A Hybrid Approach for Content Replication Improvement in Content Delivery Networks

Mostafa Moradi
Department of Computer, Baghmalek Branch, Islamic Azad University, Baghmalek, Iran
mos_moradi1359@yahoo.com

Abstract

In this paper by combining caching and replication techniques proposed a hybrid heuristic method based on the greedy algorithm to use the benefit of each other techniques. The algorithm in each interaction compares all the contents and one that made the best benefit value is selected for replication. The hybrid approach tested in a simulation environment and the results show that hybrid algorithm again stand-alone replication, reduced the average response time by 42% and compared to the pure caching, saving up 23% user requests time.

Keywords: Content Delivery Network (CDN), Hybrid Algorithm, Replica Placement

1. Introduction

Based on CISCO predictions, average of data that are accessed daily by users, is growing exponentially [1], also, traffic access to multimedia data is Exabyte’s a month [2]. Which has created new challenges in the field of electronic commerce and most popular websites are facing with heavy traffic and the emergence of bottlenecks and even failures in services related to the user. However, the content of sites load times has been improved in recent years, but in general the access time is still high. It might be thought that, make improvements in broadband Internet infrastructures, and increase server capacity have led to reduce or eliminate the access delay, but these methods have often acted in the opposite direction, and even more users suffer delays. Content delivery networks\(^1\) as an approach to improve this problem were presented in a way that, they deliver the user requested content through placement on closer surrogate servers, as reliable and on time [3]. CDN is a node-based network, which is composed of several agent servers. This agent server contains some of the content of the main server and in different geographical locations are scattered across the web. One of the main techniques for content delivery that is the case, first user request is routed to the closest agent server, in case of any content on the server, a crash occurred, and the content is delivered to users, otherwise, the request is routed to the other closest agent server, if there is no content in none of the close surrogate servers to the user, the use request is responded by main server.

The initial generations of content delivery networks were often based on the replication of the content on surrogate servers, but because of that the content that had many changes over time or referral pattern to them was different in different servers, in

\(^1\)- Content delivery networks (CDN)
the next generations, the caching techniques in the CDN are used to overcome these problems.

In most studies, these two issues will be discussed separately, in other words, the caching in proxy systems, and replication as one of the mechanisms for content delivery networks have been studied, but it can be considered a content delivery system so that agent server has both the role of cache and also the role of the repeater, as a result, it can be distribute the content on the servers with the integration of static and dynamic techniques, so that the need for caching is minimized, and if you need to cache, an efficient caching algorithm is used. In this study, by combining and replication the techniques of cache, a heuristic approach is presented, then, by the analysis of this technique, the efficiency improvement in content delivery system is investigated. This paper is organized as follows: in Section 2, the motivation for the research and a summary of previous work is evaluated. In Section 3, the modeling and assumptions related to CDN architecture is introduced, and in Section 4, based on the modeling and assumptions, the proposed hybrid method based on Greedy algorithm is presented, and then, in the simulation and training data related to the research is discussed and performance of the proposed method compared to other methods is examined.

2. Motivation and Previous Works

The most important issue in data management is placement of servers, and contents and routing decisions of requests. Traditional CDNs optimize their performance by relying on two key mechanisms: replica placement (RP) and request routing (RR) [4]. While RP mechanism decides for the locations of replicas, the RR subsystem receives the outcome of RP mechanism and redirects users to the optimum available server. In this paper, focus on the placement problem of NP-complete, and heuristic methods must be provided for it. In this regard, several heuristic methods for the placement problem is presented, and concluded that the Greedy Global algorithm is the best choice for placement, but due to the high complexity on real and large data, this method is difficult to implement. Other heuristic methods such as placement algorithms based on genetic have been suggested [6]. Usefulness of combining algorithms, placement, and caching in content delivery networks [7], compared to the same source, the importance of access on demand, and the productivity generated by the algorithm are discussed. In [5] proposed a hybrid CDN model which deploys the CDN systems in backbone networks and build the P2P regionalization networks in the access network

