EngOpt 2008 - International Conference on Engineering Optimization
Rio de Janeiro, Brazil, 01 - 05 June 2008.

Optimization in Content Distribution Networks

Tiago Araújo Neves, Luiz Satoru Ochi, Lúcia M. A. Drummond, Eduardo Uchoa e Célio Albuquerque

Universidade Federal Fluminense - UFF Niterói, RJ, Brazil - tneves@ic.uff.br

Abstract

Content Distribution Networks (CDN) are overlay networks used to place content near end clients with the aim at reducing delay, servers load and network congestion, hence improving the service quality perceived by end clients.

In traditional CDN architectures, clients requests are initially received by a central server and then redirected to another server that is close to the client and that is able to handle the request. However, in some cases, clients requests will not be served by the closest server. It may be advantageous to use a server that is a further away but that is not as loaded as the closest one.

Contents in CDN are replicated, in many cases, according to contractual clauses, because of the costs involved in the maintenance of such replicated contents. Therefore, some contents are not replicated through the entire overlay network of the CDN provider.

In this work a exact and heuristic approaches are presented in order to solve an optimization problem related to CDN management, called the Replica Placement Problem. This problem consists in finding the best set of servers to keep the replicated contents without violating QoS constraints of the requests. The overall objective is to minimize the total traffic in the overlay network.

Keywords: Content Distribution Networks, Optimization, Heuristic, Mathematical Model.

1. Introduction

With the Internet growth, the demand for contents has increased everywhere. Some of these contents require some kind of Quality of Service (QoS) such as maximum delay and minimum bandwidth so that the requirements of end clients can be satisfied. Many studies have been made about how to better serve the on growing demand for contents with QoS constraints. A Content Distribution Network (CDN) (a.k.a. Content Delivery Network) is an overlay network used to place content near end clients in order to reduce delays, server load and network congestion, hence improving the quality of service perceived by end clients. Companies like Akamai [1] have become specialized in providing a CDN architecture to other companies that need to distribute contents over the world like Adobe, Audi AG and Fox Interactive.

There are several optimization problems in CDN architectures. Among them are the Server Placement Problem, the Content Replication Problem and the Replica Placement Problem. The first consists in finding the best places in the real network to position the servers that will compose the overlay network of the CDN [6]. The second consists in deciding which contents will be replicated [13]. The last consists in finding the best servers in the overlay network to position the replicated contents [13].

The optimization problem that will be treated in this paper is an extension of the Replica Placement Problem (RPP) and consists in finding the best servers to position the replicated contents such that server capacity and the QoS constraints of the requests are not violated and the total traffic in the network is minimized.

Requests with QoS are typically found in situations where some clients are paying for a faster and/or more reliable access to the content. The models presented in this work allow the possibility of differenciating groups of clients by their QoS requirements in a natural way.

This paper is organized as follows. Section 2 brings a more detailed description of the problem. In Section 3 the recent related work is presented. Section 4 explains the computational model used in the heuristic approach. Section 5 describes the experimental scenarios. In Section 6 and 7 a mathematical model and a heuristic for the problem are presented, respectively. In Section 8 the computational results are shown and Section 9 concludes the work.

2. The Problem

The Replica Placement Problem (RPP) consists in finding the best servers to place content replicas over the CDN. In order to solve it, it is necessary to know the position of the servers in the network topology, the number of replicas of each content and to have an estimate about the distribution of the requests. As explained before, deciding on the servers placement and deciding which contents should be replicated are also optimization problems. We assume that those problems were already solved. On the other hand, the problem tackled in this work extends the RPP and is called the Replica Placement and Requests Distribution Problem (RPRDP). Besides finding the best position for the replicas in the overlay network, in the RPRDP one also wants to redistribute the requests through the servers, with the aim at reducing the network load. It is usual that the request for a content should be handled by the closest server having a replica of that content. However, in some cases, it may be advantageous to use a server that is more distant but that is not as loaded as the closest one, in order to improve the quality perceived by the client.

