Quality of service modeling for green scheduling in Clouds

Tom Guérout, Samir Medjiah, Georges da Costa, Thierry Monteil

Quality of Service modeling for green scheduling on Clouds

Tom Guérout[a,b,c], Samir Medjiah[a,d], Georges Da Costa[b], Thierry Monteil[a,c]

[a] *CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France*
[b] *IRIT/Toulouse University, 118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9*
[c] *Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France*
[d] *Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France*

Abstract

Best known Cloud providers propose services under constraints of Service Level Agreement (SLA) definitions. The SLAs are composed of different Quality of Service (QoS) rules promised by the provider. Thus, the QoS in Clouds becomes more and more important. Precise definitions and metrics have to be explained. This article proposes an overview of Cloud QoS parameters as well as their classification, but also it defines usable metrics to evaluate QoS parameters. Moreover, the defined QoS metrics are measurable and reusable in any scheduling approach for Clouds. Energy consumption is an inherent objective in Cloud Computing, thus, it is also considered. For evaluation purposes, two uncommon QoS parameters: *Dynamism* and *Robustness* are taken into account in different Cloud virtual machines scheduling approaches. Validation is done through comparison of common scheduling algorithms, including a genetic algorithm (GA), in terms of QoS parameters evolution in time. Simulation results have shown that including various QoS parameters allows a deeper scheduling algorithms analysis.

Keywords: Quality of Service, Energy-Efficiency, Multi-objectives optimization, Cloud computing, Dynamic Voltage and Frequency Scaling (DVFS), Genetic Algorithm, Simulation

1. Introduction

Cloud **Q**uality **o**f **S**ervice (QoS) [1] parameters are used by Clouds providers to establish their **S**ervice **L**evel **A**greement (SLA) contract. From the user's point of view, SLA parameters correspond to the facilities they can have using a service, mainly in terms of performance, virtual machines renting cost or availability. From the service provider point of view, the QoS analysis can be very different. Two main goals can be distinguished: Clouds providers define their SLA with the aim to satisfy their users by achieving high performance and ensuring a safe functioning. This first aim is related to the user point of view, but the second one is more dedicated to providers interests as the energy consumption can help to secure certain amount of benefits, while for example, trying to minimize the amount of virtual machine provisioning to respect SLA guarantees. Many studies have been carried out in these domains using benchmarks, metrics and technical solutions [2, 3, 4, 5] and they are also widely proposed in cloud scheduling studies.

In current Cloud quality of service researches, the main objective metrics are the energy consumption, the virtual machine provisioning, services response-time, and, from an economical view, the cost can be taken into account. Indeed, these objectives are relevant. The energy consumption is a key issue for the environment as data-centers are energy consuming, and the cost and performance of Cloud services are the most important parameters the users are looking for.

Nowadays, the analysis of Clouds quality of service parameters can be performed in depth. The main contribution of this article is the definition of Clouds QoS parameters, which are organized into four categories. Each QoS parameter is described, ranging from the standard network performance analysis to more non-common Clouds parameters. The contribution proposed in this article is threefold. First, this article proposes a large list, but also defines as much as possible metrics that allow to evaluate them. These metrics are intended to be

Email addresses: tguerout@laas.fr (Tom Guérout), smedjiah@laas.fr (Samir Medjiah), dacosta@irit.fr (Georges Da Costa), monteil@laas.fr (Thierry Monteil)

measurable and reusable by anyone who wants to take care about Cloud quality of service in research studies. Second, this article highlights how this quality of service study can be used in any scheduling approaches, and how it could help to enrich multi-objectives algorithms evaluation. Finally, the article proposes a Cloud architecture modeling, including the use of the DVFS [6] (Dynamic Voltage & Frequency Scaling) tools which give an interesting trade-off between performance and energy-efficiency.

This article is organized as follow. The related works section surveys some references of different Cloud modeling, different approaches of Clouds scheduling and SLAs definitions of the current best known Cloud providers. Section 3 describes the contributions of this article. It details the Cloud architecture model defined for this study. It also contains the enumeration of quality of service parameters, and explains how the proposed QoS metrics can improve scheduling approaches for Clouds. The validation methodology, in Section 4, presents in details the Genetic Algorithm (GA) implemented for the evaluation phase, and two other basic algorithms. In Section 5, the simulations results of these three scheduling approaches are compared and analyzed. Finally, Section 6 concludes this article and discusses future works.

2. Related work

This *Related works* Section presents previous works in three different Clouds fields: modeling, scheduling studies and an analysis of real Clouds providers SLA proposition. In the domain of Clouds modeling, the main issue is to expose a model which points out clearly the essential characteristics. In the survey [7], some models of Clouds have been presented and widely analyzed. The following Clouds modeling related works section summarizes the more relevant articles which propose both energy and quality of service models or approaches that make use of CPU frequency scaling. In Cloud scheduling studies, a detailed analysis of previous works has been carried out in [8] with a focus on works that combine complex energy-efficient solutions and QoS parameters evaluation. As this article points out Cloud quality of service modeling, an analysis of actual SaaS Clouds providers SLA propositions is also proposed.

2.1. Cloud modeling

Abdelsalam et al. [9] have analyzed the mathematical relationship between SLAs and the number of servers used. The energy consumption is also taken into account and their Cloud model uses homogeneous hosts with DVFS enabled, allowing each physical machine to use different frequencies. Their study includes the number of users and their needs, and the average response time of users requests.

Islam et al. [10] have developed an elasticity model for cloud instances. They have assumed that each resource type (CPU, memory, network bandwidth, etc.) can be allocated in units and the users are aware of the allocated resources and the relevant QoS metrics for their requests, as in the case of Amazon CloudWatch [1]. Their proposed model combines the cost of provisioned but underutilized resources and the performance degradation cost due to under-provisioning. The consumer's detriment in over-provisioning is the difference between chargeable supply and demand, while the cost of under-provisioning is quantified through the percentage of rejected requests. Authors have also assumed that the customers are able to convert the latter into the estimated their financial impact.

Gelenbe et al. [11] have formulated an optimization problem for load sharing between a local and a remote cloud service. Their study defines a multi-objective function formulated to optimize response time and energy consumption per job. Their approach also includes a Poisson arrivals jobs rate.

Baliga et al. [12] proposes a study of the energy consumption for processing large amounts of data, management, and switching of communications. Their article associated three cloud computing services, namely storage as a service, software as a service and processing as a service. The energy consumption was considered as an integrated supply chain logistics problem involving processing, storage, and transport and has been analyzed in both public and private Clouds.

Garg et al. [13] have modeled various energy characteristics, such as energy cost, carbon emission rate, workload and CPU power efficiency. Their model considers homogeneous CPUs, a cooling system which depends on a coefficient of performance (COP), and the use of the DVFS. Their performance evaluation includes many metrics and many parameters: average energy consumption, average carbon emission, profit gained, urgency class and arrival rate of applications and data transfer cost.

[1] http://aws.amazon.com/fr/cloudwatch/

Beloglazov et al. [14] have studied the performance degradation of virtual machines taking into account CPU, memory and bandwidth utilization. The virtual machine provisioning is defined as a QoS parameter, and an SLA violation occurs when a virtual machine does not have the required amount of CPU.

Mi et al. [15] have formulated the multi-constraint optimization problem of finding the optimal virtual machine consolidation on hosts while minimizing the power consumption. An application load prediction is computed and they proposed a heuristic based on genetic algorithms in order to find a near optimal reconfiguration policy. The objective function (i.e. fitness) is composed of the power consumption function and a penalty function to keep the CPU utilization between two threshold values.

2.2. Cloud green scheduling

Beloglazov et al. [16] propose a resource management system for efficient power consumption that reduces operating costs and provides quality of service. Energy saving is achieved through the continued consolidation of virtual machines according to resource utilization. The QoS is modeled by the amount of resource needed in Millions Instructions Per Second (MIPS) for the CPU, the amount of Memory (RAM), and by the network bandwidth rate. An SLA violation occurs when a VM cannot have the required three amounts.

Duy et al. [17] design, implement, and evaluate a scheduling algorithm integrating a predictor of neural networks to optimize the power consumption of servers in a cloud. The prediction of future workload is based on demand history. According to the prediction, the algorithm turns off unused servers and restarts them to minimize the number of servers running, thus also minimizing the energy consumption.

Binder and Suri [18] propose an algorithm of allocation and dispatching of tasks that minimizes the number of required active servers, managing to reduce energy consumption which is inversely proportional to the number of concurrent threads running in workloads.

Srikantaiah et al. [19] study how to obtain consolidation of energy efficiency based on the interrelationship among energy consumption, resource utilization, and performance of consolidated workloads. It is shown that there is an optimal point of operation among these parameters based on the Bin Packing problem applied to the problem of consolidation.

In [20], Dasgupta et al. pose workload normalization across heterogeneous systems, such as clouds, as a challenge to be addressed. The main differences between the proposed algorithm and other research works are the focus on heterogeneous data-centers and energy benefits provided by active cooling control for unknown sized workload processing. This study takes into account a Return On Investment (ROI) analysis, and also proposes a dynamic power saving case study using the DVFS tools.

2.3. SaaS Clouds providers SLAs propositions

2.3.1. Amazon

The recent Service Level Agreement of Amazon EC2 (Elastic Compute Cloud) and EBS (Amazon Elastic Block Store) definition is dated on June 1, 2013. From the Amazon EC2 website [2] an SLA is introduced as: *is a policy governing the use of Amazon Elastic Compute Cloud ("Amazon EC2") and Amazon Elastic Block Store ("Amazon EBS") under the terms of the Amazon Web Services Customer Agreement (the "AWS Agreement") between Amazon Web Services, Inc. ("AWS", "us" or "we") and users of AWS' services ("you"). This SLA applies separately to each account using Amazon EC2 or Amazon EBS.*

Amazon defines the terms *Region Unavailable* and *Region Unavailability*. It means that more than one *Availability Zone* in which you are running an instance, within the same Region, is *Unavailable* to you. Definitions of *Unavailable* and *Unavailability* are the following:

- For Amazon EC2, when all user's running instances have no external connectivity.
- For Amazon EBS, when all user's attached volumes perform zero read write IO, with pending IO in the queue.

Another definition concerns the *Monthly Uptime Percentage*, it is calculated by subtracting from 100% the percentage of minutes during the month in which Amazon EC2 or Amazon EBS, as applicable was in the state of *Region Unavailable*.

In the *AWE Customers agreement*, some QoS aspects are defined: *Service offering change*, *APIs update*, *Data security & privacy*, *Availability and impact of temporary service suspension*

[2] `http://aws.amazon.com/ec2/sla/`

2.3.2. Oracle

Oracle describe their SLA propositions in a document called *Oracle Cloud Services Agreements* [3] which is available in many countries. In this Cloud Services Agreements, first, there is a list of terms definition that compose their agreements. Then, this document is composed of general categories like *Rights Granted,Service Specifications*, *Use of the Services* or *Services Period, End of Services*, etc. But, more precise QoS parameter definitions can be found in the *Oracle Fusion Middleware Deployment Planning Guide for Oracle Directory Server Enterprise Edition* website [4], which also includes Cloud services. For example, System Qualities are identified in different categories:

- *Performance*: The measurement of response time and throughput with respect to user load conditions.
- *Availability*: A measure of how often a system's resources and services are accessible to end users, often expressed as the uptime of a system.
- *Scalability*: The ability to add capacity and users to a deployed system over time. Scalability typically involves adding resources to the system without changing the deployment architecture.
- *Security*: A complex combination of factors that describes the integrity of a system and its users. Security includes authentication and authorization of users, security of data, and secure access to a deployed system.
- *Latent capacity*: The ability of a system to handle unusual peak loads without additional resources. Latent capacity is a factor in availability, performance, and scalability.
- *Serviceability*: The ease by which a deployed system can be maintained, including monitoring the system, fixing problems that arise, and upgrading hardware and software components.

