OPTIMAL CLOSEST POLICY WITH QOS AND BANDWIDTH CONSTRAINTS FOR PLACING REPLICAS IN TREE NETWORKS

Veronika Rehn-Sonigo
École Normale Supérieure de Lyon
LIP, UMR CNRS-INRIA-UCBL 5668
Lyon, France
vrehn@ens-lyon.fr

Abstract This paper deals with the replica placement problem on fully homogeneous tree networks known as the REPLICA PLACEMENT optimization problem. The client requests are known beforehand, while the number and location of the servers are to be determined. We investigate the latter problem using the *Closest* access policy when adding QoS and bandwidth constraints.

In this paper, we state that the extension of *Closest*/Homogeneous with QoS to bandwidth keeps polynomial. This is an important cognition, as the postulated constraints are of different nature. QoS is a constraint that belongs to a node locally, whereas bandwidth constraints have a global influence on the resources. We propose an optimal algorithm in two passes using dynamic programming.

Keywords: Replica placement, tree networks, *Closest* policy, quality of service, bandwidth constraints.

1. Introduction

This paper deals with the problem of replica placement in tree networks with Quality of Service (QoS) guarantees and bandwidth constraints. Informally, there are clients issuing several requests per time-unit, to be satisfied by servers with a given QoS and respecting the bandwidth limits of the interconnection links. The clients are known (both their position in the tree and their number of requests), while the number and location of the servers are to be determined. A client is a leaf node of the tree, and its requests can be served by one or several internal nodes. Initially, there are no replicas; when a node is equipped with a replica, it can process a number of requests, up to its capacity limit (number of requests served by time-unit). Nodes equipped with a replica, also called servers, can only serve clients located in their subtree (so that the root, if equipped with a replica, can serve any client); this restriction is usually adopted to enforce the hierarchical nature of the target application platforms, where a node has knowledge only of its parent and children in the tree. Every client has some QoS constraints: its requests must be served within a limited time, and thus the servers handling these requests must not be too far from the client.

The rule of the game is to assign replicas to internal nodes so that some optimization function is minimized and QoS as well as bandwidth constraints are respected. Typically, this optimization function is the total utilization cost of the servers. We restrict the problem to the most popular access policy called *Closest*, where each client is allowed to be served only by the closest replica in the path from itself up to the root.

In this paper we study this optimization problem, called REPLICA PLACEMENT, and we restrict the QoS in terms of number of hops. This means for instance that the requests of a client who has a QoS range of 5 must be treated by one of the first five internal nodes on the path from the client up to the tree root.

We point out that the distribution tree (clients and internal nodes) is fixed in our approach. This key assumption is quite natural for a broad spectrum of applications, such as electronic, ISP, or VOD service delivery. The root server has the original copy of the database but cannot serve all clients directly, so a distribution tree is deployed to provide a hierarchical and distributed access to replicas of the original data.

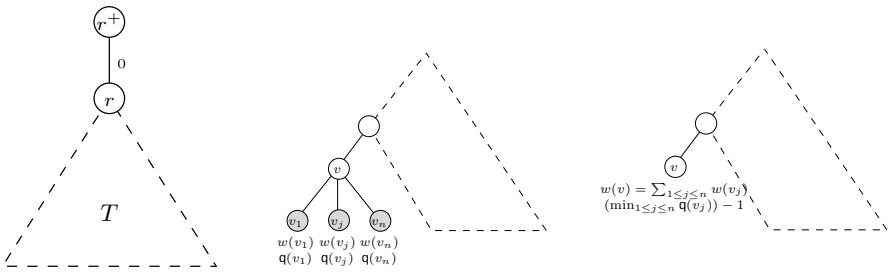
In this paper we propose an efficient algorithm called **Optimal Replica Placement** (*ORP*) to determine optimal locations for placing replicas in the REPLICA PLACEMENT problem including QoS and bandwidth. Our work provides an extension of the algorithm of Lin et al [6], which was already mentioned above. Lin et al [6]proposed an algorithm **Place-replica** to find an optimal set of replicas on homogeneous data grid trees including QoS constraints in terms of distance but without bandwidth constraints. Our approach

