Solving Replica Placement and Request Distribution in Content Distribution Networks

Tiago Araújo Neves,[1] Lúcia M. A. Drummond, Luiz S. Ochi, Célio Albuquerque and Eduardo Uchoa[2]

Instituto de Computação/Escola de Engenharia
Universidade Federal Fluminense
Niterói, Brazil

Abstract

A Content Distribution Network (CDN) is an overlay network where servers replicate contents and distribute client's requests with the aim at reducing delay, server load and network congestion, hence improving the quality of service (QoS) perceived by end clients. Because of server constraints and costs involved in the replication process, it is not reasonable to replicate the contents over the entire set of servers. In this work, exact and heuristic approaches are proposed to solve a dynamic and online problem that appears in CDN management, called the Replica Placement and Request Distribution Problem. The overall objective is to find the best servers to keep the replicas and to handle requests so that the traffic cost in the network is minimized without violating server and QoS constraints.

Keywords: Content Distribution Network, Hybrid Heuristc, Replica Placement and Request Distribution Problem, Mathematical Formulation.

[1] Email: {tneves,lucia,satoru,celio}@ic.uff.br
[2] Email: uchoa@producao.uff.br

1 Introduction

The problem tackled in this work is the Replica Placement and Request Distribution Problem (RPRDP) that appears in the management of Content Distribution Networks. This is a dynamic and online problem whose objectives are to find the best position for the content replicas and to distribute the requests through the servers, to reduce the network load without violating QoS constraints. Servers have limited capacity in bandwidth and disk space. Quality of Service (QoS) constraints are given by a required minimal bandwidth and a maximum delay in which a client's request must be served. Initially, all contents are positioned only in their origin servers. As clients requests arrive in the CDN, contents may be replicated over the network and such requests may be redistributed through the servers taking into account QoS constraints. A server is allowed to handle a request partially or totally as long as it has a replica of the desired content, i.e., a request may be served by several servers partially at the same time and may also be served in several periods of time when necessary. In this problem, there is a tradeoff between the reduction of bandwidth and free disk space in servers, and the cost of replica transmission over the network. Therefore, not only the QoS constraints but also such tradeoff must be verified. RPRDP is a dynamic and online problem, meaning that costs of communication can change, new contents and requests can come up and the future scenario is not known. This problem extends the Replica Placement Problem (RPP), which belongs to the NP-complete class [4] and consists only in finding the best set of servers to place content replicas over the CDN without considering QoS constraints.

There are a number of recent solutions proposed for CDN management problems [4] [5] [6] [7] [8] [11] . However, none of them treats important issues like QoS constraints, network capacity and server load simultaneously. This paper proposes exact and heuristic approaches to solve the RPRDP, considering all these above mentioned characteristics.

2 Mathematical Formulation

A mathematical formulation, based on the one presented in[4], that considers all the characteristics of the problem is proposed in this paper. Although the use of mathematical formulation is not the most appropriate approach for online problems, it is a key technique for obtaining bounds. It is important to notice that in the offline version of the problem, solved using the formulation, all information about future costs and requests is available, allowing a better replica allocation and reduction of operational costs.

Let R be the set of requests, S be the set of servers, C the set of contents and T the set of time periods. Also let δ be the length of a period in seconds, $origin(i)$ be the server where request i comes from, $ld(i)$ be the local delay of request i, $delay(j_1, j_2, t)$ be the delay between servers and $RTT(j_1, j_2, t)$ be the round trip time between then on period t. The demand of a request in a certain period (D_{it}) is set to the maximum bandwidth that request can bear (BX_i) while the client is being handled. Let L_k be the size of content k and B_k be the period that content k is submited to the CDN. E_k is the period that content k is removed from the CDN, O_k the origin server of content k, AS_j is the available disk space on server j, MB_j is the maximum bandwidth of server j, BR_i is the minimal bandwidth of request i, G_i is the content asked by request i, c_{ijt} is the cost of handling request i by server j, on period t, this cost is given by the equation $c_{ijt} = (RTT(origin(i), j, t) + delay(origin(i), j, t) + ld(i)) \times BR_i$, p_{it} is the backlog penalty paid for request i on period t and h_{kjlt} is the cost that server j pays for downloading content k from server l on period t. In our experiments, this cost is always set to 0 when j is equal to l and set to L_k otherwise.