2.1 Placement Algorithms Based on Replication

Surrogate placement, means the proper placement of objects on the servers, so it has the best performance when the user requests for the object are arisen. Because, proper selection of these objects in the servers is associated with the delivery of content symbols, in recent years, much research has been done in this area, as in most of them, to solve this problem, a heuristic method is used. The main objective of the optimization of the placement is reducing latency access to content, and also reducing the content transmission bandwidth consumed. In [8], a mathematical programming formulation to optimize time order by the optimal placement is presented, and then a heuristic algorithm was developed for it. Theoretical studies indicate that the algorithm performs better than the most common methods of content placement, but is more expensive.
Systemic methods discussed in [9] help the selection process of the content to repeat in order to the most appropriate replication placement is done. These discovery methods caused a change in appropriate placement algorithms, and reduced the response time of the users' requests. To this end, four methods: 1) Random, 2) being known, 3) Greedy-Single and 4) Greedy - Global for Content delivery networks have been used. Greedy – Global is an exploratory and reversal method. In this way, recursively, for all servers and objects, an object is specified, if this is specified in the agent server, the optimal performance is achieved. However, this method for the placement of large numbers of servers or objects is too complex. For example, in [9], it is mentioned that during the test on Greedy methods, given that too much memory is needed, experiments have failed to fully do. In [10], ilp2 algorithm which acts based on the load on the servers is presented. This algorithm is a Recursive algorithm, and uses two phases for the placement of objects in the cache of the agent servers. During the first phase, for each object, the best agent server is selected, so that it has the lowest latency of the network. Thus, the output of the phase will be object pairs and agent server. In the second phase, among the previous phase pairs, a pair is selected that has the highest amount of load on the server.

2.2 Caching Algorithms in Content Delivery Networks

The use of the cache algorithms in replication of the contents in [11] under the SRC algorithm is investigated. In this paper, a heuristic technique for combining these algorithms with replicated techniques is proposed. In this way, the storage space of agent servers as dynamic is partitioned. In general, this method uses similarity between object placement in once replication, and the arrangement of objects in the caching to understand the ratio of partitions. During the test results of this technique are compared with other methods and the results show the high efficiency of this method compared to other methods in terms of the average traffic load. In [12], it is mentioned that management in CDNs requires efficient algorithms for routing requests, content placement and proper policies for caching, which reduces latency in these systems. In this study, a system is modeled that the number of front nodes for resources and late nodes for caching servers is used. These nodes operate such as input and output in a switch. In this model, the queue of requests for each resource, in front of the desired source node is created, which ultimately is routed by the control pane to a cache node.

The result of improvement of this system by simulation is determined. In [13] examined the previous successful mechanisms, and they were compared. In this paper, a caching algorithm is presented, which is based on co-management. The purpose of this algorithm is to obtain the highest traffic volume in cache servers, and the lowest cost for bandwidth, which have been achieved the desired results by linear programming and simulation.

3. Research Methodology

A public infrastructure CDN, with N servers distributed in different geographical locations, is considered. Each server is shown as S (i), so that1 ≤ i ≤ N. The size of agent servers S_{(i)} in bytes is equal to s_{(i)}. For simplicity of calculations and simulation, agent servers are assumed to be homogeneous. These N servers in content delivery networks infrastructure are connected together, the cost of communications between these two servers S_{(i)} and S_{(j)} or C (i, j) is determined. This cost shows the shortest path.
between two servers based on the number of steps, so that \( C(i, j) = C(j, i) \) is satisfied. It is assumed that there are \( M \) different websites, which have been named as \((O_1, O_2, \ldots, O_M)\), and they are connected to the content delivery network service provider. The size of website \( O_i \) with \( o_j \) in bytes will be displayed. Each site \( j \) contains different object \( L_j \) which have been named as \((O_{j1}, O_{j2}, \ldots, O_{jL_j})\) and the popularity of objects follows the distribution of zipf–like with parameter \( \theta_j \) [14], it is assumed that there is a copy of each object in the network, as well \( S_{Bj} \) is site that is kept a primary copy of \( O_j \) object, and can't repeat again the objects that previously have been repeated. Each \( S_{Bj} \) keep all iteration of \( O_j \), this work is performed by gathering a list of the servers in which the \( j \)-th web is repeated. Thus, each server \( S_{(i)} \) for each site, stores a two-field record:

- **First field**: Its primary site
- **Second field**: nearest server \( SS_{j}^{(i)} \) to the server \( i \) which has a copy of the objects \( O_j \).

In other words, \( SS_{j}^{(i)} \) is a server for requests from the \( S_{(i)} \) to \( O_j \). If, the request of the same server is responded, it has the minimum cost of responding, and it would be possible that \( SS_{j}^{(i)} = S_{(i)} \) occurs, which means that \( S_{(i)} \) is a repeater \( S_{(i)} \), another situation that may occur is \( SS_{j}^{(i)} = S_{B(i)} \), which indicates that the original site is closest site to a server that keeps repeating \( O_j \). Finally, it should be noted that the storage space per server can be used as cache and repeat.