leads to two extensions. First of all, we separate the set of clients from the set of servers. Lin et al also suppose clients to be leaf nodes, but with the double functionality of a server and client. Our separation allows that client nodes do not have to offer the possibility to place replicas on them, which demands less assumptions on leaf nodes. However our model can simulate the latter model while the converse is not true. Indeed, we can model client-server nodes by inserting a fictive node before the client which can take the role of a server. The approach of Lin et al in contrast does not offer the possibility to model clients without server functionality. Our second contribution is the introduction of bandwidth constraints. This is an important modification of the requirements as QoS and bandwidth are of a completely different nature. QoS is a constraint that belongs to a node locally, hence each client has to cope with its own limitation. Bandwidth constraints in contrast have a global influence on the resources as a link may be shared by multiple clients and consequently all of them are concerned. Therefore it is not obvious whether the problem with these completely different constraint types would remain polynomial or would become NP-hard.

The rest of the paper is organized as follows. Section 2 introduces our main notations used in REPLICA PLACEMENT problems. Section 3 is dedicated to the presentation of our polynomial algorithm: the proper terminology of the algorithm is introduced in Section 3.1. The subsections 3.2 and 3.3 treat the different phases. Some related work can be found in Section 4. Complexity and optimality are subject of Section 3.4. Section 5 finally summarizes our work.

2. Notations

This section familiarizes with our basic notations. We consider a distribution tree $\mathcal{T}$ whose nodes are partitioned into a set of clients $\mathcal{C}$ and a set of internal nodes $\mathcal{N}$ ($\mathcal{N} \cap \mathcal{C} = \emptyset$). The clients are leaf nodes of the tree, while $\mathcal{N}$ is the set of internal nodes. Let r be the root of the tree. The set of tree edges (links) is denoted as $\mathcal{L}$. Each link l owns a bandwidth limit $\mathsf{BW}(l)$ that can not be exceeded. A client $v \in \mathcal{C}$ is making $w(v)$ requests per time unit to a database. Each client has to respect its personal *Quality of Service* constraints (QoS), where $\mathsf{q}(v)$ indicates the range limit in hops for v upwards to the root until a database replica has to be reached. A node $j \in \mathcal{N}$ may or may not have been provided with a replica of the database. Nodes equipped with a replica (*i.e.* servers) can process up to W requests per time unit from clients in their subtree. In other words, there is a unique path from a client v to the root of the tree, and each node in this path is eligible to process all the requests issued by v when provided with a replica. We denote by $R \subseteq \mathcal{N}$ the entire set of nodes equipped with a replica.



(a) Appearance of T^* (b) Node v before ... (c) ... and after suppression.

Figure 1. Transformations.

3. Optimal Replica Placement Algorithm (*ORP*)

In this section we present *ORP*, an algorithm to solve the REPLICA PLACEMENT problem using the *Closest* policy with QoS and bandwidth constraints. For this purpose, we modify an algorithm of Lin, Liu and Wu [6]. Their algorithm **Place-replica** is used on homogeneous conditions with QoS constraints but without bandwidth restrictions. Hence to be able to use the algorithm, we have to modify the original platform. We transform the tree T in a tree T^* by adding a new root r^+ as father of the original root r (see Figure 1(a)). r^+ is connected to r via a link l_0, where $\mathsf{BW}(l_0) = 0$. As the bandwidth is limited to 0, no requests can pass above r, so that this artificial transformation for computation purposes can be adapted to any tree-network. We make this changing to be able to model whether the original root r is equipped with a replica or not.

