Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 273537, 9 pages
http://dx.doi.org/10.1155/2014/273537

Research Article

A Green Strategy for Federated and Heterogeneous Clouds with Communicating Workloads

Jordi Mateo,[1] Jordi Vilaplana,[1] Lluis M. Plà,[2] Josep Ll. Lérida,[1] and Francesc Solsona[1]

[1] *Department of Computer Science & INSPIRES, University of Lleida, Jaume II 69, 25001 Lleida, Spain*
[2] *Department of Mathematics, University of Lleida, Jaume II 73, 25001 Lleida, Spain*

Correspondence should be addressed to Francesc Solsona; francesc@diei.udl.cat

Received 28 July 2014; Accepted 10 September 2014; Published 11 November 2014

Academic Editor: Wei-Chiang Hong

Providers of cloud environments must tackle the challenge of configuring their system to provide maximal performance while minimizing the cost of resources used. However, at the same time, they must guarantee an SLA (service-level agreement) to the users. The SLA is usually associated with a certain level of QoS (quality of service). As response time is perhaps the most widely used QoS metric, it was also the one chosen in this work. This paper presents a green strategy (GS) model for heterogeneous cloud systems. We provide a solution for heterogeneous job-communicating tasks and heterogeneous VMs that make up the nodes of the cloud. In addition to guaranteeing the SLA, the main goal is to optimize energy savings. The solution results in an equation that must be solved by a solver with nonlinear capabilities. The results obtained from modelling the policies to be executed by a solver demonstrate the applicability of our proposal for saving energy and guaranteeing the SLA.

1. Introduction

In cloud computing, SLA (service-level agreement) is an agreement between a service provider and a consumer where the former agrees to deliver a service to the latter under specific terms, such as time or performance. In order to comply with the SLA, the service provider must monitor the cloud performance closely. Studying and determining SLA-related issues are a big challenge [1, 2].

GS is designed to lower power consumption [3] as much as possible. The main objective of this paper is to develop resource scheduling approaches to improve the power efficiency of data centers by shutting down and putting idles servers to sleep, as Intel's Cloud Computing 2015 Vision [4] does.

At the same time, GS is aimed at guaranteeing a negotiated SLA and power-aware [3] solutions, leaving aside such other cloud-computing issues as variability [2], system security [3], and availability [5]. Job response time is perhaps the most important QoS metric in a cloud-computing context [1]. That is also why the QoS parameter is chosen in this work. In addition, despite good solutions having been presented by some researchers in the literature dealing with QoS [6, 7] and power consumption [8, 9], the model presented aims to obtain the best scheduling, taking both criteria into account.

This paper is focused on proposing a static green alternative to solving the scheduling problem in cloud environments. Many of the cited solutions consist of creating dynamically ad hoc VMs, depending on the workload, made up of independent tasks or a parallel job composed of communicating or noncommunicating tasks. This implies constantly creating, deleting, or moving VMs. These processes consume large amounts of time. Low ratios for return times and VM management should lead to proposing more static scheduling methods between the existing VMs. The solution described in this paper goes on this direction. However, this solution can be merged and complemented with dynamic proposals.

Our additional contribution with respect to [10] is that our solution tries to optimize 2 criteria at the same time: scheduling tasks to VMs, saving energy, and consolidating VMs to the nodes. Providing an efficient NLP solution for this problem is a novelty challenge in the cloud computing research field.

Another important contribution of this paper is method used to model the power of the virtual machines in function of their workload. Relying on the work done in [11], where the authors formulate the problem of assigning people from various groups to different jobs and who may complete them in the minimum time as a stochastic programming problem, the job completion times were assumed to follow a Gamma distribution. To model the influence of the workload, the computing power of the virtual machine is weighted by a load factor determined by an *Erlang* distribution (equivalent to a Gamma). Finally, a stochastic programming problem is obtained and transformed into an equivalent deterministic problem with a nonlinear objective function.

The remainder of the paper is organized as follows. Our contribution is based on the previous work presented in Section 2. In the GS section (Section 3), we present our main contributions, a sort of scheduling policy. These proposals are arranged by increasing complexity. The experimentation showing the good behavior of our cloud model is presented in the Results section (Section 4). Finally, the Conclusions and Future Work section outlines the main conclusions and possible research lines to explore in the near future.

2. Related Work

There is a great deal of work in the literature on linear programming (LP) solutions and algorithms applied to scheduling, like those presented in [12, 13]. Another notable work was performed in [14], where authors designed a Green Scheduling Algorithm that integrated a neural network predictor in order to optimize server power consumption in cloud computing. Also, the authors in [15] proposed a genetic algorithm that takes into account both makespan and energy consumption.

Shutting down servers when they are not being used is one of the most direct methods to reduce the idle power. However, the authors in [16] state that a power-off requires an additional setup cost, resulting in long system delays. Shutting down servers may sacrifice quality of service (QoS) levels, thus violating the SLA. They put the server work at a lower service rate rather than completely stopping work during idle periods. This drawback can be reduced if scheduling is performed for a large enough number of tasks, as in our case.

In [17], the authors treat the problem of consolidating VMs in a server by migrating VMs with steady and stable capacity needs. They proposed an exact formulation based on a linear program described by too small a number of valid inequalities. Indeed, this description does not allow solving, in a reasonable time or an optimal way, problems involving the allocation of a large number of items (or VMs) to many bins (or servers).

