Sustainable Computing: Informatics and Systems 5 (2015) 41–53

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Power-aware replica placement in tree networks with multiple servers per client

Guillaume Aupy[a,*], Anne Benoit[a], Matthieu Journault[b], Yves Robert[a,c]

[a] LIP, École Normale Supérieure de Lyon, CNRS, Université de Lyon & INRIA, France
[b] École Normale Supérieure de Cachan, France
[c] University of Tennessee, Knoxville, USA

ARTICLE INFO

Article history:
Received 1 March 2014
Received in revised form 30 June 2014
Accepted 12 August 2014

Keywords:
Replica placement
Tree network
Multiple servers
Power-aware algorithm
Complexity
Discrete speeds

ABSTRACT

In this paper, we revisit the well-studied problem of replica placement in tree networks. Rather than minimizing the number of servers needed to serve all client requests, we aim at minimizing the total power consumed by these servers. In addition, we use the most general (and powerful) server assignment policy, where the requests of a client can be served by multiple servers located in the (unique) path from this client to the root of the tree. We consider multi-modal servers that can operate at a set of discrete speeds, using the dynamic voltage and frequency scaling (DVFS) technique. The optimization problem is to determine an optimal location of the servers in the tree, as well as the speed at which each server is operated. A major result is the NP-completeness of this problem, to be contrasted with the minimization of the number of servers, which has polynomial complexity. Another important contribution is the formulation of a Mixed Integer Linear Program (MILP) for the problem, together with the design of several polynomial-time heuristics. We assess the efficiency of these heuristics by simulation. For mid-size instances (up to 30 nodes in the tree), we evaluate their absolute performance by comparison with the optimal solution (obtained via the MILP). The most efficient heuristics provide satisfactory results, within 20% of the optimal solution.

1. Introduction

In this paper, we revisit the well-studied problem of replica placement in tree networks. Replica placement in tree networks is an important problem [1–3], with a broad spectrum of applications, such as electronic, ISP, or VOD service delivery (see [4,1,5] and additional references in [2]). The problem is the following: one is given a tree-shaped network where clients are periodically issuing requests to be satisfied by servers. The clients are known (both their position in the tree and their number of requests per time unit), while the number and location of the replicas (also called servers) are to be determined. Clients are leaves of the tree, and requests can be served by one or several internal nodes. Note that the distribution tree (clients and nodes) is fixed in the approach.

Initially, there is no replica; when a node is equipped with a replica, it can process a number of requests, up to its capacity limit. 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. More precisely, there are three classical policies to serve the requests of a client [3]: (i) *Closest*: All requests of a client must be served by the first server located in the path from this client to the root; (ii) *Single*: All requests of a client must be served by a single server, located anywhere in the path from this client to the root; and (iii) *Multiple*: The requests of a client can be served by several servers, all located in the path from this client to the root. For instance in the *Multiple* policy, half the requests of a client can be served by one server, and the other half by another server located higher in this path. In this paper, we study the *Multiple* policy, because it is the most flexible, hence it will lead to the most efficient solution in terms of both the number of servers and total consumed power. The *Multiple* strategy can safely be used in all applications, because in the classical replica placement problem, it is assumed that (i) the servers are identical and capable of serving any client; and (ii) all requests have same size and cost.

* Corresponding author at: Laboratoire LIP, ENS Lyon, 69364 Lyon Cedex 07, France. Tel.: +33 472728352; fax: +33 472728080.
E-mail addresses: Guillaume.Aupy@ens-lyon.fr, guillaume.aupy@ens-lyon.org (G. Aupy).

http://dx.doi.org/10.1016/j.suscom.2014.08.013
2210-5379/© 2014 Elsevier Inc. All rights reserved.

The usual optimization objective in the literature is the number of servers needed to serve all requests. However, minimizing the total power consumed by the servers has recently become a very important objective, both for economic and environmental reasons [6]. To help reduce power dissipation, multi-modal servers are used: each server has a *discrete* number of predefined speeds, which correspond to different voltages that the server can be subjected to. State-of-the-art processors can only be operated with a restricted number of voltage levels, hence with a few speeds [7,8]. The power consumption is the sum of a static part (the cost for a server to be on and operated) and a dynamic part. This dynamic part is a strictly convex function of the server speed, so that the execution of a given amount of work costs more power if a server runs at a higher speed [8]. More precisely, a server operated at speed s dissipates s^3 watts [9–11]. Faster speeds allow servers to handle more requests per time unit, but at the price of a much higher (supra-linear) power consumption.