The formulation uses the following variables: x_{ijt} is the fraction of content asked by request i handled by server j in period t; y_{kjt} assumes value 1 if content k is replicated on server$j$ in period t, and 0 otherwise; b_{it} represents the backlog of request i on period t; w_{kjlt} assumes value 1 if content k is copied by server j from server l on period t, and 0 otherwise.

$$(1) \quad Min \sum_{i \in R} \sum_{j \in S} \sum_{t \in T} c_{ijt} x_{ijt} + \sum_{i \in R} \sum_{t \in T} p_{it} b_{it} + \sum_{k \in C} \sum_{j \in S} \sum_{l \in S} \sum_{t \in T} h_{kjlt} w_{kjlt}$$

$S.t.$

$$(2) \quad \sum_{j \in S} L_{G_i} x_{ijt} - b_{i(t-1)} + b_{it} = \delta D_{it} \forall i \in R, \forall t \in \{B_{G_i}, E_{G_i}\}$$

$$(3) \quad \sum_{i \in R} L_{G_i} x_{ijt} \leq \delta MB_j \ \forall j \in S, \forall t \in T$$

$$(4) \quad \sum_{j \in S} L_{G_i} x_{ijt} \leq \delta BX_i \ \forall i \in R, \forall t \in T$$

$$(5) \quad \sum_{j \in S} \sum_{t \in T} x_{ijt} = 1 \ \forall i \in R$$

$$(6) \quad y_{G_i jt} \geq x_{ijt} \ \forall i \in R, \forall j \in S, \forall t \in T$$

$$(7) \quad \sum_{j \in S} y_{kjt} \geq 1 \ \forall k \in C, \forall t \in [B_k, E_k]$$

(8) $y_{kjt} = 0 \ \forall k \in C, \forall j \in S, \forall t \notin [B_k, E_k]$

(9) $y_{kO_kB_k} = 1 \ \forall k \in C$

(10) $y_{kjB_k} = 0 \ \forall k \in C, \forall j \in \{S | j \neq O_k\}$

(11) $y_{kj(t+1)} \leq \sum_{l \in S} w_{kjlt} \ \forall k \in C, \forall j \in S, \forall t \in T$

(12) $y_{kjt} \geq w_{kljt} \ \forall k \in C, \forall j, l \in S, \forall t \in T$

(13) $\sum_{k \in C} L_k \, y_{kjt} \ \leq \ AS_j \ \forall j \in S, \forall t \in T$

(14) $x_{ijt} \in [0, 1] \ \forall i \in R, \forall j \in S \forall t \in T$

(15) $y_{kjt} \in \{0, 1\} \ \forall j \in S, \forall k \in C \forall t \in T$

(16) $b_{it} \geq 0 \ \forall i \in R, \forall t \in T$

(17) $w_{kjlt} \in \{0, 1\} \ \forall j, l \in S, \forall k \in C \forall t \in T$

The objective function (1) minimizes the operational costs and the backlog. Constraints (2) relate demand and backlog. Servers bandwidth are controlled by constraints (3). Constraints (4) prevent that a request receive more than it can bear. Constraints (5) guarantee that every request is fully handled. Constraints (6) impose that a request must be handled by a server that has a replica of the desired content. Constraints (7) and (8) control the number of replicas of active contents. Constraints (9) and (10) make sure that only origin server has a replica of a content on the submission period. Constraints (11) guarantee that all replications create a new replica. Constraints (12) make sure that a replication occurs from a server that has the content. The servers disk space are controlled by constraints (13). The remaining constraints are the integrality and non-negativity constraints.This formulation, called *FD*, is used to solve several instances of the problem, and its performance is shown in Section 6.

3 The Hybrid Constructive Heuristic (*HC*)

The *HC* heuristic proceeds as follows. For all periods, the algorithm solves the Request Distribution Problem (RDP) using a mathematical formulation extracted from *FD* and described in details in [10].

To solve the replica positioning a greedy heuristic is used. The heuristic tries to insert in each server the contents that clients of this server demands more. If there is enough disk space, the replicas are simply placed. Otherwise, the heuristic tries to remove replicas with lesser demand from the server. At the end, the solutions for all periods are concatenated in a single solution to the original problem.