A further, only formal transformation, consists in the suppression of clients from the tree and hence the consideration of their parents as leaves in the following way: for every parent p who has only leaf-children $v_1, .., v_n$, we assign the sum of the requests of the v_j as its requests $w(p)$, i.e., $w(p) = \sum_{1\leq j\leq n} w(v_j)$. The associated QoS is set to $(\min_{1\leq j\leq n} \mathsf{q}(v_j)) - 1$. (Figures 1(b) and 1(c) give an illustration). This transformation is possible, as we use the *Closest* policy and hence all children have to be treated by the same server. From those parents who have some leaf-children $v_1, .., v_n$, but also non-leaf children $v_{n+1}, .., v_m$, the clients can not be suppressed completely. In this case the leaf-children $v_1, .., v_n$ are compressed to one single client c with requests $w(c) = \sum_{1\leq j\leq n} w(v_j)$ and QoS $\mathsf{q}(c) = \min_{1\leq j\leq n} \mathsf{q}(v_j)$. Once again this compression is possible due to the restriction on the *Closest* access policy.

ORP works in two phases. In the first phase so called Contribution Functions are computed which will serve in the second phase to determine the optimal replica placements. In the following some new terms are introduced and then the two phases are described in detail.

3.1 Terminology

Working with a tree T^* with root r^+, we note $t(v)$ the subtree rooted by node v, and $t'(v) = t(v) - v$, i.e. the forest of trees rooted at v's children. The i'th ancestor of node v, traversing the tree up to the root, is denoted by $a(v, i)$.

Using these notations, we denote $m(T^*)$ the minimum cardinality set of replicas that has to be placed in tree T such that all requests can be treated by a maximum processing capacity of W (respecting QoS and bandwidth constraints). In the same manner $m(t(v))$ denotes the minimum number of replicas that has to be placed in $t'(v)$, such that the remaining requests on node v are within W. For this purpose we define a contribution function C. $C(v, i)$ denotes the minimum number of requests on node $a(v, i)$ contributed by $t(v)$ by placing $m(t(v))$ replicas in $t'(v)$ and none on $a(v, j)$ for $0 \leq j < i$. The computation is presented below (Cf. Section 3.2). But before we need a last notation. The set $e(v, i)$ denotes the children of node v that have to be equipped with a replica such that the remaining requests on node $a(v, i)$ are within W, there are exactly $m(t(v))$ replicas in $t'(v)$ and none on $a(v, j)$ for $0 \leq j < i$ and the contribution $t(v)$ on $a(v, i)$ is minimized. The computation formula is also given below.

3.2 Phase 1: Bottom up computation of set e, amount m and contribution function C

The computation of e, m and C is a bottom up process, distinguishing two cases.

1. v **is a leaf.** In this case we do not need e and m and we can directly compute the contribution function. $C(v, i)$ is $w(v)$ when $(i \leq \mathsf{q}(v) \wedge w(v) \leq \min_{\mathsf{BW}} \mathsf{path}[v \rightarrow a(v, i)])$, and infinity otherwise.

We point out that there is no solution if any of the leaves has more requests than W or if the bandwidth of any of the clients to its parent is not sufficiently high.

2. v **is an internal node with children** $v_1, \ldots, v_n$**.**
$i = 0$**:** If the contribution on v of its children, i.e. the incoming requests on v is bigger than the processing capacity of inner nodes W, we know we have to place some replicas on the children to bound the incoming requests on W. To find out which children have to be equipped with a replica, we take a look at the $C(v_j, 1)$-values of the children. The set $e(v, 0)$ is used to store the v_j's that are determined to be equipped with a replica. Hence the procedure is the following:

$e(v, 0) = \emptyset$
while $\sum_{v_j \notin e(v,0)} C(v_j, 1) > W$ **do**
add $v_j \in \mathcal{N}$ with biggest $C(v_j, 1)$ to $e(v, 0)$

Note that the set $\mathcal{N}$ used in the procedure still corresponds to the set of internal nodes of the original tree T. So we can add leaf nodes of T^* that are inner nodes in T, but we can not add compressed client nodes. Note furthermore that there is no client that is added to $e(v,0)$. Besides we remark that there is no valid solution within W and the present QoS and bandwidth constraints, when all children $v_j \in \mathcal{N}$ of v are equipped with a replica and the incoming requests do not fit in W. Of course this holds also true in the case $i > 0$. Subsequently, the value of $m(t(v))$ is determined easily: $m(t(v)) = \sum_{1 \leq j \leq n} m(t(v_j)) + |e(v,0)|$. We remind that $m(t(v))$ indicates the minimum number of replicas that have to be placed in $t'(v)$ to keep the number of contributed requests inferior to W. Finally, the computation of the contribution function : $C(v,0) = \sum_{v_j \notin e(v,0)} C(v_j, 1)$.

