HAL
open science

An Intelligent Approach to Allocating Resources within an Agent-Based Cloud Computing Platform

Fernando de La Prieta, Sara Rodríguez-González, Pablo Chamoso, Yves Demazeau, Juan M. Corchado

To cite this version:

Fernando de La Prieta, Sara Rodríguez-González, Pablo Chamoso, Yves Demazeau, Juan M. Corchado. An Intelligent Approach to Allocating Resources within an Agent-Based Cloud Computing Platform. Applied Sciences, 2020, Multi-Agent Systems 2020, 10 (12), pp.4361. 10.3390/app10124361 . hal-02882975

HAL Id: hal-02882975

https://hal.science/hal-02882975v1

Submitted on 28 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

applied sciences

MDPI

Article

An Intelligent Approach to Allocating Resources within an Agent-Based Cloud Computing Platform

Fernando De la Prieta [1,*], Sara Rodríguez-González [1], Pablo Chamoso [1,2], Yves Demazeau [3] and Juan Manuel Corchado [1,2,4,5]

1 BISITE Research Group, University of Salamanca, 37007 Salamanca, Spain; srg@usal.es (S.R.-G.); chamoso@usal.es (P.C.); corchado@usal.es (J.M.C.)
2 Air Institute, IoT Digital Innovation Hub, 37188 Salamanca, Spain
3 Centre National de la Recherche Scientifique, Laboratoire d'Informatique de Grenoble, Bâtiment IMAG-700 avenue Centrale, Domaine Universitaire—CS 40700, F-38058 Grenoble, France; yves.demazeau@imag.fr
4 Department of Electronics, Information and Communication, Faculty of Engineering, Osaka Institute of Technology, Osaka 535-8585, Japan
5 Pusat Komputeran dan Informatik, Universiti Malaysia Kelantan, Karung Berkunci 36, Pengkaan Chepa, Kota Bharu 16100, Kelantan, Malaysia
* Correspondence: fer@usal.es; Tel.: +34-677-522-678

Received: 29 May 2020; Accepted: 22 June 2020; Published: 25 June 2020

Abstract: The cloud computing paradigm has the ability to adapt to new technologies and provide consistent cloud services. These features have led to the widespread use of the paradigm, making it necessary for the underlying computer infrastructure to cope with the increased demand and the high number of end users. Platforms often use classical mathematical models for this purpose, helping assign computational resources to the services provided to the final user. Although this kind of model is valid and widespread, it can be refined through intelligent techniques. Therefore, this research presents a novel system consisting of a multi-agent system, which integrates a case-based reasoning system. The resulting system dynamically allocates resources within a cloud computing platform. This approach, which is distributed and scalable, can learn from previous experiences and produce better results in each resource allocation. A model of the system has been implemented and tested on a real cloud platform with successful results.

Keywords: cloud computing; multi-agent systems; case-based reasoning; resource allocation

1. Introduction

Cloud computing (CC) has undergone rapid growth since users began to notice its great advantage over traditional IT systems. This is because this paradigm facilitates the development of distributed computing systems, data management and computing resources through a scalable network, data processing centers and web services [1]. Hence, this technology is leading the revolution of distributed computing and has resulted in the rapid growth of private and public platforms [2–5]. There is no doubt that the general social acceptance of this paradigm [6] has largely contributed to its development, due to the economic interests of large technology companies that underline the purely technical aspects [7,8].

The marketing model used in the CC paradigm is also innovative, as it is based on a pay-as-you-go concept [7] for utility services such as water, gas, etc. In the utility computing paradigm [9], users are required to first negotiate and establish a service level agreement (SLA) to have access to the goods [10]. Having established a contract for the computing-based goods, both users (who pay fees on a regular basis) and the CC system (through service maintenance) are obliged to comply with the agreement. This marketing model contractually requires CC platforms to maintain the quality of services (QoS) as

previously agreed upon, which makes it necessary for the internal architecture to monitor and adapt to the demand dynamically. In this respect, the novelty is reflected in the range of innovative underlying technologies, such as web services and microservices, virtualization, dynamic resource allocation or service farms, among others, which have made it possible for services to be offered dynamically, regardless of existing user demand [11–13].

The resources of a CC environment are distributed among all of the services. Each service has specific characteristics and requirements; thus, computational resources are assigned dynamically to each service in accordance with the demand. In order to do so, these platforms use a system for the management (monitoring and control) of the underlying computational infrastructure and for the implementation of algorithms that allow for greater elasticity in the amount of resources dedicated to each service, depending on the instant demand. Therefore, if a specific service has, for example, established a specific set of SLAs with its end-users, the system must be able to satisfy the pre-established agreement regardless of existing demand. That is, it must be able to add new resources as demand increases and, likewise, to remove these resources when the demand decreases. In other words, the main objective of elastic algorithms is to allocate the appropriate amount of computational resources to a specific service according to the instantaneous demand of those services.

At the technological level, a CC platform is mainly based on virtualization technology, which is a software layer that can abstractly visualize the underlying infrastructure as virtual computing nodes. The main advantage of virtualization is that the assignment of computational resources can be dynamically reconfigured, giving rise to an abstract or virtual view, which is homogeneous and can be easily controlled, unlike the real hardware environment [14]. Therefore, given that the services offered by a CC environment are distributed in different computational nodes, the assignment or withdrawal of resources is based on adding or eliminating computational nodes to/from a specific service. In any case, although the process of managing the infrastructure is simplified as a result of virtualization, the new generation of algorithms should not only provide an elastic model (increasing and decreasing) for allocating resources, but it should also reduce operational costs in the CC environment. Current state-of-the-art research is based on methods that use centralized algorithms focused on mathematical and heuristic models [15–17], neither of which ensures the efficiency of the system, or even its availability in the case of a system failure. In general terms, these algorithms add or eliminate the resources assigned to a particular computational service, according to the SLA established with the end user, the consumption of energy in the CC environment, and the actual state (available resources) of the CC environment [18].

Because of these shortcomings derived from centralization (unavailability, bottlenecks, bandwidth consumption, etc.), new techniques need to be explored for the development of CC models with elasticity of services. This research suggests that the application of models based on artificial intelligence (AI) as a CC is distinguished by its large distribution, its heterogeneity and its high level of uncertainty, and it is precisely in this area that the application of AI techniques offers considerable opportunities. The incorporation of proactivity, self-adaptation and learning capacity is the reason for the evolvement of such elastic management algorithms for the allocation of computational resources. In this research, agents and multi-agent systems (MAS) [19] have been selected from among all the available AI techniques due to their distributed nature and ability to adapt to different environments, such as CC systems, which are open systems. MAS are considered particularly effective because they can provide advanced and innovative solutions [20] by exploiting differentiating elements as part of the strategy defined for the provision of customer satisfaction, providing flexibility, capability and response speed [21]. This approach allows existing models to evolve toward a model in which different distributed agents are represented within an uncertain environment, causing them to interact and share information with their peers [22]. This would make it possible for the algorithms that manage the resources to be distributed over the whole system, which would ease their use regardless of the size of the data center. The MAS-based structure proposed in in this paper offers a self-adaptive and dynamic model for the deployment of computational resources in a CC. This framework has learning capabilities

through the implementation of a case-based reasoning (CBR) system [23], a model which, to the best of our knowledge, has not been applied before in distributed systems with these characteristics. These reasoning systems use past experience to solve new problems; their intelligent model resembles human intelligence.

Consequently, we have developed a computational resource distribution model to be used in distributed environments, capable of managing resources according to past experiences, and of dynamically adjusting the resources allocated to each service. Using such a model makes it possible to achieve appropriate responses to demand and to increase the degree of effectiveness/efficiency of the solution and the state of the CC. Therefore, managing the functions of the nucleus of a CC system through an agent-based model and a CBR approach makes it possible to create a much more efficient, scalable and adaptable platform than the current ones.

This paper is structured as follows: the next section describes the context of the current work and the state-of-the-art approaches related to the presented one. Section 3 focuses on the MAS-based platform that supports the technical proposal. Section 4 presents the resource allocation algorithms applied in each agent, while Section 5 presents the evaluation and validation of the proposal. Finally, the last section draws conclusions from the conducted research.