In [18], the authors presented a server consolidation (Sercon) algorithm which consists of minimizing the number of used nodes in a data center and minimizing the number of migrations at the same time to solve the bin (or server) packing problem. They show the efficiency of Sercon for consolidating VMs and minimizing migrations. Despite our proposal (based on NLP) always finding the best solution, Sercon is a heuristic that cannot always reach or find the optimal solution.

The authors in [19] investigated resource optimization, service quality, and energy saving by the use of a neural network. These actions were specified in two different resource managers, which sought to maintain the application's quality service in accordance with the SLA and obtain energy savings in a virtual servers' cluster by turning them off when idle and dynamically redistributing the VMs using live migration. Saving energy is only applied in the fuzzy-term "*intermediate load*," using fewer resources and still maintaining satisfactory service quality levels. Large neural network training times and their nonoptimal solutions could be problems that can be overcome by using other optimization techniques, such as the NLP one used in this paper.

In [10], the authors modelled an energy aware allocation and consolidation policies to minimize overall energy consumption with an optimal allocation and a consolidation algorithm. The optimal allocation algorithm is solved as a bin-packing problem with a minimum power consumption objective. The consolidation algorithm is derived from a linear and integer formulation of VM migration to adapt to placement when resources are released.

The authors of [20] presented an effective load-balancing genetic algorithm that spreads the multimedia service task load to the servers with the minimal cost for transmitting multimedia data between server clusters and clients for centralized hierarchical cloud-based multimedia systems. Clients can change their locations, and each server cluster only handled a specific type of multimedia task so that two performance objectives (as we do) were optimized at the same time.

In [21], the authors presented an architecture able to balance load into different virtual machines meanwhile providing SLA guarantees. The model presented in that work is similar to the model presented in this paper. The main difference is in the tasks considered. Now, a more complex and generalized model is presented. In addition, communicating and heterogeneous tasks as well as nondedicated environments have been taken into account.

Our proposal goes further than the outlined literature. Instead of designing a single criteria scheduling problem (LP), we design an NLP scheduling solution which takes into account multicriteria issues. In contrast to the optimization techniques of the literature, our model ensures the best solution available. We also want to emphasize the nondedicated feature of the model, meaning that the workload of the cloud is also considered. This also differentiates from the related work. This consideration also brings the model into reality, providing more reliable and realistic results.

3. GS Model

The NLP scheduling solution proposed in this paper models a problem by giving an objective function (OF). The equation representing the objective function takes various performance criteria into account.

GS tries to assign as many tasks as possible to the most powerful VMs, leaving the remaining ones aside. As we will consider clouds made up of various nodes, at the end of the scheduling process, the nodes all of whose VMs are not assigned any task can then be turned off. As SLA based on the minimization of the return time is also applied, the model also minimizes the computing and communication time of the overall tasks making up a job.

3.1. General Notation. Let a job made up of T communicating tasks (t^i, $i = 1, \ldots, T$), and a cloud made up of N heterogeneous nodes ($\text{Node}_1, \ldots, \text{Node}_N$).

The number of VMs can be different between nodes, so we use notation v_n ($v_n = 1, \ldots, V_n$, where $n = 1, \ldots, N$) to represent the number of VMs located to Node_n. In other words, each Node_n will be made up by VMs $\text{VM}_{n1}, \ldots, \text{VM}_{nV_n}$.

Task assignments must show the node and the VM inside the nodes task t^i is assigned to. In doing so, Boolean variables will also be used. The notation $t^i_{nv_n}$ is used to represent the assignment of task t^i to Node_n VM VM_{nv_n}.

The notation $M^i_{nv_n}$ represents the amount of *Memory* allocated to task t^i in VM VM_{nv_n}. It is assumed that Memory requirements do not change between VMs, so $M^i_{nv_n} = M^i_{nv'_n}$ $\forall n \leq N$, and $v_n, v'_n \leq V_n$. The Boolean variable $t^i_{nv_n}$ represents the assignment of task t^i to VM_{nv_n}. Once the solver is executed, the $t^i_{nv_n}$ variables will inform about the assignment of tasks to VMs. This is $t^i_{nv_n} = 1$ if t^i is assigned to VM_{nv_n}, and $t^i_{nv_n} = 0$ otherwise.

3.2. Virtual Machine Heterogeneity. The *relative computing power* (Δ_{nv_n}) of a VM_{nv_n} is defined as the *normalized score* of such a VM. Formally, consider

$$\Delta_{nv_n} = \frac{\delta_{nv_n}}{\sum_{i=1}^{V} \sum_{k=1}^{V_i} \delta_{iv_k}}, \tag{1}$$

where $\sum_{i=1}^{V} \sum_{k=1}^{V_i} \Delta_{iv_k} = 1$. δ_{nv_n} is the score (i.e., the computing power) of VM_{nv_n}. Although δ_{nv_n} is a theoretical concept, there are many valid benchmarks it can be obtained with (i.e., Linpack (Linpack. http://www.netlib.org/linpack/) or SPEC (SPEC. http://www.spec.org)). Linpack (available in C, Fortran and Java), for example, is used to obtain the number of floating-point operations per second. Note that the closer the *relative computing power* is to one (in other words, the more powerful it is), the more likely it is that the requests will be mapped into such a VM.