$i > 0$**:** Treating node v, we want to compute the contribution on $a(v,i)$. As for $i = 0$, we start computing the set $e(v,i)$:

$e(v,i) = \emptyset$
while $\sum_{v_j \notin e(v,i)} C(v_j, i+1) > W$ **do**
add $v_j \in \mathcal{N}$ with biggest $C(v_j, i+1)$ to $e(v,i)$

The computation of the contribution function follows a similar principle:

$$C(v,i) = \begin{cases} \sum_{v_j \notin e(v,i)} C(v_j, i+1), & \text{if } |e(v,i)| = |e(v,0)| \\ \infty, & \text{otherwise} \end{cases} \quad (1)$$

$C(v,i)$ is set to ∞, when the number of $|e(v,0)|$ replicas placed among the children of v is not sufficient to keep the contributed requests on $a(v,i)$ within W.

Example of Phase 1. Consider the tree in Figure 2 and a processing capacity of inner nodes fixed to $W = 15$. The tree has already been transformed. So nodes x and y are compressed client-leaves (grey scaled in the figure), whereas all other leaves correspond to servers (former inner nodes, hence nodes that are within $\mathcal{N}$). We start with the computation of all $C(v,i)$-values of all leaves. Leaf l for example has $C(l,0) = 3$ as it holds 3 requests. As the link from l to e has a bandwidth of 4, and the QoS is 2, the requests of l can ascent to node e and hence the contribution of l's requests on node e, $C(l,1)$, is 3. In the same manner, $C(l,2)$, i.e. the contribution of l's requests on node b is 3 as well. But then the QoS range is exceeded and hence the requests of l can not be treated higher in the tree. Consequently the contributions on nodes a and a^+ ($C(l,3)$ and $C(l,4)$) are set to infinity.

Table 2 is used for the computation of e, m and C values of inner nodes. During the computation process it is filled by main columns, where one main column consists of all inner nodes of the same level in the tree. So we start with node e. The contribution of its child l, $C(l,1)$, is 3. As it is the only

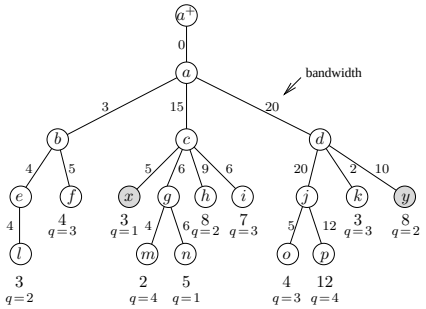


Figure 2. Example

Table 1. Computation of $C(v,i)$-values of leaves.

	l	*f*	*x*	*m*	*n*	*h*	*i*	*o*	*p*	*k*	*y*
$C(v,0)$	3	4	3	2	5	8	7	4	12	3	8
$C(v,1)$	3	4	3	2	5	8	∞	4	12	∞	8
$C(v,2)$	3	∞	∞	2	∞	8	∞	4	12	∞	8
$C(v,3)$	∞	∞	∞	2	∞	∞	∞	4	12	∞	∞
$C(v,4)$	∞			∞	∞			∞	∞		