2. Related Technologies and State-of-the-Art Approaches

Usually, in a CC environment, the term hardware infrastructure refers to the virtualized infrastructure [24,25], which means that there is a layer of abstraction between the actual hardware infrastructure and the computation nodes. In this way, each of the offered services is deployed in virtual nodes belonging to that layer of abstraction, and these are called virtual machines (VMs). Therefore, services are usually distributed and this distribution of services in the different nodes requires a system that provides the necessary load balance to distribute the requests among the different computer nodes that attend them.

Each VM has a set of dedicated hardware resources and is completely independent, making it possible to run different operating systems, where the software is completely independent. With this hardware in any given physical equipment, the virtualization process allows one to share and encapsulate the physical resources among a set of VMs by following a pattern similar to the master-slave model [26], in which the (real) server hosts different VMs. This capability, made possible by the virtualization technology, allows the computational resources in the virtualization layer to be considered unlimited resources, while the physical resources or hardware infrastructure are limited to the real infrastructure. The only disadvantage of this technology is that the management of the virtual environment requires a highly-specialized software called hypervisor [27]. This component is responsible for managing the hardware abstraction in each physical node, and this also consumes resources [28]. Nevertheless, this computational consumption depends on the selected virtualization model. Currently, the additional cost does not pose any significant problem as it does not exceed 2% of the computational power of the physical environment [24,29]. However, it is necessary to have dedicated hardware, which is also widely extended as a result of technologies such as INTEL-VT or AMD-V [25].

Virtualization techniques greatly simplify both the management and control of IT resources at the infrastructure level, supporting the dynamic creation or removal of VMs on demand or even enabling the migration of a VM from one physical machine to another at runtime, without stopping or pausing the machine or service. Therefore, thanks to virtualization technology, this complex problem is actually simple to solve, since it is only based on the efficient redistribution of physical (real) resources among the different computational (virtual) nodes. Due to its complexity and its contemporaneity, this problem has attracted the attention of the scientific community, which has led to the proposal of many different solutions.

Current Approaches to Allocated Computational Resources

There are two main approaches to resource distribution in the state-of-the-art systems [8]. The first one involves the search for the cheapest and most efficient provider in terms of resource usage. Thus, this approach follows a model in which a broker or resource manager, usually an external one, is in permanent and simultaneous contact with various providers, and selects the most appropriate one for each client at any given moment. The key technologies employed in this approach are service discovery, negotiation, etc. However, this approach is not the focus of this research and thus will not be considered any further.

The second approach is within the scope of this study, as it is concerned with the efficiency of resources within a CC provider's data centers. In this case, the problem can be contemplated from two different perspectives [30,31]. On the one hand, a large-scale CC resource provider with different distributed data processing centers throughout the world must process client requests according to two variables: the distance from the data center, and the current workload at each center. This makes it easier to reduce the latency of giving a response to clients. This model is known as a time-drive adaptive mechanism [31]. On the other hand, a small-scale resource provider is one that distributes resources from an individual data center, regardless of its size. The task of distributing resources is performed by a component usually referred to as the resource allocation system (RAS). Regarding this perspective, the state-of-the-art systems presents two major approaches [8]:

- QoS-aware, or the market-oriented approach [8]. This approach is related to a customer-oriented resource and service distribution model, seeking to minimize computing risks in order to distribute computing resources, following the SLA and a pay-per-use model. In this model, computer resource management techniques aim to comply with these agreements at all times, thus providing the quality of service requested by the end user. In line with this approach, the state-of-the-art systems include proposals such as RAS-M [15], which is a model based on the market economy. Its aim is to redistribute resources that will lead to a fair market price. Other studies address the distribution of resources on a mathematical basis, such as game theory [32,33], or the maximization of an optimization function, based on linear programming techniques, given that it simply involves the optimization of resources [34–36].
- Energy-aware approach [8]. In this second case, the distribution of resources considers both energy consumption and the pre-established SLA, which implies compliance with both. This approach has less published research because it is more recent. This includes a variety of techniques such as the application of energy savings policies in physical machines and VMs [16,37], the efficient redistribution of VMs according to the consumption of each physical machine [38], or models that are based on optimization techniques [17,39]. Each of these can be found in an incipient state of development, which makes their application in large computing centers more complicated.

In the light of the examination of the most recent studies, it is essential to build a model for the distribution of computer resources that will consider energy consumption as a key variable. The objective of including this variable is to seek to reduce energy consumption to the point that it satisfies the SLAs that have been established with the users of the environment. Moreover, all algorithms presented in this section follow centralized approaches, which use mathematical and economic theory models to provide the best suited algorithms depending on the type of problem. This research follows a novel approach, different from the usual ones, as it is based on AI and optimization techniques, which allows one to distribute resources following a distributed and scalable model, thus allowing the system to learn during a long period of time. To this end, the system is based on a MAS architecture, which makes it possible to interact among the different platform components and permits the introduction of advanced reasoning algorithms to facilitate the development of autonomous and distributed models able to dynamically self-adapt to changes in the environment.

3. Proposed Intelligent Model

On the basis of the analysis described above, where the strengths and weaknesses of the related work have been analyzed, we intend to propose a completely different model, based on CBR and with a MAS architecture to allocate resources and manage CC. The MAS architecture is called +Cloud (multi-agent system cloud) and it is designed specifically for the monitoring and control of a CC infrastructure. This MAS has been designed to leverage some of the most important characteristics of this type of system [39,40] (autonomy, pro-activity, intelligence, learning, organization, mobility, etc.) with the aim of implementing a distributed control strategy that allows for the design of decentralized algorithms for the management and control of cloud infrastructures.

+Cloud has been described at length in previous works [40,41] and it has been validated that it is an adequate system for the implementation of decentralized algorithms that allow for the allocation of computational resources [42]. Thus, this article is only going to focus on the contributions we add to this architecture. Specifically, a decentralized algorithm has been developed that incorporates a case-based reasoning model. This model gives priority to the distribution of responsibilities and to the limited knowledge of the system regarding each of the MAS components. Thus, this section describes the key components that allow to extend the operation of the elastic algorithms to resource distribution.

However, it should be noted that since the proposed MAS is a distributed system by its own definition, all the agents involved in resource distribution tasks may be located making efficient use of the CC environment. In other words, a distributed approach is used in the monitoring and control of the CC system. The cloud environment obtains the data from all of the CCs, both from the services it provides and from the infrastructure itself, thus having a distributed model oriented towards monitoring, which allows existing resources to be instantly adapted to the characteristics sought for in the CC environment. In this way, the demands of each service are satisfied in an agile way, meeting the objective of reducing energy consumption and SLA agreements.

Figure 1 presents the primary agents in charge of the distribution of computational resources. The main difference between a CC environment and previous technologies is that it has the ability to offer the demanded services through a pay-per-use model [7]. +Cloud has been designed using the GORMAS methodology [43]. The model followed in the design of this MAS differs from the traditional control models employed in the development of this type of platforms, where decisions are usually taken centrally [8]. In this regard, monitoring and decision-making related responsibilities have been distributed throughout all the components of the platform (servers, services, etc.). Thanks to this model, it is possible to decide where the information is collected on the basis of local knowledge, which has allowed for the design of agile control processes based on uncertainty and interaction between peers.

In general terms, when a service k is demanded by a process or user, the service must fulfill it with the agreed SLA. This issue is controlled by the two types of agents associated with each service (service monitor and service supervisor) and with the sub-organization of resource consumption. The service monitor agent (SMA) is in charge of monitoring each of the services offered by the system, collecting data regarding the requests being made and measuring parameters of their quality, performance, errors, etc., in addition to having access to the demand history. There is an agent for each of the services offered by the CC and it is located in the node that balances the requests. The service supervisor agent (SSA) ensures that the previously established SLA agreements are being complied with, taking appropriate action in the event of detecting deficiencies. The SSA is also responsible for ensuring the high availability of the service, making sure that there are at least a certain number of nodes working in independent physical teams.

