Computers & Operations Research 75 (2016) 12–27

Contents lists available at ScienceDirect

Computers & Operations Research

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

A branch and price approach for deployment of multi-tier software services in clouds

Anders N. Gullhav*, Bjørn Nygreen

Department of Industrial Economics and Technology Management, The Norwegian University of Science and Technology, NO-7491 Trondheim, Norway

ARTICLE INFO

Article history:
Received 12 May 2015
Received in revised form
11 May 2016
Accepted 12 May 2016
Available online 13 May 2016

Keywords:
Branch and price
Shortest path problem with resource constraints
Multi-tier service
Replication
Cloud computing

ABSTRACT

This paper considers a service deployment problem that combines service placement and replication level decisions in a cloud computing context. The services are composed of multiple components that are to be placed on nodes in the private cloud of the service provider or, if the private cloud has limited capacity, partly in a public cloud. In the service delivery, the provider has to take into account the quality of service guarantees offered to his end-users. To solve the problem, we develop a branch and price algorithm, where the subproblems both are formulated as a linear mixed integer program and a shortest path problem with resource constraints (SPPRC) on a network with a special structure. The SPPRC can be solved by an exact label-setting algorithm, but to speed up the solution process, we develop a heuristic label-setting algorithm based on a reduced network and simplified dominance rule. Our results show that using the heuristic subproblem solver is efficient. Furthermore, the branch and price algorithm performs better than a previously developed pre-generation algorithm for the same problem. In addition, we analyze and discuss the differences in solutions that utilize resources in a public cloud to different degrees. By conducting this analysis we are able to identify some essential characteristics of good solutions.

1. Introduction

In this work, we are considering a service deployment problem of a software-as-a-service (SaaS) provider that offers a set of services to his end-users. In the provisioning, the SaaS provider (SP) must scale his services according to the performance and availability guarantees specified in the service level agreements (SLAs) contracts that define the services in terms of functionality and quality. Furthermore, the SP also has to decide where to run the services. A typical objective in this problem is to minimize the cost of provision, while fulfilling the service quality guarantees.

In principle, this decision problem can be solved statically or dynamically. Herein, we consider the demand to vary over time, but within certain periods, the demand is stationary in a stochastic sense. When these periods are sufficiently long and recurring, that is, they might reflect working hours, evenings, etc., it is possible to compute a stationary deployment solution for each period, and apply the appropriate solution when one enters a new period. If the infrastructure running the services is failure-prone, it is necessary to complement the stationary solution with a strategy to return from a failed state back to the stationary solution. This strategy can be based on migrating (by e.g., live migration [7]) or restarting service components on preallocated backup locations, as proposed in [5], or based on activating standby service components [8]. However, in cases where the demand is constantly fluctuating and not in a stationary state, it is necessary to solve the deployment problem dynamically, but such cases are not considered here.

The different SaaS services offered by the SP are represented by multi-tier services, which are services composed of several components collaborating to provide a service to the end-users. An example of a multi-tier service is a three-tier web service composed of a web server tier, an application tier and a database tier. Fig. 1 illustrates the structure of a three-tier web service. Each tier corresponds to a software component, referred to as component throughout this paper, which runs on one or more virtual machines (VMs), which in turn run on physical servers. E.g. in periods of low service demand, the web server component of a service might run on only one VM, while with increased demand, the component is deployed by running multiple identical VMs in order to provide a service in accordance with the SLA. While a component might run on multiple VMs, a VM will only run one component. The considered SP owns and operates a limited set of servers, forming a private cloud, which are capable of running the VMs of the service components. An important operational cost component in a data center is the cost of energy, and a strategy used to minimize the cost of energy usage in the VM scheduling is to turn off servers that are not required with the current demand [16]. For the SP, there might be periods where the service demand is too high to be able to provide the services from the private cloud

* Corresponding author.
E-mail addresses: anders.gullhav@iot.ntnu.no (A.N. Gullhav), bjorn.nygreen@iot.ntnu.no (B. Nygreen).