3.3. Task Heterogeneity. In order to model task heterogeneity, each task t^i has its *processing cost* $P^i_{nv_n}$, representing the execution time of task t^i in VM_{nv_n} with respect to the execution time of task t^i in the least powerful VM_{nv_n} (in other words, with the lowest Δ_{nv_n}). It should be a good choice to maximize $t^i_{nv_n} P^i_{nv_n}$ to obtain the best assignment (in other words, the OF) as follows:

$$\max \left(\sum_{n=1}^{N} \sum_{v_n=1}^{V_n} \sum_{i=1}^{T} t^i_{nv_n} P^i_{nv_n} \right). \tag{2}$$

However, there are still a few criteria to consider.

3.4. Virtual Machine Workload. The performance drop experienced by VMs due to workload saturation is also taken into account. If a VM is underloaded, its throughput (tasks solved per unit of time) will increase as more tasks are assigned to it. When the VM reaches its maximum workload capacity, its throughput starts falling asymptotically towards zero. This behavior can be modeled with an *Erlang* distribution density function. *Erlang* is a continuous probability distribution with two parameters, α and λ. The α parameter is called the shape parameter, and the λ parameter is called the rate parameter. These parameters depend on the VM characteristics. When α equals 1, the distribution simplifies to the exponential distribution. The *Erlang* probability density function is

$$E(x; \alpha, \lambda) = \lambda e^{-\lambda x} \frac{(\lambda x)^{\alpha - 1}}{(\alpha - 1)!} \quad \forall x, \lambda \geq 0. \tag{3}$$

We consider that the *Erlang* modelling parameters of each VM can easily be obtained empirically. The *Erlang* parameters can be obtained by means of a comprehensive analysis of all typical workloads being executed in the server, supercomputer, or data center to be evaluated. In the present work, a common PC server was used. To carry out this analysis, we continuously increased the workload until the server was saturated. We collected measurements about the mean response times at each workload variation. By empirical analysis of that experimentation, we obtained the *Erlang* that better fitted the obtained behaviour measurements.

The *Erlang* is used to weight the *Relative computing power* Δ_{nv_n} of each VM_{nv_n} with its associated workload factor determined by an *Erlang* distribution. This optimal workload model is used to obtain the maximum *throughput* performance (number of task executed per unit of time) of each VM_{nv_n}. In the case presented in this paper, the x-axis (abscisas) represents the sum of the *Processing cost* $P^i_{nv_n}$ of each t^i assigned to every VM_{nv_n}.

Figure 1 shows an example in which we depict an *Erlang* with $\alpha = 76$ and $\lambda = 15$. The *Erlang* reaches its maximum when $X = 5$. Provided that the abscissas represent the workload of a VM, a workload of 5 will give the maximum performance to such a VM in terms of throughput. So we are not interested in assigning less or more workload to a specific VM_{nv_n} because otherwise, this would lead us away from the optimal assignment.

Given an *Erlang* distribution function with fixed parameters α and λ, it is possible to calculate the optimal workload in which the function reaches the maximum by using its derivative function:

$$E(x; \alpha, \lambda)' = e^{-\lambda - 1 * x} * x(\lambda - 1 * x - \alpha - 1) \quad \forall x, \lambda \geq 0. \tag{4}$$

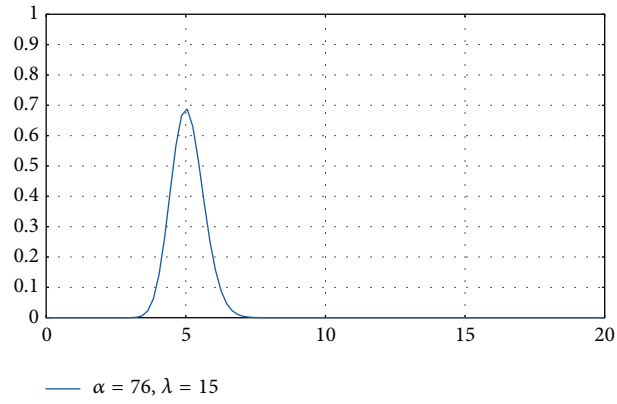

Figure 1: *Erlang* plots for different α and λ values.

Accordingly, the optimal workload of the *Erlang* example in Figure 1, with $\alpha = 76$ and $\lambda = 15$, is

$$E(x;76,15)' = e^{-14*x} * x(14x - 75) = 0$$
$$x = \frac{75}{14}. \tag{5}$$

Finally, in our case, provided that the VM workload, defined as the sum of the processing costs ($P^i_{nv_n}$) of the tasks assigned to a particular VM, must be an $\mathbb{N}$ number, the optimal workload (x) is

$$x = \text{round}\left(\frac{75}{14}\right) = 5. \tag{6}$$

Provided that Boolean variable $t^i_{nv_n} = 1$ is a boolean variable informing of the assignment of t^i to VM_{nv_n}, and $t^i_{nv_n} = 0$ if not, the *Erlang*-weighted Δ_{nv_n} would be

$$\Delta_{nv_n} E\left(\sum_{i=1}^{T} P^i_{nv_n} t^j_v; \alpha, \lambda\right). \tag{7}$$

3.5. Task Communication and VM Selection. In this section, the VM selection is also considered. In doing so, each VM can be selected from a range of N nodes, forming a federated cloud. We want to obtain an OF that considers the scheduling of heterogeneous and communicating tasks to N heterogeneous nodes made up of different numbers of heterogeneous VMs.

The *communication cost* (in time) between tasks t^i and t^j when in the same VM is denoted by C^{ij} and should be passed to the solver as an argument. For reasons of simplicity, all communication links are considered to have the same bandwidth and latency. Notation $C^{ij}_{nv_n}$ represents the *communication cost* between task t^i residing in VM_{nv_n} with another task t^j (located in the same VM or elsewhere). Provided equivalent bandwidth between any two VMs, $C^{ij}_{nv_n} = C^{ji}_{nv'_n}$ $\forall v, v' \leq V$. In other words, the *communication cost* does not depend on the VM or the links used between the VMs.

