

A MODELING STRATEGY FOR CLOUD INFRASTRUCTURE PLANNING CONSIDERING PERFORMANCE AND COST REQUIREMENTS

Erica Sousa¹; Fernando Lins¹; Eduardo Tavares²; Paulo Maciel²

Department of Statistical and Informatics, Federal Rural University of Pernambuco, Recife, PE, Brazil¹

Center of Informatics, Federal University of Pernambuco, Recife, PE, Brazil²

erica.sousa@deinfo.ufrpe.br; fernando.aires@deinfo.ufrpe.br; eagt@cin.ufpe.br; prmm@cin.ufpe.br

Abstract

Cloud computing is a model that allows resources to be offered as services over the Internet. Due to dynamic and virtualized nature of cloud environments and diversity of client requests, providing the expected service quality while avoiding over-provisioning is not a simple task. To ensure that the provisioned service is acceptable, providers must exploit techniques and mechanisms that guarantee a minimum level of service quality. The performance evaluation of cloud infrastructures has been receiving considerable attention by providers as a prominent activity for improving service quality. This paper presents a methodology, representation models and an optimization model for cloud infrastructures planning. The proposed methodology and models aim to provide support for the planning and selection of cloud infrastructures according to performance and cost requirements. The optimization model generates cloud infrastructures with different hardware and software configurations and these infrastructures are represented using SPN and cost models. Using the proposed technique, performance metrics allows the selection of cloud infrastructures that meet the client requirements. A case study based on Eucalyptus platform is adopted to demonstrate the feasibility of the methodology and models.

Keywords: Cloud Computing; Performance Evaluation; Cost Evaluation; Stochastic Petri Net; Greedy Randomized Adaptive Search Procedure

1. INTRODUCTION

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (NIST, 2015).

Cloud computing is also a combination of technologies that have been developed over the last several decades, which includes virtualization, grid computing, cluster computing and utility computing. Due to market pressure, cloud computing technologies have rapidly evolved in order to let users focus on business aspects rather than infrastructure issues, hence fostering business competitiveness (Velte *et al.*, 2010; Chee and Franklin, 2009).

Cloud providers are not only required to supply services properly, but also, to meet their expectations in the context of performance and cost. Indeed, cloud computing services have been massively expanding, thus, demanding companies to offer services with reliability, high availability, performance, scalability and security at affordable costs. Performance evaluation is a prominent

activity for improving service quality, infrastructure planning, and for tuning system components (Jain, 1991; Menasce and Almeida, 2004).

This paper extends a previous work (Sousa, 2014) in which a modeling strategy for cloud infrastructure planning with different software and hardware configurations, according to performance and cost requirements is proposed. The current paper proposes a methodology, representation models and an optimization model for cloud infrastructure planning. The methodology allows the generation of cloud infrastructures through an optimization model and the representation of these cloud infrastructures through the performance and cost models. These models are evaluated and the results are used for the selection of cloud infrastructures with different hardware and software configurations according to performance and cost requirements. Different from the previous work, this work provides the generation of cloud infrastructures and the selection of these infrastructures through an optimization model.

An optimization model based on Greedy Randomized Adaptive Search Procedure (GRASP) (Feo and Resende, 1995) allows the generation of various cloud infrastructures through the assignment of software sets to hardware sets of these infrastructures. A performance model based on SPN (Murata, 1989) and a cost model based on an equation

represent these cloud infrastructures and calculate the response time, processor utilization, memory utilization, equipment cost and software cost of these infrastructures.

This paper is organized as follows. Section 2 presents related works. Section 3 introduces basic concepts. Section 4 shows the adopted methodology. Section 5 presents the optimization model. Section 6 shows the performance and cost models. Section 7 describes a case study and some experimental results. Section 8 provides concluding remarks and future works.

