Sustainable Computing: Informatics and Systems 9 (2016) 20–32

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

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

A packing problem approach to energy-aware load distribution in Clouds

Thomas Carli [a], Stéphane Henriot [a,b,1], Johanne Cohen [c], Joanna Tomasik [a,*]

[a] SUPELEC, Computer Science Dpt., 91192 Gif-sur-Yvette, France
[b] INSA-Rouen, 76800 St.-Etienne-du-Rouvray, France
[c] LRI, University Paris Sud, 91190 Gif-sur-Yvette, France

ARTICLE INFO

Article history:
Received 21 September 2014
Received in revised form 28 April 2015
Accepted 6 August 2015
Available online 15 August 2015

Keywords:
Service virtualization
Cloud computing
Energy consumption minimization
Packing problems
Problem approximability
Approximation algorithm

ABSTRACT

The Cloud Computing paradigm consists in providing customers with virtual services of the quality which meets customers' requirements. The efficiency of infrastructure exploitation may be expressed by the electrical energy consumption of computing centers, amongst others.

We propose to model the energy consumption of private Clouds by a variant of the Bin Packing problem which we analyze next from a theoretical point of view. We advance on-line and off-line approximation algorithms to solve our problem to balance the load either on-the-fly or at the planning stage. In addition to the computation of the approximation factors of these two algorithms, we evaluate their performance experimentally. The quality of the results is encouraging, which makes a packing approach a serious candidate to model energy-aware load balancing in Cloud Computing.

1. Introduction

The Cloud Computing paradigm consists in providing customers with virtual services of the quality which meets customers' requirements. A cloud service operator is interested in using his infrastructure in the most efficient way while serving customers. Namely, he wishes to diminish the environmental impact of his activities by reducing the amount of energy consumed in his computing servers. Such an attitude allows him to lower his operational cost (electricity bill, carbon footprint tax, etc.) as well.

Three elements are crucial in the energy consumption on a Cloud platform: computation (processing), storage, and network infrastructure [1–4]. Our long-term goal is to study different techniques to reduce the energy consumption regarding these three elements. This document describes our attempt to address the first two by consolidating applications on servers to keep their utilization at a hundred per cent. The use of a private Cloud we assume here gives a complete control over servers and it assures a fixed network topology and bandwidth.

The consolidation problem was discussed in [5] through an experimental approach based on intuition as its authors did not propose any formal problem definition. Another attempt to task consolidation through a "partition-like" method is described in [6]. That work resulted in an on-line implementation of task scheduling, EAGLE. Its evaluation through simulation exhibited a considerable reduction of the energy consumption obtained by lowering the number of servers running. The consolidation paradigm was also used in [7] which works with the energy consumption described as a piecewise linear function of CPU utilization. We also base ourselves on the principle that the reduction of the energy expenses may be reached by cutting down on the number of servers activated.

In this paper we address the challenge of the minimization of energy required for processing by means of proper mathematical modeling and we propose algorithmic solutions to minimize the energy consumption on Cloud Computing platforms. On this stage we are interested in selecting servers according to their power requirements. We remind that we address here a private Cloud infrastructure which operates with knowledge of resource availability.

* Corresponding author. Tel.: +33 01 69 85 14 79; fax: +33 01 69 85 14 99.
E-mail addresses: Thomas.Carli@supelec.fr (T. Carli), Stephane.Henriot@ens-cachan.fr (S. Henriot), Johanne.Cohen@lri.fr (J. Cohen), Joanna.Tomasik@supelec.fr (J. Tomasik).

[1] Stéphane Henriot participation in this work was partially financed by the grant 2013-22 of PRES UniverSud Paris.

http://dx.doi.org/10.1016/j.suscom.2015.08.001
2210-5379/

We study a theoretical problem adjacent to the minimization of energy required to execute computational tasks. Our working hypotheses are as follows:

1 any computational task is parallelizable, i.e. it may be executed on several servers; there is, however, a restriction on the number of servers on which a task can be launched,
2 available servers have different computing capacities,
3 the computation cost of a server in terms of its energy consumption is monotone, i.e. a unity of computation power is cheaper on a voluminous server than on a less capacious one.

We justify the latter by the use a simple but realistic, because validated through experiments, formula to express the server power consumption. This consumption is a sum of a constant part which is due to cooling equipment, network devices, disc systems, and a variable part modeling energy expenses principally due to CPU utilization which depends upon a current load [8–10]. According to the bibliographical sources, the constant part may represent as much as 60% of the energy consumption of an entirely full server [4]. In this study we stay within the general concept, well established in the literature, which separates idle and busy states from the energy consumption perspective [11,5]. We argue that the infrastructure expenses, independent of loading and required to keep a computation unity operational, on the voluminous servers are greater than or equal to those on small ones.

The assumption that all tasks are divisible (point 1 above) may sound unrealistic as in practice some tasks cannot be split arbitrarily or even at all. We make this hypothesis in order to formulate theoretical problems and analyze them. In the real world scenario one will cope with jobs which either cannot be cut at all, which can be cut once, twice, up to D times, or where the number of cuts allowed is less than D because task pieces are not of arbitrary size. Such a situation corresponds to a problem which is "somewhere between" two extreme cases: no jobs can be split or all jobs can be split D times. As the reader will notice going through this paper, the "real life" problem performance bounds can be deduced from those of the extreme problems.