2.3.3. Microsoft Azure

Microsoft Azure [5] diffuse their SLA in a document called *Windows Azure Cloud Services, Virtual Machines, and Virtual Network Service Level Agreement (SLA)*. This document details: Definitions of terms, Service Credit Claims, SLA exclusions, Service Credits and Service Level which is divided in three parts containing information about *Monthly Connectivity Uptime Service Level* of: Cloud Services, Virtual Machines and Virtual Network.

Unlike SLA parameters described by SaaS Clouds providers, as described just above, which tackle various domains of quality of service aspects, actual Cloud scheduling or modeling studies do not contain models, definitions or metrics that include this kind of parameters. This ascertainment leads to focus this article on the analysis of Cloud QoS parameters, their definition, and metrics definition when it is relevant.

3. Contributions

This section details how the problem has been analyzed and modeled. The first subsection explains how it can be relevant to consider more QoS parameters, based on the QoS analysis contribution of Section 3.3, for scheduling algorithms. The second subsection is dedicated to the Cloud architecture model used in this article, in which each component is clearly defined. The last subsection contains an extensive list of QoS parameters for Clouds which compose the main contribution of this article.

3.1. How to enrich green scheduling

Most of the time, cloud scheduling algorithms are based on energy and response-time optimization. In the field of Clouds, another common metric is to take into account, from an end-user view, the cost induced by the renting of the services (several virtual machines during several hours), or from the provider view, the benefit that it can take out from this process.

The analysis of Cloud providers SLAs propositions in Section 2 allows to understand that even if one or two new quality of service parameters was integrated in scheduling studies that meets the service provider interests, it will be very helpful to any scheduling approach. For example, if the *Latent capacity* cited in the Oracle SLA definition was integrated as an optimized metric, in addition to the energy consumption and the response-time, it could give an interesting trade-off between the green efficiency, the wanted live performance and the services

[3] `http://www.oracle.com/us/corporate/contracts/cloud-services/index.html`
[4] `http://docs.oracle.com/cd/E29127_01/doc.111170/e28974/service-level-agreements.htm`
[5] http://www.windowsazure.com/fr-fr/support/legal/sla/

ability to quickly react if a peak load occurs. Indeed, taking more parameters into account in green scheduling may lead to choose a temporary virtual machines consolidation configuration that is less efficient regarding one or two metrics, but it ensures a very high level for the other metrics.

3.2. Cloud Architecture Models

The Cloud model defined for this article is presented in tables, one for each component of the model. This Section is organized into two subsections. The first one explains as clearly as possible the adopted notations. The second one presents characteristics of each component.

3.2.1. Notations

First, the Cloud architecture proposed in this article is composed of:

Virtual Machine (v), **Elementary Service** (S_e), **Host** (h), **Cluster** (k) and **Data-Center** (d).

Some of these components can use resources of other components. For example, a virtual machine can take certain percentage of a host resources (memory, CPU, etc.). Indeed, the notion of resources types has to be introduced too. Two types of resources are defined and used in the notations:

- *Fluid*: is a resource which is also called *non-critical*, because its modification can not lead the system in an error state, but can only affect the performance of the system. The *fluid* resource considered in this article is the CPU of a host.
- *Rigid*: is a resource with a more critical use which can lead to destroy or definitively affect the system state. The *rigid* resource considered in this article is the Memory of a host.

The adopted notation in this article is as follows: $X_{Y,N}^{R_1,R_2}$, with:
- X is the set of variable used in models
- $Y = \{f, g\}$ is the type of the resource
- $N = \{cpu, mem\}$ is the name of the resource
- $R_1 = \{v, S_e, h, k\}$ is the component that consumes resources of another one
- $R_2 = \{h, k, d\}$ is the component on which the resources are consumed

The set X can have three different meaning:
- The maximum fraction of resource asked or owned by the component, denoted by ξ
- The fraction of resource really allocated, denoted by α
- The fraction of resource really used, denoted by ω

For the sets Y and N, the notation $*$ denote any values of these sets. Here is two notations' examples:
- $\alpha_{g,mem}^{v,h}$: means the fraction of *rigid* resource, i.e. the Memory that has been allocated to virtual machine v running on the host h.
- $\omega_{*,*}^{v,h}$, means the fraction of any resource, used by the virtual machine v running on the host h.

3.2.2. Components models description

Knowing these notations, the following tables (Table 1 to Table 5) describe all parameters of each component of the whole Cloud model. For each parameter, four informations are given: the 1st column specifies the notation used for the parameter, in concordance with the quality of service Section 3.3, the 2nd column indicates if the variable is an input parameter or not, the 3rd column gives the type of the variable: Integer or Float and the last column contains a description of what the variable means.

An elementary service runs in one and only one VM, and it is dedicated to a specific function: Load Balance service, Computation service, Web Server service, Data base service or Storage service for example. In this modeling, only Elementary Services are considered, each one running on one virtual machine, without any communication between them.

Variable	Input parameter	Type	Description
$\xi_{f,cpu}^{v,h}$	yes	Float	Maximum CPU fraction the VM v is allowed to ask on PM h
$\xi_{g,mem}^{v,h}$	yes	Float	Maximum Memory fraction the VM v is allowed to ask on PM h
$\alpha_{f,cpu}^{v,h}(t)$	no	Float	CPU fraction given to the VM v on PM h at time t
$\alpha_{g,mem}^{v,h}(t)$	no	Float	Memory fraction given to the VM v on PM h at time t
$\omega_{f,cpu}^{v,h}(t)$	no	Float	CPU fraction used by the VM v on PM h at time t
$\omega_{g,mem}^{v,h}(t)$	no	Float	Memory fraction used by the VM v on PM h at time t
$Trcf_{*,*}^{v,h}$	yes	Float	Time need for the VM v, running on PM h, to change its configuration (increase or decrease the CPU or Memory fraction given to it).
$Tmig_v^{h_j,h_k}$	yes	Float	Time needed for the VM v to migrate from PM h_j to PM h_k

Table 1: Virtual Machine model description and notations

Variable	Input parameter	Type	Description
v^{s_e}	no	$\oslash$	VM v dedicated to the Elementary Sevice s_e
$\tau^{s_e}(t)$	no	Integer	Utilization rate of the Elementary Service s_e at time t, in requests per second

Table 2: Elementary service model description and notations

Variable	Input parameter	Type	Description
F^h	yes	$\oslash$	Set of CPU frequencies available on PM h
n_F^h	yes	Integer	Number of frequencies available on PM h
$\xi_{f,cpu}^{h,k}(t)$	no	Integer	Maximum CPU capacity of PM h at time t
$\xi_{g,mem}^{h,k}$	yes	Integer	Maximum Memory capacity of PM h at time t
$\omega_{f,cpu}^{h,k}(t)$	no	Float	CPU fraction used on PM h at time t
$\omega_{g,mem}^{h,k}(t)$	no	Float	Memory fraction used on PM h at time t
V^h	no	$\oslash$	Set of VMs running on PM h
n_V^h	no	Integer	Number of VM in the set V^h
Ty^h	yes	Integer	Indicate the type of the host (heterogeneity in Power consumption)
$P_{min}^{h,k}(F_i, Ty^h)$	yes	Integer	Given power (W) by the PM h when using 0% of CPU when using the frequency F_i
$P_{max}^{h,k}(F_i, Ty^h)$	yes	Integer	Given power (W) by the PM h when using 100% of CPU when using the frequency F_i

Table 3: Physical machine model description and notations

Variable	Input parameter	Type	Description
H^k	yes	$\oslash$	Set of PM in Cluster k
n_H	yes	Integer	Number of PM in the set H^k
$P^k(t)$	no	Integer	Given Power (W) by the Cluster k at time t

Table 4: Cluster model description and notations

Variable	Input parameter	Type	Description
K^d	yes	$\oslash$	Set of Cluster in the Data-Center K
n_K	yes	Integer	Number of Cluster in the set H^K
$P^d(t)$	no	Integer	Given power by the Cluster K at time t

Table 5: Data-Center model description and notations

3.3. SaaS Clouds Quality of Service Models

Cloud services providers may propose various solutions to companies wishing to shift to Cloud Computing. It is easily to note that every solution may propose different performance levels in terms of response time, precision, or provided features. It is then up to these companies to compare the different offers, to understand

how their applications may run over each cloud, and finally choose the best offer that fits their needs.

An SLA is a contract that is established between the cloud user and the service provider. It describes the prestation level, the utilization and execution conditions, and responsibilities of each party (user/service provider). The service provider also provides more precise information named SLO (Service Level Objective) that the provider will try to satisfy in order to guarantee a certain level of QoS to the user. The expression of QoS in a cloud computing environment can be done through high level parameters. In this section, different QoS parameters encountered in Cloud Computing are defined. These parameters are then classified into four categories: **Performance**, **Dependability**, **Security & Data** and **Cost**.

Multiple studies have been carried out in the field of QoS, especially for web services [21, 22]. Two types of QoS parameters can be distinguished [23, 24]:

Functional: A functional requirement is used to describe a behavior of a system. This behavior may be expressed through services, tasks or a set of functions the system is expected to provide. It is important to distinguish basic functionalities that can be common to various systems, and more specific functionalities that each system is providing in such a way that the user is able to put them on competition.

Non-Functional: A non-functional requirement specifies criteria that may be utilized for judging a system functioning rather than a specific behavior. Non-functional requirements are also called "qualities", "constraints", "quality attributes", "quality objectives", or "non-behavioral requirements". The non-functional requirements may be divided into two categories: (1) *Execution qualities*: such as security and usability which can be observed at execution time. And (2) *Evolving qualities*: such as testability, maintainability, extensibility and scalability which are incorporated into the static structure of the software system.

3.3.1. Cloud QoS Parameters

In cloud computing, the most important issue from the end user point of view is the capability to select the service that meets his/her needs. From the service provider perspective, it is important to improve the attractiveness of its offers by exhibiting the efficiency of its platform. To this end, it becomes necessary to define global Cloud QoS parameters in order to compare, analyze and classify the different cloud service providers.

- 1st category: Performance

- **Execution Time:**
  Execution time [25] depends on the complexity of the request *req* to be executed (i.e. number of instructions, denoted $\xi_{f,cpu}^{req,s}$) and the VM capacity, denoted $\omega_{*,*}^{v,h}$, where the request is processed. Execution time is equal to: $T_{ex}^{req,s} = \frac{\xi_{f,cpu}^{req,s}}{\omega_{*,*}^{v,h}}$.