A major contribution of this paper is to show that minimizing power consumption is an NP-complete problem, even if the servers are already placed in the network (and without static power). This is to be contrasted with the polynomial complexity of minimizing the number or servers [3]. Another major contribution is the design of a set of heuristics to minimize power consumption. These heuristics work in two steps: (i) server placement and (ii) request assignment. The placement step relies on an interesting theoretical result: given a fixed set of servers that should all be used, and assuming continuous speeds, it is possible to optimally assign the requests to these servers in polynomial time. We can therefore easily derive a greedy algorithm to place the servers in the continuous case, because for a given placement, we can directly compute the corresponding optimal power consumption. Of course, assuming continuous speeds is not realistic, but it is a handy simplification of the problem: with continuous speeds, once requests are assigned to servers, each server can operate just at the right speed, namely the sum of its requests, so that selecting the server speeds is immediate. With discrete speeds, the problem is more challenging and may well lead to re-assign the requests, for a given placement of servers. To see this, we start from the solution with continuous speeds (including the greedy placement and the optimal request assignment). Let r be the number of requests processed by a given server N in the solution with continuous speeds. With discrete speeds, we have to use the smallest speed s that is larger than r, thereby losing a lot of power if the difference $s - r$ is large. If it is the case, we can try and re-assign some requests to another server N' located upper in the path from N to the tree root. There would then remain only s' requests to be served by N, where s' is the largest speed that is smaller than r: this saves power locally by avoiding the large $s - r$ gap, but we have to re-assign $r - s'$ requests to another server, and this has a cost that should be balanced with the local gain. Such trade-off decisions are exactly those taken in the request assignment step of the heuristics.

To the best of our knowledge, this paper is the first to propose heuristics for power minimization with multiple servers, hence we cannot use any heuristics from the literature as reference. However, we have derived a Mixed Integer Linear Program (MILP) to compute the optimal solution to the power minimization problem. Using this linear program has (potentially) an exponential cost, but it enables us to assess the absolute performance of the heuristics, at least for small-size problems.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 is devoted to a precise statement of the framework. Section 4 assesses the complexity of the power minimization problem, through an intricate NP-completeness proof. This section also provides the MILP to compute the optimal solution. Section 5 introduces several heuristics to solve the problem. The placement step is an incremental greedy procedure, whose evaluation is based on the optimal solution for request assignment with fixed servers, when assuming continuous speeds. Section 6 reports simulation results and comparisons of the heuristics, together with their absolute performance evaluation: the distance to the optimal solution is computed through the linear program for instances with up to 30 servers.

2. Related work

Many papers considering the replica placement problem deal with general graphs, while we focus in this work on tree networks. In the problem with a general graph network, it is already difficult to decide which spanning tree to use, in order to optimize some global objective function. A survey of work targeting performance issues can be found in [12]. Recently, some work start to tackle energy-related problems. For instance, in [13], the authors discuss thermal and power-aware task scheduling and data placement heuristics, in the context of a Hadoop system. All problems are NP-hard, and there is no tree structure but rather a set of racks, and a set of data nodes per rack.

For tree networks, a large effort has been spent to optimize the performance of replica placements, assuming that the spanning tree was given, or that the network had a tree structure initially. Most work has focused on the *Closest* policy, where a client has to be served by the closest server on the path towards the root of the tree, see for instance [1,5]. Kalpakis et al. [4] studied a variant with bi-directional links, and therefore the tree structure may not be respected anymore, and a client may be served by a node that is not its ancestor in the tree. While the problem with a tree structure has polynomial complexity, the bi-directional problem becomes NP-complete.

Following this line of work, we had investigated in our previous work [14] the complexity of the power-aware replica placement problem with the *Closest* policy, and proved that the problem becomes NP-complete when the objective is to minimize the total power consumption. We considered servers with several distinct possible speeds, and a server operating at a given speed consumes a power composed of a static part and a dynamic part proportional to the cube of the speed. We keep the same model in this paper, because it is a classical model extensively used when considering dynamic voltage and frequency scaling (DVFS) technique [9–11,15].

The *Multiple* policy is more flexible than *Closest* because it loosens placement rules: the requests of a client can be processed by several servers located anywhere in the path from the client to the root. Similarly to the *Closest* policy, the problem of minimizing the cost of the replica placement can be solved in polynomial time [3]. However, we are not aware of any other work aiming at optimizing the power consumption on tree networks for this *Multiple* policy.

3. Framework

This section is devoted to a precise statement of the framework. We start with a description of the replica placement problem. Then we detail the power consumption model. Finally, we state the objective function, and the corresponding optimization problems. Notations are summarized in Table 1.

3.1. Replica placement

We consider a distribution tree whose nodes are partitioned into a set of clients $\mathcal{C}$, and a set of nodes, $\mathcal{N}$. The clients are leaf nodes of the tree, while $\mathcal{N}$ is the set of internal nodes. Each client $i \in \mathcal{C}$ (leaf of the tree) is sending r_i requests per time unit to a database object. Internal nodes equipped with a replica (also called *servers*) can process requests from clients in their subtree. If a server $j \in \mathcal{N}$