To redistribute requests in RPRDP, some factors should be taken into account. A request may be redirected to a server only if that server has a replica of the content specified in the request. Also, the QoS constraints of the request must be verified. If these constraints are violated, redirecting this request for that server will produce an unfeasible solution. In some cases, to optimize the request distribution, it is necessary to change the replica positioning first.

The positioning of replicas is subjected to factors related to the servers, like storage capacity and distance between servers. In the beginning, all replicas are positioned in the same server, possibly the main server in the CDN architecture, and then, they are distributed over the network. As there may exist costs associated with the transmission of these replicas, the tradeoff between the reduction of load in the network and the cost of transmission of the replica must be analyzed before migrating a replica.

Information about the overlay network is also needed. As the positioning of servers is out of the scope of this work, it is considered that such information is an input for the problem tackled in this paper. Therefore, information like distance between servers and the delay in the network are simply input data, and are not optimized.

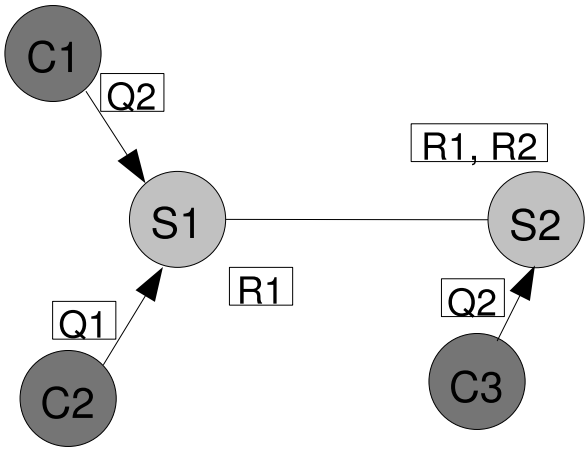

Figure 1: Replica Placement and Request Distribution Problem

Figure 1 shows an example of the RPRDP in a CDN. The figure shows two servers and three clients. Server 1 has a replica $R1$ of content 1, server 2 has one replica of content 1 and one of content 2 , and each client sends a request (Q) for a content in the CDN. Note that the solution shown in the figure is unfeasible because the request from client 1 is being handled by a server that does not have a replica of the requested content. A possible change to make the solution feasible is to change the request of client 1 from server 1 to server 2.

As described above, the problem can be seen as a Multi-Commoditty Capacitated Facility Location

Problem with Unsplitable Demand [10, 9]. This problem is a generalization of the Capacitated Facility Location Problem (CFLP) [9] that can be described as follows. Given a set C of clients, a set F of possible locations to open facilities, determine a subset of F named F_{opened} of size less or equal to a parameter k so that the cost of handling all the clients in C with the facilities in F_{opened} is minimized.

By establishing a relationship between servers and facilities and a relationship between clients and requests, the described problems become similar.

The unsplitable demand of the problem is an extra constraint that impose that a client (request) must be handled by single facility (server). The capacitated constraint limits the capacity of the facilities, and the Multi-commodity permits more than one commodity (content) to be handled at the same time.

3. Related Work

In [13], the Content Replication Problem and the Replica Placement Problem are introduced. Three replication algorithms are proposed, one that creates replicas based on communications costs, another that determines the number of replicas for a content based on its popularity, and the last one, that creates replicas based on a weighted round-robin. For the Replica Placement, two algorithms are proposed, a round-robin, and another one that places replicas with greater communication costs in servers with smaller load. The author also proposed a simulated annealing algorithm for the problem.

In [6], the authors deal with many problems related to CDN, including the Server Placement Problem, the Content Replication Problem and the Replica Placement Problem. A non linear mathematical model and a linearization for the presented model are proposed. They also introduced a heuristic algorithm and an exact algorithm, that is based on the problem decomposition.

Huang *et al.* [8] also proposed a distributed and a centralized approaches for the problem. The distributed approach is based on the graph domination strategy. The objective is to build a P2P network with the graph vertices, but, on the contrary of what is generally observed in P2P networks, the vertices have considerable knowledge about the graph topology. The algorithms were compared with other graph domination algorithms, showing that the proposed distributed approach can find solutions that present a very low replication cost and also respect the delay requirements.