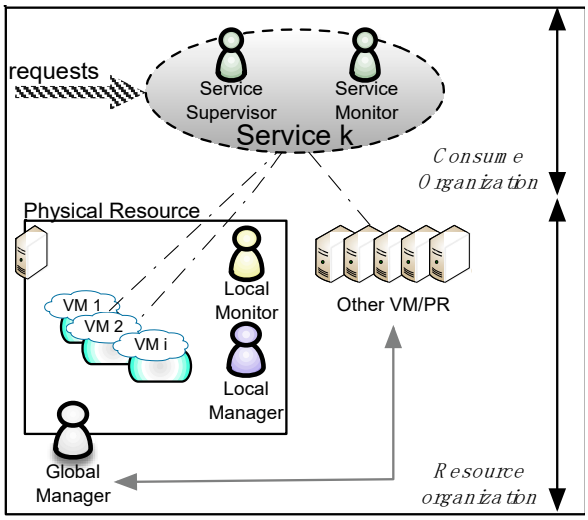

Figure 1. Distribution of agents on the cloud computing platform.

The services are deployed on servers, which represent the system's computer resources. Three roles (local monitor, local supervisor and global manager) associated with the sub-organization that provides resources are also deployed here. The local monitor agent (LMA) is in charge of collecting data regarding the state of the local resources of each physical server—including its virtual machines—(PR_i) and transforming these data into adequate information for decision making. The local supervisor agent (LSA) is in charge of the control and distribution of the physical machine's computer resources, and is able to redistribute resources among the instances of execution, launch or shut down of virtual machines. Its objective is, therefore, the efficient management of the individual resources of the physical server between the different nodes it hosts, maximizing its use, but without diminishing the quality of the services being provided; and, ensuring that the physical machine always has the minimum resources to perform the tasks of coordination and control. Its work is carried out in close collaboration with the server's local monitor agent. The SLA and the local monitor have total knowledge of each individual server, but at the same time uncertainty regarding the rest of the infrastructure. The global manager (GMA) is the role in charge of making decisions about how to distribute the computer resources among several nodes of the CC platform, and not only at a local level as in the case of the LSA's role. In order to provide assistance to the decision-making process regarding the means of distributing resources, they use a partial knowledge base (provided by the LMA role) and past experiences stored in the usage history repository, which is achieved by relying on the CBR model [23]. The GMA's role is to determine how and with what characteristics the new service-associated virtual machines will be instantiated. Additionally, it is also responsible for the process of compacting the existing virtual machines into a smaller number of servers, so that it is possible to shut down physical servers and reduce energy consumption. Given that it is possible to stop and start execution nodes, it is also possible that these agents enter and leave the system dynamically, linking their life cycle to that of the server they monitor or control.

All these agents have a local knowledge base, but they also store historical information on a centralized nonSQL server, based on MongoDB. When the agents enter the system, they retrieve specific information about the history of the specific service or server where they are located. Concretely, there are three centralized repositories (resources, SLAs and historical), as can be seen in the structural view according to the GORMAS methodology given in Figure 2. This figure also shows the detail of the activity model according to GORMAS for the infrastructure control service.

The next section provides a detailed description of the reasoning model designed to allocate computational resources by means of agents within a distributed and heterogeneous environment where the level of uncertainty is high.

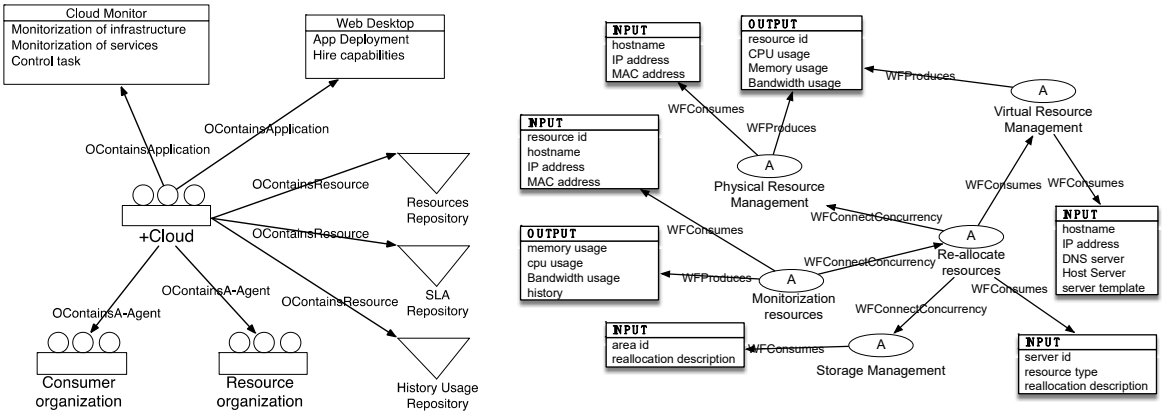

Figure 2. (**Left**) The structure of the suborganizations (consumption and supply), data repositories and control devices; (**Right**) Shows the activity of the infrastructure control service.

4. Intelligent Model for Allocating Resources

The redistribution of resources is performed by the GMAs located in each physical server, which have greater authority than the LSAs and can request them to add/remove virtual nodes to a specific service *k* with specific characteristics. The GMA is a highly specialized agent which is in charge of providing the CBR architecture [23,44]. This reasoning process is performed simultaneously in all physical machines with available resources. In the last part of the distributed algorithm, a new VM, with specific characteristics of virtual CPUs (*vcpu*) and Memory (*M*) is instantiated to meet current demand (see Figure 3). The part associated with the reasoning of CBR at each physical node is based on the experience gained in storing similar cases. The knowledge base (*KB*)—or case memory—is shared with the whole CC environment; the global knowledge of the system can be shared with each of the GMAs. Since this memory can grow considerably as a maintenance strategy, a high-speed, schema-free database, based on MongoDB, is used to provide rapid access to the stored data.

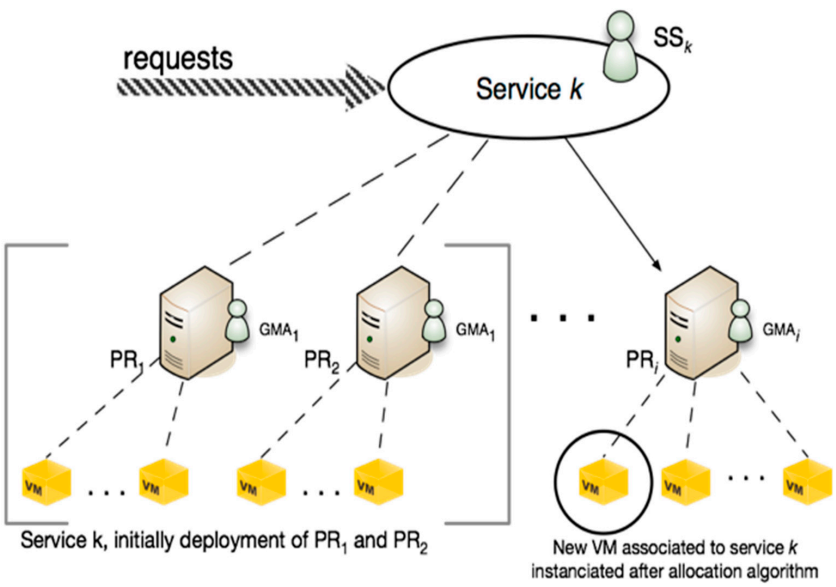

Figure 3. When a service experiences difficulty, more resources must be added by means of a new virtual machine (VM).

The reallocation process is initiated by the SSA associated with the service that is experiencing difficulties in responding to demand, as shown in Figure 4. This figure shows a hypothetical case of communication among the agents presented in Figure 3, where service *k* is deployed on virtual nodes hosted by three servers ($PR_{1,2 \text{ and } i}$) that have been represented as vertical lines. The servers host

their agents and processes as described. The communication between agents and other processes is indicated by arrows. The aim of the communication is to find a solution—S(P)—to service demand.

