Contents lists available at ScienceDirect

Applied Soft Computing

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

A low-level resource allocation in an agent-based Cloud Computing platform

Javier Bajo[b], Fernando De la Prieta[a,*], Juan M. Corchado[a], Sara Rodríguez[a]

[a] *Department of Computer Science and Automation Control, University of Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain*[1]
[b] *Department of Artificial Intelligence, Technical University of Madrid Bloque 2, Despacho 2101, Campus Montegancedo, Boadilla del Monte, Madrid 28660, Spain*[2]

ARTICLE INFO

Article history:
Received 18 November 2014
Received in revised form 17 April 2016
Accepted 31 May 2016
Available online 5 August 2016

Keywords:
Allocation computation resources
Load balancing
Multiagent systems
Virtual organizations
Linear programming

ABSTRACT

The distribution of computational resources in a Cloud Computing platform is a complex process with several parameters to consider such as the demand for services, available computational resources and service level agreements with end users. Currently, the state-of-the-art presents centralized approaches derived from previous technologies related to cluster of servers. These approaches allocate computational resources by means of the addition/removal of (physical/virtual) computational nodes. However, virtualization technology currently allows for research into new techniques, which makes it possible to allocate at a lower level. In other words, not only is it possible to add/remove nodes, but also to modify the resources of each virtual machine (*low level resource allocation*). Thus, agent theory is a key technology in this field, allowing decentralized resource allocation. This innovative approach has undeniable improvements such us computational load distribution and reduced computation time. The evaluation was carried out through experiments in a real Cloud environment, thus proving the validity of the proposed approach.

1. Introduction

The technology industry is presently making great strides in the development of the paradigm of *Cloud Computing* (CC) [28]. As a result, the number of closed and open source platforms, both of which have similar architectures, has been rapidly increasing [20,30]. From an external point of view, the three most widely known services are *Software* (*SaaS*), *Platform (PaaS)* and *Infrastructure* (IaaS) [32]. From an internal point of view, the services generally offered are considered elastic [16]; in other words, it is possible to have the same level of quality (response time, quality of results, etc.) regardless of the instant of demand. To do so, the amount of resources allocated to these services needs to be variable.

This new model of production and distribution of services is largely possible due to the high number of underlying technologies (virtualization, server farms, web services, web portals, etc.) which have reached their prime [13]. In fact, the hardware infrastructure of a CC platform is one of the most complex and unpredictable technological systems in existence, housing varied equipment such as servers, network modules, uninterrupted power supply units and a long high performance computer infrastructure. Additionally, these components tend to be highly heterogeneous due to the incessant advances in electronics [39], which give way to a new generation approximately every 12–18 months [6]. In order to work, the heterogeneity and intrinsic complexity of this type of computing environments make them difficult to manage and maintain over time [7].

In order to deal with these issues, this type of environment currently uses virtualization technology [52] to abstract the inherent complexity. Virtualization makes it possible to consider the complexity of the underlying hardware abstractly, in the form of virtual machines. A virtual machine is an abstract computational resource that emulates a physical machine with specific software and hardware characteristics. In other words, it makes it possible to create an abstract or virtual, but homogeneous and controllable view of the real hardware [5].

As stated above, these new capabilities derived from virtualization lead to the birth of a new concept: *Elasticity* [16]. This concept is based on the *just-in-time* production method [25], which references the manner in which the (computational) services and the resources they require are produced. Thus, the services produced within the framework of CC only receive the amount of

* Corresponding author at: Department of Computer Science and Automation Control, Plaza de la Merced s/n, 37008 Salamanca, Spain.
E-mail address: fer@usal.es (F. De la Prieta).
[1] {fer, corchado, srg}@usal.es.
[2] {jbajo}@fi.upm.es.

http://dx.doi.org/10.1016/j.asoc.2016.05.056
1568-4946/© 2016 Elsevier B.V. All rights reserved.

resources they need to maintain the level of quality, which is previously agreed to by contract with end users [49,46]. As a result, the main objective within this context is to maintain the level of Quality of Services (QoS) [2], regardless of demand. This objective implies a need to dynamically manage the computational resources assigned to each service offered to the user. In order to do so, these platforms predominantly use a management system (monitor and control) for the underlying computational infrastructure [28].

In this sense, virtualization technology has recently begun to advance at great speeds, allowing the physical characteristics of the virtual machines (memory, CPU, storage space, etc.) to be changed dynamically, even when the machine is running [15]. Nevertheless, existing models have not yet incorporated this new capability offered by the technology [29,38]. These models use a classic approach in which elasticity is based on modifying the number of nodes that attend to a particular service. The hypothesis of the present research work is based on the premise that since technology is able to offer new capabilities, it is necessary to design new resource allocation models that take these new characteristics of the technology into account. In this context, the *Theory of Agents* and *Multiagent Systems* (MAS) [50] can provide a new route for managing CC systems based on the distribution of responsibilities, flexibility and autonomy. Managing the functions of the nucleus of a CC system through an agent-based model allows the resulting platforms to be much more efficient, scalable and adaptable than they currently are [43,41].