In [4], the authors tackled the Replica Placement Problem. They also proposed the use of a multicast tree to deliver the contents to the end clients. Although the delivery costs are considered, the quality perceived by the end clients is not taken into account.

In [5], the Dynamic Replica Placement Problem is treated. The difference between this problem and the usual RPP is that, in the first, new requests can arrive along the time, and some requests can be removed from the system, indicating the end of the request. The problem is modelled as a Markovian process, which is able to deal with dynamic systems by using mathematical distributions. Requests are modelled as an incoming rate, that is an interval indicating how many requests the system receives per time unit. With this request modelling and an initial state of replica positioning, the decision problem is to choose the best change to perform in such state.

In [12], the authors also used concepts of P2P networks to solve the problem. Servers communicate among each other, exchanging information about local traffic and load. Servers use such information to locally take decisions of creating, deleting and migrating replicas, making the CDN infrastructure more robust, efficient and fault tolerant.

In [7] a hybrid method is proposed to solve the dynamic problem. According to the authors, although the centralized approaches can provide the optimal placement, they are not scalable for large networks. In this approach, all the servers run a copy of the placement algorithm taking in account only the local neighborhood. After that, the servers decide locally where to place the replicas. This hybrid approach presents results that are compatible with the centralized approaches and also keeps the advantages of distributed approaches like scalability and fault tolerance.

In [11] a centralized and a distributed approaches are proposed to solve the dynamic problem. The algorithms take into account ratios that were calculated based on the replicas access frequency, communication and processing cost. The proposed algorithms are compared with each other and the distributed one presented the best results.

Although the attempts to optimize several aspects related to the RPP and other aspects related to the

CDN architecture, fewer works consider the perceived quality at the end clients during the optimization process.

4. Computational Model

The system modelling for the problem is based on three entities: Server, Replica and Request. The Server entity represents the real servers in the overlay network. They have information like storage capacity and bandwidth. The distances (RTT) between each pair of servers are previously computed and stored in a matrix. These distances will be used during the evaluation of a solution. The Servers, in our modelling, are responsible for storing and managing replicas.

The Replica represents a copy of a content. The number of copies of the contents are ruled, in many cases, by contractual clauses, and, because of that, the contents are not necessarily replicated in the whole network. A copy of a content is called, in CDN vocabulary, Replica. These Replicas have information like content size and content identifier.

Clients requests are represented by the Request entity. It contains the content desired by the client, request source and local delay and requirements of maximum delay, minimal bandwidth.

The number of clients is much greater than the number of servers in network environments. To simplify the model and make it scalable we considered that each client is connected to one of the CDN servers. Our model takes into account only servers information in the optimization process. It means that only distances between pairs of servers are known. So it is easy to calculate the time and delay for a request. The links that connect clients to a same server may present different delays. Thus, a field, called local delay, was included in each request, representing such delay. This makes possible to calculate the cost to handle requests individually and also makes the model more scalable.

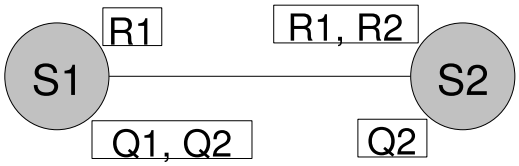

Figure 2: Computational model.

Figure 2 shows a representation of the computational model. This figure shows the same scenario of Figure 1. The servers have the same replicas but now, they also have the requests. Requests contain the source server that forwarded the request and the distance between that server and the client (local delay).

A solution in the model is an assignment of requests to servers, that indicates which servers will be used and which requests they will handle. Note that every request will be associated with some server but, a server may be not used.

In order to evaluate a solution, an objective function that aims at reducing the bandwidth consumption in the network, the delay perceived by the clients and the cost of the initial migration of the replicas was created.