One may find, for instance in [12,13], that Clouds users communicate their requirements for high performance computing or data processing application through tools such as EC2 [14] or hadoop [15]. The overhead due to hypervisor activity is, according to [16] for Xen [17], limited. The dispatching of resources demanded on virtual machines would be done with acceptable delay while keeping in mind that the more tasks run on a server, the greater the time overhead is, as reported in [13].

The three assumptions above lead us to formulate a generalization of the Bin Packing problem [18], which we refer to as the Variable-Sized Bin Packing with Cost and Item Fragmentation Problem (VS-CIF-P) for which the packing costs corresponds to a power required. In the considered case a cost of packing is monotone. This problem models a distribution of computational tasks on Cloud servers which ensures the lowest energy consumption because it expresses the tendency of the task consolidation on servers which require less power. Its definition is given in Section 3.2. We point out that this problem, to the best of our knowledge, has not been treated yet.

Confronted with numerous constraints of the VS-CIF-P we decided to start, however, by studying in Section 3.1 a less constrained problem, without an explicit cost function, the Variable-Sized Bin Packing and Item Fragmentation Problem (VS-IF-P). This problem has not yet been studied either. This gradual approach allows us to deduce several theoretical properties of the VS-IF-P which can be then extended to the principal problem.

In Section 4 we propose customized algorithms to solve the VS-CIF-P. Thinking ahead of their application in the Cloud load planning we already consider two scenarios despite the fact that this study does not cover the temporal issues. Willing to treat users' demands in bulk, what corresponds to regular dispatching of collected jobs (for instance, hourly) we propose an off-line method (Section 4.2). An on-line algorithm, dealing with demands on-the-fly is given in Section 4.3. This treatment allows one to launch priority jobs which have to be processed upon their arrivals. Expecting an important practical potential of the VS-CIF-P we also furnish results concerning the theoretical performance bounds of the algorithms we elaborated.

Despite the fact that the problem is approximate with a constant factor, we go further with the performance evaluation of the algorithms we come up with. The empirical performance evaluation is discussed in Section 5.

The list of our contributions given above also partially constitutes the description of the paper's organization. We complete this description by saying that in Section 2 we present a survey of related works concerning definitions of the family of bin-packing problems together with their known approximation factors. We also give there an outline of algorithmic approaches used to solve packing problems. Our special attention is put on those which inspired us in our study. We point out that the notation used in the article is also introduced in that section while carrying out our survey.

After giving our contributions in the order announced above we draw conclusions and give directions of our further work.

2. Related works

Let L be a list of n items numbered from 1 to n, $L=(s_1, s_2, \ldots, s_n)$, where s_i indicates an item size. Let us also assume for a moment that for all $i=1, 2, \ldots, n$ $s_i \in [0, 1]$. The classical Bin Packing Problem (BPP) consists in grouping the items of L into k disjoint subsets, called bins, $B_1, B_2, \ldots, B_k$, $\bigcup_{l=1}^{m} B_l = L$, such that for any j, $j=1, 2, \ldots, k$, $\sum_{l \in B_j} s_l \leq 1$. The question 'Can I pack all items of L into K, $K \leq k$, bins?' defines the BPP in the decision form. Put differently, we ask whether a *packing* $B_1, B_2, \ldots, B_k$ for which k is less or equal to a given value K exists. The corresponding optimization problem aims to find the minimal k. Due to its numerous practical applications the BPP, which is NP-hard, was studied exhaustively.

Note that in this paper, we consider the efficiency of the on-line and off-line algorithms. On the one hand, an off-line algorithm $\mathcal{A}$ for a problem has an *approximation ratio* of α if, for any instance I, the cost C of the solution produced by the algorithm is within a factor of α of the cost $OPT(I)$ of an optimal solution: $\frac{A(I)}{OPT(I)} \leq \alpha$. On the other hand, the efficiency of an on-line algorithm $\mathcal{A}$ is given by its *competitive ratio*, which measures the cost of $\mathcal{A}$ relative to the optimal cost of a "global" algorithm, which knows the entire sequence in advance. The algorithm's competitive ratio on an instance I is $\frac{A(I)}{OPT(I)}$, where $\mathcal{A}(I)$ is the cost of the solution computed by $\mathcal{A}$ for instance I. Our algorithms are assessed according to one of two ratios depending on the variant chosen (off-line or on-line).

The current basic on-line approaches, Next Fit (NF) and First Fit (FF), give satisfactory results. The asymptotic approximation factor for any on-line algorithm cannot be less than 1.54 [18]. A widely used off-line approach consists in sorting items in decreasing order of their size before packing them. The tight bound for First Fit Decreasing (FFD) is given in [19,20].

2.1. Variable-sized bin packing with cost

In the initial problem the capacity of all bins is unitary. The problem may thus be modified by admitting different bin capacities. A bin can have any of m possible capacities b_j, $B=(b_1, b_2, \ldots, b_m)$. In other words, we have m bin classes.