### 3.1 Formulation of the Problem

Suppose that \( R_{j}^{(i)} \) is the number of requests for \( O_j \) from users over a period determined in the server \( S_{(i)} \), the goal is to minimize the total cost during transferring the object. Also, assume that \( R_{j}^{(i)} \) is the overall cost of requests to the server \( S_{(i)} \) to \( O_j \) site, which is directed to the closest server \( SS_{j}^{(i)} \). The cost is expressed by the following equation:

\[
R_{j}^{(i)} = C(i, SS_{j}^{(i)}) [r_{j}^{(i)} - L_{j}^{(i)}]
\]  

(1)

\( L_{j}^{(i)} \) is the number of requests that locally by \( S_{(i)} \) are serviced.

Note that if \( SS_{j}^{(i)} = S_{(i)} \), for example, \( S_{(i)} \) is a repeater for \( O_j \), in this case, \( r_{j}^{(i)} - L_{j}^{(i)} \), \( R_{j}^{(i)} = 0 \), in other words, \( L_{j}^{(i)} \) presents the total number of requests answered with local cache, so the total cost by \( C_t \) and is shown as follows:

\[
C_t = \sum_{i=1}^{N} \sum_{j=1}^{N} R_{j}^{(i)}
\]  

(2)

Iterative matrix \( n \times m \) is defined as \( X \). An element \( X_{ij} \) of this matrix is equal to 1, if the \( O_j \) in \( S_{(i)} \) is repeated, and otherwise it is zero, then the placement problem is formulated as follows:

1. Transferring from the values 0 and 1 in the matrix \( X \) is found that minimizes \( C_t \).
2. Storage capacity would be limited as follows.

\[ \sum_{j=1}^{M} x_{ij} \leq S_{i} \quad \forall \; i=1, 2, \ldots, N \]  

(3)

### 3.2 Cache Hit Ratio

To calculate being beneficial the use of cache as part of an analysis CDN architecture, it is required that the hit rate under different system parameters be predicted. Initially, a simple caching policy, such as LRU as an accessible way for cache hit rate in a simple CDN for a particular website is assumed. However, a public instance of the server \( S_{(i)} \), and the object \( O_{j} \) is considered, cache LRU, as a buffer that can store a limited number of object \( B \) in itself, is modeled. Since the size of the object for a file Mai website is variable, \( B \) as \( \frac{C_{(i)}}{o_{i}} \) is estimated that \( C_{(i)} \) is the amount of memory considered for caching and \( o_{i} \) is average requests' size. When an object is stored in the cache \( O_{jk} \) for the first time, is transmitted to the end of the buffer. If this object is not requested for long-term, gradually is moved to the front of the buffer, and finally, after \( K \geq B \) request for the object, is evicted from the cache, if before leaving the cache, \( O_{jk} \) again is requested, it returns to the front of the buffer. Any request for an object, without regard to other requests will be considered. However, the probability of survival of a particular object \( O_{jk} \) in the server cache \( S_{(i)} \) is computed, in survival mode, the desired object in interval of mean \( \bar{h} \) is in the cache, and also \( \bar{m} \) is an interval when the desired object is not in the cache. These two periods are as follows:

\[ \bar{h} = \sum_{i=1}^{K} (1 + \bar{h}) p^{i-1} (1-p) + \sum_{i=K+1}^{\infty} K p^{i-1} (1-p) = \frac{p^{K-1}}{1-p} \]  

(4)

\[ \bar{m} = \sum_{i=1}^{\infty} i p^{i-1} (1-p) = \frac{1}{1-p} \]  

(5)

Where, \( p \) is the probability \( O_{jk} \), which is not requested. Therefore, the probability of survival \( H_{jk}^{(i)} \) to be \( O_{jk} \) in LRU server cache is:

\[ H_{jk}^{(i)} = \frac{\bar{h}}{\bar{h} + \bar{m}} = 1 - p^{K} \]  

(6)

In fact, the probability that an object at least once in the \( K \) successive interval has requested, therefore, the hit rate for a site like \( O_{j} \) is:

\[ H_{j}^{(i)} = \sum_{k=1}^{L} \left[ 1 - \left( 1 - p_{j}^{(i)} \right)^{a_{j}/k_{j}^{(i)}} \right] \frac{a_{j}}{k_{j}^{(i)}} \]  

(7)
Where, \( p_j^{(i)} = \frac{r_j^{(i)}}{\sum_{k=1}^{M} r_k^{(i)}} \) is popularity of object \( O_j \) in set \( S^{(i)} \) and \( \alpha_i \) is balancing factor for distribution of Zipf-like.