The objective function, for didactic reasons, was split into two parts. Let R be the set of requests and C the replicas set. Let $o(k)$, $h(k)$, $ld(k)$ and $nb(k)$ be functions that return, respectively, the server that is the source of the request k, the server that is currently handling the request k, the local delay between the client that sent request k and the server that is considered its source, and the bandwidth requirement of request k.

$$\sum_{k \in R} (RTT_{o(k),h(k)} + Delay_{o(k),h(k)} + ld(k)) \times nb(k) \tag{1}$$

The first part of Objective Function, shown in (1), is the sum for each request k, of the RTT

between the server that originated the request and the server where the request will be handled, the delay between these two servers and the local delay (between the client and the server to which it is connected), multiplied by the bandwidth requirement of the request.

The second part of the Objective Function corresponds to the additional cost of sending the replicas through the network. This cost has to be included in 1 in order to model a real CDN.

$$\sum_{c \in C} RTT_{0,server(c)} \times size(c) \tag{2}$$

The second part of the Objective Function 2 is the sum for each replica, of the RTT between the main CDN server (for simplification, the main server in the CDN is the server with index 0) and the server that the replica is currently positioned, multiplied for the size of this replica. This part of the objective function rises because, in some cases, there are costs to send contents through the network and these costs are proportional to the size of the contents.

5. Scenarios

To generate the information about the network topology, the topology generator BRITE [2] was used. Four topologies were built with 20, 30, 40 and 50 servers. The link delay was set to vary between 15 and 20 ms. The RTT was set to be the double of the delay value (unloaded network). Default values were used to all other parameters in BRITE. Once the topology is created, the delay and RTT between each pair of servers is calculated using the Dijkstra algorithm.

The number of contents was set to three, but the model is general enough to deal with any number of contents.

The servers are heterogeneous, having an output bandwidth that varies between 1 and 1.8 Gb/s and a storage capacity between 100 and 200 GB.

It is known that some contents are more popular than others. This implies in a higher number of requests for more popular contents, and also a higher number of replicas. As the number of contents was set to three, the popularity of a content was modelled through the number of requests for that content. So, for the tests in this work, the number of requests for content 1, 2, and 3 is, respectively, 1/6, 2/6 and 3/6 of the total number of requests.

The number of replicas for each content follows the same popularity idea of the requests. That means that 1/6 of replicas are of content 1, 2/6 of content 2 and 3/6 of content 3. The size of a replica varies between 1 and 1.5 GB. As determining the ideal number of replicas is out of the scope of this work, it is assumed that there may exist unused replicas, but, at least one replica of each content must exist. In other words, a request for a content that does not have a replica in the CDN can not exist. In the tests include 36 replicas, 6 of content 1, 12 of content 2 and 18 of content 3.

The requests were created using the Normal distribution of probability varying from 15 to 500 requests per server. Each request has a delay restriction that varies between 400 and 800 ms and a needed band that varies from 1 to 2 Mb/s. Each request has a local delay that varies between 50 and 100 ms.

6. Mathematical Model

To the best of our knowledge, no mathematical formulation for the RPRDP exist in the literature. We propose the following one. Let R be the set of requests, S the servers set in the CDN, C the set of replicated contents. Let T_k, NR_k, BR_i, BS_j and E_j be the size of content k, the number of replicas available for content k, the needed band of request i, the outgoing bandwidth of server j and the available disk space in server j, respectively. Let c_{ij} be the cost for server j to handle the request i. This cost is previously calculated for each pair *(request, server)* taking into account factors such as distance between the server that the request came from and the server j, the local delay of request i, and the delay restriction of request i.

The following variables and notations were defined:

- $y_{jk} = \begin{cases} 1 & \text{if content } k \text{ is replicated in server } j \\ 0 & \text{otherwise} \end{cases}$
- $x_{ij} = \begin{cases} 1 & \text{if request } i \text{ is handled by server } j \\ 0 & \text{otherwise} \end{cases}$
- c_{ij} - the cost for server j to handle request i
- R - the set of requests to handle.
- S - the servers set in the CDN.
- C - the replicated contents set.
- T_k - the size of content k.
- E_j - the available disk space in server j.
- BS_j - the outgoing bandwidth of server j.
- BR_i - the needed bandwidth of request i.
- NR_k - the number of replicas available for content k.
- $K(i)$ - function that determines which content is asked by request i.

