that the proposed approach achieves a delicate balance between replication and caching towards improving the CDN's performance. Furthermore, the R-P approach captures in an efficient way the sudden changes of Web users' request streams, which usually occur in streaming media Web sites.

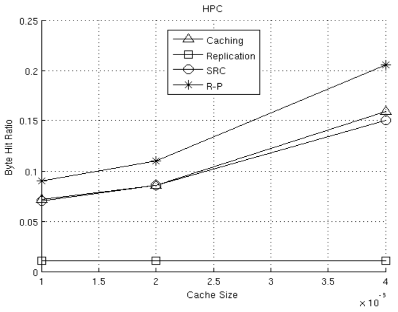

Figure 4. BHR vs. Cache Size

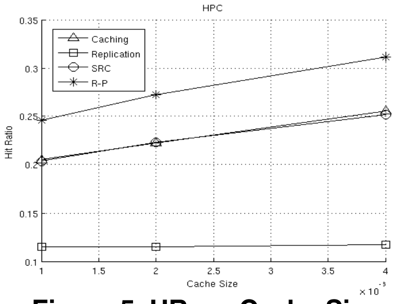

Figure 5. HR vs. Cache Size

6. Acknowledgments

This work has been supported partially by the Marie Curie EU project (Contract No. 042467) and by the Hrakleitos project under the EPEAEK Framework.

7. References

[1] S. Bakiras, T. Loukopoulos: Combining replica placement and caching techniques in content distribution networks. *Computer Communications,* 28(9): 1062-1073, 2005.

[2] C.M. Bishop: Neural Networks for Pattern Recognition. Oxford University Press, 1995.

[3] Y. Chen, L. Qiu,W. Chen, L. Nguyen, R. H. Katz: Efficient and adaptive Web replication using content clustering. *IEEE Journal on Selected Areas in Communications*, 21(6): 979-994, 2003.

[4] T. Koskela, J. Heikkonen, K. Kaski: Web cache optimization with nonlinear model using object features. *Computer Networks*, 43(6): 805-817, 2003.

[5] G. Pallis, K. Stamos, A. Vakali, A. Sidiropoulos, D. Katsaros, Y. Manolopoulos: Replication based on objects load under a content distribution network. Proceedings of the 2nd WIRI (In conjunction with ICDE'06), Atlanta, Georgia, USA, Apr. 2006.

[6] K. Stamos, G.Pallis, C.Thomos, A.Vakali: A similarity based approach for integrated Web caching and content replication in CDNs. Proceedings of the 10th IDEAS, New Delhi, India, Sep. 2006.

[7] W. Tang, Y. Fu, L. Cherkasova, A. Vahdat: Modeling and generating realistic streaming media server workloads. *Computer Networks*, 51(1): 336-356, 2007.