Computers & Operations Research 69 (2016) 48–55

Contents lists available at ScienceDirect

Computers & Operations Research

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

Application placement and backup service in computer clustering in Software as a Service (SaaS) networks

Ali Amiri

Department of MSIS, College of Business, Oklahoma State University, Stillwater, OK 74078, USA

ARTICLE INFO

Available online 23 December 2015

Keywords:
Application placement
Backup service
Software as a service
Cloud computing
Integer programming

ABSTRACT

This paper studies the reliable application placement problem encountered in computer clustering in Software as a Service (SaaS) networks. The problem involves deciding which software applications to install on each computer cluster of the provider and how to assign customers to the clusters in order to provide primary and backup service to customers in case of a cluster failure, while minimizing total cost. Given the complexity of the reliable application placement problem, we propose two algorithms to solve it. The first one is a probabilistic greedy algorithm and the second one is based on a reformulation of the problem where each cluster is to be assigned an application configuration from among all possible configurations or from a properly generated subset of configurations. Results of an extensive computational study show that the two algorithms are more effective than a standard branch-and-bound procedure based on the linear programming relaxation of the problem in solving problem instances with large sizes.

1. Introduction

One major service provided by cloud computing is Software as a Service (SaaS) [3,8,10]. SaaS is associated with business software applications which are hosted by providers and accessed by users over the web/Internet [6]. SaaS offers a variety of potential advantages, including reduced time to start using the software by the user, lower initial and maintenance costs per user as these costs are shared by all users of the software, scalability and integration of SaaS platform, and fast access of users to upgrades of the software at no direct cost and effort associated with upgrades to them [6,8]. The success of a SaaS provider depends heavily on the proper design and configuration of its computing system. The need for backup service should be considered since it is impossible to design a reliable system that provides only primary service to users. This paper studies the reliable application placement problem (RAPP) encountered in computer clustering in SaaS networks [2]. The problem involves deciding which software applications (or applications for short) to install on each computer cluster of the provider and how to assign customers to the clusters in order to provide primary and backup service to customers in case of a cluster failure, while minimizing total cost.

More specifically, the input of RAPP is the set of customers, the set of applications requested by each customer, the resource requirements (i.e., demand in terms of transactions/second) associated with each application requested by each customer for both primary and secondary (i.e., backup) service, and the available capacities of the computer clusters. The goal is to assign each customer to two distinct clusters and to decide which applications to install on each cluster such that if a customer is assigned to a particular cluster, then all applications requested by the customer should be installed on that cluster. The objective is to minimize the total cost of installing, running, and maintaining the applications on the clusters. The two distinct clusters that a customer is assigned to are called the primary and secondary or backup clusters. Primary clusters alone are not sufficient to guarantee service availability to all customers without interruption because if a customer is assigned to only a primary cluster it will be denied service if the connection between the customer and the cluster fails or the cluster itself fails. The level of backup service can be set as part of the agreement between the customer and the provider. This level indicates the percentage of the customer demand which will be handled by the secondary cluster in case of a failure of the primary cluster. A 100% backup level ensures that the entire customer demand will be served by the secondary cluster in case of a failure of the primary cluster.

While some methods for planning a SaaS network without back-up service have been developed, the search continues for more realistic models and practical solution methods. The goal of this study is to make a contribution in this area by presenting a

E-mail address: amiri@okstate.edu

http://dx.doi.org/10.1016/j.cor.2015.12.003
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

more realistic model for the reliable application placement problem and developing effective heuristic solution procedures for the model.

As mentioned by Gaast et al. [2], the simple application placement problem (APP) (i.e., backup service is not considered in this problem) is closely related to several classical problems in the areas of Operations Research and Computer Science. In particular, APP is related to the multiproduct capacitated facility location problem (MPCFL) which consists of deciding the locations of facilities to open, the products to manufacture at each open facility, and the allocation of customers to the facilities. This problem has been studied extensively [11]. Two key differences between APP and MPCFL are: (i) a customer in APP is assigned to exactly one cluster; whereas, a customer in MPCFL can be assigned to more than one facility; and (ii) the capacity of a facility is defined for each product manufactured at the facility; whereas, the capacity of a cluster in APP is defined for the entire cluster as the maximum amount of demand (e.g., transactions per second) of all customers assigned to the cluster for the applications installed on the cluster.