The proposed Mathematical model is represented by:

$$Min \quad \sum_{i \in R} \sum_{j \in S} c_{ij}\ x_{ij} + \sum_{j \in S} \sum_{k \in C} T_k\ y_{jk} \tag{3}$$

$$S.a.$$

$$\sum_{j \in S} x_{ij} = 1\ \forall i \in R \tag{4}$$

$$\sum_{i \in R} BR_i\ x_{ij}\ \leq\ BS_j\ \forall j \in S \tag{5}$$

$$x_{ij} \leq y_{jK(i)}, \quad \forall i \in R\ \ \forall j \in S \tag{6}$$

$$\sum_{j \in S} y_{jk}\ \leq\ NR_k \forall k \in C \tag{7}$$

$$\sum_{k \in C} T_k\ y_{jk}\ \leq\ E_j, \forall j \in S, \tag{8}$$

$$x_{ij} \in \{0,1\}, \forall i \in R, \forall j \in S \tag{9}$$

$$y_{jk} \in \{0,1\}, \forall j \in S, \forall k \in C \tag{10}$$

7. Constructive Heuristic

In this work a constructive heuristic is proposed with the objective of providing faster way to obtain good solutions for large instances of the problem.

The proposed heuristic takes into account the fact that the problem can be split into smaller problems, i.e., an instance of the RPRDP can be split into several instances of the CFLP (each content and the requests associated to that content compose an instance of the CFLP).

The idea is to solve the problem considering one single content at each step. After obtaining the result, the bandwidth consumed by the requests and the amount of space in disk used to store the

replicas are subtracted from the used servers. The servers with updated bandwidth and storage capacity are used to solve the problem for the next content.

After solving the problems for all contents, the individual solutions are merged into a single solution for the RPRDP.

The pseudo-code of the constructive heuristic, called PartialConstructive, or PC, is shown in Algorithm 1

Algorithm 1 Procedure PartialConstructive()

```
1: Split the problem into k sub problems
2: Sort the problems following some criteria
3: for all sub problem do
4:    PartialSolution ps = SolveCFLP(servers)
5:    Store ps
6:    UpdateStorageBandwidthCapacity(servers, ps)
7: end for
8: Merge the partial solutions into a single solution finalSolution
9: Return finalsolution
```

The PartialConstructive algorithm proceeds as follows. First, it splits the RPRDP into k instances of the CFLP, where k is the number of contents in the CDN. In the second step, the problems are sorted according to a criterion. After that, all instances of the CFLP are solved, in the order established by the previous step, using a solver (in our case, we used CPLEX [3]). Finally, the solutions of the CFLP problems are stored. Note that the set of possible servers to position the replicas and their capacities, are passed as parameters to the solver. For each CFLP solved, the used servers in the solution ps are updated. If a server is used in the solution of the current CFLP to handle three requests that consume 10 Kb/s each, and the current content has a size of 200 MB, then 30 Kb/s and 200 MB are subtracted from the capacity of the used server, and the new capacity is passed as a parameter to the next CFLP.

Although the CFLP instances are independent from each other, the order that the instances are evaluated in the heuristic has a significant influence in the quality of the produced solution. In our preliminary tests, the order that produced the best results, was the one that solves first the instances with fewer requests associated with them. In other words, by solving the smaller problems first, we obtained the best results in our tests.

8. Experimental Results

In this section the result obtained with the mathematical formulation of Section 6 and with the PartialConstructive heuristic of Section 7 are presented.

The formulation and the PartialConstructive heuristic were compiled using the *G++* compiler, version 4.0.3, and the well known solver CPLEX [3], version 10.

The computer used in the tests was a *Pentium D* with 3,0 GHz and 2 GB of RAM memory, with a Linux operational system with kernel version 2.6.15.