2. RELATED WORK

In the last few years, some works evaluated the performance of public clouds for scientific applications. Ostermann *et al.* (Ostermann *et al.*, 2010) present a performance analysis of Amazon EC2 for scientific computing using specific benchmarks, such as LMBench, Bonnie, CacheBench and HPCC. Similarly, in (Iosup *et al.*, 2010), the authors evaluate the performance of public clouds, more specifically, Amazon EC2, GoGrid (GG), Elastic Hosts (EH) and Mosso. In (Shafer, 2010), the Eucalyptus platform is tested in a variety of configurations to determine its suitability for applications with high I/O performance requirements, such as the Hadoop MapReduce framework for data-intensive computing. In (Ghoshal *et al.*, 2010), the authors evaluate I/O performance using IOR benchmarks, which is a set of tools for understanding the I/O performance of high-performance parallel file systems.

Xiong *et al.* (Xiong and Perros, 2009) present a queuing network model for calculating the relationship among the maximal number of customers, the minimal service resources and the highest level of services in order to deliver QoS guaranteed services in cloud environments. Ghosh *et al.* (Ghosh *et al.*, 2011) propose an approach based on Markov models to assess the public clouds, such as Amazon EC2 and IBM Smart Cloud Enterprise. The performance evaluation is based on the response time and service time provided by public clouds. The response time is the time between the requests processing and use of a pre-configured image to create a virtual machine. This paper also presents classes of cloud computing systems, which physical machines are configured according to the availability of its resources and virtual machines are provisioned with multiple images, aiming to minimize the response time. Physical machines are partitioned into three pools: hot, warm and cold. In the hot pool, pre-instantiated virtual machines are provisioned with a minimum response time. In the warm pool, virtual machines are instantiated with a response time corresponding to the provisioning duration. In the cold pool, virtual machines need a start-up time before being provisioned. Experimental results demonstrate the effect of each pool class in performance of public clouds.

Other works propose the cost evaluation of cloud infrastructures. Martens *et al.* (Martens *et al.*, 2012) show that the analysis of relevant cost types and factors of cloud computing services is an important pillar of decision-making in cloud computing management. Thus, such paper presents a total cost of ownership (TCO) approach for cloud computing services. Li *et al.* (Li *et al.*, 2009) provide metrics and equations for calculating the cloud total cost of ownership (TCO) and utilization cost, considering the elastic feature of cloud infrastructure and the adopted virtualization technology. That paper provides a foundation for evaluating economic efficiency of cloud computing and indications for cost optimization of cloud computing infrastructures.

Some works present an optimization model for data center planning that compose the cloud infrastructure. Callou (Callou *et al.*, 2014) proposes an integrated approach to evaluate and optimize the dependability, cost and sustainability issues of data center infrastructures. The authors adopt reliability block diagram (RBD), stochastic Petri nets (SPN) and energy-flow (EFM) models, as well as an optimization method based on GRASP. Ferreira (Ferreira *et al.*, 2013) proposes a power load distribution algorithm (PLDA) based on the Ford-Fulkerson algorithm to optimize energy distribution of data center power infrastructures.

Differently from previous studies, this paper proposes a methodology, representation models and an optimization model for cloud infrastructure planning. The methodology permits the creation of various cloud infrastructures through an optimization modeling and the generation of performance and cost models in relation to these cloud infrastructures. The methodology also allows the selection of cloud infrastructures according to performance and cost requirements.

3. PRELIMINARIES

This section presents an overview of relevant concepts for a better understanding of this work.

3.1 CLOUD PLATFORM

Eucalyptus is an open-source cloud computing platform that allows the creation of private clusters in enterprise datacenters (Eucalyptus, 2015). Eucalyptus provides API compatibility with the most popular commercial cloud computing infrastructure, namely, Amazon Web Services (AWS) (Eucalyptus, 2015), which allows management tools to be adopted in both environments. This framework is designed for compatibility across a broad spectrum of Linux distributions (e.g., Ubuntu, RHEL, OpenSUSE) (Eucalyptus, 2015) and virtualization hypervisors (e.g., KVM, Xen) (Eucalyptus, 2015). Figure 1 shows the Eucalyptus architecture.