A comprehensive set of results for this heuristic is documented in [10] and shows that, on the considered instances, *HC* presented an average gap of 5.5% from the optimal solutions obtained by *FD*. From the presented results it is not possible to determine if these gaps are caused by the difference between the online and offline versions of the problem or if it is caused by the natural difference between exact and heuristic approaches. In order to explain the reason of the gaps, two heuristics, Hybrid Constructive with Future knowledge (HCFK) and Period Solution Heuristic (PSH), are proposed.

4 The *HCFK* and *PSH* heuristics

The *HCFK* heuristic is similar to *HC*. The difference between them is that *HCFK* knows the real demand of next period and does not need to estimate it. The *PSH* solves the RPRDP exactly for each period individually. Then, at the end, those individual solutions are concatenated to build a solution to the original problem. Although the solution for each period is optimal, the final one produced by this heuristic may not be globally optimal. This happens because the knowledge of the future attributes may lead to a cost reduction that can not be achieved without such *a priori* knowledge.

The formulation used in *PSH* to solve the subproblem presented in each period is based on *FD*. The main difference of this formulation and *FD* is that it does not have a time dimension. Thus, all t indexes from variables and constraints are removed. The constraints (5) are no longer necessary because this formulation deals with a single period. The constraints (9) and (10) are also excluded and may be replaced by the constraints (18) which are applied whenever a content k is submitted in the current period. The remaining notation keep the same meaning.

$$w_{kjl} = 0 \; \forall j, l \in S, j \neq l, \forall k \in \{C | B_k = t\} \tag{18}$$

The solutions produced by both of these heuristics can not be used in practice, since the first uses future data and the second does not consider the time needed to move contents. However, these solutions can be used for gap analysis. By comparing the results obtained by these two methods and *FD*, it is possible to verify the impact of using a greedy approach instead of an exact one and the impact that the *a priori* knowledge of all attributes has in the solutions quality. In the case the results of *PSH* and *FD* are close, it could be concluded that the future knowledge of attributes has a small influence in the solutions quality, meaning that the online and offline versions of the problem present similar difficulty. In the case the results of *HC* and *HCFK* are very different, it could be concluded that the future knowledge has great influence in solutions quality.

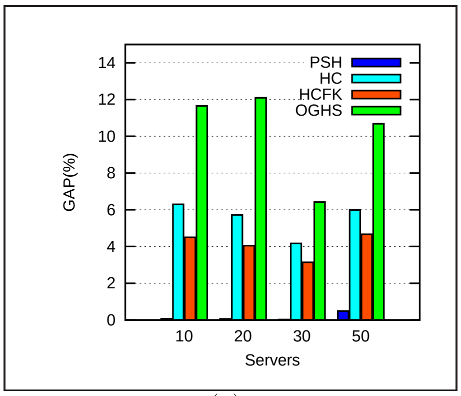
5 Global Hosting System Heuristic (*GHS*)

The *GHS*, a patented solution of a CDN provider, is presented in [9]. This heuristic replicates a content based simply on current demand, without considering the historical data. Every time a request is received it will be handled only by the origin server of this request. If the origin does not have a replica of the content, instead of forwarding the request to another server where the content is replicated, it will first download the content and then handle the request. A "Last Recent Used" (LRU) scheme is used to discard replicas in case of lack of storage space in the origin server. This heuristic was not able to achieve competitive results when compared with *PSH* and *HC*. Thus, in order to have a fair comparison, a hybridization of this heuristic was also proposed. The new hybrid heuristic, called *OGHS*, consists in using the formulation used in *HC* to deal with the RDP and use the method described in [9] to deal with the Replica Placement Problem.

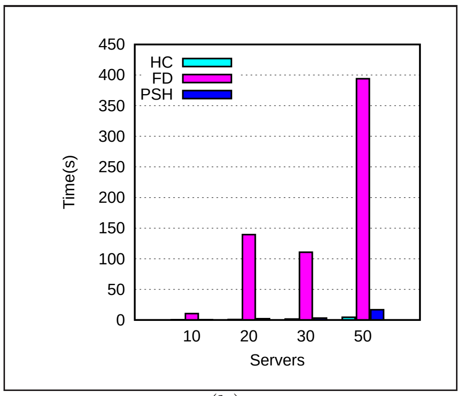
6 Results

The algorithms presented on previous sections were implemented in C++ using g++ version 4.3 and executed on a Quad-Core with 2.83 GHz/core, 8 Gigabytes of RAM using Linux (kernel 2.6). To solve all the formulations the solver CPLEX 11.2 [3] was used. To the best of authors knowledge there are no instances for this problem available, so 60 instances were generated (15 for each number of servers) with different numbers of servers (10, 20, 30 and 50), contents (4 - 15) and requests (445 - 3762). These instances were generated by using BRITE topology generator [1], that tries to generate topologies similar to real networks, and are available on the LABIC [2] website.