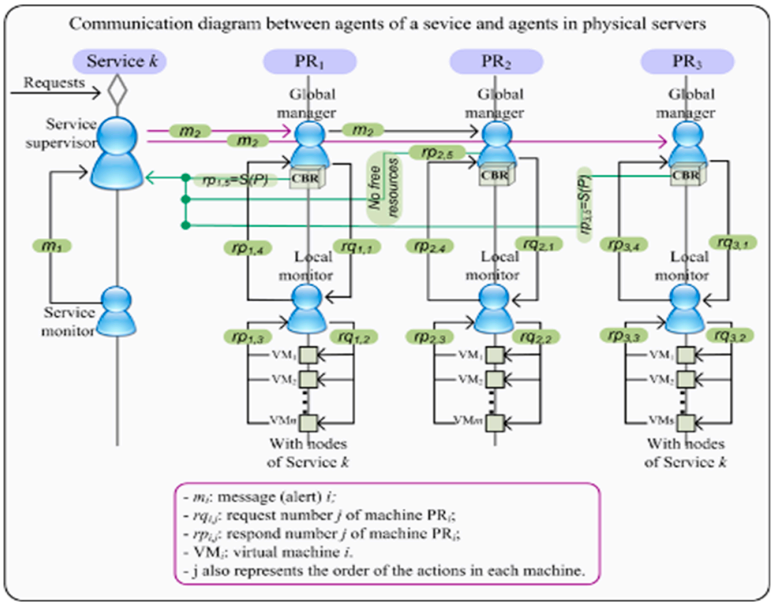

Figure 4. Diagram of the collaborative agents shown in Figure 2 communicating to reach a solution S(P) to the request carried out by the service k.

When a service k is experiencing difficulties, the associated SSA alerts all physical machines (that host the nodes of the service) that a new allocation of resources is required to handle the demand, as shown in the messages m_2 sent from the SSA to the remaining agents of the service in Figure 4. The message is received by the GMA from each physical server of the service, GMAs of PR_1, PR_2 and PR_i in Figure 4. The GMAs that received the message alert the remaining GMAs in the CC environment, as shown in the message m_2 sent from the GMA of PR_1 to PR_2 in Figure 4.

These agents ask the LMA of each machine to evaluate the level of usage of the internal resources. Each LMA generates a matrix ($I^t_{PR_i}$) with the information for the time t in order to evaluate the amount of available resources. Communication occurs between the GMA and LMA of each PR_i, which implies the message sequence $\{rq_{1,1}, rq_{1,2}, rp_{1,3}, rp_{1,4}\}$ for the case of PR_1 in Figure 4. This snapshot structure gathers instantaneous information ($I^t_{PR_i}$) on the complete state of resource assignment at a specific time t in the resource PR_i (See Table 1).

Table 1. Summary of information on physical servers.

	Processing			Memory			
	$vcpu_{max}$	$vcpu_{min}$	$vcpu_{used}$	M_{max}	M_{min}	M_{used}	$M_{asigned}$
VM_1							
VM_2							
VM_m							
PR^{e_1}							

With this information, the GMA determines whether the physical machine can enter the decision process. If the amount of resources in each physical machine is greater than the minimum indispensable resources required to instantiate a node associated with the service, then the reasoning process is initiated to determine the amount of resources that can be reserved for each execution node as requested by the SSA for that service. Figure 4 shows the messages $rp_{1,5}$ and $rq_{2,5}$ sent by CBRs in GMAs from PR_1 and PR_2, respectively, to the SSA. However, if there are no available resources, or if there are fewer

resources than those demanded by the service, the GMA determines that the physical node is not part of the global assignment process (see message $rq_{3,5}$ of unavailable resources sent from the GMA in PR_3 to the SSA, Figure 4). If there is no physical machine that could respond to the increased service needs, a new physical server is requested to start up in order to instantiate a VM according to the minimum characteristics defined in the service level.

When the machine has enough computational resources (cases PR_1 and PR_2 in), the GMA initiates the process based on the definition of the concept of the following case: $C = \{P,\ S(P),\ E\}$ where:

- P represents the description of the problem, with a matrix-matched representation directly related to the instantiation of resource use, $I^t_{PR_i}$, along with the description of the service level that is instantiated and the temporary indicator (*timestamp*) that identifies the instant in which the problem has been detected, where VM^k_t is the description of the minimum resources, in terms of memory M and *vcpu*, that are needed by the service.

$$P = \left(I^t_{PR_i}, VM^k_t, timestamp\right) \tag{1}$$

- S(P) represents the solution to the *P: S(P) = (M, vcpu)* in terms of *vcpu* and memory.
- Finally, E represents the efficiency which is measured from two perspectives: micro and macro.
  - Firstly, the efficiency at the micro level (E_m) is associated with the level of the efficiency of the solution proposed within the physical server where the VM has been deployed. The LMA proposes this level of efficiency according to the processor usage rates and the allocated memory.

$$E_m = \left(\frac{\overline{p_{vcpu}}, M_{used}}{M_{assigned}}\right), \tag{2}$$

    where M_{used} and $M_{assigned}$ are the used and total allocated memory, respectively.
  - The efficiency at the macro level (E_M) is associated with the degree of efficiency from the point of view of the service and is calculated if the proposed solution requires the process of infrastructural resource distribution to be initiated at a macro level. In this sense, the degree of efficiency measures the number of additional nodes n required by the service.

$$E_M = n \tag{3}$$

Therefore, the efficiency is given by the following expression $E = (E_m, E_M)$

The CBR (*Case-Based Reasoning*) starts by retrieving similar cases from the case memory. The most similar cases are selected as explained below, and the formal algorithm is given in Algorithm 1:

1. Select the cases with similar characteristics to the physical machines and a degree of efficiency greater than 90%. The similar physical machines are evaluated according to the *benchmark* parameter, which characterizes each physical machine.
2. On the basis of this subset of retrieved cases, a vector C_i is configured for each case that contains the same number of VMs that are in the case, and the free resources available $Ci = (n, M, vcpu)$.
3. The cases selected from this subset are those that had previously used the same service, and for a similar period of time. This is determined by analyzing a network usage pattern over a period of one week.

During the reuse phase, a solution to the problem is prepared, based on the cases that have been retrieved:

- If the case base does not contain a similar past case, the solution to the problem is associated with the minimum resources determined at the level where the service is instantiated:

$$S(P) = (M_{min}, vcpu_{min})$$

- If similar cases were recovered, the solution to the problem would be the case closest to the new one, multiplied by the efficiency of the most similar case:

$$S(P) = (M' \times (E_m\ (1) + E_M\ (1)), vcpu' \times (E_m\ (2))) \quad (4)$$

- In the case where there are not many resources available and the values assigned to the previous solution are higher than the values assumed by the machine, the result of the case would be the maximum amount of resources available in the machine.

$$S(P) = (PR\ (Mmax), PR\ (vcpumin)) \quad (5)$$

When the solution to the case is calculated, it is sent to the SSA. This agent reactively chooses the node that would provide a higher amount of resources at the VM level. In the subsequent review stage, the new node is deployed, and its use is evaluated from both a micro and macro perspective. In this way, the efficiency of the solution is obtained. Finally, in the last step of the CBR, both the case and the value of the efficiency are stored so that they can be reused in future executions.

The proposed adaptation model is distributed, which makes it possible to improve the high availability of the system, since the decision-making process is made throughout the entire CC system. Furthermore, this model can distribute the strength of the calculation, which requires obtaining the solution; as a result, the impact that the search for a solution has on the CC environment is reduced.

Algorithm 1. Steps performed by the case-based reasoning algorithm involved in each global manager to allocate resources demanded by a service.

CBR Algorithm
INPUT: P, the description of the problem (case) to solve for a service *k*.
OUTPUT: S(P), the solution to problem P.
REQUIRE: *KB*: the knowledge base, *d*: dissimilarity metric of cases, E: compute the solution efficiency of a case, *#vm*: VM number for a case, *ar*: the available resource amount, *ur*: the used resource amount, *Service*: used service, T: service use period. ρ : dissimilarity metric of the service use period

```
% Retrieve all cases similar to the case with feature P in a
% set SC1(P)
SC1(P) := {Pi ∈ KB d(P, Pi) ≤ mp AND E(Pi) > 90}% where mp is a
% threshold (i.e., midpoint, mean, etc.) stating similarity
% between cases.
% Select all cases in SC1 with the same number of virtual
% machines and with available resources.
Assign a vector vi := (ni, mi, vcpui) to each Pi of the set:
SC2(P) := {Pi ∈ SC1(P)/#vm (P) = #vm(Pi)AND ar ≥ ur (P)}
% Select cases that have used the current service k and for a
% period similar to case P.
SC3(P) := {Pi ∈ SC2(P)/Service (Pi) = k AND p(T(P), T(Pi) ≤ t} % where t
% is the threshold stated for period similarity.
If SC3(P) = ∅ then S(P) := (Mmin, vcpumin)
  else
    Q := arg_i min{d(P, Pi)/Pi ∈ SC3(P)}
    M := M(P) := M(Q) × (Em(1) + EM(1))
    vcpu := vcpu(P) := vcpu (Em(2))
    If M(P) ≤ M(PRi) and vcpu (P) ≤ vcpu (PRi) then S(P) := (M, vcpu)
     else S(P) := (PRi(Mmax), PRi(vcpumin))
   end else
end if
end.
```

Compaction

The proposed model uses the key characteristics of virtualization technology, which involves migration of machines between physical servers. In fact, this characteristic is very effective when compacting VMs into the smallest possible number of servers. This allows one to turn off/switch to sleep mode the set of physical servers that have not yet been assigned a machine, which significantly increases energy efficiency.