- **Latency:**
  Latency [26] represents the necessary time for a user's request to be handled by the relevant service. This time only depends on the performance and the condition of the network when the user's request is sent to the service provider. It is usual [27] to define the necessary time T_{net} to send a message with a payload of n bytes as follows: $T_{net}(n) = l + \beta n$, where l is a constant (i.e. Latency) and β a gradient (i.e. Throughput).

- **Throughput:**
  The throughput [28] is expressed as the number of bytes per second transmitted from the source to the destination across the network. Considering the latency formula, and in order to understand the meaning of β in the formula $T_{net}(n)$, the throughput is defined as $B(n)$ and measured for the transmission of n bytes. Thus, the throughput is equal to the number of transmitted bytes in time: $B(n) = \frac{n}{T_{net}(n)} = \frac{n}{l+\beta n} \cong \frac{1}{\beta}$, which is the maximum value possible to get for the throughput.

- **Response Time:**
  A service efficiency can be measured through its *response time*, denoted as T_{rep}. It represents the elapsing time between the reception of the user's request and the moment he/she receives a response. The *response time* of a service depends on many sub-factors such as the *average response time*, the *maximum response time* guaranteed by the service provider or the probability that the response time will not be satisfied (i.e. response time failure).

- The *average response time*, denoted T_{rep}^{avg}, is given as: $\sum_{i=1}^{n} \frac{T_{rep_i}}{n}$, where T_{rep_i} is the elapsed time between the request sending time and the response reception time, and n is the total number of transmitted requests.

- The *maximum response time*, denoted $T_{rep_i}^{max}$ is the maximum response time clamed by the cloud service provider.

- *Response time failure* is expressed as a percentage of cases where the effective response time T_{rep_i} exceeds the maximum response time $T_{rep_i}^{max}$ promised by the service provider. It is denoted as σ_{rep}, and it is defined as follows: $\sigma_{rep} = \frac{n'}{n} \times 100$, where n is the total number of service accesses and n' is the number of times where the response time exceeds the expected maximum value ($T_{rep_i} > T_{rep_i}^{max}$).

Formally, in the throughput formula $B(n) \cong \frac{1}{\beta}$, the time $T_{net}(n)$ to send n bytes over the network, as detailed by Hockney [27], can be expressed as follows: $T_{net}(n) = l + \frac{n}{\rho}$, where $\rho = \frac{1}{\beta}$. The response time is thus the sum of the sending time (of both the request and the response) and the processing time of the request *req*: $T_{rep}^{req} = 2 \times T_{net}^{req}(n) + T_{ex}^{req}$

- 2nd category: Dependability

- **Accuracy:**
The accuracy of a service, denoted Ac, is defined as two parameters function: the *accuracy frequency*, denoted ϕ_S, and the *pertinence value*, denoted τ_S. It indicates how close are the resulting values to those expected by the user. It is denoted as: $Ac = f(\phi_S, \tau_S)$. In the case of storage service, it is the deviation of the average read/write time. In the case of a computing service, the accuracy is defined as the difference between the achieved performance and the specified performance in the SLA. In the case of computing resources such as virtual Machines, the most important accuracy indicator for a service is a the number of times, the SLA is violated. Let $n_{val'}^{u}$ be the number of times a service provider cannot guarantee the promised precision level. Thus, the *accuracy frequency* of the service is defined as: $\phi_S = \sum_{u=1}^{n_u} \frac{n_{val'}^{u}}{n_{val}^{u}}$, where n_u is the number of users, n_{val}^{u} is the amount of expected values by user u, and $n_{val'}^{u}$ is the amount of values actually received by user u. Another performance indicator of a service is the *pertinence value* which is defined as: $\tau_S = \frac{\sum_{u=1}^{n_u} \sum_{i=1}^{n_\varsigma} \left| \frac{\varsigma_i'^{u} - \varsigma_i^{u}}{\varsigma_i^{u}} \right|}{n_u}$, where ς is the service unit (processing, network, storage, ...), n_ς is the number of expected values, ς_i^{u} is the service unit value expected by user u, and $\varsigma_i'^{u}$ is the service unit value sent back to the user u.

- **Reliability**
The reliability [29], denoted Re, represents the capacity of a service to achieve its functions under certain conditions and in a given period of time. The global reliability measure for a service is related to the number of times (per day, week, month or year) the service is not able to fulfil its duty. The reliability of a service reflects its capacity to function without any failure during a given period and under given constraints. This characteristic is thus based upon the "functioning duration before failure" or the "functioning duration between failure" of the service, denoted $MTTF$ [30], but also on the basis of the user's failures. The $MTTF$ may be seen as a function of "failure density", denoted as $f_{dp}(t)$: $MTTF = \int_0^\infty t f_{dp}(t) dt$, where, $\int_0^\infty f_{dp}(t) dt = 1$. Therefore, λ which represents the "failure rate" is equal to: $\lambda = \frac{1}{MTTF}$. If it is considered that service failures are not predictable and randomly occur, then the reliability of the service in time can be expressed as follows: $Re(t) = e^{-\lambda t}$.

- **Availability**
The availability, denoted as Av, is the percentage of time during which the user can access the service. This availability is the probability that the system is working. Thus, it is related to "failure rate" or "failure density" and it is denoted as $f_{dp}(t)$. **The failure rate curve of an hardware component is shown in Figure 1**. The availability in time can be expressed as follows: $Av = \int_0^\infty \frac{1}{f_{dp}(t)} dt$

- **Capacity**
The capacity ξ is defined as the upper limit of the number of request (per second) a service is able to

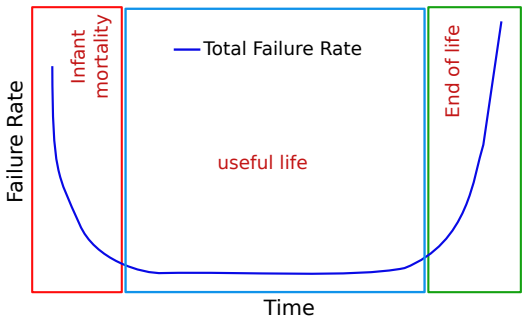
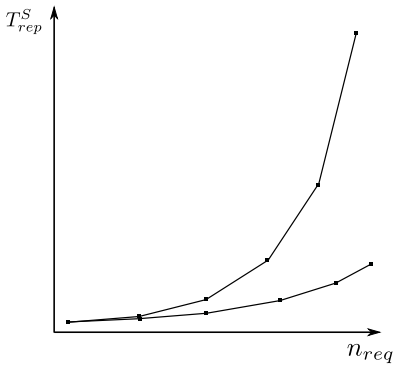


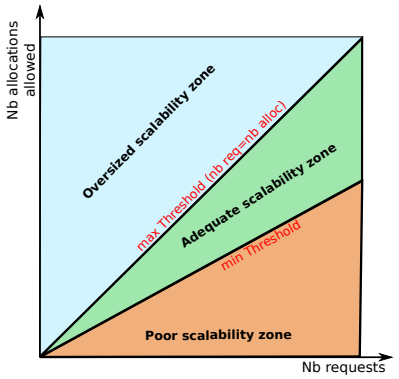
Figure 1: Failure rate of an hardware component during its life cycle (bathtub curve)

process. In terms of computing, the capacity depends on the virtual machine performance being used by this service and the physical machines (cluster component) hosting these VMs. Let k be a cluster composed of a set H of n_h hosts (homogeneous or heterogeneous), such as $H = \{h_1, ..., h_{n_h}\}$. Moreover, let $\xi^{h_i}_{*,*}$ be the capacity of each host h_i (for example, in terms of MIPS). Thus, the total capacity, denoted $\xi^{k}_{*,*}$ is equal to: $\xi^{k}_{*,*} = \sum_{i=0}^{n_h} \xi^{h_i}_{*,*}$, with: $\xi^{h}_{*,*} = \sum_{i=0}^{n_c} \xi^{c_i}_{*,*}$, where $\xi^{c}_{*,*}$ is the capacity of one CPU core of the machine h. Let s be a service which uses V virtual machines, such as $V = \{v_1, v_2, ..., v_n\}$. The capacity of a virtual machine is $\alpha^{v,h}$. Thus, the capacity of a service $\xi^{s}_{*,*}$ is equal to: $\xi^{s}_{*,*} = \sum_{i=0}^{n_v} \alpha^{v_i,h}, v_i \in V$.

- **Scalability**
  In a cloud platform, it is difficult to estimate the number of users at a given instant because it is neither predefined nor constant. Thus, without these information, it is impossible to predict the necessary resources for the service operation. Users arrival and their use of the service may quickly vary in time. Consequently, it may be very difficult to satisfy all the users at the same time. The scalability represents the capability to increase the computing capacity, and so the overall capacity of the system to process a huge number of users' requests in a given period of time. Scalability is directly related to performance. It is measured against increasing the number of users (i.e. increasing the number of service requests per second) while checking if performances characteristics that are described in the SLA are still met. Most of the time, the scalability, denoted S, is measured by the response time of a service (T^S_{rep}) in terms of number of requests n_{req} the service receives, as shown in figure 2(a), according to: $S = f(\frac{T^S_{rep}}{n_{req}})$



(a) Example of scalability curve, measured as service response time in terms of number of requests to process



(b) Relationship between number of users requests and possible number of allocations by the service provider

Figure 2: Scalability figures

Consequently, scalability is a desirable property for a cloud services provider. It guarantee the services stability in case of heavy load (increase in number of requested services or users requests) in a seamless way for the users. However, it is desirable for the service provider to not over provision the computing capacity of its machines nor to use these machines at 100% of their capacity for energy consumption reasons, and in order to remain in an acceptable "scalability zone" as shown in figure 2(b).

- **Reactivity**
  Reactivity or elasticity, denoted Ra is the capability of a service to "scale up" during an attendance peak. **Let denote by s_c, a complex service composed of multiple elementary services**. Then Ra is defined by two variables: the necessary time $Trcf^{s_c}_{*,*}$ to increase or decrease the service and the maximum capacity variation of the service, denoted $\delta\xi^{s_c}_{*,*}$ and it is defined as: $\sum_{s_e \in S^{s_c}} \delta\xi^{s_e}_{*,*}$. The maximum processing capacity of a service is considered as the number of computing units and their respective maximum capacity that can be allocated during an attendance peak. The Reactivity can also be considered as the system "speed" to increase its performance level while remaining within its acceptable scalability zone. If a system reacts too slowly to an attendance peak, then the users needs will not be satisfied and so the system will fail. On the other hand, if a system reacts quickly, then it will process the users requests in time. However, it is important to not overestimate the necessary reconfiguration and reach the "oversized scalability" zone. Let $Trcf^{v,h}_{*,*}$ be the necessary time to increase or decrease the capacity of a virtual machine (for example, in terms of MIPS), and let $Trcf^{s}_{*,*}$ be the necessary time to increase or decrease the capacity of all the virtual machines being used by the service s. Following the complexity of the reconfiguration to perform (VMs reconfiguration, starting of VMs, starting of physical machine, and VM migration), $Trcf^{s}_{*,*}$ can be differently defined. For example, if the only parameter considered is the VM reconfiguration time, then $Trcf^{s}_{*,*}$ will be defined as: $Trcf^{s}_{*,*} = \max_{v \in V^s}\left(Trcf^{v,h}_{*,*}\right)$. The capacity variation of the virtual machine v between two instants t_1 and t_2, such as $Trcf^{v,h}_{*,*} = t_2 - t_1$, is defined as $\Delta\xi^{v}_{*,*} = \xi^{v}_{*,*}(t_2) - \xi^{v}_{*,*}(t_1)$.
  The total capacity variation of a service s is defined as the sum of the capacity variation of all its virtual machines: $\Delta\xi^{s}_{*,*} = \sum_{v=1}^{|V^s|} \Delta\xi^{v}_{*,*}$. Consequently, the reactivity of a service s is estimated by the ratio of the capacity variation to the previous capacity, divided by the necessary time for this reconfiguration: $Ra = \frac{\Delta\xi^{s}_{*,*} \times \frac{1}{\xi^{s}_{*,*}(t_1)}}{Trcf^{s}_{*,*}}$.