child, we have that the contributed requests on e are less than the processing capacity $W = 15$ and hence we do not need to place a replica on its child l. Corresponding we get $m(t(e)) = 0$ and a contribution $C(e,0) = 3$. $e(e,1)$ and $C(e,1)$ are computed in the same manner, taking into account $C(l,2)$. Computing $e(e,2)$, i.e. the nodes that have to be equipped with a replica if we want to minimize the contribution on node $a(e,2) = a$ by placing replicas on the children of e but none on e up to a. For this purpose we use $C(l,3)$, the contribution of l on a and remark that it is infinity. Hence we have to equip l with a replica, and as now the set $e(e,2)$ has a higher cardinality than $e(e,0)$, we know that this solution is not optimal anymore and we set the contribution of $C(e,2)$ to infinity (Eq. 1). Taking a look at node j: In the computation of $e(j,0)$, we have a total contribution of its children of 16, which exceeds the processing power of $W = 15$ (bandwidth and QoS are not restricting here). Indeed we have to equip one of the children with a replica, and we choose the one with the highest contribution on j: node p. Consequently, we get $m(t(j)) = 1$ as we have to place one replica on the children. The contribution $C(j,0)$ consists in the 4 remaining contributed requests of node o. Once we have finished all computations for this level, we start with the computations of the next level, which can be found in the next main column of the table.

Table 2. Computation of e, m and C for internal nodes.

	e	*g*	*j*	*b*	*c*	*d*	*a*	a^+
$e(v,0)$	$\emptyset$	$\emptyset$	$\{p\}$	$\emptyset$	$\{g,i\}$	$\{k\}$	$\{b,c\}$	$\{a\}$
$m(t(v))$	0	0	1	0	2	2	6	7
$C(v,0)$	3	7	4	9	11	12	12	∞
$e(v,1)$	$\emptyset$	$\{n\}$	$\{p\}$	$\{e\}$	$\{g,i\}$	$\{k\}$	$\{b,c,d\}$	
$C(v,1)$	3	∞	4	∞	∞	12	∞	
$e(v,2)$	$\{l\}$	$\{n\}$	$\{p\}$	$\{e,f\}$	$\{g,i\}$	$\{j,k\}$		
$C(v,1)$	∞	∞	4	∞	∞	∞		
$e(v,3)$	$\{l\}$	$\{m,n\}$	$\{o,p\}$					
$C(v,1)$	∞	∞	∞					

3.3 Phase 2: Top down replica placement

The second phase uses the precomputed results of the first phase to decide about the nodes on which to place a replica. The goal is to place $m(T^*) = m(t(r^+))$ replicas in $t'(r^+)$. Note that this means that there is no replica on r^+ and hence only the original tree T will be equipped with replicas. If the number of contributed requests on node r is within W, we have a feasible solution.

Phase 2 is a recursive approach. Starting with $i = 0$ on node $v = r^+$, all nodes that are within $e(v,i)$ are equipped with a replica. In this top down approach, i indicates the distance of node v to its first ancestor up in the tree that is equipped with a replica and hence the set $e(v,i)$ denotes the set of children of v that have to be equipped with a replica in order to minimize the contribution of v on $a(v,i)$. Next the procedure is called recursively with the appropriate index i. Algorithm 1 gives the pseudo-code for the top down placement phase, which is the same as the one in [6].

Algorithm 1 Top down replica placement

```
procedure Place-replica (v, i)
if v ∈ C then
  return
place a replica at each node of e(v,i)
for all c ∈ children(v) do
  if c ∈ e(v,i) then
    Place-replica(c,0)
  else
    Place-replica(c,i+1)
```

3.4 Complexity and Optimality

Due to lack of space, we discuss only sketchy complexity and optimality. A detailed disquisition with proofs can be found in our research report [7]. We state a total complexity of $LN \log N$, where N is the number of nodes in the tree and L the maximum range limit among all nodes. Optimality is subject of the following theorem:

Theorem 1. *Algorithm ORP returns an optimal solution to the* REPLICA PLACEMENT *problem with fixed W, QoS and bandwidth constraints, if there exists a solution.*

To prove optimality we perform an induction over levels, where we transform an optimal solution R_0 in the solution found by Algorithm *ORP*. We consider any tree T^* of hight $n + 1$ and start at level 0, which consists in the artificial root r^+. At each step i of the induction we change the placement of the replicas in the i-th level of the solution R_i such that the new placement corresponds to the solution of *ORP*. We prove then that this new solution R_{i+1} is still optimal.