VM communication links are considered with the same bandwidth capacity. Depending on its location, we multiply the communication cost between tasks t^i and t^j by a given *communication slowdown*. If t_i and t_j are located in the same VM, the *communication slowdown* (denoted by Cs_{nv_n}) is 1. If t^j is assigned to another VM in the same node than t^i ($t^j_{n\overline{v_n}} = 1$, the *Communication slowdown* ($\text{Cs}_{n\overline{v_n}}$) will be in the range $[0, \ldots, 1]$. Finally, if t^j is assigned to another VM located in another node ($t^j_{\overline{nv_n}} = 1$), the corresponding *communication slowdown* term ($\text{Cs}_{\overline{nv_n}}$) will also be in the range $[0, \ldots, 1]$. $\text{Cs}_{n\overline{v_n}}$ and $\text{Cs}_{\overline{nv_n}}$ should be obtained with respect to Cs_{nv_n}. In other words, $\text{Cs}_{n\overline{v_n}}$ and $\text{Cs}_{\overline{nv_n}}$ are the respective reduction (in percentage) in task communication between VMs located in the same and different nodes compared with task communication inside the same VM. To sum up, $\text{Cs}_{nv_n} = 1 \geq \text{Cs}_{n\overline{v_n}} \geq \text{Cs}_{\overline{nv_n}} \geq 0$.

According to task communication, the idea is to add a component in the OF that penalizes (enhances) the communications performed between different VMs and different nodes. Grouping tasks inside the same VM will depend on not only their respective *processing cost* ($P^i_{nv_n}$) but also the *communication costs* $C^{ij}_{nv_n}$ and *communication slowdowns* Cs_{nv_n}, $\text{Cs}_{n\overline{v_n}}$, and $\text{Cs}_{\overline{nv_n}}$. We reward the communications done in the same VM but less so the ones done in different VMs while still in the same node. Finally, communications between nodes are left untouched, without rewarding or penalizing.

In the same way, if we modelled the OF in function of the tak heterogeneity in (2), the communication component will be as follows (only the communication component of the OF is shown):

$$\max\left(\sum_{n=1}^{N}\sum_{v_n=1}^{V_n}\sum_{i=1}^{T}\left(t^i_{nv_n}\sum_{j<i}^{T}C^{ij}_{nv_n}\left(t^j_{nv_n}\text{Cs}_{nv_n} + t^j_{n\overline{v_n}}\text{Cs}_{n\overline{v_n}} + t^j_{\overline{nv_n}}\text{Cs}_{\overline{nv_n}}\right)\right)\right). \tag{8}$$

And the OF function will be

$$\max\left(\sum_{n=1}^{N}\sum_{v_n=1}^{V_n}\left(\left(\sum_{i=1}^{T}t^i_{nv_n}\left(P^i_{nv_n} + \sum_{j<i}^{T}C^{ij}_{nv_n}\left(t^j_{nv_n}\text{Cs}_{nv_n} + t^j_{n\overline{v_n}}\text{Cs}_{n\overline{v_n}} + t^j_{\overline{nv_n}}\text{Cs}_{\overline{nv_n}}\right)\right)\right) \times \Delta_{nv_n} E\left(\sum_{j=1}^{T}t^j_{nv_n}P^j_{nv_n}; \alpha, \lambda\right)\right)\right). \tag{9}$$

3.6. Choosing SLA or Energy Saving. It is important to highlight that the optimization goal is two criteria (SLA and energy in our case). Thus, the user could prioritize

the criteria. Assignments are performed starting from the most powerful VM. When this becomes saturated, task assignment continues with the next most powerful VM, regardless of the node it resides in. When this VM resides in another node (as in our case), the energy-saving criteria will be harmed. It would be interesting to provide a means of increasing criteria preferences in the model presented.

In order to highlight specific criteria (i.e., energy saving), one more additional component must be added to the OF. This component must enhance the assignment of tasks to the same node by assigning tasks to the most powerful nodes and not only to the most powerful VMs as before. This is the natural procedure to follow, because the OF is a maximum. Thus, the likelihood of less powerful nodes becoming idle increases and this gives the opportunity to power them off, hence saving energy.

The additional component can be defined in a similar way as for the relative *VM computing power* (Δ_{nv_n}) of a VM_{nv_n}. Instead, we obtain the relative *node computing power* of a Node_n (Θ_n) as the normalized summatory of their forming VMs. Θ_n will inform about the computing power of Node_n.

For N nodes, Θ_n is formally defined as

$$\Theta_n = \frac{\sum_{v_n=1}^{V_n} \Delta_{nv_n}}{\sum_{n=1}^{N} \sum_{v_n=1}^{V_n} \Delta_{nv_n}}, \tag{10}$$

where $\sum_{n=1}^{N} \Theta_n = 1$. To obtain Θ_n, the parallel Linpack version (HPL: high performance Linpack) can be used. It is the one used to benchmark and rank supercomputers for the TOP500 list.