The only new variable in the above formula is \( K \) which specifies the number of intervals, which may an object is placed in them before leaving the cache, and has never requested. In the simulation experiments, the granularity of \( K \) was set to 450 time slots.

It is assumed that each website \( O_j \) is an estimation of \( \lambda_j \) that presents a request for non-cached content. The values of \( \lambda_j \) can be calculated by examining the file Mai Log in CDN servers, then the reality hit rate \( h_j^{(i)} \) can be calculated by multiplying by the value \( (1 - \lambda_j) \).

3.3 Proposed Approach

Widespread changes in exploration algorithms to provide a more appropriate method, according to numerous algorithms which have already been provided are actually unnecessary. Hence, in this study proposed a hybrid algorithm to be added the benefit of caching to it in addition to the advantages of iterative methods. The core of the proposed method is based on Greedy algorithm, which in [15, 14] the benefits of it are mentioned. Greedy is a heuristic and recursive method and provide sufficient solutions with lower computation cost. This algorithm chooses \( M \) servers among \( N \) potential sites. In first iteration, the cost associated with each site is computed in the first iteration. It is assumed that access from all clients converges to the site under consideration. Hence, the lowest cost site is chosen. In the second iteration, the greedy algorithm searches for a second site (yielding the next lowest cost) in conjunction with the site already chosen. The iteration continues until \( M \) servers have been chosen. The greedy algorithm works well even with imperfect input data. In this way, recursively, for all servers and objects an object is identified, if this object in an agent server is repeated, the optimal performance is achieved. Thus, by implementation of this approach in content delivery networks, objects are placed in the agent server that lead to increase of performance. However, this method cannot be applied as quite practical on content delivery networks, because this model for placement the large number of servers and objects has a very high complexity.

The proposed hybrid method is more efficient in content delivery networks, which just keep the original Mai repeating. In this case, at each iteration, all the possible contents are assessed and the best desired Mai content is selected and is repeated. Assessments include the calculation of the usefulness of the new iteration, and compare it with the cost which is added due to limited cache space. For example, evaluation of increase of hit rate with the addition of a new iteration. Whatever the hit rate is higher, the higher the object is likely to be repeated at the end of the iteration, the best content is created for replication and selection. In Figure 1, the start of algorithm shown is related to the Initialization and basic operations of the algorithm. Initially, it is assumed that the total storage space is allocated to the cache. At this stage, hit rate and total cost are calculated.
The main part of the algorithm consists of a repeat loop in Figure 2 is shown. In the beginning, the local utility for server $S_i$ is calculated. Then, the total benefit amount is calculated. Because the new iteration decreases the size of the buffer LRU, effectively decreases hit Rate of all the objects not repeated in the $S_i$.

**Figure 1. Start of algorithm - the Initialization and basic operations**

The main part of the algorithm, includes a number of organizations operating for the new iterative because after selecting the object and the server with the maximum benefit, the objects in the cache should be replaced with the new value and updates the status of nearest servers (Figure 3).

**Figure 2. Main part of the algorithm**

The final state of the algorithm, includes a number of organizations operating for the new iterative because after selecting the object and the server with the maximum benefit, the objects in the cache should be replaced with the new value and updates the status of nearest servers (Figure 3).

**Figure 3. Final state of algorithm**
3.4 Simulation of Algorithm

In this section, a framework to simulate the structure and content delivery framework is presented. By providing this environment, these networks can be tested and evaluated, and details of the results obtained from user data, the servers and networks can be analyzed. In order to simulate the proposed approach, the CDNSim program which is a simulation tool for content delivery networks has been used (Figure 4). CDNSim is able to provide libraries, tools, and a web interface for CDN in web, and calculate all main components of these networks, such as Select a surrogate server, content delivery, users, queuing, bottlenecks and delays in processing dynamically, and it is able to provide full details about the protocol TCP / IP, how the routing of packets, retransmission of packets and other. To implement other components of the simulation environment structure, C++ according to the compiler of g++ and python in Ubuntu 12.04 Linux operating system is used.

![Figure 4. CDNSim Environment](image)

3.4.1 Simulation Settings

Creation of Artificial Website

Each hypothetical website has at least 2200 objects that the size of the sum of these objects is 1.5 GB. Also, for each object, a size must be allocated. To do so, the distribution of log–t, which is explained in [16], is used.