4. Related work

Many authors deal with the REPLICA PLACEMENT optimization problem. Most of the papers neither deal with QoS nor with bandwidth constraints. Instead they consider average system performance as total communication cost or total accessing cost. Please refer to [2]for a detailed description of related work with no QoS constraints.

Cidon et al [3]studied an instance of REPLICA PLACEMENT with multiple objects, where all requests of a client are served by the closest replica (*Closest* policy). In this work, the objective function integrates a communication cost, which can be seen as a substitute for QoS. Thus, they minimize the average communication cost for all the clients rather than ensuring a given QoS for each client. They target fully homogeneous platforms since there are no server capacity constraints in their approach. A similar instance of the problem has been studied by Lin et al [6], adding a QoS in terms of a range limit, and whose objective is to minimize the number of replicas. In this latter approach, the servers are homogeneous, and their capacity is bounded. Both [3],[6]use a dynamic programming algorithm to find the optimal solution.

Some of the first authors to introduce actual QoS constraints in the problem were Tang and Xu [8]. In their approach, the QoS corresponds to the latency requirements of each client. Different access policies are considered. First, a replica-aware policy in a general graph with heterogeneous nodes is proven to be NP-complete. When the clients do not know where the replicas are (replica-blind policy), the graph is simplified to a tree (fixed routing scheme) with the

Closest policy, and in this case again it is possible to find an optimal dynamic programming algorithm.

Bandwidth limitations are taken into account when Karlsson et al [5],[4]compare different objective functions and several heuristics to solve NP-complete problem instances. They do not take QoS constraints into account, but instead integrate a communication cost in the objective function as was done in [3]. Integrating the communication cost into the objective function can be viewed as a Lagrangian relaxation of QoS constraints. Please refer to [1]for more related work dealing with QoS constraints.

5. Conclusion

In this paper we dealt with the REPLICA PLACEMENT optimization problem with QoS and bandwidth constraints. We restricted our research on *Closest*/Homogeneous instances. We were able to prove polynomiality and proposed the optimal algorithm *ORP*. This algorithm extends an existing algorithm in two important areas. First the set of clients and the set of servers can be distinct now and does not require exclusively double-functionality nodes anymore. The other contribution is the expansion to the interplay of different nature constraints. QoS, which is a proper constraint for each client, and bandwidth, a global resource limitation, subordinate to a common optimization function. This accomplishment completes furthermore the study on complexity of *Closest*/Homogeneous in tree networks.

References

[1] A. Benoit, V. Rehn, and Y. Robert. Impact of QoS on Replica Placement in Tree Networks. Research Report 2006-48, LIP, ENS Lyon, France, Dec. 2006. To appear in ICCS'2007.

[2] A. Benoit, V. Rehn, and Y. Robert. Strategies for Replica Placement in Tree Networks. In *HCW'2007*. IEEE Computer Society Press, 2007.

[3] I. Cidon, S. Kutten, and R. Soffer. Optimal allocation of electronic content. *Computer Networks*, 40(2):205–218, 2002.

[4] M. Karlsson and C. Karamanolis. Choosing Replica Placement Heuristics for Wide-Area Systems. In *ICDCS'04*, pages 350–359, Washington, DC, USA, 2004. IEEE Computer Society.

[5] M. Karlsson, C. Karamanolis, and M. Mahalingam. A Framework for Evaluating Replica Placement Algorithms. Research Report HPL-2002-219, HP Laboratories, Palo Alto, CA, 2002.

[6] P. Liu, Y.-F. Lin, and J.-J. Wu. Optimal placement of replicas in data grid environments with locality assurance. In *ICPADS*. IEEE Computer Society Press, 2006.

[7] V. Rehn. Optimal Closest Policy with QoS and Bandwidth Constraints for Placing Replicas in Tree Networks. Research Report 2007-10, LIP, ENS Lyon, France, Mar. 2007.

[8] X. Tang and J. Xu. QoS-Aware Replica Placement for Content Distribution. *IEEE Transactions on Parallel and Distributed Systems*, 16(10):921–932, 2005.