However, problems occur when the machines are widely dispersed, which can happen when a *SSA* associated with a specific service detects various nodes that are at the highest priority level (they have a high quality of results) and requests the random elimination of one of these nodes.

The compaction process is simple and is applied by the specialized *GMA* as follows:

1. A virtual server with a very low number of VMs (the number is determined by the system administrator) asks other servers to host its VMs so that it can go into sleep mode and not use any resources.
2. The nodes with available resources at the time of the request evaluate the snapshop ($I^t_{PR_i}$) provided by the *LMA* to determine whether it can host another machine with the given characteristics.
3. If there are resources available, the configuration is sent to the *GMA* that has made the request, and the migration process is initiated. If this agent receives various confirmations simultaneously, it randomly initiates the migration process with one of the machines. The process is random since the agent does not know all the internal details of the machine that hosts the new virtual node.

This simple process makes it possible to compact the set of VMs without affecting the quality of service, since the individual resources of each machine are not modified. The system then goes into a compact state.

5. Evaluation of the Proposed Model

In order to evaluate and validate the model proposed in this article, a CC platform designed and developed by the BISITE research group (http://bisite.usal.es.) has been used. This CC platform was deployed in the HPC (High Performance Computing) environment, which offers numerous services and is composed of 15 machines that support virtualization in hardware with the use of Intel-VT technology and the KVM (Kernel-based Virtual Machine) virtualization system. MAS is implemented using the Python programming language because of its power, ease of maintenance and flexibility in handling data structures. In this regard, the data exchange format is based on JSON (JavaScript Object Notation). The web service layer has been implemented using the Tornado web development framework because of its ability to manage a large number of client connections.

With regard to the distribution of the initial resources (see Figure 5), a Cloud service—the file storage service—is deployed in different virtual nodes (VM_1 and VM_2), each one hosted by a different physical machine (PR_1 and PR_2, respectively). The result obtained with this deployment is that the service is highly available (deployed in two servers), and it is also deployed in two physical machines with different computational loads, something that happens in real environments, since the two physical machines host other virtual machines that correspond to other services of the CC platform. In other words, the physical server PR_1 has many available and unallocated resources, while PR_2 has no available resources and the machines it hosts have a high computational load.

An amount of 10 to 40 threads were launched which, every 3 s, consulted specific methods of the service (*GetSize* and *GetFolderContent*). The acceptable QoS level for the *GetSize* function in this experiment remains at 1.5 s, while the threshold QoS level for *GetFolderContent* is set at 0.5 s.

The process starts once the *Service Supervisor* detects a decrease in performance, at this time it directly executes the adaptation process. This type of adaptation occurs when the demand for the service is much greater; an increase in the computational load results in a more rampant increase in the load. The process for exchanging messages among agents during the adaptation of the infrastructure is shown in Figure 6. The specialized *Service Supervisor* agent that initiates the service also sends

an alert (step 1, Figure 6) to the *Global Manager* agent for each of the physical machines that host the service nodes. It should be noted that the GMA is a specialized agent using a CBR reasoning process [23,44] in order to allocate resources at the macro level. After receiving the first alert, the agents forward the alert message to the rest of the GMA in the CC (step 2, Figure 6).

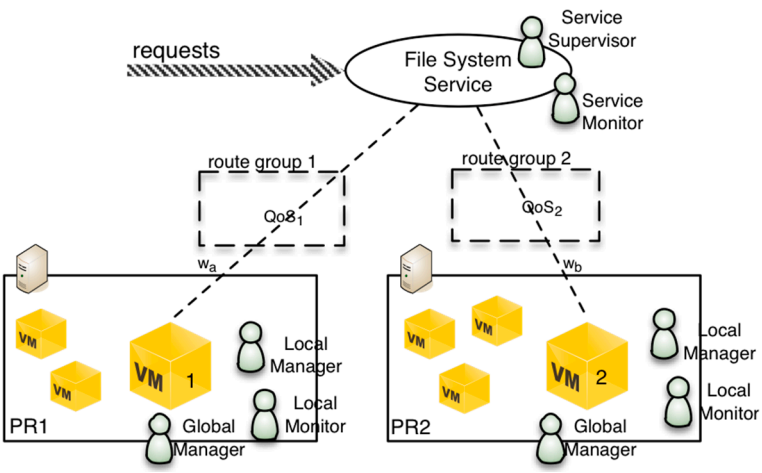

Figure 5. Evaluation of the case study—status during the initial period.

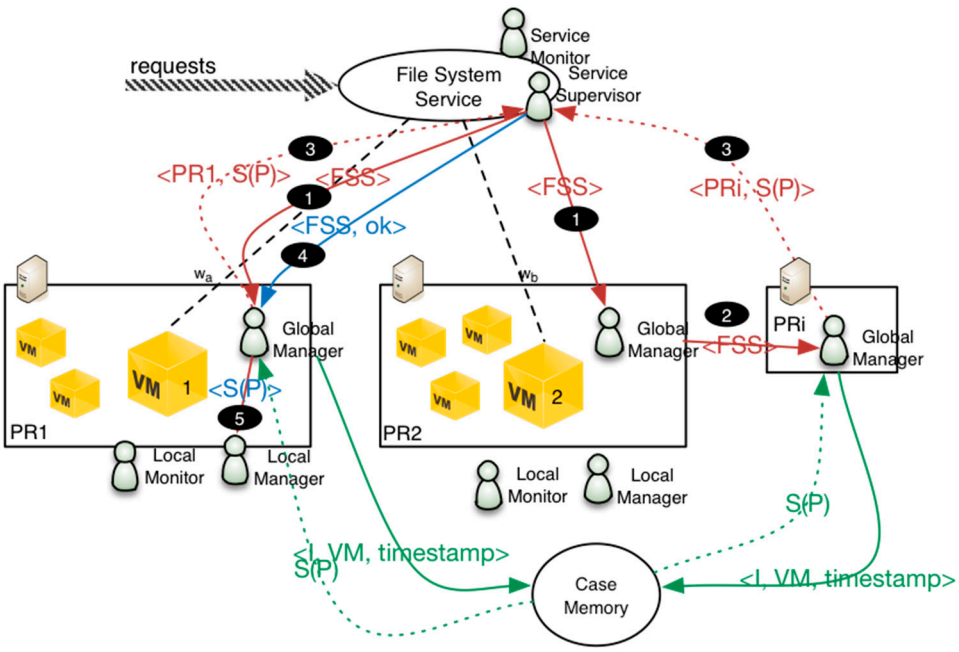

Figure 6. Exchange of messages in the infrastructure distribution during the experiment.

The next process is carried out in parallel in each physical node of the CC system. Each GMA that has received an alert message requests the *LMA* of the machine in which they are located to instantiate the state of the system (I^t_{FR}). The agent uses this information to determine if there are available resources to instantiate a new node associated with the service requesting resources. If the machine, in response to the instantiation, detects that the physical server does not have any available resources, it does not perform any action. This process is conducted in parallel for each of the machines that have available resources.

This problem description is used to retrieve similar cases from the case memory. Each *GMA* individually determines which resources can relinquish to the new node that has just been instantiated. This new provision of resources is itself the solution to the current problem, *S(P)*. The solution

provided by each of the CBR agents from each physical server with available resources is then sent to the *Service Supervisor* agent that initiated the process because of its own performance difficulties (step 3, Figure 6). Once it has received the set of proposed solutions from the different CBR-BDI agents, the agent overseeing the service sends an acceptance message to the *GMA* that offers the greatest amount of resources for the new node that must be instantiated (step 4, Figure 6). Finally, the *GMA* that receives the request asks the *LMA* (step 5, Figure 6) from its machine to instantiate a new virtual node on the level of the VM associated with the service, according to the proposed problem solution.