Table 1: Results Formulation X PC

	Formulation		PC	
Instance	GAP (%)	Time(s)	GAP (%)	Time(s)
1	0	7.8	0	9.3
2	0	2.7	0	3.2
3	0	0.6	0	0.8
4	0	531.2	0	293.6
5	0	2430.6	0	1717.9
6*	0.04	10800	0.9	3226
Average	0.01	2295	0.15	875

Table 1 summarizes the obtained results. The column *GAP* shows the distance, in percentage, from the solution obtained to the best Solution. The column *Time* shows the CPU time for each method. Note that the instance 6 is marked with an $*$. This mark means that the best solution is not known for this instance, so the gap was calculated using the best fractional solution found by CPLEX in this case.

It can be observed that for the smaller instances (*1, 2* and *3*), both methods were able to find the best solution but, the formulation got the best results in time. This was already expected since the initialization of the CPLEX algorithms has a well known overhead [3]. For the larger instances (*4, 5* and *6*) the PC heuristic got the best results in time and was able to find the optimal solution in almost all cases.

9. Conclusions and Future Work

This work presented the Replica Placement an Request Distribution Problem, that aims at reducing the network traffic and improving the quality perceived at end clients by introducing QoS constraints in the clients requests. Two distinct approaches to solve it were also presented. One of these approaches is a mathematical model, that was able to find the optimal solution for some instances of the problem. The other approach, a heuristic, was able to find the optimal solution in almost all cases and is much less time consuming than the exact approach for larger instances.

The presented results show that for small instances the mathematical model is the best approach to solve the problem.

As future work, the study of the dynamic RPRDP, where the request set changes along the time, rises as the natural option.

The use of metaheuristics or hybrid methods, which use mathematical and heuristic techniques at the same time, also seems to be a promising way to solve the RPRDP and the dynamic RPRDP as well.

References

[1] AKAMAI. World Wide Web, www.akamai.com. 03/2008.

[2] BRITE. World Wide Web, http://www.cs.bu.edu/brite/. Acessado em Junho em 2007.

[3] ILOG S.A., CPLEX 10 user's manual, 2006.

[4] Almeida, J. M., Eager, D. L., Vernon, M. K., Wright, S. J. Minimizing delivery cost in scalable streamingcontent distribution systems. *IEEE TRANSACTIONS ON MULTIMEDIA*, 2004, 6, 356–365.

[5] Bartolini, N., Presti, F., Petrioli, C. Optimal dynamic replica placement in content delivery networks. *Proc. of The 11th IEEE International Conference on Networks 2003. ICON2003* , 2003, p. 125–130.

[6] Bektas, T., Oguz, O., Ouveysi, I. Designing cost-effective content distribution networks. *Computers & Operations Research*, 2007, 34, 2436–2449.

[7] Coppens, J., Wauters, T., Turck, F. D., Dhoedt, B., Demeester, P. Design and performance of a self-organizing adaptive content distribution network. *Proc. of Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, 2006, p. 534–545.

[8] Huang, C., Abdelzaher, T. Towards content distribution networks with latency guarantees. *Proc. of Twelfth IEEE International Workshop on Quality of Service, 2004. IWQOS 2004*, 2004, p. 181–192.

[9] Korupolo, M. R., Plaxton, C. G. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 2000, 37, 146–188.

[10] Pirkul, H., Jayaraman, V. A multi-commodity, multi-plant capacitated facility location problem: Formulation and efficient heuristic solution. *Computers & Operations Research*, 1998, 25(10, 869–878.

[11] Tenzekhti, F., Day, K., Ould-Khaoua, M. Replication algorithms for the world-wide web. *Journal of Systems Architecture*, 2004, 50, 591 – 605.

[12] Wauters, T., Coppens, J., Dhoedt, B., Demeester, P. Load balancing through efficient distributed content placement. *Proc. of Next Generation Internet Networks*, 2005, p. 99–105.

[13] Zhou, X., Xu, C.-Z. Efficient algorithms of video replication and placement on a cluster of streaming servers. *Journal of Network and Computer Applications*, 2007, 30, 515–540.