- **Dynamism**
  Dynamism or "Latent Capacity", denoted Dyn, represents the amount of computational capacity available without turning on new hosts. It leads the Cloud provider to not consolidate virtual machines on hosts, and so keep a certain amount of capacity available in case of a peak load. The Dynamism can be evaluate as the average of free CPU capacity of all running hosts: $Dyn = \sum_{i=0}^{n_H} \frac{\xi^{h_i,k}_{f,cpu} - \omega^{h_i,k}_{f,cpu}}{n_H}$, where n_H only includes hosts powered ON.

- **Robustness**
  The Robustness, denoted Rob, can be interpreted as the probability of a service to be affected by a failure a component of the Cloud. Or in the provider point of view the number of services affected by an host failure. The more the number of service affected is small, the more the Cloud provider can insure to these users a high level of robustness. The Robustness can simply be represented as the average number of services on hosts: $Rob = \sum_{i=0}^{n_H} \frac{n^{h_i}_{V}}{n_H}$

- **Stability**
  The stability of a service, denoted St, is defined as its performance variation in time. For example, in the case of a storage service, it is the variation of the average read/write time. In the case of a computing service, the stability is defined as the difference between the achieved performance and the specified performance in the SLA: $St(t_1, t_2) = \frac{\sum_{u=1}^{n_u} \frac{\varsigma^{u}_{avg}(t_1,t_2) - \varsigma^{u}_{sla}}{\Delta T}}{n_u}$, where ς is a unit of computing, network or resource storage, $\varsigma^{u}_{avg}(t_1, t_2)$ is the average performance of the requested service between the time instants t_1 and t_2, ς^{u}_{sla} is the promised performance level in the SLA, $\Delta T = t_2 - t_1$ is the service time, and n_u is the total number of users.

- **Fault tolerance**
  The fault tolerance of a service, denoted FTs, represents its capability to remain reachable and working when anomalies occur. Anomalies are subject to:

- The error rate of a physical machine, denoted τ_M

- The network error rate, denoted τ_N

- The software error rate: invalid, incomplete or contradictory inputs, denoted τ_{SW}

A cloud service must ensure its proper functioning when facing these different anomalies. Thus, the fault tolerance of a service is a function of these three parameters: $FTs = f(\tau_M, \tau_N, \tau_{SW})$. If it is assumed that the physical machine and network errors make the service unreachable, then only software errors (τ_{SW}) should be taken into account.

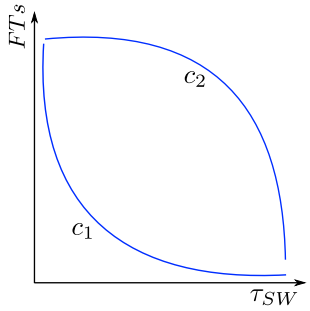


Figure 3: C1 and C2 represent two different fault tolerances curves by taking into account a software error rate denoted by τ_L.

A service fault tolerance curve approximating the curve c_1 from figure 3 indicates that the service is not very robust, unlike a service whose curve is similar to c_2 from figure 3 and which supports a high error until the service becomes unusable.

- **Sustainability**
  Sustainability can be defined in terms of service life cycle or in terms of the environmental impact of the services being used. This parameter can be split into two sub-attributes: service sustainability and environmental sustainability.

  – Service sustainability can be defined as the number of components that are reusable without any modification despite the evolution of the user requirements. A service is said to be very sustainable if it has more usable characteristics than what the user currently needs. In this case, if the user requirements evolve (i.e. increase in the number of requested characteristics), then the user can still user the service simply by using more characteristics. Consequently, the user does not have to look for another service that meets his/her new requirements. The sustainability of a service is defined as the ratio of "*number of characteristics provided by the service*" to the "*number of characteristics requested by the user*".
    Let $C^u_{needed} = \{c^u_1, ..., c^u_n\}$ be the set of characteristics requested by the user, and let $C^S_{provid} = \{c^S_1, ..., c^S_n\}$ be the set of the characteristics provided by the service where $C^u_{needed} \subset C^S_{provid}$. Let N^u_{used} and N^S_{provid} respectively be the number of characteristics of the subset C^u_{needed} and the number of characteristics of the set C^S_{provid}. The possible evolution E (in %) of the user requirements in terms in of characteristics is thus equal to: $E = \frac{(N^S_{provid} - N^u_{used})}{N^u_{provid}}$

  – Environmental sustainability can be appreciated as the "carbon footprint" of the service. The measure of the carbon footprint is very complex and depends on many parameters. For these reasons, it is preferable to measure the PUE (Power Usage Effectiveness) [31].

- **Agility**
  The Agility can be defined by two different characteristics' types: (1) the variation possibility in the availability of hosts made available for the cloud users, and (2) the easiness for a service provider to accept or not to renegotiate the terms of the SLA. For (1), the advantage of a cloud environment comes from the integration of a level of agility in the proposed services. Hence, the considered environment may rapidly evolve and grow without involving insurmountable costs, and is measured as the variation rate of various metrics that demonstrate the rapidity of the the system to integrate new features. It is related

to which extent the desired service is elastic, portable, adaptable and flexible. In this case, the possible variations of the system that define its agility are:

- Number of resources allocated to a user.

- Flexibility of virtual machine capacities (computing, memory, etc.)

- Use of an additional service

In the case of the second characteristic (redefinition of thresholds initially specified in the SLA), the parameters to be taken into account are the following:

- Increase in the rental time of virtual machines

- Increase in the number of machines made available to the user

- Increase in the maximum allowed capacity (computing, memory, etc.)

- **Insurance**
  This parameter indicates the probability of a cloud service to respect the terms of the SLA. Companies seek the development of their activities and the provision of better services to their customers. Thus, reliability, resistance and stability parameters are carefully considered before shifting all of the services to the cloud.

- **Usability**
  For a quick adoption of cloud services, usability plays a very important role. The more the cloud services are easier to master, the more likely a company is willing to shift to Cloud Computing. The usability of a cloud service includes many parameters such as the ease of access, setup, and utilization.

- **Customization**
  SaaS services need to be adaptable according to the type of user. Every user should have the possibility to choose and save its own settings (appearance, logo, font, etc.) and configuration files. Thus, the SaaS application being used is adapted to every user like all the traditional heavy weight applications.

- **Automatic Update**
  A key advantage for a SaaS provider is the ability to regularly update its services. Thus, the platform should function with the last innovations in terms of service, and so letting the users directly benefit from these updates. From a user perspective, one of the key advantages is the automatic updates, not alone the seamless installing of these updates. Moreover, if the user is working collaboratively with another user on the same application, he/she does not have to worry about compatibility issues which is the primary source of errors in such situations.

- 3rd category: Security & Data

Despite the importance of the performance of cloud services, the security [32] used for access, user services, and data storage [33] is also one of the major user concerns. Data protection is a key issue for all companies. Hosting the company data "outside" is critical. Thus, cloud services providers are required to set up very efficient security policies. Data confidentiality [34] is equally essential: the use of storage proof techniques [35] allows to check that the Cloud does not experience DoS (Denial of Service) attacks and all the users data are safe. Dynamic data may also be verified with a similar approach [36]. These techniques require very low computational time while only little data is being transferred between the client and the server.

- **Authentication:**
  Traditional authentication means are no longer relevant in a cloud infrastructure. For this reason, more and more cloud services providers are looking for an efficient authentication service, dubbed Authentication-as-a-Service (AaaS), in order to enhance the security and to ease the handling of users authentication. Strong authentication can nowadays be ensured everywhere a password is used by employing industry standards such as RADIUS (**R**emote **A**uthentication **D**ial-**I**n **U**ser **S**ervice) and SAML [37](**S**ecurity **A**ssertion **M**arkup **L**anguage) and API availability for other applications. The use of authentication token ("Cloud Token" [38] method) prevents the users from losing their token when migration from one platform to another is decided. Moreover, the automation of the authentication management allows to considerably reduce the management and administration costs.

- **Authorization:**
  Medium and large companies have generally specific requirements of authorization characteristics for their cloud users [39], i.e. users' privileges or rights granting based on their roles and needs. In some cases, a company application may require RBAC (Role Based Access Control) [40]. With RBAC, authorization are structured in such a way to satisfy the requirements of functional roles within the company. To this day, the setup of authorizations and the capacities management in a cloud service are poor and they cannot be handled with a satisfying granularity. Most of cloud services providers take into account at least two roles (privileges): *administrator* and *user*. This allows to grant service administration rights in order to manage the users profiles and service access policies, or to allow or deny access to the service based on users provenance.

- **Integrity:**
  The capacity to save the users data integrity is to ensure that the content of their data has not been modified without their knowledge. One of the means to check the integrity of content is to compare the data state against a past known state (at $t = t_b$) and between these instants there were no changes made by the user to this content. Let D_u^t be the set of data composed of n_d elements such as $D_u^t = \{data_0, ..., data_{n_d}\}$ which represents all the data belonging to user u at time t. The data integrity is ensured by $D_u^{t_b} \equiv D_u^t$, such as every element $data_i \in D_u^t$ is equal to the element $data_i \in D_u^{t_b}$.

- **Confidentiality:**
  Different issues such as software bugs, operator errors, or external attacks may compromise the confidentiality [34] of the applications data in Cloud environments. Thus, these data become vulnerable to malicious actions of certain users to whom access should have been denied. Protection techniques [41] against attacks that target data confidentiality can help the users to verify the obtained data consistency. These techniques require only little exchange of information such as the root of a Merkle [42] hash tree.

- **Accountability:**
  "Accountability" in Cloud is associated to "verification chain" or "control chain" of different parameters the service provider implements within its Cloud. Thus, service providers implement verification mechanisms [43], essentially in order to ensure a security level over users data. Such mechanisms are implemented in order to provide the users control and monitoring over their data stored in the Cloud. One example of the control chain elements is shown in figure. A control chain of cloud services provider allows to ensure access to the following tools:

  - Tools allowing users to have control over their data and a view on how their data are used, and so the insurance that their data are processed and protected the way they want.

  - Tools allowing users to make choices on the way cloud service providers can use and protect their data, and to be aware of the risks and consequences resulting from these choices.

  - Tools to verify that users expectations in terms of security are respected, and there is no interference to rules defined in the SLA.

  - Tools to explain and help informing the user on the way surveillance and control of their data are made by the cloud service provider from an ethical point view.