Depending on the importance of the energy saving criteria, a weighting factor should be provided to Θ_n. We simply call this factor energy Energy Ξ. The Ξ will be in the range $(0, \ldots, 1]$. For an $\Xi 0$, our main criteria will be energy saving, and for $\Xi = 1$, our goal is only SLA. Thus, the resulting energy component will be $\Theta_n \Xi$. Thus, for a given Node_n with Θ_n, we must weigh the energy saving criteria of such a node by the following factor:

$$\sum_{v_n=1}^{V_n} \sum_{i=1}^{T} t^i_{nv_n} \Theta_n \Xi. \tag{11}$$

The resulting OF function will be

$$\begin{aligned} \max \Bigg(& \sum_{n=1}^{N} \Bigg(\sum_{v_n=1}^{V_n} \sum_{i=1}^{T} t^i_{nv_n} \Theta_n \Xi \Bigg) \\ & \times \sum_{v_n=1}^{V_n} \Bigg(\Bigg(\sum_{i=1}^{T} t^i_{nv_n} \Bigg(P^i_{nv_n} + \sum_{j<i}^{T} C^{ij}_{nv_n} \left(t^j_{nv_n} \text{Cs}_{nv_n} + t^j_{n\overline{v}_n} \text{Cs}_{n\overline{v}_n} + t^j_{\overline{nv}_n} \text{Cs}_{\overline{nv}_n} \right) \Bigg) \Bigg) \\ & \times \Delta_{nv_n} E \left(\sum_{j=1}^{T} t^j_{nv_n} P^j_{nv_n}; \alpha, \lambda \right) \Bigg) \Bigg) \Bigg). \end{aligned} \tag{12}$$

3.7. Enforcing SLA. For either prioritized criteria, SLA or energy saving, there is a last consideration to be taken into account.

Imagine the case where tasks do not communicate. Once they are assigned to a node, one would expect them to be executed in the minimum time. In this case, there is already no need to group tasks in the VM in decreasing order of power in the same node, because this node is no longer eligible to be switched off. A better solution in this case would be to balance the tasks between the VMs of such a node in order to increase SLA performance. Note that this not apply in the communicating tasks due to the communication slowdown between VMs.

To implement this, we only need to assign every noncommunicating tasks without taking the *relative computing power* (Δ_{nv_n}) of each VM into account.

We only need to replace Δ_{nv_n} in (12) by Δ, defined as

$$\Delta = \text{if } \left(\sum_{j<i}^{T} C^{ij}_{nv_n} \geq 0 \right) \Delta_{nv_n} \quad \text{else } 1. \tag{13}$$

For the case of noncommunicating tasks, by assigning a $\Delta = 1$, all the VMs have the same *relative computing power* Δ_{nv_n}. Thus, tasks are assigned in a balanced way.

3.8. Model Formulation. Finally, the OF function and their constraints are presented. The best task scheduling assignment to VMs which takes all the features into account (*GS* policy) is formally defined by the following nonlinear programming model:

$$\begin{aligned} \max \Bigg(& \sum_{n=1}^{N} \Bigg(\sum_{v_n=1}^{V_n} \sum_{i=1}^{T} t^i_{nv_n} \Theta_n \Xi \Bigg) \\ & \times \sum_{v_n=1}^{V_n} \Bigg(\Bigg(\sum_{i=1}^{T} t^i_{nv_n} \Bigg(P^i_{nv_n} + \sum_{j<i}^{T} C^{ij}_{nv_n} \left(t^j_{nv_n} \text{Cs}_{nv_n} + t^j_{n\overline{v}_n} \text{Cs}_{n\overline{v}_n} + t^j_{\overline{nv}_n} \text{Cs}_{\overline{nv}_n} \right) \Bigg) \Bigg) \\ & \times \Delta E \left(\sum_{j=1}^{T} t^j_{nv_n} P^j_{nv_n}; \alpha, \lambda \right) \Bigg) \Bigg) \end{aligned} \tag{14a}$$

$$\text{s.t.} \quad \sum_{i=1}^{T} {Mnv_n}^i \leq Mnv_n \quad \forall n \leq N,\ v_n \leq V_n \tag{14b}$$

$$\sum_{n=1}^{N} \sum_{v_n=1}^{V} t^i_{nv_n} = 1 \quad \forall i \leq T. \tag{14c}$$

Equation (14a) is the objective function (OF) to be maximized. Note that OF is an integer and nonlinear problem. Inequality in (14b) and equality in (14c) are the constraints of the objective function variables. Given the constants T (the total number of requests or tasks), and M_v for each VM_{nv_n}, the solution that maximizes OF will obtain the values of the variables $t^i_{nv_n}$, representing the number of tasks assigned to VM_{nv_n}. Thus, the $t_{nv_n i}$ obtained will be the assignment found by this model.

OF takes into account the *processing costs* ($P^i_{nv_n}$) and the *communication times* ($C^{ij}_{nv_n}$) of the tasks assigned to each VM_{nv_n} and the *communication slowdowns* between VMs $\mathrm{Cs}_{\overline{nv_n}}$ and nodes $\mathrm{Cs}_{\overline{\overline{nv_n}}}$. $\mathrm{Cs}_{nv_n} = 1$. Δ is defined in Section 3.7. And $E(\sum_{j=1}^{T} t^j_{nv_n} P^j_{nv_n}; \alpha, \lambda)$ represents the power slowdown of each VM due to its workload (defined in Section 3.4).

To sum up, for the case when the workload is made up of noncommunicating tasks, if we are interested in prioritizing the SLA criteria, OF 3.5 should be applied. If, on the contrary, the goal is to prioritize energy saving, OF (14a) should be used instead.

4. Results

In this section, we present the theoretical results obtained from solving the scheduling problems aimed at achieving best task assignment. Two representative experiments were performed in order to test the performance of GS.