Once the new execution node has been launched, it is then necessary to evaluate the proposed solution. The *LMA* evaluates the solution according to the degree of underused resources in the node that has just been instantiated. If a new resource distribution process must be conducted at the macro level, the SSA completes the evaluation performed by the *LMA* in order to penalize the solution. In both cases, the efficiency of the proposed solution is evaluated according to the amount of underused resources of the new instantiated node.

Results in terms of QoS are represented in Figures 7 and 8, showing also the increase in quality once the adaptation is finished.

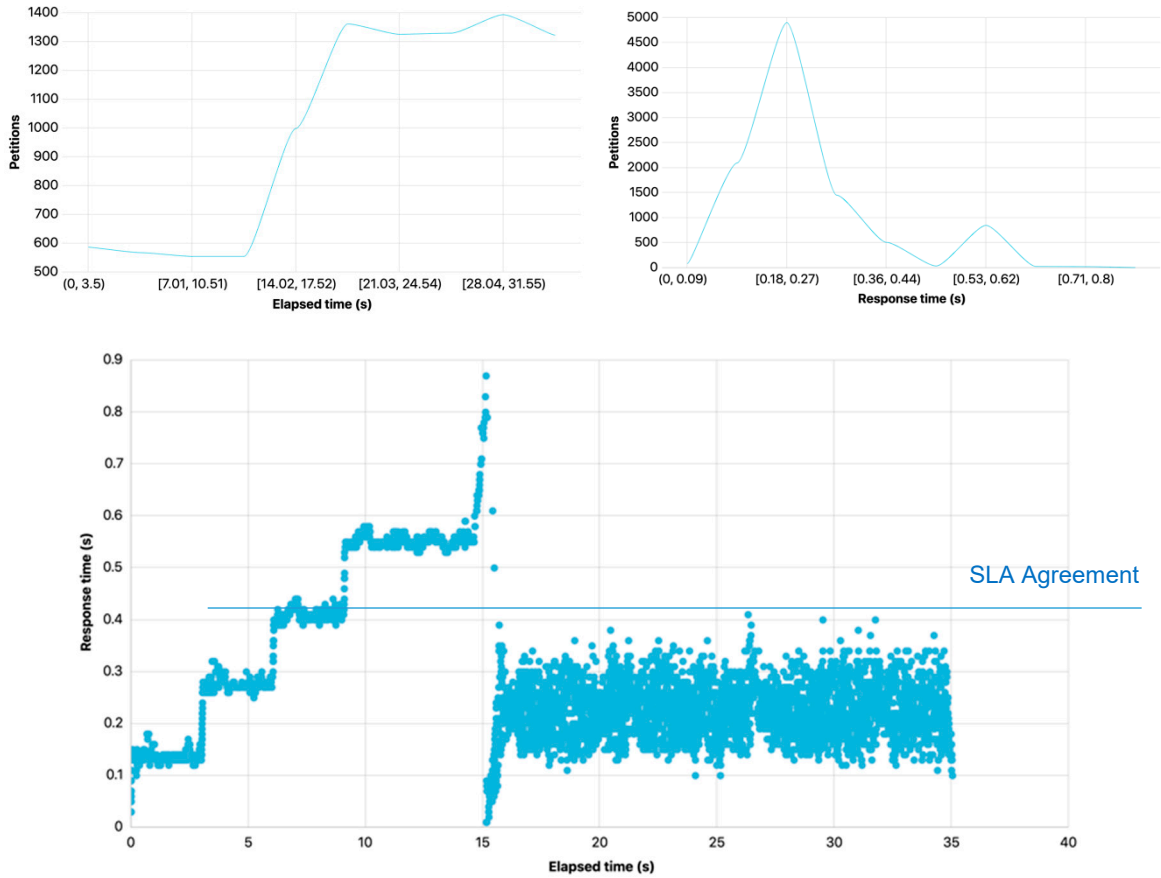

Figure 7. Experiment 1: Readjustment of the infrastructure resources for the method "GetFolderContent" ((**top left**) increase in request; (**top right**) time response of request; (**bottom**) the whole adaptation process).

Subsequently, another set of experiments has been conducted for the distribution of infrastructure resources. In these experiments, we intended to perform numerous consecutive executions in the adaptation process, because a single execution of the proposed algorithm to perform the adaptation automatically could not satisfy the demand for the services. The result of the tests carried out from the point of view of the adaptation model was positive since it showed that it performed correctly, always within the limits of the case study and keeping the SLA level within the established limits.

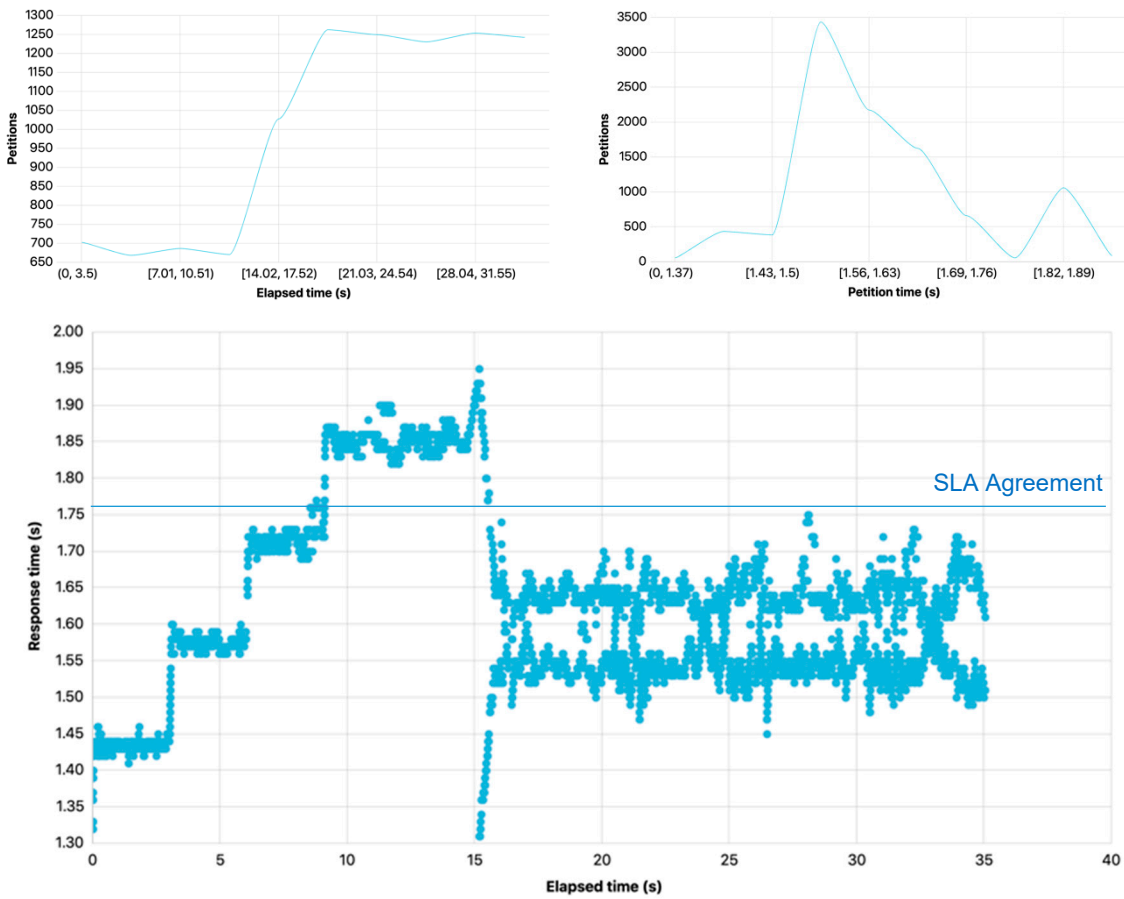

Figure 8. Experiment 2: Readjustment of the infrastructure resources for the method "GetSize" ((**top left**) increase in request; (**top right**) time response of request; (**bottom**) whole adaptation process).