- **Traceability:**
  In general, traceability is defined as follows: *"The traceability indicates the situation where necessary and sufficient information is available to know (eventually in a retrospective way) the composition of a product throughout its production and distribution chain."* Within a cloud environment, the traceability of service usage allows the user to precisely know how the service is being used. Any information related to interventions on the used services must be made available to every user. Indeed, the impossibility to know the functioning state or the data location is a real barrier to cloud services adoption [44]. Thus, it is important to inform the users, through a log journal (written traces), which type of automatic or human actions have been made, and where the data are stored[6] [45]. A traceability platform allows to tackle these no-transparency issues, and to keep an events chain that indicates, for example, human

[6] http://www.hpl.hp.com/techreports/2011/HPL-2011-38.pdf

actions, file transfers, and automatic processes activity, but also information from related systems such as authentication systems or equipment management systems.

- **Encryption:**
  Encryption[46] is a process used for protecting the information transfer or storage. It is the conversion of data into cryptograms that can only be read by person that own the data or are authorized to read its content. Encryption is commonly used to protect sensitive data stored and used in networks, mobile or wireless devices. In Cloud, ciphering algorithms are used to protect outgoing data in order to make them invulnerable once they are outside the company they belong to. The ciphering is often used to meet the companies regulation policies. HIPAA (Health Insurance Portability and Accountability Act Compliance Report) and PCI DSS (Payment Card Industry - Data Security Standards) are essential tools for securing data in Clouds for companies that uses SaaS applications such as Salesforce.com or Oracle CRM on Demand.

- **Isolation:**
  In Clouds, the challenge is to allow authenticated users to view their own data from outside the cloud while preventing any malicious user, that may have exploited any service provider vulnerability, to access the users' data. To avoid such problem, the setup of a secure logging service can help isolating data among different users.

- **Data Life cycle:**
  Oracle [47] defines the data life cycle by the access frequency to the data. As time goes on, the access frequency decreases which leads to the archiving these data. The life cycle includes three states: active, less active, and archived. The Cloud Security Alliance [48] has proposed the concept of data security life cycle which summarizes the data security life cycle in six processes: Production, Conservation, Utilization, Sharing, Archiving and Destruction.

- **Non-Repudiation:**
  Non-repudiation implies that the contract (a.k.a. security contract) cannot be questioned by either two parties (user & provider).

- 4$^{\text{th}}$ category: Cost

- **Service Cost**
  Before adopting the Cloud Computing, it is interesting to see if it is really profitable? Consequently, the cost is clearly one of the most important attribute for the Cloud user (individual user or professional user). The cost tends to be an easily quantifiable parameter but it is important to be able to express it through relevant characteristics regarding the the organization to which it is addressed. For a Cloud service provider, the cost can be expressed as follows: γ_v^{cap}, the hourly renting cost of a virtual machine v having capacity *cap*. ΔT_{loc}, the service use duration in number of hours and N_v the Number of requested virtual machines. Thus, the final cost the user has to pay can be defined as follows: $\gamma_{tot} = \gamma_v^{cap} \times N_v \times \Delta T_{loc}$

- **Energy cost**
  In this article, the affine energy model has been chosen. Even though it is a simple model, its efficiency has been proven since quite a long time. As it is described in Section 3.2, the hosts allow several frequencies of CPU functioning. Indeed, at fixed frequency F_i the affine model is:

  $P(F_i) = \alpha\,(P_{full}(F_i) - P_{idle}(F_i)) + P_{idle}(F_i)$, **where α is the CPU load**, $P_{full}(F_i)$ and $P_{idle}(F_i)$ are the power given by the host, using the F_i frequency, at 100% and 0% of CPU utilization, respectively. Moreover, this model was already used by the authors in a previous work [49] that presents in details the validation of this model using DVFS tools.

- **Carbon cost**

  The Carbon cost, also called Carbon footprint [50] represents the environmental impact of a system. In Clouds, values and costs associated to it vary between Clouds providers. According to the Open Data Center Alliance [7], these variations can be due to:

[7] `http://www.opendatacenteralliance.org/docs/Carbon_Footprint_and_Energy_Efficiency_Rev2.0.pdf`

- Energy efficiency of hardware used
- Settings of hardware and systems software
- Degree of virtualization and efficiency of its management
- Ambient environmental conditions (e.g., Nordic vs. Mediterranean)
- Efficiency of data center infrastructure and housing (i.e., the power usage effectiveness or PUE)
- Source of electricity used (e.g., coal vs. nuclear from grid, local generators)
- Carbon offsetting and trading options deployed
- National or regional regulation or tax arrangements

And the amount of carbon produced can expressed as: $CFP = IT_{eq} \times E_{eq} \times E_{ovh} \times (Ca_{em}(src) + Lo)$, where, IT_{eq} is the number of equipment used, E_{eq} the energy consumption associated to these equipment, E_{ovh} is the energy overhead, $Ca_{em}(src)$ is the carbon emission of electricity sources, also called CEF (Carbon Emission Factor), and Lo are the losses due to the electricity transport from the source to the destination.

3.3.2. Summary table

Category	Name	Notation	Functional
PERFORMANCE	Execution Time	T_{ex}	no
	Latency	l	no
	Throughput	B	no
	Response Time	T_{rep}	no
DEPENDABILITY	Accuracy	$Ac = f(\phi_S, \tau_S)$	no
	Reliability	Re	no
	Availability	Av	no
	Capacity	ξ	no
	Scalability	S	no
	Reactivity	Ra	no
	Dynamism	Dyn	no
	Robustness	Rob	no
	Stability	$St(t_1, t_2)$	no
	Fault Tolerance	FTs	no
	Sustainability	E	no
	Agility	$\oslash$	yes
	Insurance	$\oslash$	yes
	Usability	$\oslash$	yes
	Customization	$\oslash$	yes
	Automatic Update	$\oslash$	yes
SECURITY & DATA	Authentication	$\oslash$	yes
	Authorization	$\oslash$	yes
	Integrity	$\oslash$	yes
	Confidentiality	$\oslash$	yes
	Accountability	$\oslash$	yes
	Traceability	$\oslash$	yes
	Encryption	$\oslash$	yes
	Data Life Cycle	$\oslash$	yes
	Non-Repudiation	$\oslash$	yes
COST	Service Cost	ω^u_{tot}	no
	Energy Cost	$P^h(F_i)$	no
	Carbon Cost	CFP	no

Table 6: Summary table of all Quality of Service parameters detailed in this Section 3.3.

4. Validation Methodology

This validation methodology Section describes the scheduling approach chosen to validate the quality of service models and metrics proposed in Section 3.3.

Four quality of service parameters have been chosen: the Energy consumption to have a relevant green approach and to tackle environmental issues, the Reponse-Time to have a pure performance metric which is important for users, the Robustness that reassures both providers and users on the probability of failure of their services, and the Dynamism which ensures a certain amount of performance reserve in case of a peak of traffic, and so ensures that currently used services are not affected by this peak load.

The complexity of scheduling problems constitutes an important research domain since many years [51, 52, 53]. It is now well known that a scheduling problem, as simple as it can be, is NP-complete [54]. To get an approximation of a good solution, different heuristics can be used [55, 56].

In this article, the Genetic Algorithm allows to have an approach which is able to take into account a complex optimization function. Here, the aim is also to show that the quality of service models are relevant, and their metrics can be used in a multi-objectives scheduling optimization. The GA, by its intrinsic functioning allows to integrate these metrics in its objective function, and also to prove that the given scheduling is in correlation with the QoS metrics used.

4.1. Quality of Service parameters

The chosen QoS parameters for these studies are the four following:

- *Energy Consumption*: The Energy consumption computation as it is explained in the *Cost* category in Section 3.3.

- *Response Time*: The Response Time, is considered as the execution time of VMs, knowing the MIPS capacity of these VMs and the number of instructions (in Million of Instructions) that VMs have to execute.

- *Robustness*: The Robustness is interpreted as how many VMs should be disposed if a host failure happens.

- *Dynamism*: The Dynamism (or Latent Capacity) is the average amount of free MIPS on each host that can be used in a case of a peak of request rate arrival.

4.2. Genetic Algorithm

A Genetic Algorithm [57] is an optimization heuristic that mimics the natural evolution. GA considers a population (a set) of individuals, where each individual, also called *chromosome*, represents a solution to the problem being solved. Each individual is assigned a score (or fitness) according to an objective function. The main idea of the GA is to evolve an initial population, through a number of generations, to a state where all the individuals have the best scores. From one generation to another, specific operators (genetic operators) are applied on each individual in order to explore new possible solutions. In each generation, and after genetic operators application, initial and new generated individuals are sorted based on their fitness, and only a subset of the top ranked individuals are kept to compete for the next generation. Consequently, after every generation, the population tends to a better state. In this article, and since the considered problem is a multi-objective optimization problem, the individual's fitness is composed of a value for each metric being optimized.

The GA implemented for this article is dedicated to virtual machines scheduling on Clouds.

4.2.1. Modelling

- A *chromosome* represents a solution of VMs placement, as depicted in Figure 4(a)

- A *gene* represents a VM

- The value of a *gene* represents the index of the host on which the VM is allocated

- In parallel, the characteristics of the VMs and the PMs are stored in order to compute metric values and *chromosomes* fitness values.

4.2.2. Operators

The common operators of GA have been used :

- The *mutation* operator, applied on a fixed number of *chromosomes*, randomly chooses a *gene* and changes its value. This means that a VM has been allocated to another host, the so obtained *chromosome* represents an new scheduling solution. The new *chromosome* is added to the population.

- The *crossover* operator, also applied on a fixed number of *chromosomes*, interchanges parts of two *chromosomes* and so generates two new solutions. It is applied using two points of crossing. An example of this type of crossover operator is given in Figure 4(b).

- The *selection* operator is in charge of reducing the newly extended population to its original size, by only keeping the best individuals.

4.2.3. Stopping criteria

The end of the GA can be decided in two manners:

- with an improvement threshold that allow to decide if the new best *chromosome* of the current generation is enough better than the previous one. If the difference between the two are smaller than the threshold, the GA is stopped. It means that the evolution between the two generations is not enough high to continue.

- with a fixed number of generations

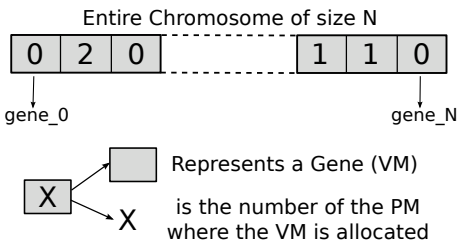
4.2.4. Chromosome validity

Another important step of the GA process is how each *chromosome* is decided valid or not. Indeed, a *chromosome* is allowed if it respects all the model constraints. In this study, it leads to check resources (both CPU and Memory) usage of hosts. If a *chromosome* is not valid, it is not added to the current population and a new one is generated.