The experiments were performed by using the AMPL (AMPL. A Mathematical Programming Language. http://ampl.com) language and the SCIP (SCIP. Solving Constraint Integer Programs. http://scip.zib.de) solver. AMPL is an algebraic modeling language for describing and solving high-complexity problems for large-scale mathematical computation supported by many solvers. Integer and nonlinear (our model type) problems can be solved by SCIP, one of the solvers supported by AMPL.

Throughout all the experimentation, the *Erlang arguments* were obtained empirically by using the strategy explained in Section 3.4.

As the objective of this section is to prove the correctness of the policy, only a small set of tasks, VMs, and nodes was chosen. The size of the experimental framework was chosen to be as much representative of actual cases as possible, but at the same time, simple enough to be used as an illustrative example. So, the experimental framework chosen was made up of 2 different nodes: one of them comprised 3 VMs and the other 1 VM; see Figure 2. The objective of this simulation was to achieve the best assignment for 3 tasks. Table 1 shows the *processing cost* $P^i_{nv_n}$, relating the execution times of the tasks in each VM. To show the good behavior of the model presented, each task has the same *processing cost* independently of the VM. The model presented can be efficiently applied to real cloud environments. The only weak point is that the model is static. That means that homogenous and static workload conditions must be stable in our model. Job executions in different workload sizes can be saved in a database system, providing a means for determining the SLA of such a job in future executions.

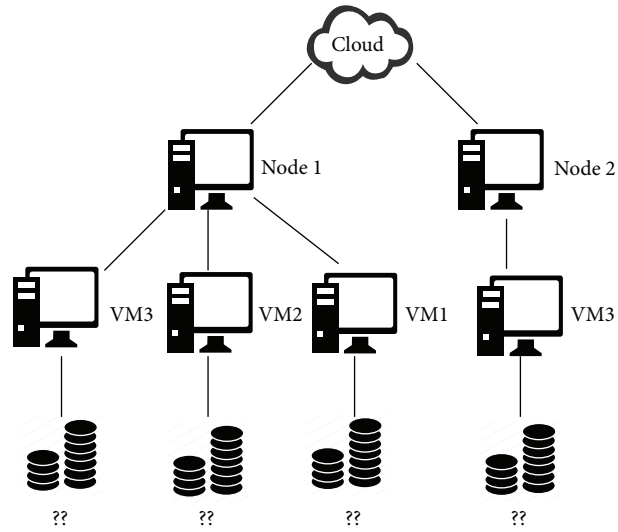

Figure 2: Cloud architecture.

Table 1: Task processing costs.

Task t^i	Processing costs $P^i_{nv_n}$	Value
t^1	$P^1_{11}, P^1_{12}, P^1_{13}, P^1_{21}$	**1**
t^2	$P^2_{1}, P^2_{12}, P^2_{13}, P^2_{21}$	**5**
t^3	$P^3_{11}, P^3_{12}, P^3_{13}, P^3_{21}$	**1**
Total processing cost $\left(\sum_i P^i_{nv_n}\right)$		**7**

4.1. Without Communications. In this section, a hypothetical situation without communication between tasks is evaluated. In this situation, our scheduling policy tends to assign the tasks to the most powerful set of virtual machines (i.e., with the higher *relative computing power* Δ_{nv_n}, considering their individual saturation in this choice). This saturation becomes critical when more and more tasks are added. Here, the most important term is the *Erlang* function, since it models the behaviour of every virtual machine. Thus, taking this into account, our scheduler knows the exact weight of tasks it can assign to the VMs in order to obtain the best return times. This phenomenon is observed in the following examples.

4.1.1. Without Communications and High Optimal Erlang. Table 2 shows the parameters used in the first example. The amount of *Memory* allocated to each task t^i in every VM M_{nv_n} (as we supposed this amount to be equal in all the VMs, we simply call it M^i). The *relative computing power* (Δ_{nv_n}) of each VM and finally the α and λ *Erlang arguments*. Note that all the VMs have the same *Erlang* parameters. The parameters were chosen this way because any VM saturates with the overall workload assigned. In other words, the total *processing cost* 7 is lower than the optimal *Erlang* workload 16.

Table 3 shows the solver assignment results. The best scheduling assigns all the tasks to the same VM (VM_{21}, the only VM in node 2), because this VM has the biggest *relative*

Table 2: Without communications. High optimal *Erlang*. VM configurations.

Node	VM_{nv_n}	M^i	Δ_{nv_v}	*Erlang*
1	VM_{11}	**10**	**0.75**	$\alpha = 3, \lambda = 8$
1	VM_{12}	**10**	**0.35**	$\alpha = 3, \lambda = 8$
1	VM_{13}	**10**	**0.1**	$\alpha = 3, \lambda = 8$
2	VM_{21}	**10**	**0.85**	$\alpha = 3, \lambda = 8$

Table 3: Without communications. High optimal *Erlang*. Solver assignment.

Node	VM_{nv_n}	Task assignment
1	VM_{11}	**0**
1	VM_{12}	**0**
1	VM_{13}	**0**
2	VM_{21}	t^1, t^2, t^3

computing power (Δ_{nv_v}). This result is very coherent. Due to the lack of communications, the model tends to assign tasks to the most powerful VM while its workload does not exceed the *Erlang* optimum (a workload of 10 tasks). As in our case, the total workload is 7, and VM_{21} could host even more tasks.