It is not possible to make an empirical comparison of the proposed model with other existing approaches in the state-of-the-art systems, as it is difficult to recreate the computer and/or simulation environments in which they have been evaluated. However, it is possible to make a theoretical comparison of the proposed approach with respect to other existing works in the state-of-the-art approaches. Firstly, it is observed that the proposed model follows a distributed approach to solving the problem, which totally differentiates it from existing works in the state-of-the-art approaches [16,38]. This approach, which has been shown to be valid for the distribution of computational resources in this type of CC environment, has advantages with respect to their availability, since there is no single component in charge of the distribution of resources, but rather the system (society) itself is reorganized as a whole through the individual adaptation of its components (agents).

In the state-of-the-art approaches, as detailed in particular by Goudarzi and Pedram [45], the execution of allocation algorithms is a complex task that requires a large amount of computational power and time. However, the proposed model simplifies the search for an adequate solution to the problem, as (i) the computational needs are distributed among different nodes; (ii) the value space to be considered is smaller since each node must only consider the data regarding its own resources and, furthermore, it is not necessary to have global knowledge of the platform; and, finally, (iii) each node can autonomously apply a partial solution to the problem, eliminating the coordination needs at the global level of the platform.

Finally, the clear advantage of the proposed model with respect to other works is its capacity for learning. In the macro-level infrastructure distribution model, it has been used as an adaptation algorithm based on a case-based reasoning system that is integrated into a specialized agent in the +Cloud organization. Thanks to this approach, as has been demonstrated, it is possible for

the system to learn from past experiences, achieving greater efficiency in adaptations as it learns, memorizing both positive and negative experiences. Among the state-of-the-art approaches, there is no similar approach in which the distribution of computational resources is based on the results obtained in past adaptation processes.

6. Conclusions

This research is one of the first to propose the use of a MAS in the framework of control and surveillance systems in CC environments. The main result of this work is the proposal of a new architectural model based on a MAS with a clearly integrating purpose. To achieve this, a set of algorithms for the distribution of computing resources in CC environments has been developed, evaluated and assessed. The main innovation of the proposal lies in the dynamic capacity of the system to adapt autonomously to demand and to learn from previous experiences.

With the new proposal, it has been possible to demonstrate that a control and surveillance system in CC environments can be designed using MAS as the basis, and it is also the key element of the architecture. This is clearly due to the distributed nature of MAS, which makes it possible to implement elastic algorithms to support the services that require distribution. Another key aspect is the distribution of responsibilities. Thanks to the use of this type of algorithm, it is possible not only to make decisions in the place where the problems actually arise, but also to distribute the necessary calculation capacity to obtain an efficient solution among all the instances that comprise the CC environment. This type of solution directly contradicts the way in which current centralized models solve the problem of elasticity. As this study has demonstrated, the use of a distributed approach is undoubtedly a viable option that should be considered when designing elastic algorithms.

This strategy ensures that each process is independent when making decisions in the software layers where many actions are running. It is undeniable that if the capabilities offered by the underlying technology change, it will also require that certain aspects be changed in the reasoning models used, just as would be done in any traditional strategy. However, in this case, the MAS architecture overcomes these challenges; thanks to its adaptive design, it is possible to modify the individual agents that carry out specific actions in the MAS. This approach ensures independence between the software layers in which decisions are made and those in which the decisions are executed. In a CC environment, such separation of responsibilities is particularly important because today's platforms are highly dependent on the technological environment (virtualization tools, load balancers, distributed file systems, etc.). This dependence constitutes a great limitation and hinders the platform's evolution, since a change in some of the hardware or software components also makes it necessary to alter the algorithms and techniques that make the system elastic. In the proposed MAS, implementation is made using communication ports according to the GORMAS methodology. Thanks to this methodology, dependence on the environment is limited to the port itself, i.e., interface for communication with the environment. There is no doubt that a change in the capabilities offered by the underlying technology would also make it necessary to modify the proposed reasoning models, as in a traditional design approach. However, organizational models also offer an adequate response to this difficulty. In a system such as that being proposed, the agents who are part of society are given one or more specific roles. Each role is an abstract definition of the objectives, responsibilities and privileges of the individual who assumes it. Following this high-level definition, in the case of the introduction of new technological capacities, the work of adaptation would consist in modifying the individual or individuals who carry out the concrete tasks and those who play roles in the organization. Therefore, since it would not be necessary to alter society as a whole and only the individual entities; the proposed architecture is also above other existing platforms in the market.

The proposed model has been designed to address the problem of excess energy consumption by proposing solutions that consider the degree of efficiency. As part of the problem, this study proposes a compaction model of VMs. This model makes it possible to define the problem of excess energy consumption within the framework of the CC platform. However, the learning capability

of the proposed adaptation model is undoubtedly the key characteristic that can cogently improve the existing state-of-the-art approaches: while current models use mathematic algorithms and statistical systems, the proposed system develops specific solutions to a given problem on the basis of the degree of efficiency that the same or similar solutions have had in the past. Given that the system has learning capacities, its response and ability improve over time. Thus, this approach can continually increase the efficiency of the solutions it proposes. Moreover, the system's learning abilities are extremely important in an environment where there is some uncertainty, as is the case of CC. This is because when the context or environment of the CC platform changes at any given time, the adaptation model must likewise evolve and adapt all the proposed solutions so that the efficiency of the proposed solution is maximized.

In conclusion, the use of MAS enables us to continue researching techniques, tools and methodologies that will have intelligent characteristics, such as autonomy and pro-activity. In this regard, future lines of research will be proposed to continue extending the capacities of the proposed system, firstly, by incorporating the capacity provided by containers to improve the granularity in the distribution of resources and to achieve a much more precise resource allocation. Likewise, we also intend to extend the capacities of the MAS to distribute computing responsibilities beyond the cloud computing environment, within the framework of the edge computing paradigm, so that it can be integrated in an Internet of things environment and communication can be carried out at the edge.

Author Contributions: Conceptualization, F.D.l.P. and S.R.-G.; methodology, S.R.-G. and J.M.C.; software, F.D.l.P.; validation, J.M.C. and Y.D.; formal analysis, F.D.l.P. and S.R.-G.; investigation, F.D.l.P. and S.R.-G.; resources, S.R.-G.; data curation, F.D.l.P.; writing—original draft preparation, F.D.l.P. and S.R.-G.; writing—review and editing, F.D.l.P. and Y.D.; visualization, P.C.; supervision, Y.D. and J.M.C.; project administration, S.R.-G.; funding acquisition, J.M.C., F.D.l.P. and S.R.-G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Spanish government and European FEDER funds, project InEDGEMobility: Movilidad inteligente y sostenible soportada por Sistemas Multi-agentes y Edge Computing (RTI2018-095390-B-C32).

Acknowledgments: MDPI for APC removal.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shyamala, K.; Sunitha Rani, T. An Analysis on Efficient Resource Allocation Mechanisms in Cloud Computing. *Indian J. Sci. Technol.* **2015**, *8*, 814–821. [CrossRef]
2. Fisher, P.; Pant, R.; Edberg, J. *Cloud Computing: Assessing Azure, Amazon EC2, Google App Engine and Hadoop for IT Decision Making and Developer Career Growth*; Apress: New York, NY, USA, 2011.
3. Luo, J.-Z.; Jin, J.-H.; Song, A.-b.; Dong, F. Cloud computing: Architecture and key technologies. *J. Commun.* **2011**, *7*, 3–21.
4. Laszewski, G.v.; Diaz, J.; Wang, F.; Fox, G.C. Comparison of Multiple Cloud Frameworks. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, 24–29 June 2012; pp. 734–741.
5. Wen, X.; Gu, G.; Li, Q.; Gao, Y.; Zhang, X. Comparison of open-source cloud management platforms: OpenStack and OpenNebula. In Proceedings of the 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, Chongqing, China, 29–31 May 2012; pp. 2457–2461.
6. Leavitt, N. Is Cloud Computing Really Ready for Prime Time? *Computer* **2009**, *42*, 15–20. [CrossRef]
7. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I.; et al. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]
8. Buyya, R.; Beloglazov, A.; Abawajy, J. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. In Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010), Las Vegas, NV, USA, 12–15 July 2010.