Figure 1 summarizes the obtained results. Figure 1(a) presents the average gaps for 15 instances of each size obtained by *HC*, *PSH*, *HCFK* and *OGHS* heuristics when compared to *FD*. Although *HC* does not have any knowledge about the future attributes, the average gap was only about 5.5% of the optimal solution, as shown in Figure 1(a). The gaps obtained by *PSH* and *HCFK* indicate that the gaps produced by *HC* are mainly caused by the natural gap between exact and heuristic approaches, although the difference between the online and offline versions of the problem also contributes for this gap. Fig-



(a)



(b)

Fig. 1. Gap and Time analysis

ure 1(b) shows the averages of computational times expressed in seconds. We decided to not plot the times for *OGHS* and *HCFK* because these times are quite similar to the ones observed for *HC*, since these heuristics have almost the same behaviour. The Figure 1(b) also shows that the time for executing *FD* increases in a non linear way as the instance size grows, and that the performance of the heuristics are much better in terms of time. The average execution time *FD* for instances with 20 servers is higher than the average time for instances with 30 servers because there was one instance, in which CPLEX spent more than 600 seconds to prove optimality. Thus, if execution time of such instance was not regarded, the average time for instances with 30 servers would be higher than the average time spent for instances with 20 servers. For detailed results the reader is referred to [10].

7 Conclusion and Acknowledgments

This paper presents a brief description of the Replica Placement and Request Distribution Problem, proposes a mathematical formulation (*FD*) to the offline version of the problem and a hybrid heuristic called (*HC*) for the online version. Besides that, two heuristics, *HCFK* and *PSH*, were developed in order to analyse the reason of the gaps between *HC* and *FD*. We implemented a heuristic used by real CDN providers [9] but as this heuristic was not proposed to scenarios with QoS constraints, it produced solutions with very poor quality, so a hybridization of this heuristic, called *OGHS* was also proposed. The results show that *HC* can reach good results in much less computational time when compared with *FD*, observing that *FD* is applied to the offline version of the problem. The analysis of the gaps by using *PSH* and *HCFK*, indicates that the gaps between *HC* and *FD* are mainly caused by the natural difference between exact and heuristic methods. It is also clear that *HC* outperforms *OGHS*, since *HC* produce solutions with lower gaps in quite similar times. All proposed approaches consider minimal bandwidth, maximum delay, better use of network capacity and servers load. To the best of authors knowledge, this is the first work to deal with this problem considering so many realistic details.

The authors would like to thank the support of CAPES, CNPq and FAPERJ.

References

[1] *BRITE*. World Wide Web, http://www.cs.bu.edu/brite/. 06/2007.

[2] *LABIC*. World Wide Web, http://labic.ic.uff.br/. 02/2009.

[3] *ILOG Inc., CPLEX 11 user's manual*, 2008.

[4] Aioffi, W., Mateus, G., Almeida, J., Loureiro, A. *Dynamic Content Distribution for Mobile Enterprise Networks*. IEEE Journal on Selected Areas in Communications **23**, 10 (2005), 2022–2031.

[5] Bartolini, N., Presti, F., Petrioli, C. *Optimal Dynamic Replica Placement in Content Delivery Networks*. In *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 **34** (2007), 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*. In *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*. In *Twelfth IEEE International Workshop on Quality of Service, 2004. IWQOS 2004* (2004), p. 181–192.

[9] Leighton, F. T., Lewin, D. *Global Hosting System*, August 2000. US Patent:US006108703.

[10] Neves, T., Drummond, L., Satoru, L., Albuquerque, C., Uchoa, E. *Hybrid and Exact Approaches for Replica Placement in Content Distribution Networks*. Technical Report, UFF, 2009. http://labic.ic.uff.br/.

[11] Tenzakhti, F., Day, K., Ould-Khaoua, M. *Replication algorithms for the World-Wide Web*. Journal of Systems Architecture **50** (2004), 591–605.