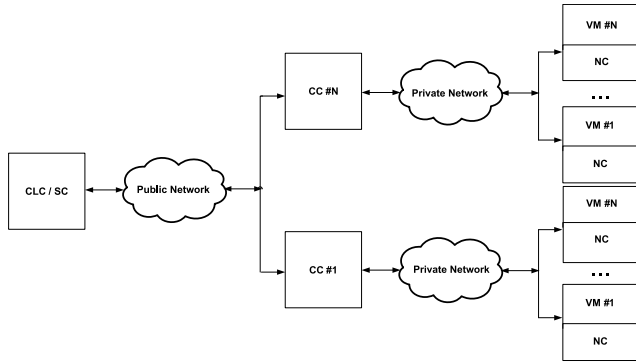


Figure 1. Eucalyptus Platform

Eucalyptus system is composed of several components that interact through interfaces (Eucalyptus, 2015). There are four major components, each with a service-based interface (Eucalyptus, 2015):

- Cloud Controller (CLC) - The CLC is the entry-point into the cloud computing for users and administrators. It queries node managers for information about resources, performs high-level scheduling decisions and implements them by making requests to cluster controllers.
- Cluster Controller (CC) - The CC acts as a gateway between the CLC and individual nodes in the data center. This component collects information on schedules and execution of virtual machine (VM) on specific node controllers and manages the virtual instance network. The CC must be in the same Ethernet broadcast domain as the nodes it manages.
- Node Controller (NC) - The NC contains a pool of physical computers that provides generic computational resources to the cluster. Each of these physical machines contains a node controller service that is responsible for controlling the execution, inspection and termination of VM instance. The NCs also configure the hypervisors and host OSs as defined by the CC. Each NC executes in the host domain (in KVM) or driver domain (in Xen) (Eucalyptus, 2015).
- Storage Controller (SC) - The SC is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data (Eucalyptus, 2015).

3.2 STOCHASTIC PETRI NET

Petri nets (PN) (Murata, 1989) are a family of formalisms very well suited for modeling several system types, since concurrency, synchronization, communication mechanisms as well as deterministic and probabilistic delays are naturally represented. In general, Petri nets are a bipartite directed graph, in which places (represented by circles) denote local states and transitions (depicted as

rectangles) represent actions. Arcs (directed edges) connect places to transitions and vice-versa.

This work adopts a particular extension, namely, Stochastic Petri Nets (SPN) (Marsan *et al.*, 1998), which allows the association of probabilistic delays to transitions using the exponential distribution, and the respective state space is isomorphic to continuous-time Markov chains (Trivedi, 2002).

Figure 2 depicts the simple component model using SPN. The delay assigned to s-transition F is the MTTF (Mean Time To Failure) and the delay of s-transition R is the MTTR (Mean Time To Repair). SPN allows the adoption of simulation techniques for obtaining performance metrics, as an alternative to the Markov chain generation.

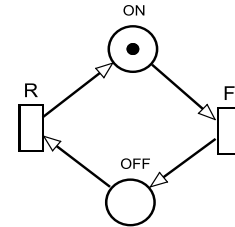


Figure 2. Simple Model - SPN

Phase approximation technique can be applied for modeling non-exponential activities. A variety of performance activities can be constructed in SPN models by using throughput subnets and s-transitions, as depicts in Figures 3, 4 and 5. This throughput subnets and s-transitions represent polynomial-exponential functions, such as the Erlang, Hypoexponential and Hyperexponential distributions (Desrochers and Al-Jaar, 1995).

Measured data from a system (empirical distribution) with an average μ_D and a standard deviation σ_D must adjust their stochastic behaviour through the phase approximation technique. The inverse of the variation coefficient (Equation 1) of measured figure allows the selection of which distribution matches it best. In this work, the adopted distribution for moment matching are the Erlang, Hypoexponential and Hyperexponential distributions.

$$\frac{1}{CV} = \frac{\mu_D}{\sigma_D}$$

Equation 1.