9. Ross, J.W.; Westerman, G. Preparing for utility computing: The role of IT architecture and relationship management. *IBM Syst. J.* **2004**, *43*, 5–19. [CrossRef]
10. Alhamad, M.; Dillon, T.; Chang, E. Conceptual SLA framework for cloud computing. In Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies, Dubai, UAE, 13–16 April 2010; pp. 606–610.
11. Liu, F.; Tong, J.; Mao, J.; Bohn, R.; Messina, J.; Badger, L.; Leaf, D. NIST cloud computing reference architecture. *NIST Spec. Publ.* **2011**, *500*, 292.
12. Wang, L.; von Laszewski, G.; Younge, A.; He, X.; Kunze, M.; Tao, J.; Fu, C. Cloud Computing: A Perspective Study. *New Gener. Comput.* **2010**, *28*, 137–146. [CrossRef]
13. Zhang, Q.; Cheng, L.; Boutaba, R. Cloud computing: State-of-the-art and research challenges. *J. Internet Serv. Appl.* **2010**, *1*, 7–18. [CrossRef]
14. Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauer, R.; Pratt, I.; Warfield, A. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.* **2003**, *37*, 164–177. [CrossRef]
15. You, X.; Xu, X.; Wan, J.; Yu, D. RAS-M: Resource Allocation Strategy Based on Market Mechanism in Cloud Computing. In Proceedings of the 2009 Fourth ChinaGrid Annual Conference, Yantai, China, 21–22 August 2009; pp. 256–263.
16. Raghavendra, R.; Ranganathan, P.; Talwar, V.; Wang, Z.; Zhu, X. No "power" struggles: Coordinated multi-level power management for the data center. *SIGARCH Comput. Archit. News* **2008**, *36*, 48–59. [CrossRef]
17. Kusic, D.; Kephart, J.O.; Hanson, J.E.; Kandasamy, N.; Jiang, G. Power and Performance Management of Virtualized Computing Environments Via Lookahead Control. In Proceedings of the 2008 International Conference on Autonomic Computing, Chicago, IL, USA, 2–6 June 2008; pp. 3–12.
18. Buyya, R. Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility. In Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai, China, 18–21 May 2009; p. 1.
19. Wooldridge, M.; Jennings, N.R. Intelligent agents: Theory and practice. *Knowl. Eng. Rev.* **1995**, *10*, 115–152. [CrossRef]
20. Dignum, M. *A Model for Organizational Interaction: Based on Agents, Founded in Logic*; Utrecht University Repository: Utrecht, The Netherlands, 2004.
21. Schertler, W. *Virtual Enterprises in Tourism: Folklore and Facts—Conceptual Challenges for Academic Research*; Springer: Vienna, Austria, 1998; pp. 278–288.
22. Modoni, G.E.; Veniero, M.; Trombetta, A.; Sacco, M.; Clemente, S. Semantic based events signaling for AAL systems. *J. Ambient Intell. Humaniz. Comput.* **2018**, *9*, 1311–1325. [CrossRef]
23. Corchado, J.M.; Glez-Bedia, M.; De Paz, Y.; Bajo, J.; De Paz, J.F. Replanning mechanism for deliberative agents in dynamic changing dynamic changing environments. *Comput. Intell.* **2008**, *24*, 77–107. [CrossRef]
24. Che, J.; Shi, C.; Yu, Y.; Lin, W. A Synthetical Performance Evaluation of OpenVZ, Xen and KVM. In Proceedings of the 2010 IEEE Asia-Pacific Services Computing Conference, Hangzhou, China, 6–10 December 2010; pp. 587–594.
25. Chen, W.; Lu, H.; Shen, L.; Wang, Z.; Xiao, N.; Chen, D. A Novel Hardware Assisted Full Virtualization Technique. In Proceedings of the 2008 the 9th International Conference for Young Computer Scientists, Hunan, China, 18–21 November 2008; pp. 1292–1297.
26. Sullivan, M.; Anderson, D. Marionette: A system for parallel distributed programming using a master/slave model. In Proceedings of the 9th International Conference on Distributed Computing Systems, Newport Beach, CA, USA, 5–9 June 1989; pp. 181–188.
27. Sahoo, J.; Mohapatra, S.; Lath, R. Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues. In Proceedings of the 2010 Second International Conference on Computer and Network Technology, Bangkok, Thailand, 23–25 April 2010; pp. 222–226.
28. McDougall, R.; Anderson, J. Virtualization performance: Perspectives and challenges ahead. *SIGOPS Oper. Syst. Rev.* **2010**, *44*, 40–56. [CrossRef]
29. Che, J.; He, Q.; Gao, Q.; Huang, D. Performance Measuring and Comparing of Virtual Machine Monitors. In Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Shanghai, China, 17–20 December 2008; pp. 381–386.

30. Anuradha, V.P.; Sumathi, D. A survey on resource allocation strategies in cloud computing. In Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, India, 27–28 February 2014; pp. 1–7.
31. Jung, G.; Sim, K.M.; Kwok, P.C.; Zhang, M. A time-driven adaptive mecahnism for cloud resource allocation. In Proceedings of the 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology, Shenzhen, China, 28–30 October 2011; pp. 441–446.
32. Rohaninejad, M.; Tavakkoli-Moghaddam, R.; Vahedi-Nouri, B. Redundancy Resource Allocation for Reliable Project Scheduling: A Game-theoretical Approach. *Procedia Comput. Sci.* **2015**, *64*, 265–273. [CrossRef]
33. Wei, G.; Vasilakos, A.V.; Zheng, Y.; Xiong, N. A game-theoretic method of fair resource allocation for cloud computing services. *J. Supercomput.* **2010**, *54*, 252–269. [CrossRef]
34. Dobrovolskienė, N.; Tamošiūnienė, R. Sustainability-oriented financial resource allocation in a project portfolio through multi-criteria decision-making. *Sustainability* **2016**, *8*, 485. [CrossRef]
35. Van, H.N.; Tran, F.D.; Menaud, J. Autonomic virtual resource management for service hosting platforms. In Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, Vancouver, BC, Canada, 23–23 May 2009; pp. 1–8.
36. Van, H.N.; Tran, F.D.; Menaud, J. SLA-Aware Virtual Resource Management for Cloud Infrastructures. In Proceedings of the 2009 Ninth IEEE International Conference on Computer and Information Technology, Xiamen, China, 11–14 October 2009; pp. 357–362.
37. You, C.; Huang, K.; Chae, H.; Kim, B. Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 1397–1411. [CrossRef]
38. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [CrossRef]
39. Legrain, A.; Jaillet, P. A stochastic algorithm for online bipartite resource allocation problems. *Comput. Oper. Res.* **2016**, *75*, 28–37. [CrossRef]
40. De la Prieta, F.; Rodríguez, S.; Bajo, J.; Corchado, J.M. +Cloud: A Virtual Organization of Multiagent System for Resource Allocation into a Cloud Computing Environment. In *Transactions on Computational Collective Intelligence XV*; Nguyen, N.T., Kowalczyk, R., Corchado, J.M., Bajo, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 164–181. [CrossRef]
41. De la Prieta, F.; Rodríguez, S.; Bajo, J.; Corchado, J.M. A Multiagent System for Resource Distribution into a Cloud Computing Environment. In *Advances on Practical Applications of Agents and Multi-Agent Systems*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7879, pp. 37–48.
42. Bajo, J.; De la Prieta, F.; Corchado, J.M.; Rodríguez, S. A low-level resource allocation in an agent-based Cloud Computing platform. *Appl. Soft Comput.* **2016**, *48*, 716–728. [CrossRef]
43. Argente, E.; Botti, V.; Julian, V. Gormas: An organizational-oriented methodological guideline for open mas. In *International Workshop on Agent-Oriented Software Engineering*; Springer: Heidelberg, Germany, 2009; Volume 6038, pp. 32–47.
44. Corchado, J.M.; Laza, R. Constructing deliberative agents with case-based reasoning technology. *Int. J. Intell. Syst.* **2003**, *18*, 1227–1241. [CrossRef]
45. Goudarzi, H.; Pedram, M. Multi-dimensional SLA-Based Resource Allocation for Multi-tier Cloud Computing Systems. In Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, Washington, DC, USA, 4–9 July 2011; pp. 324–331.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).