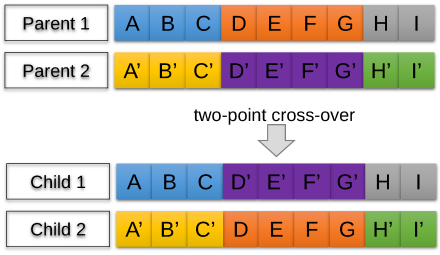
4.2.5. Metrics and Fitness values

Each metric has its own real value. In order to compute the value of the objective function, that gives the fitness value for each *chromosome* of the population, a standardization of these values is needed. Indeed, each metric does not have the same range of value. The standardization of each metric value, in order to compute an efficient *chromosome* fitness value has been done with the "center and reduce" formula (which gives a set of values with an average value of 0, a variance value of 1 and standard deviation value of 1): $v_{std} = \frac{v-\mu}{\sigma}$, where, v is the value to standardize, μ is the average and σ is the standard deviation. The fitness is then equal to a linear formula involving these standardized values. Consequently, all fitness values of one generation can be compared together.

The objective function has been defined as: $F_{obj} = \alpha_1 \times E + \alpha_2 \times RespT + \alpha_3 Rob - \alpha_4 Dyn + \alpha_5 NbMig$, where α_1, α_2, α_3, α_4 and α_5 are the coefficients of the Energy (E), the Response-Time (RespT), the Robustness (Rob), the Dynamism (Dyn), and the number of migrations (NbMig), respectively. These coefficients can be tuned (one greater than the other) to advantage the optimization of the corresponding metric. The first three metrics (energy, response time and robustness) have to be minimized, contrary to the Dynamism metric which has to be maximized (this explains the minus $\alpha_4 Dyn$). The use of the fifth metric *NbMig* is explained in Section 5.1.



(a) A *chromosome* is composed of N *genes* (i.e. N virtual machines). Each value assigned to a *gene* is the index of the host where the VM has been allocated.



(b) Two-point Crossover operator applied on GA Chromosomes

Figure 4: Genetic Algorithm's chromosome and crossover figures

4.3. Basic Algorithms

4.3.1. Round-Robin

The Round-Robin (RR) algorithm was empirically used in network scheduler, with any priority on the chosen destination, when the virtualization did not exist yet. Its use in the context of this article can appear

quite strange because of the behaviour of this algorithm does not have any intelligence to take benefit of the virtualization, and so also of the consolidation concept, currently commonly used in Cloud data-centers to minimize the energy consumption. Thus, its use in this article is analyzed with more metrics than only the energy side, and allows to show its advantages when considering **different kind of QoS metrics**.

The Round-Robin algorithm sorts hosts based on their types. This sorting becomes less efficient when many virtual machines have to be allocated, due to the intrinsic behavior of the RR algorithm. Since the number of VMs decreases (some VMs' execution is finished), this sort becomes more interesting. Indeed, the less the number of the virtual machines to allocate, the more the efficient hosts are chosen.

4.3.2. Best-Fit Sorted

The Best-Fit Sorted (BFS) algorithm is more known thanks to its good efficiency on a virtualized environment. In accordance with one or more sorts algorithms applied on hosts or virtual machines characteristics, the BFS achieves good results regarding the power consumption.

The implemented Best-Fit Sorted algorithm performs two sorts of hosts. First, a sort on hosts type (detailed in Section 5.1) is applied. When this first sort is done, then a sort on free available MIPS on hosts is done, this for each type of hosts. This allows the BFS to allocate efficiently virtual machines on hosts and also on the hosts which are more able to execute them.

4.4. Virtual Machines reconfiguration

The virtual machine reconfiguration is independent of the algorithm used. A reconfiguration of a virtual machine allows to decrease the amount of their *fluid* resource (CPU capacity) and is applied when the virtual machine is going to be run on a host. The virtual machines reconfiguration allows to better consolidate them on hosts. Details on the application of virtual machines reconfiguration process are given on Section 5.1.

5. Evaluation

5.1. Methodology

To be able to compare different executions, the same set of virtual machine and hosts configuration is used. The results shown in the following section have been done with 110 hosts and 400 virtual machines. The hosts have a CPU capacity of 2000 MIPS and 2500 Mo of Memory. Regarding CPU and Memory capacities of VM, four values for both resources have been used: 200, 400, 600 and 800 (MIPS or Mo for CPU and Memory, respectively). This results in 16 types of VMs. Each VM has a fixed number of instructions to execute, also randomly generated, between 10 and 110 times the size of the VM considered. The virtual machine migration time depends on the network bandwidth which has been set to 800Mo/s, this to have maximum migration time up to 1 second. Knowing that the total execution time is about 120 seconds, the migration time of 1 second corresponds to a migration time of 30 seconds for an utilization of one hour (if the VM used has to be migrated), which is approximately a real migration time value. Moreover, during a migration, a virtual machine consumes the same amount of power on the source and destination hosts. Of course, a more precise model for virtual machines could be used, as in [8], but this is not the main focus of this evaluation phase.

The power characteristics are those of the Grid'5000 Reims site described in Table 8. The type of hosts are randomly assigned to the hosts, but are also the same between different runs.

Regarding the scheduling method, for the three used algorithms, simulations have been done in two manners:

- The first one is a simple virtual machine allocation at $t = 0$
- The second method, is a periodic scheduling, each 20 seconds, which leads to 5 reallocation during the simulation.

As it is described in Section 3.2, an heterogeneity is considered in terms of the host power consumption. It means that each host has the same capacity of CPU computation, but not all the same P_{Idle} and P_{Full} values. **This heterogeneity defines five types (Ty^h) of hosts and their characteristics are shown in Table 7.**

Finally, the configuration of the GA for VMs' allocation at $t = 0$, is the following:

- Random initial population size: 1500
- Work population size: 120
- Number of *mutations* operation: 100
- Number of *crossovers* operation: 90
- A fixed number of generation: 400

Host Type	0	1	2	3	4
% of Power Heterogeneity	-20	-10	0	+10	+20

Table 7: An host type corresponds to a percentage Power heterogeneity, which means that the host consumes more or less than the power model given in Table 8. In this table, the type 2 is no different than the basic model, the types 0 and 1 consume less, and the types 3 and 4 consume more.

Grid'5000 Reims site						
Available Frequencies (GHz)		0.8	1.0	1.2	1.5	1.7
Power (W)	***Idle***	140	146	153	159	167
	Full	228	238	249	260	272

Table 8: Grid'5000 Reims site available frequencies, and power measurements at *Idle* and *Full* CPU states, which both use 0% and 100% of CPU utilization, respectively.

The coefficients $[\alpha_1...\alpha_4]$ of the objective function have been all set to 1, thus, the GA does not optimize any metric over another one. For the first allocation at $t = 0$ the coefficient α_5 corresponding to the number of migration is set to 0.

The coefficients configuration for a reallocation are little different: $[\alpha_1, \alpha_2]$ are set to 1, and $[\alpha_3, \alpha_4, \alpha_5]$ are set to 0.5. These last three coefficients, of Robustness, Dynamism and the number of migration tend to perform against virtual machines consolidation, and therefore against virtual machine migrations, even if the number of VMs decreases. The choice of the value 0.5 is a good trade-off between metrics importance and migration penalty.

5.2. Simulations

The simulator used for all the simulations is CloudSim [58]. In a previous article, an extension of CloudSim has been presented in [49] by adding the DVFS tools. The whole methodology and evaluation of this implementation has been demonstrated in this previous article by different use cases. This DVFS version of CloudSim has been used to conduct these simulations, and it is updated by including the computing of QoS parameters described in Section 3.3.

Regarding scheduling algorithms, the BFS and RR has been directly implemented into the simulator, contrary to the GA which is an independent C++ program. This leads to a small difference to call these algorithms during the simulation. Indeed, during the simulation a new scheduling of virtual machines which are still alive is done every 20 seconds. For the BFS and RR algorithms it is done with an internal function call, but for the GA it requires to execute the dedicated program outside the simulator. This outside call needs to save the current situation (VMs alive, remaining Instructions of each VM), to have them as input parameters of the GA. After the GA has computed the new scheduling, the new virtual machines allocation are given the CloudSim using files, this new placement is compared with the old one to determine which virtual machines migrations have to be done. These periodic scheduling allows to see the evolution of the QoS parameters over the time, and the effect of virtual machine reallocation during the simulation phase. In the next Section5.3, results for reallocation method for all these algorithms are called: BFS-ReAlloc, RR-ReAlloc and GA-ReAlloc. Simulations have also been done without reallocation, but with only one allocation at $t = 0$. Results for these simulation are simply called: BFS, RR and GA.

5.3. Results & Comparisons

Simulations results are illustrated in the following graphs, showing the evolution of the different metrics in time. Moreover the comparisons of the four QoS metrics chosen, a comparison of the number of hosts used is proposed, and also an histogram of the number of virtual machines migrations triggered by each algorithm during reallocations. For each algorithm, there are results for simulation of allocation and for reallocation. Of course, between $t = 0$ and $t = 20$ results are the same for both.

First, the comparison of the *Number of Hosts* (Figure 5) used allows to see that, at $t = 0$ the virtual machines allocation given by the GA (and GA-Realloc, which is the same at $t = 0$) uses the whole 110 PMs. As a reminder, the GA gives a solution that optimizes all the metrics. Indeed, the *Robustness* and the *Dynamism*

lead to have the lowest number of VM on host, respectively to decrease the risk of failure of each service, and to maximize the free CPU capacity on each host. After the simulation starts, the GA significantly decreases the number of used hosts and becomes the best on this parameter, showing that the first allocation, is also efficient after the number of VMs decreases. For $t > 40$s and after the second reallocation, the GA-ReAlloc uses the least number of hosts. This demonstrates its efficiency in terms of VMs consolidation. The RR uses also all the hosts, this result is logical because of the intrinsic behavior of RR, that allocates VM on hosts one by one. This behavior can also be easily seen at $t = 80$ seconds, the reallocation of RR has the effect of dispatching alive VMs over all hosts. Finally, at $t = 0$ the BFS (and BFS-ReAlloc) uses the less number of hosts, and are no better than GA results. However, it is interesting to note that reallocation, done every 20 seconds, tends to decrease the number of used hosts. It is more obvious at $t = 80$ seconds that the reallocation has really helped to have a better virtual machines consolidation.

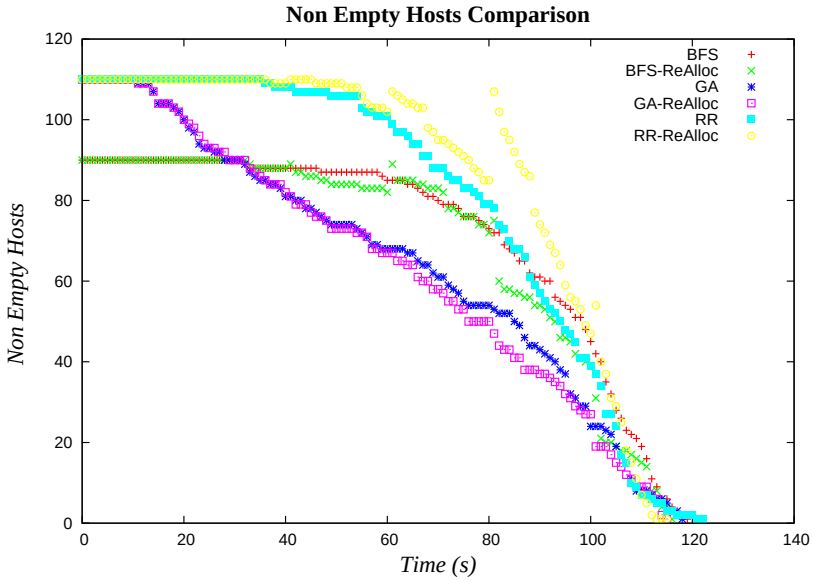


Figure 5: Evolution of the number of hosts used during simulations for the six algorithms.