When the inverse of the variation coefficient ($1/CV$) is a whole number and different from one, the empirical figure should be characterized by an Erlang distribution, represented in SPN by a sequence of exponential transitions whose length (γ) is calculated by Equation 2.

$$\gamma = \left(\frac{\mu}{\sigma}\right)^2$$

Equation 2.

The rate (λ) of each exponential transition is calculated by Equation 3.

$$\lambda = \left(\frac{\gamma}{\mu}\right)$$

Equation 3.

The Petri Net model depicted in Figure 3 represents an Erlang distribution.

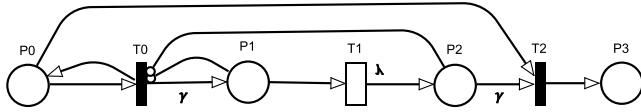


Figure 3. Erlang Distribution Net

When the inverse of the variation coefficient ($1/CV$) is a number larger than one (but not an integer), the empirical figure is represented by a hypoexponential distribution which is represented by a SPN composed of a sequence whose length (γ) is calculated by Equation 4.

$$\left(\frac{\mu}{\sigma}\right)^2 \leq \gamma < \left(\frac{\mu}{\sigma}\right)^2$$

Equation 4.

The transition rates (λ_1 and λ_2) of exponential transitions are calculated by Equations 5 and 6.

$$\lambda_1 = \left(\frac{1}{\mu_1}\right)$$

Equation 5.

$$\lambda_2 = \left(\frac{1}{\mu_2}\right)$$

Equation 6.

The average time (expected values) (μ_1 and μ_2) assigned to the exponential transitions are calculated by the Equations 7 and 8.

$$\mu_1 = \mu \mp \frac{\sqrt{\gamma(\gamma + 1)\sigma^2 - \gamma\mu^2}}{\gamma + 1}$$

Equation 7.

$$\mu_2 = \gamma\mu \pm \frac{\sqrt{\gamma(\gamma + 1)\sigma^2 - \gamma\mu^2}}{\gamma + 1}$$

Equation 8.

The model presented in Figure 4 is a net that depicts a hypoexponential distribution.

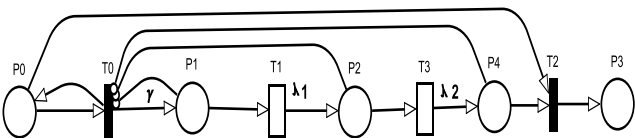


Figure 3. HypoExponential Distribution Net

When the inverse of the variation coefficient ($1/CV$) is a number smaller than 1, the empirical distribution should be represented by an hyperexponential distribution. The exponential transition rate (λ_h) should be calculated by Equation 9.

$$\lambda_h = \left(\frac{2\mu}{\mu^2 + \sigma^2}\right)$$

Equation 9.

The weights (ϖ_1 and ϖ_2) of immediate transitions are calculated by Equations 10 and 11.

$$\varpi_1 = \left(\frac{2\mu^2}{\mu^2 + \sigma^2}\right)$$

Equation 10.

$$\varpi_2 = 1 - \varpi_1$$

Equation 11.

The Petri Net model that represents this hyperexponential distribution is depicted in Figure 5.

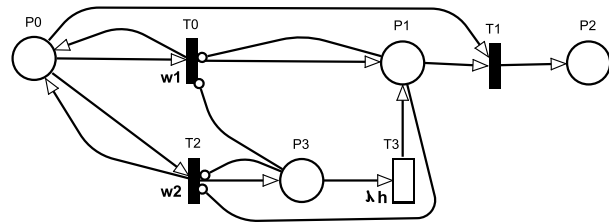


Figure 4. HyperExponential Distribution Net

3.3 GRASP METAHEURISTIC

The GRASP metaheuristic consists of an iterative process in which each interaction provides an improved solution to the optimization problem. Each iteration consists of the construction phase and local search phase (Feo and Resende, 1995).