4.1.2. Without Communications, Low Optimal Erlang, and Preserving SLA. In this example (see Table 4), the VMs have another *Erlang*. However, the *task processing costs* do not change, so they remain the same as in Table 1. In this case, each VM becomes saturated when the assignment workload weight is higher than 5 (because 5 is the optimal workload).

The best assignment in this case is the one formed by the minimum set of VMs with the best *relative computing power* Δ_{nv_v} (see Table 5 column Task Assignment SLA). The assignment of the overall tasks to only one VM (although it was the most powerful one) as before will decrease the return time excessively, due to its saturation.

4.1.3. Without Communications, Low Optimal Erlang, and Preserving Energy Saving. Provided that the most important criterion is the energy saving, the assignment will be somewhat different (see Table 5, column Task Assignment Energy). In this case, OF (14a) with $\Xi = 1$ was used. Then, as expected, all the tasks were again assigned to VM_{21} of $Node_2$.

4.2. With Communications. Starting from the same VM configuration shown on Table 4, in this section we present a more real situation where the costs of communications between tasks are also taken into account. It is important to highlight that in some situations, the best choice does not include the most powerful VMs (i.e., with the highest *relative computing power* Δ_{nv_n}). Thus, the results shown in this section must show the tradeoff between *relative computing power* of VM, workload scheduling impact modeled by the *Erlang* distribution, and communication efficiency between tasks.

4.2.1. High Communication Slowdown. This example shows the behaviour of the model under large communication costs

Table 4: Without communications. Low optimal *Erlang*. Preserving SLA. VM configurations.

Node	VM_{nv_n}	M^i	Δ_{nv_v}	*Erlang*
1	VM_{11}	**10**	**0.75**	$\alpha = 5, \lambda = 1$
1	VM_{12}	**10**	**0.35**	$\alpha = 5, \lambda = 1$
1	VM_{13}	**10**	**0.1**	$\alpha = 5, \lambda = 1$
2	VM_{21}	**10**	**0.85**	$\alpha = 5, \lambda = 1$

Table 5: Without communications. Low optimal *Erlang*. Preserving SLA. Solver assignment.

Node	VM_{nv_n}	SLA Task assignment	Energy Task assignment
1	VM_{11}	t^1, t^3	t^2
1	VM_{12}	**0**	t^1, t^3
1	VM_{13}	**0**	**0**
2	VM_{21}	t^2	**0**

Table 6: High slowdown. Communication configurations.

Task t^i	Task t^j	Communication costs C^{ij}
t^1	t^2	**0.2**
t^1	t^3	**0.6**
t^2	t^3	**0.3**
	$Cs_{n\overline{v}_n}$	**0.2**
	$Cs_{\overline{nv}_n}$	**0.1**

between VMs (see Table 6). This table shows the *communication costs* between tasks (C^{ij}) and the *communication slowdown* when communications are done between VMs in the same node ($Cs_{n\overline{v}_n}$) and the penalty cost when communications are performed between different nodes ($Cs_{\overline{nv}_n}$). Note that the penalties are very high (0.2 and 0.1) when the communications are very influential.

The solver assignment is shown in Table 7. In order to avoid communication costs due to slowdowns, the best assignment tends to group tasks first in the same VM and second in the same node. Although VM_{21} should become saturated with this assignment, the high communication cost compensates the loss of SLA performance, allowing us to switch off node 1.

4.2.2. Low Communication Slowdown. Now, this example shows the behaviour of our policy under more normal communication conditions. Here, the communication penalties between the different VMs or nodes are not as significant as in the previous case because $Cs_{n\overline{v}_n}$ and $Cs_{\overline{nv}_n}$ are higher. Table 8 shows the communication costs between tasks and the penalty cost if communications are performed between VMs in the same node ($Cs_{n\overline{v}_n}$) or between different nodes ($Cs_{\overline{nv}_n}$).

In this case, the solver got as a result two hosting VMs (see Table 9) formed by the VM_{11} (with the assigned tasks t^2 and t^3) and VM_{21} with task t^1. In this case, due to the low

Table 7: High slowdown. Solver assignment.

Node	VM_{nv_n}	Task assignment
1	VM_{11}	**0**
1	VM_{12}	**0**
1	VM_{13}	**0**
2	VM_{21}	t^1, t^2, t^3

Table 8: Low slowdown. Communication configurations.

Task t^i	Task t^j	Communication costs C^{ij}
t^1	t^2	**0.2**
t^1	t^3	**0.6**
t^2	t^3	**0.3**
$Cs_{n\overline{v_n}}$		**0.9**
$Cs_{\overline{nv_n}}$		**0.8**

Table 9: Low slowdown. Solver assignment.

Node	VM_{nv_n}	Task assignment
1	VM_{11}	t^1, t^3
1	VM_{12}	**0**
1	VM_{13}	**0**
2	VM_{21}	t^2

differences between the different communication slowdowns, task assignment was distributed between the two nodes.

4.2.3. Moderate Communication Slowdown. We simulated a more normal situation, where the communication slowdown between nodes is higher than the other ones. From the same example, it was only reduced $Cs_{\overline{nv_n}}$ to 0.4 (see Table 10). As expected, the resulting solver assignment was different from that in the previous case. Theoretically, this assignment should assign tasks to the powerful unsaturated VMs, but as much as possible to the VMs residing on the same node. The solver result was exactly what was expected. Although the most powerful VM is in node 2, the tasks were assigned to the VMs of node 1, because they all fit in the same node. That is the reason why a less powerful node like node 1, but one with more capacity, is able to allocate more tasks than node 2 due to the communication slowdown between nodes. These results are shown in Table 11.