The *Robustness* graph (Figure 6) represents the average number of services on hosts (empty hosts excluded). Indeed, the higher this number is, the higher is the risk to have a service affected by a failure. This metric really tends to have a schedule which distributes virtual machines over a large number of hosts. This leads to an antagonist behavior compared to energy consumption optimization, for example. This is the reason why this metric has a big importance in this evaluation phase. Knowing the meaning of this metric, it is easy to realize that the GA (and GA-ReAlloc) has very poor results compared to the two other algorithms, even if the *Robustness* metric is included in its objective function. For the BFS (and BFS-ReAlloc) it is interesting to note the correlation between its bad *Robustness* value at $t = 0$ which is totally linked with the small number of hosts used commented in the previous paragraph. Regarding the RR (and RR-ReAlloc), the modeling of this metric allows to notice that, an algorithm which is totally inefficient on many points of view, can be the best one. The fact of distributing the virtual machines over all the hosts produces logically a small average of services on each hosts. The impact of the reallocation on this metric is not so bad. At each scheduling time, a peak can be observed but its effect is very brief, after this moment for the BFS-ReAlloc and RR-ReAlloc the value can be better than the corresponding values in the allocation simulation.

The next simulation results comparison, in Figure 7, concerns the *Dynamism* metric. *Dynamism* value depends on the number of used hosts when it is computed. This comparison allows to remark that the GA/GA-ReAlloc is always the worst, the RR/RR-ReAlloc is always the best and BFS/BFS-ReAlloc is between the two, the GA gets the worst value than the others, especially when compared to BFS/BFS-ReAlloc, when the BFS/BFS-ReAlloc uses less hosts than the GA (until $t > 40$s for the same number of virtual machines to allocate). This ascertainment permits to note that the GA uses more hosts to satisfy as best as possible all the metrics of its objective function while choosing a much better frequency of CPU than BFS/BFS-ReAlloc can do, which lead to very small average of free MIPS available on hosts. This behavior of the GA-ReAlloc,

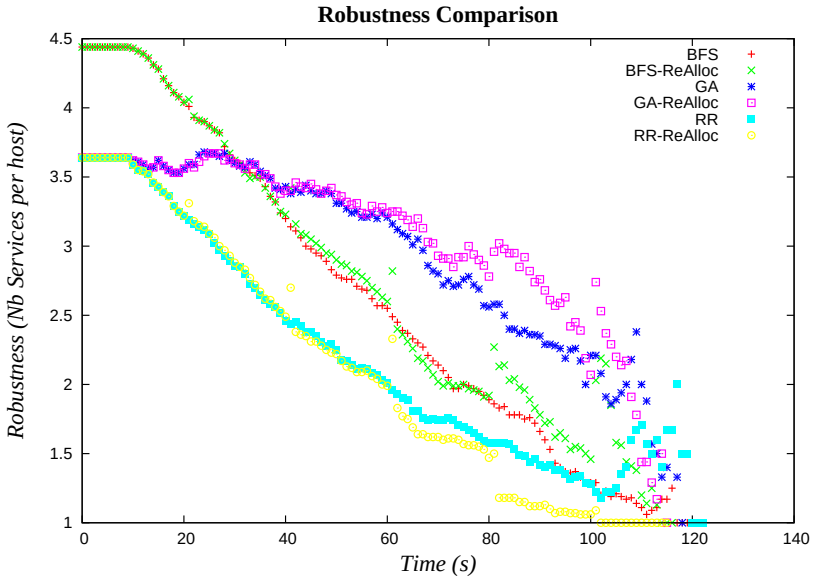


Figure 6: Evolution of the Robustness metric during simulations for the six algorithms.

tends to really decrease the *Dynamism* during reallocation, it is remarkable when comparing the two *Dynamism* curves of GA and GA-ReAlloc where the GA-ReAlloc curve is always beneath the GA curve, and peak decrease can be observed at each reallocation. This second remark means that a GA reallocation consolidate virtual machines on hosts much better than the other algorithms which decreases the *Dynamism*. The last analysis is about RR/RR-ReAlloc, this algorithm, which performs worst in many metrics, is here broadly the best one. Indeed, even if the lowest possible frequency is used, depending on the number of VMs allocated on each host, RR/RR-ReAlloc distributes VMs over all hosts. Thus, the number of used hosts is higher and the *Dynamism* is also higher. This analyses of the *Dynamism* allows to see that an efficient algorithm in terms of energy consumption can perform better when taking into account other metrics as the *Dynamism*.

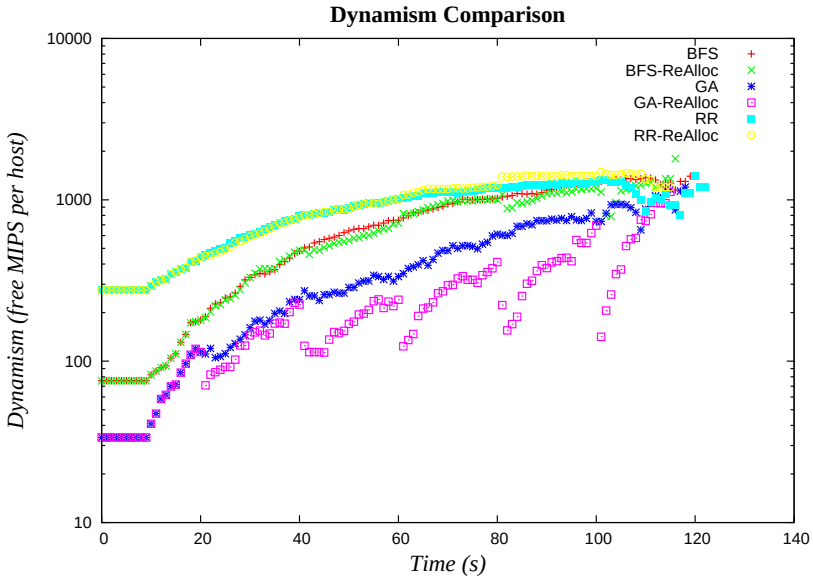


Figure 7: Evolution of the Dynamism metric during simulations for the six algorithms.

The Figure 8 concerns the performance and the green efficiency results, using the *Response-Time* values and the *Energy consumption* computation. The first thing to note is that all reallocation simulations give

better results on both *Response-Time* and *Energy consumption* metrics, except for the RR-ReAlloc which constitutes a particular case. These results justify the use of reallocation. The BFS/BFS-ReAlloc give good results. Compared to the GA results, the BFS algorithm consumes 3.5% more energy with a *Response-Time* only slightly slower. The same behavior is observed when comparing BFS-ReAlloc and GA-ReAlloc. However, RR-ReAlloc gives a worst result in terms of *Energy consumption* than RR while having a better *Response-Time*. This can be explained by the number of used hosts during the entire simulation, and also by the number of performed migrations as shown in Figure 9. Indeed, when a reallocation is done, and if the number of VMs has relatively decreased from the last reallocation, RR generates many migrations. This can be seen at each reallocation time, and results in 470 migrations. Moreover, regarding the number of migration, the placement done by RR-ReAlloc after each reallocation does not consolidate VMs on hosts, avoiding so to slow down the VMs execution, but a huge amount of *Energy* is consumed.

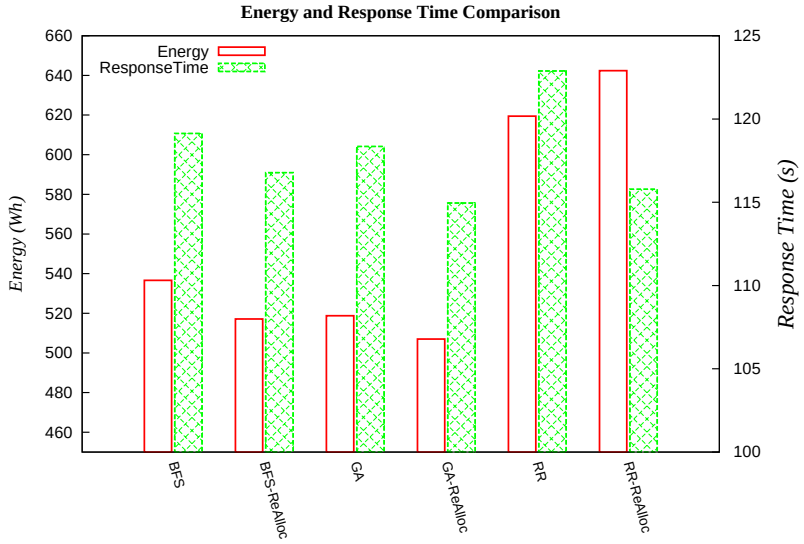


Figure 8: Energy consumption and Response-Time result values of simulations for the six algorithms. Energy consumption values have to be read on the left Y axis, and the Response-Time values on the right Y axis.

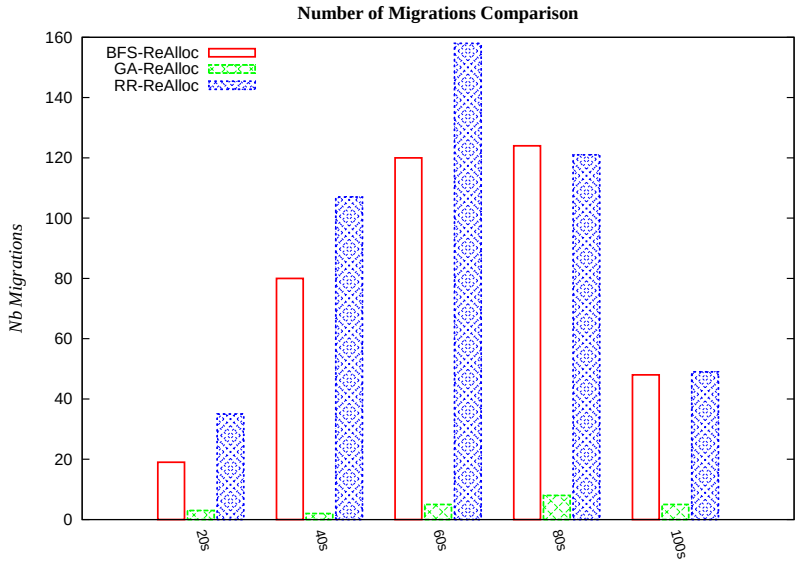


Figure 9: Number of migrations (at each reallocation time) during simulations for the three reallocation algorithms.

6. Conclusion

This article presents analysis of studies on Clouds modeling, Clouds scheduling, and actual SLAs of SaaS providers. Based on these analysis, this article proposes a Cloud architecture modeling that includes the DVFS, but especially a modeling of Clouds Quality of Service parameters. This list contains definitions, measurable and reusable metrics for *non-functionnal* parameters. It aims to allow a better analysis of Clouds QoS, and allows to be closer to Clouds providers needs while keeping a green approach.