The construction phase generates a random element to be basis for improving the solution to the problem. Next, the local search phase investigates the neighborhood of the solution in order to obtain an improved solution. Moreover, each element is randomly selected from a restricted candidate list (RCL) and added to the solution set. The stopping criterion of the iterations is based on the maximum number of iterations and best solution found (Feo and Resende, 1995).

A feasible solution is iteratively constructed until the solution is complete. The elements that compose the solution candidates are ordered in the candidate list (CL) which contains all the candidates. This candidate list is ordered by a deterministic function that measures the benefit of the selected element in relation to the constructed solution. A subset called restricted candidate list (RCL) is composed by the best elements that compose the candidates

list. The best solutions of the iterations is returned as a result with a greater availability and a lower cost (Feo and Resende, 1995).

The local search phase investigates the neighborhood of the solution. If an improvement is found regarding the current solution, this solution is updated and the neighborhood around the new solution is investigated. The process repeats until no improvement is found (Feo and Resende, 1995).

4. METHODOLOGY FOR CLOUD INFRASTRUCTURE PLANNING

The cloud infrastructures planning is an essential activity because it enables providers to have sufficient resources to dynamically allocate and release resources, when subjected to different client requests. This planning also allows the scaling of the cloud infrastructures to support high workload levels with acceptable response times. The performance evaluation of cloud infrastructures allows the attendance of the client requests while maintaining the service quality offered (Xiong and Perros, 2009).

This section presents the proposed methodology (see Figure 6) to cloud infrastructures planning considering different hardware and software configurations, according to performance and cost requirements. Basically, the methodology is divided into five activities: Performance and Cost Planning; Method for Performance Evaluation; Method for Cost Evaluation; Analysis of Performance and Cost Scenarios; and Selection of Performance and Cost Scenarios.

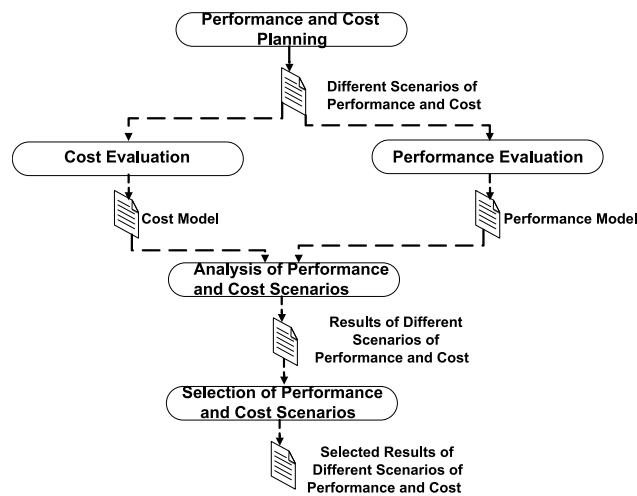


Figure 5. Methodology for Cloud Infrastructure Planning

The first activity aims to plan cloud infrastructures with different hardware and software configurations through an optimization model. This model is presented in Section 5. As an example, there are scenarios with different software configurations (cloud platforms, operating systems,

databases and web servers) and hardware configurations (processor, physical memory and secondary memory).

The Performance Evaluation activity provides the performance parameters and generation of a stochastic model for performance evaluation of cloud infrastructures with different hardware and software configurations. The performance model and metrics are presented in Section 6.1. This performance model also evaluates cloud infrastructure planning considering future demands. For a better understanding, this activity is detailed in the Section 4.1.

The Cost Evaluation activity proposes equations for evaluating the equipment and software costs of cloud infrastructures. These equations are presented in Section 6.2.

The Analysis of Performance and Cost Scenarios activity allows the evaluation of performance and cost models in relation to cloud infrastructures with different hardware and software configurations, taking into account response time, processor utilization, memory utilization, equipment cost and software cost.