This work proposes a low-level resource distribution model in which a level of resources is associated with each individual virtual machine. This calculation model is based on agent technology and, as such, is a distributed model, thus making it possible to distribute computational resources throughout the entire CC environment and allow the distribution of its complexity and associated computational costs. This new approach in monitoring and, in particular, controlling the CC system, makes it possible to incorporate the new characteristics, as previously mentioned, that virtualization has to offer.

This document is organized as follows: the following section provides a detailed description of the problem, Section 3 presents the current literature related to the problem. Then, Section 4 proposes a solution based on multiagent systems, while the evaluation and validation of these systems are presented in Section 5. Finally, the last section presents the conclusions of the research.

2. Problem statement

2.1. Cloud environment overview

Given the complexity of the environment, as well as the different artificial and human components involved in this context, it is necessary to define how the services will be offered at a technical level. To do so, a CC is presented in Fig. 1, in which each software service for the platform, at the PaaS or SaaS level, can be deployed simultaneously on various virtual machines (computational nodes or *workers*).

Each Physical Resource or server (PR) will usually host a set of Virtual Machines (VM). This will allow every physical server in the system to have an associated matrix at all times with information regarding the state of the real hardware (namely, the physical machine) as well as the different virtual machines that it hosts at any given time. This matrix, shown in Eq. (1), is a representation of the state of each physical server that has been launched and can be used to analyze the allocation of resources to each service and the amount of resources to be lent.

$$e_i = \left\{ \begin{array}{ccc} & \mathrm{PR}^{\mathrm{e}} & \\ \mathrm{VM}_1 & \ldots & \mathrm{VR}_\mathrm{m} \end{array} \right\} where$$
$$PR^e = \left\{ hostname, IP, mac, state, M_{\max}, vcpu_{max}, M_{min}, vcpu_{min}, benchmark \right\}$$
$$VM_i = \left\{ IP, state, M, vcpu, M_i, \overline{p_{cpu}} \right\} \tag{1}$$

The intrinsic characteristics of the service are therefore determined by the template, which is used to instantiate a virtual machine that offers a concrete service and contains the software and the minimum hardware characteristics of each node associated to the concrete service. Once the virtual node has been instantiated or created using the template, it is possible to modify, in execution time, the resources associated with a specific service. Consequently, every template for a virtualized node associated to any service k (VM_t^k), will be described through a set of properties as seen in Eq. (2): identifier, minimum assignable memory, minimum number of assignable CPUs, type (hardware or software service) and state, which determine whether it is balanceable:

$$VM_t^k = \left\{ ID^k, M_{\mathrm{min}}, vcpu_{min}, type, state \right\} \tag{2}$$

This deployment model of the hardware infrastructure, which is based on two levels of abstraction (real and virtual), makes it possible to provide any type of computer services. While the user will consume these services, it will be necessary to have previously reached an agreement about the QoS, through a Service Level Agreement (SLA). Formally, as shown in Eq. (3), the SLA is formalized for any given user j ($ServA^j$) as the combined set of user agreements established for each service i on an individual basis (SLA_i^j) [35,19].

$$\mathrm{ServA}^{\mathrm{j}} = \cup_{i=o}^{i=m} \mathrm{SLA}_{\mathrm{i}}^{\mathrm{j}} \tag{3}$$

Using this simple expression, the goal for achieving an adequate model of the context consists of measuring the quality of services offered. Various related studies can be found in the current state of the art [2,21,15]. Within the scope of this work, the metrics that are directly dependent on the underlying computational resources will be used by the service, and the response time for each request will be selected. In other words, for a service k with a set of methods (r_i^k) which make up the service API, the quality of each request that forms the service API is determined by Eq. (4) for the size of the response (s_i^k) and the corresponding transmission time (t_i^k):

$$\mathrm{QoS}^{\mathrm{k}} = \left\{ \overline{\mathrm{r}_1^{\mathrm{k}}} \cdots \overline{\mathrm{r}_{\mathrm{i}}^{\mathrm{k}}} \cdots \overline{\mathrm{r}_{\mathrm{n}}^{\mathrm{k}}} \right\} \mathrm{donde}\, \overline{\mathrm{r}_{\mathrm{i}}^{\mathrm{k}}} = \frac{1}{\mathrm{m}} \sum_{i=1}^{m} \mathrm{s}_{\mathrm{i}}^{\mathrm{k}} / \mathrm{t}_{\mathrm{i}}^{\mathrm{k}} \tag{4}$$

Using this very characteristic deployment model, the control system in a CC environment should vary the computational resources assigned to each service according to the demand that exists at any given time, making it possible to maintain the QoS levels in each of those services. In this regard, the greatest advantage of virtualization is that the assignment of resources at any instance of execution can be reconfigured dynamically, which makes it possible to elastically modify the amount of resources associated with each service in execution time. In terms of requests for a specific service, the demand is balanced among the different virtual machines that are associated to the service.

The distribution of resources can be presented from different points of view according to the capabilities permitted by the underlying technology. First, *Distribution at the service level* requires a simple balance of the workload among the worker nodes that are offering a specific service. This balancing has been highly extended from the birth of cluster computational systems in HPC (*High performance Computing*) environments [1,10,44]. Second, *Distribution at the infrastructure level* is a more complex and novel method which