Generating users’ surveying

After creating a website, a workload should be created on it. The workload is a sequence of users’ requests to the website. After creating these transactions and using the stages described in [17], users' request file for a specified time interval is achieved.
Creating a network topology and websites

For this purpose, the manufacturer of the network topology GT-TM [15,18] is used. The manufacturer, to create two types of random planar graph Mai N - level and the transit - stub is used. This manufacturer to create a random planar graph Mai in two types of N - level and transit-stub is used. By using this tool, a graph of nodes with a structure of transit - stub randomly is created. This topology determines the orientation of the routers in the network. The number of Mai nodes of the topology in this simulation is equal to 1464, according to the figure (5) comments. the number of agent servers is assumed equal to N = 60, which do tasks related to the hosting m=200 websites.

```
## Comments :
## <#method keyword> <#number of graphs> [<#initial seed>]
## <#stubs/xit> <#t-s edges> <#s-s edges>
## <#n> <#scale> <#edgemethod> <#alpha> [<#beta>] [<#gamma>]
## number of nodes = 3*8* (1 + 3*20) = 1464
ts 10 47
   3 0 0
   3 20 3 1.0
   8 20 3 0.8
```

*Figure 5. Creating a network topology with GT-ITM*

4. Efficiency Evaluation

To measure the efficiency, the proposed method is compared with other heuristic methods, for this purpose, the comparison action takes place in the following Mai mode:

- **Caching**: Caching lead to store objects (which recently are requested) in desired server cache to respond to future requests immediately. The user request to the Agent server is routed. It operates in case of lack of storage space on an agent server to store the object according to the LRU policy.

- **Replication**: This is a simple method to repeat which uses the Greedy Global algorithm, and the objects are repeated with respect to the frequency of replication, load and delay generated in the servers.

- **Hybrid**: This method is a proposed Algorithm which is a combination of the cache and repeat.

4.1 The Results of Evaluation

4.1.1 First Experiment (λ = 0)

In this test, a case is considered that all requests for objects are allowed to cache in the CDN servers. (λ = 0) determines the results of this test, the upper limit of the efficiency in the once cache mode. Figure 6 shows the average delay for a request in a variety of content delivery Mai mechanisms. In different Mai modes, limit the cache space, a hybrid method compared with other Mai strategies has had better efficiency. In general, compared with a simple iteration algorithm, the average
response time is reduced by approximately 42%, as compared with pure caching technique, although there is not much difference in efficiency, but the decrease of approximately 15% of the response time for servers with large storage space, can be useful. Figures 7,8 show the other Mai mode of efficiency in different mechanisms, in particular, these graphs show the cumulative distribution function of time-passer (CDF), means the percentage of requests which in a particular time interval have been answered.

![Figure 6. Average response time (λ=0)](image)

![Figure 7. Compare the effectiveness in different content delivery mechanisms (λ=0, Capacity=15%)](image)

![Figure 8. Compare the effectiveness in different content delivery mechanisms (λ=0, Capacity=20%)](image)

In this experiment, the efficiency of the mechanisms in case that is closer to reality has been studied, as has been assumed that 10% of requests to objects cannot be cached. Details of this experiment in Figure 9 and Figure 10, 11are shown. Although the relative decline in the efficiency of both cache and hybrid algorithms is created, but
the hybrid method again is better than the other two methods in terms of efficiency, and the average response time compared with to the once replication by about 35% and compared, once caching technique by about 23 per cent is decreased. In total, due the ability of the proposed algorithm to select appropriate surrogate servers, the benefits of caching and replication have been exploited.

![Figure 9. Average response time (λ=10%)](image)

**Figure 9. Average response time (λ=10%)**

![Figure 10. Compare the effectiveness in different content delivery mechanisms (λ=10%, Capacity=15%)](image)

**Figure 10. Compare the effectiveness in different content delivery mechanisms (λ=10%, Capacity=15%)**

![Figure 11. Compare the effectiveness in different content delivery mechanisms (λ=10%, Capacity=20%)](image)

**Figure 11. Compare the effectiveness in different content delivery mechanisms (λ=10%, Capacity=20%)**
5. Conclusion

In this paper, an improvement of the efficiency in content delivery networks by providing a hybrid algorithm was evaluated. The purpose was to find a placement of the objects in agent servers, and to optimize replication and caching operations in these systems, so that, hit rate increases, the average response time decreases, and users can receive content from the closest agent server. Since, the content placement using once replication and caching methods, in addition to its benefits, has some problems, the proposed algorithm which was based on Greedy, in addition to being free from the problems of these two methods, taking advantage of their benefits well could enhance efficiency. In a simulation environment, the algorithm in different percentages related to the allocation of storage space for caching, as well as the amount of non-cached data (λ) were tested, and it was shown that this method significantly increases the efficiency.

References