5. Conclusions and Future Work

This paper presents a cloud-based system scheduling mechanism called GS that is able to comply with low power consumption and SLA agreements. The complexity of the model developed was increased, thus adding more factors to be taken into account. The model was also tested using the AMPL modelling language and the SCIP optimizer. The results obtained proved consistent over a range of scenarios. In all the cases, the experiments showed that all the tasks were assigned to the most powerful subset of virtual machines by keeping the subset size to the minimum.

Table 10: Moderate slowdown. Communication configurations.

Task t^i	Task t^j	Communication costs C^{ij}
t^1	t^2	**0.2**
t^1	t^3	**0.6**
t^2	t^3	**0.3**
$Cs_{n\overline{v_n}}$		**0.9**
$Cs_{\overline{nv_n}}$		**0.4**

Table 11: Moderate slowdown. Solver assignment.

Node	VM_{nv_n}	Tasks assignment
1	VM_{11}	t^2, t^3
1	VM_{12}	t^1
1	VM_{13}	**0**
2	VM_{21}	**0**

Although our proposals still have to be tested in real scenarios, these preliminary results corroborate their usefulness.

Our efforts are directed towards implementing those strategies in a real cloud environment, like the OpenStack [22] or OpenNebula [23] frameworks.

In the longer term, we consider using some statistical method to find an accurate approximation of the workload using well-suited *Erlang* distribution functions.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the MEYC under Contracts TIN2011-28689-C02-02. The authors are members of the research groups 2009-SGR145 and 2014-SGR163, funded by the Generalitat de Catalunya.

References

[1] R. Aversa, B. Di Martino, M. Rak, S. Venticinque, and U. Villano, *Performance Prediction for HPC on Clouds. Cloud Computing: Principles and Paradigms*, John Wiley & Sons, New York, NY, USA, 2011.

[2] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '11)*, pp. 104–113, May 2011.

[3] J. Varia, *Architection for the Cloud: Best Practices*, Amazon Web Services, 2014.

[4] Intel cloud computing 2015 vision, http://www.intel.com/content/www/us/en/cloud-computing/.

[5] M. Martinello, M. Kaâniche, and K. Kanoun, "Web service availability—impact of error recovery and traffic model," *Reliability Engineering & System Safety*, vol. 89, no. 1, pp. 6–16, 2005.

[6] J. Vilaplana, F. Solsona, F. Abella, R. Filgueira, and J. Rius, "The cloud paradigm applied to e-Health," *BMC Medical Informatics and Decision Making*, vol. 13, no. 1, article 35, 2013.

[7] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 492–507, 2014.

[8] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," *Concurrency Computation Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[9] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[10] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: exact allocation and migration algorithms," in *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid '13)*, pp. 671–678, May 2013.

[11] M. F. Khan, Z. Anwar, and Q. S. Ahmad, "Assignment of personnels when job completion time follows gamma distribution using stochastic programming technique," *International Journal of Scientific & Engineering Research*, vol. 3, no. 3, 2012.

[12] J. L. Lérida, F. Solsona, P. Hernández, F. Giné, M. Hanzich, and J. Conde, "State-based predictions with self-correction on Enterprise Desktop Grid environments," *Journal of Parallel and Distributed Computing*, vol. 73, no. 6, pp. 777–789, 2013.

[13] A. Goldman and Y. Ngoko, "A MILP approach to schedule parallel independent tasks," in *Proceedings of the International Symposium on Parallel and Distributed Computing (ISPDC '08)*, pp. 115–122, Krakow, Poland, July 2008.

[14] T. Vinh, T. Duy, Y. Sato, and Y. Inoguchi, "Performance evaluation of a green scheduling algorithm for energy savings in cloud computing," in *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW '10)*, pp. 1–8, Atlanta, Ga, USA, April 2010.

[15] M. Mezmaz, N. Melab, Y. Kessaci et al., "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 11, pp. 1497–1508, 2011.

[16] Y. C. Ouyang, Y. J. Chiang, C. H. Hsu, and G. Yi, "An optimal control policy to realize green cloud systems with SLA-awareness," *The Journal of Supercomputing*, vol. 69, no. 3, pp. 1284–1310, 2014.

[17] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. de Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.

[18] A. Murtazaev and S. Oh, "Sercon: server consolidation algorithm using live migration of virtual machines for green computing," *IETE Technical Review (Institution of Electronics and Telecommunication Engineers, India)*, vol. 28, no. 3, pp. 212–231, 2011.

[19] A. F. Monteiro, M. V. Azevedo, and A. Sztajnberg, "Virtualized Web server cluster self-configuration to optimize resource and power use," *Journal of Systems and Software*, vol. 86, no. 11, pp. 2779–2796, 2013.

[20] C.-C. Lin, H.-H. Chin, and D.-J. Deng, "Dynamic multiservice load balancing in cloud-based multimedia system," *IEEE Systems Journal*, vol. 8, no. 1, pp. 225–234, 2014.

[21] J. Vilaplana, F. Solsona, J. Mateo, and I. Teixido, "SLA-aware load balancing in a web-based cloud system over OpenStack," in *Service-Oriented Computing—ICSOC 2013 Workshops*, vol. 8377 of *Lecture Notes in Computer Science*, pp. 281–293, Springer International Publishing, 2014.

[22] OpenStack, http://www.openstack.org/.

[23] OpenNebula, http://www.opennebula.org.