Then, the article exposes how the use of these QoS metrics can improve the functioning, but also the analysis, of Clouds scheduling approaches. In this article, the aim is to show the relevance of this analysis, and the ability to use these metrics in green scheduling. The *Validation Methodoly* (Section 4) highlights the genetic algorithm implemented for this study, which includes energy-efficiency, performance and QoS metrics in its objective function. The evaluation phase includes six simulations of six different algorithms. Three simulations use only simple VM allocation (BFS,RR and GA), and the three others use reallocation during the simulation (BFS-ReAlloc, RR-ReAlloc and GA-ReAlloc). Results of these simulations allow to analyze metrics including: *Energy consumption*, *Response-Time*, *Robustness*, and *Dynamism*. The number of migrations induced by reallocation is also considered.

This evaluation phase highlights that the analysis of very basic algorithms, as Round-Robin, can be seen differently when more than only the two common *Energy consumption* and *Response-Time* metrics are considered. The *Robustness* and the *Dynamism* enrich the analysis of these results, and show that RR is much better than the other algorithms in terms of these two metrics. Another conclusion on having more QoS parameters included in scheduling concerns the tuning of one or more metrics. For example, the GA configuration allows to advantage the performance or the ability of the Cloud to quickly react to users requests, or otherwise promote the Cloud to ensure a low level of services' failure.

Proposed perspectives first include the introduction of the concept of *complex services* which will be composed of multiple *elementary services* executed with dependencies between them, as a DAG (Direct Acyclic Graph). This step will allow to have a more detailed response-time computing, with delays between *elementary service* and also will complicate the use of the DVFS. Indeed, if one service of the DAG has to be slowed down, the other services will have to run at the same frequency. These services' configuration will lead to use other quality of service metrics to adapt the GA's to this kind of services. Another perspective is to compare the GA with a custom algorithm in which all QoS parameters will be integrated, and also with an LP (Linear Programming) resolution which returns the optimal solution for a given multi-objective configuration.

7. Acknowledgement

The experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

The work presented in this paper has been funded by the ANR in the context of the project SOP, ANR-11-INFR-001.

References

[1] K. Kritikos, B. Pernici, P. Plebani, C. Cappiello, M. Comuzzi, S. Benrernou, I. Brandic, A. Kertész, M. Parkin, M. Carro, A survey on service quality description, ACM Comput. Surv. 46 (1) (2013) 1:1–1:58.

[2] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, et al., A taxonomy and survey of energy-efficient data centers and cloud computing systems, Advances in Computers 82 (2) (2011) 47–111.

[3] Z. Li, L. O'Brien, H. Zhang, R. Cai, On a catalogue of metrics for evaluating commercial cloud services, in: Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on, 2012, pp. 164–173.

[4] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of internet services and applications 1 (1) (2010) 7–18.

[5] B. Rimal, E. Choi, I. Lumb, A taxonomy and survey of cloud computing systems, in: INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, 2009, pp. 44–51.

[6] T. Kolpe, A. Zhai, S. Sapatnekar, Enabling improved power management in multicore processors through clustered dvfs, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2011, 2011, pp. 1 –6.

[7] G. Sakellari, G. Loukas, A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing., Simulation Modelling Practice and Theory 39 92–103.

[8] D. G. d. Lago, E. R. M. Madeira, L. F. Bittencourt, Power-aware virtual machine scheduling on clouds using active cooling control and dvfs, in: Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science, MGC '11, ACM, New York, NY, USA, 2011, pp. 2:1–2:6.

[9] H. S. Abdelsalam, K. Maly, R. Mukkamala, M. Zubair, D. Kaminsky, Analysis of energy efficiency in clouds, in: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:, IEEE, 2009, pp. 416–421.

[10] S. Islam, K. Lee, A. Fekete, A. Liu, How a consumer can measure elasticity for cloud platforms, in: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE '12, ACM, New York, NY, USA, 2012, pp. 85–96.

[11] E. Gelenbe, R. Lent, M. Douratsos, Choosing a local or remote cloud, in: Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on, IEEE, 2012, pp. 25–30.

[12] J. Baliga, R. Ayre, K. Hinton, R. Tucker, Green cloud computing: Balancing energy in processing, storage, and transport, Proceedings of the IEEE 99 (1) (2011) 149–167.

[13] S. K. Garg, C. S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of {HPC} applications on distributed cloud-oriented data centers, Journal of Parallel and Distributed Computing 71 (6) (2011) 732 – 749, special Issue on Cloud Computing.

[14] A. Beloglazov, R. Buyya, Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers, in: Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, ACM, 2010, p. 4.

[15] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers, in: Services Computing (SCC), 2010 IEEE International Conference on, IEEE, 2010, pp. 514–521.

[16] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, 2010, pp. 826–831.

[17] T. V. T. Duy, Y. Sato, Y. Inoguchi, Performance evaluation of a green scheduling algorithm for energy savings in cloud computing, in: Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, IEEE, 2010, pp. 1–8.

[18] W. Binder, N. Suri, Green computing: Energy consumption optimized service hosting, in: Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 117–128.

[19] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of the 2008 Conference on Power Aware Computing and Systems, HotPower'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 10–10.

[20] G. Dasgupta, A. Sharma, A. Verma, A. Neogi, R. Kothari, Workload management for power efficiency in virtualized data centers, Commun. ACM 54 (7) (2011) 131–141.

[21] T. Rajendran, P. Balasubramanie, R. Cherian, An efficient ws-qos broker based architecture for web services selection, International Journal of Computer Applications 1 (9) (2010) 110–115.

[22] International Journal of Web Services Practices (IJWSP), Vol. 5, 2010.

[23] I. B. Areas, Functional and nonfunctional requirements.

[24] L. Chung, B. Nixon, E. Yu, J. Mylopoulos, Non-functional requirements, Software Engineering.

[25] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, P. Stenström, The worst case execution time problem overview of methods and survey of tools, ACM Trans. Embed. Comput. Syst. 7 (3) (2008) 36:1–36:53.

[26] V. Raghunathan, S. Ravi, G. Lakshminarayana, High-level synthesis with variable-latency components, in: VLSI Design, 2000. Thirteenth International Conference on, 2000, pp. 220–227.

[27] R. W. Hockney, The communication challenge for mpp: Intel paragon and meiko cs-2, Parallel Comput. 20 (3) (1994) 389–398.

[28] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling tcp throughput: a simple model and its empirical validation, in: Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM '98, ACM, New York, NY, USA, 1998, pp. 303–314.

[29] Y.-S. Dai, B. Yang, J. Dongarra, G. Zhang, Cloud service reliability: Modeling and analysis, in: The 15th IEEE Pacific Rim International Symposium on Dependable Computing, 2009.

[30] J. D. Kalbfleisch, R. L. Prentice, The statistical analysis of failure time data, Vol. 360, John Wiley & Sons, 2011.

[31] M. Patterson, Energy efficiency metrics, Energy Efficient Thermal Management of Data Centers, Springer, 2012.

[32] S. Subashini, V. Kavitha, A survey on security issues in service delivery models of cloud computing, Journal of Network and Computer Applications 34 (1) (2011) 1 – 11.

[33] D. Elgesem, The structure of rights in directive 95/46/ec on the protection of individuals with regard to the processing of personal data and the free movement of such data, Ethics and Information Technology 1 (4) (1999) 283–293.

[34] K. P. Puttaswamy, C. Kruegel, B. Y. Zhao, Silverline: toward data confidentiality in storage-intensive cloud applications, in: Proceedings of the 2nd ACM Symposium on Cloud Computing, ACM, 2011, p. 10.

[35] A. Juels, B. S. Kaliski Jr, Pors: Proofs of retrievability for large files, in: Proceedings of the 14th ACM conference on Computer and communications security, ACM, 2007, pp. 584–597.

[36] C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, in: Proceedings of the 16th ACM conference on Computer and communications security, ACM, 2009, pp. 213–222.

[37] F. Nie, F. Xu, R. Qi, Saml-based single sign-on for legacy system, in: Automation and Logistics (ICAL), 2012 IEEE International Conference on, 2012, pp. 470–473.

[38] A.-R. Sadeghi, T. Schneider, M. Winandy, Token-based cloud computing, in: A. Acquisti, S. Smith, A.-R. Sadeghi (Eds.), Trust and Trustworthy Computing, Vol. 6101 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 417–429.

[39] T. Mather, S. Kumaraswamy, S. Latif, Cloud security and privacy: an enterprise perspective on risks and compliance, O'Reilly, 2009.

[40] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, Role-based access control models, Computer 29 (2) (1996) 38–47.

[41] J. Li, M. N. Krohn, D. Mazières, D. Shasha, Secure untrusted data repository (sundr), in: OSDI, Vol. 4, 2004, pp. 9–9.

[42] R. C. Merkle, A digital signature based on a conventional encryption function, in: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, CRYPTO '87, Springer-Verlag, London, UK, UK, 1988, pp. 369–378.

[43] R. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, B. S. Lee, Trustcloud: A framework for accountability and trust in cloud computing, in: Services (SERVICES), 2011 IEEE World Congress on, 2011, pp. 584–588.

[44] Z. Xiao, Y. Xiao, Security and privacy in cloud computing, Communications Surveys Tutorials, IEEE 15 (2) (2013) 843–859.

[45] S. Nakahara, H. Ishimoto, A study on the requirements of accountable cloud services and log management, in: Information and Telecommunication Technologies (APSITT), 2010 8th Asia-Pacific Symposium on, 2010, pp. 1–6.

[46] R. M. Needham, M. D. Schroeder, Using encryption for authentication in large networks of computers, Communications of the ACM 21 (12) (1978) 993–999.

[47] A. O. W. paper, Information lifecycle management for business data, `http://www.oracle.com/us/026964.pdf` (2007).

[48] C. S. Alliance, Security guidance for critical areas of focus in cloud computing v3.0, `https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf`.

[49] T. Guérout, T. Monteil, G. Da Costa, R. Neves Calheiros, R. Buyya, M. Alexandru, Energy-aware simulation with dvfs, Simulation Modelling Practice and Theory 39 (2013) 76–91.

[50] T. Wiedmann, J. Minx, A definition of'carbon footprint', Ecological economics research trends 2 (2007) 55–65.

[51] R. E. Miller, J. W. Thatcher, Complexity of computer computations, Springer, 1972.

[52] R. M. Karp, Reducibility among combinatorial problems, Springer, 1972.

[53] C. H. Papadimitriou, Computational complexity, John Wiley and Sons Ltd., 2003.

[54] M. R. Garey, D. S. Johnson, Computer and intractability, A Guide to the NP-Completeness. Ney York, NY: WH Freeman and Company.

[55] D. E. Goldberg, The existential pleasures of genetic algorithms, Genetic Algorithms in Engineering and Computer Science, Winter G ed. New York: Wiley (1995) 23–31.

[56] E.-G. Talbi, Metaheuristics: from design to implementation, Vol. 74, John Wiley & Sons, 2009.

[57] M. Srinivas, L. Patnaik, Genetic algorithms: a survey, Computer 27 (6) (1994) 17–26.

[58] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41 (1) (2011) 23–50.