http://dx.doi.org/10.1016/j.cor.2016.05.007
0305-0548/

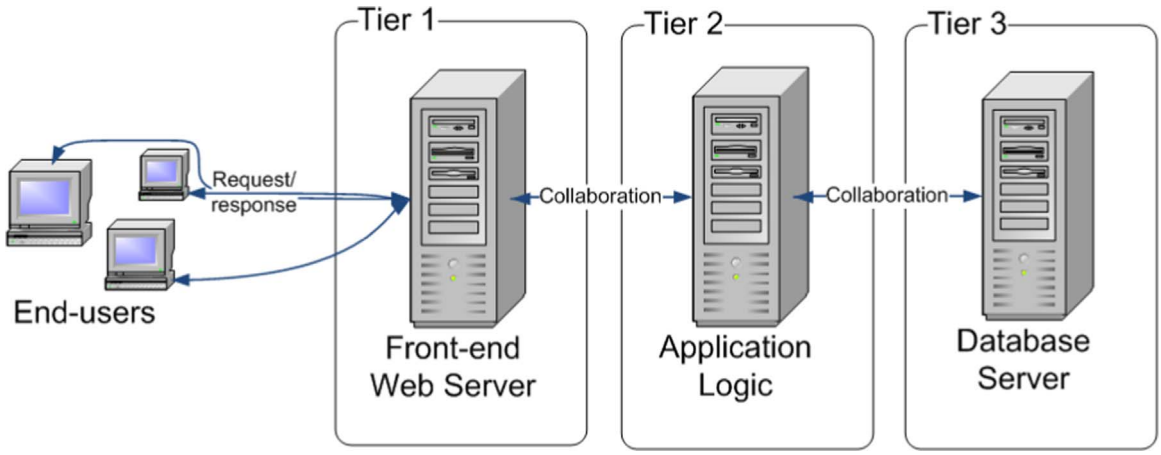

Fig. 1. Illustration of a three-tier web service.

alone. In such cases the SP has the option to lease resources from a public infrastructure-as-a-service (IaaS) provider (e.g. Amazon [1]), denoted a public cloud provider. When the infrastructure used by the SP to provide his services is composed of both a private and a public cloud, this infrastructure is referred to as a hybrid cloud [22]. In cases where the SP maintains a private infrastructure, it is often desirable to fully utilize this infrastructure before leasing capacity from an IaaS provider.

The private and public clouds typically consist of a large amount of cheap, off-the-shelf hardware, which make the services prone to failures, and hence, make fault tolerance an important consideration in the deployment of cloud services. When a VM fails, one has to provide a new VM to maintain service. A technique to improve the fault tolerance is standby redundancy, that is, the principle of allocating standby resources that can be activated in case of a failure. The use of standby redundancy reduces the time from a failure until the moment the new VM is up and running. This time is commonly denoted failover time. Undheim et al. [28] present different ways to implement standby redundancy in a cloud context. Moreover, software systems and conceptual frameworks that employ standby redundancy by running passive standby VMs on the infrastructure are proposed by Cully et al. [8] and Distler et al. [9].

In [12], we present a novel optimization model of the service deployment problem of the SP that includes decisions both related to the replication of the components of multi-tier services and related to the placement of the replicas of the components. In the model, each component of a service could be replicated into a number of load-balanced replicas, referred to as *active replicas*, and in addition, *passive replicas* are used to improve the fault tolerance of the component. However, since we are interested in the performance and fault tolerance of the whole service, not only the components, the selection of replication levels of the different components of a service is linked. In cases where different services interact through their placement (e.g., by running on the same servers), the most cost-efficient way to replicate the components of a given service is dependent of the replication of the components of other services. This is reflected in the model. The replication level decisions, i.e., deciding the number of active and passive replicas of each component of each service, are somewhat similar to the decisions of an optimization problem referred to as the redundancy allocation problem [20]. This problem consists of allocating parallel replicas to different subsystems in series so that the reliability is over a given threshold, while minimizing the cost.

The service deployment problem was modeled as a linear mixed-integer program (MIP), and solved using a commercial MIP solver in [12]. We also reformulated the problem and obtained a pattern-based formulation. The linear relaxation of the reformulation was shown to be much stronger than that of the former, *direct MIP formulation*. Nevertheless, the number of variables in the reformulation grew exponentially with the size of the problem, and we could only optimize over a small subset of the variables. Since we seek to find a stationary solution, we argue that one can spend some time searching for a near-optimal or optimal solution. If the solution quality is of more importance than the time to find a solution, we suggest using an exact solution method. Here, we propose a branch and price (B&P) algorithm, in which the master problem is based on the mentioned pattern-based formulation. The subproblem of the B&P is solved using a MIP solver and a label-setting algorithm. The latter seeks to find the shortest path in a network, which to our knowledge has a novel structure. While developing an exact label-setting algorithm, we observed that this algorithm has deficiencies related to its dominance rule.

A contribution of this paper is an efficient heuristic label-setting algorithm based on a reduced network and simplified dominance rule. Using this heuristic in conjunction with an exact MIP solver speeds up the B&P algorithm. However, in some nodes of the enumeration tree, no improving columns can be found, and hence, using the heuristic algorithm is ineffective. Another contribution of this paper is a simple rule to decide whether the heuristic algorithm should be used in a node, or the exact MIP solver should be called directly. A major question we seek to answer in our computational study is by which methods the subproblem should be solved. The paper also provides a discussion on how the size of the private cloud, relative to the service's resource requirements, affects the solution structure.

The outline of the paper is as follows. Next, in Section 2, we present some works related to the service deployment problem, and in Section 3, we describe the problem in more detail. Two variants of the problem are formulated in Section 4, before the B&P algorithm is explained in Section 5. More details of the algorithm are found in Appendix A. The numerical results of our experiments with the algorithm, along with a discussion of the results, are presented in Section 6. Finally, we conclude the paper in Section 7.

2. Related work

As stated in the introduction, the part of the problem that regards the replication level decisions is related to the redundancy allocation problem, which has applications in many areas including in the design of computer systems. Ashrafi et al. [3] present optimization models with the goal to optimize the reliability of a software system. More recently, Meedeniya et al. [21] use multi-objective optimization to explore the trade-off between reliability and energy consumption when building redundancy into an embedded system. While the