Another problem closely related to our RAPP is the problem encountered by a platform provider to maximize the number of applications requested by application providers (i.e., customers) that can be placed on a set of servers with limited capacities (max-APP) [9,12]. Each application can include several components that can be placed on different servers. Components cannot be shared among applications and components of the same application can be installed on different servers. The number of applications that can be placed on the servers while satisfying their resource requirements [12] is used as a measure of the revenue that the platform provider can generate from the hosted applications. A key difference between problem RAPP and the other problem max-APP is that the goal in RAPP is to minimize the cost of the applications to be installed on the servers; whereas the goal in max-APP is to maximize the number of applications to be installed on the servers as a proxy for revenue to be generated.

More recently, Gaast et al. [2] studied an extended version of APP where the number of computer clusters and their capacities are decisions to be made. The goal is to minimize total cost made of cost of clusters to open and cost of applications installed on the opened clusters. System reliability to provide backup service is not considered in this version. The authors developed a Tabu Search heuristic to solve the problem. The authors [2] conducted computational experiments using small problem instances with up to 60 customers. In our current study, we assume that the number of computer clusters and their capacities are given. Hence, the decisions to be made in the RAPP in the current study are the assignment of applications to clusters and the allocation of customers to clusters for both primary and backup services.

Given the complexity of the reliable application placement problem, we propose two algorithms to solve it. The first one is a probabilistic greedy algorithm which includes randomization and perturbation features to avoid getting trapped in a local optimum. The second algorithm is based on a reformulation of the problem where each cluster is to be assigned an application configuration from among all possible configurations or from a properly generated subset of configurations. We conducted an extensive computational study using large data sets with up to 180 customers and 50 applications. The results show that both algorithms outperform a standard branch-and-bound procedure based on the linear programming relaxation of the problem for problem instances with large sizes. The probabilistic greedy algorithm is shown to be the most efficient in solving the problem.

2. Integer linear programming formulations

The following notation is used:

N	set of customers
M	set of clusters
A	set of applications
A_i	set of applications needed by customer $i \in N$
d^p_{ik}	demand for primary service for application $k \in A$ by customer $i \in N$
d^s_{ik}	demand for backup/secondary service for application $k \in A$ by customer $i \in N$
Q_j	capacity of cluster $j \in M$
C_k	cost of installing and running application $k \in A$ on a cluster

The decision variables can be defined as follows:

$$X^p_{ij} = \begin{cases} 1 \text{ if customer } i \in N \text{ is assigned to cluster } j \in M \text{ for primary service} \\ 0 \text{ otherwise} \end{cases}$$

$$X^s_{ij} = \begin{cases} 1 \text{ if customer} i \in N \text{ is assigned to cluster } j \in M \text{ for secondary service} \\ 0 \text{ otherwise} \end{cases}$$

$$Y_{kj} = \begin{cases} 1 \text{ if application} k \in A \text{ is installed on cluster} j \in M \\ 0 \text{ otherwise} \end{cases}$$

Based on the above notation, the problem can be formulated as follows:

$RAPP_1$:

$$\min \sum_{k \in A} \sum_{j \in M} C_k Y_{kj} \tag{1}$$

s.t.

$$\sum_{j \in M} X^p_{ij} = 1 \quad \forall\ i \in N \tag{2}$$

$$\sum_{j \in M} X^s_{ij} = 1 \quad \forall\ i \in N \tag{3}$$

$$X^p_{ij} + X^s_{ij} \le Y_{kj} \quad \forall i \in N,\ k \in A_i\ ,\ j \in M \tag{4}$$

$$\sum_{i \in N} \sum_{k \in A_i} (d^p_{ik} X^p_{ij} + d^s_{ik} X^s_{ij}) \le\ Q_j \quad \forall j \in M \tag{5}$$

$$X^p_{ik},\ X^s_{ik}, Y_{kj} \in \{0, 1\} \quad \forall i \in N,\ k \in A,\ j \in M \tag{6}$$

The objective function (1) minimizes the total cost of installing and running the applications on the clusters. Constraints (2) and (3) ensure that each customer is assigned to exactly two clusters for primary and secondary services, respectively. Constraints (4) ensure that a cluster cannot be used to process both the primary and secondary demands of the same customer. In addition, constraints (4) require that if a customer is assigned to a cluster for primary or secondary service, then all the applications needed by the customer are installed on the cluster. Constraints (5) ensure that the total demand of the customers assigned to a cluster cannot exceed the capacity of the cluster. Constraints (6) enforce integrality restrictions on the decision variables.

This model is a large 0–1 integer programming problem. An instance of the model with 30 customers, 20 applications, and 10 clusters will have 800 binary variables and 3070 constraints (assuming that each customer requires 10 applications). Furthermore, the problem can be shown to be NP-hard [2]. Therefore, it very unlikely that general purpose integer programming packages can solve problem instances of realistic size in reasonable computing times. Next, a new formulation of the reliable application