The Selection of Performance and Cost Scenarios activity presents the selected cloud infrastructures according to performance and cost requirements. The response time, processor utilization, memory utilization, equipment cost and software cost results are adopted as selection criteria.

4.1 PERFORMANCE EVALUATION

Performance Evaluation (see Figure 7) is composed of Understanding the Environment, Preparing the Environment, Workload Generation, Measurement, Abstract Model Generation, Qualitative Analysis of Abstract Model, Refined Model Generation, Qualitative Analysis of Refined Model, Mapping Metrics and Quantitative Analysis of Refined Model.

- Understanding the Environment includes the identification of the cloud platform, system architecture and performance and cost requirements that must be considered.
- Preparing the Environment provides the cloud platform and client system configuration.
- Workload Generation defines the workload levels that should be considered for evaluation of the cloud environment. As an example, JMeter (JMeter, 2015) generates different levels of workload to Moodle (Moodle, 2015) through the creation of activities in this VLE.
- Measurement defines the measurement process that consists in the choice of the metrics, measurement tool, statistical methods for data analysis and treatment. The metrics are chosen according to the representation level in relation to the cloud environment. After the metrics measurement, the collected data is analyzed in order to obtain the average (μ_D) and standard deviation (σ_D). These statistics provide the selection of the polynomial-

exponential distribution that best fits the empirical distribution (collected data).

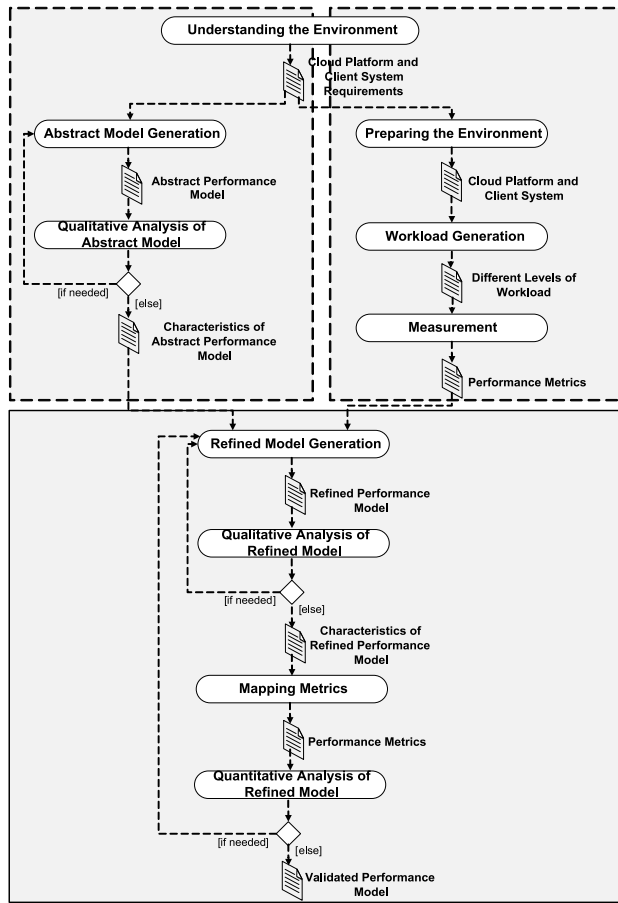


Figure 6. Performance Evaluation Activity

- Abstract Model Generation allows the generation of the performance model that is used to estimate the performance of the cloud computing which is submitted to different workloads or variations in the cloud infrastructure.
- Qualitative Analysis of Abstract Model corresponds to the verification of the qualitative properties (e.g., liveness and boundedness) (Murata, 1989) and the validation of the abstract performance model. The INA (INA, 2015) evaluates if the abstract performance model is deadlock free, for example.
- Refined Model Generation provides the generation of the performance model considering the abstract model and statistics obtained in the measurements. These statistics suggest the polynomial-exponential distribution that best fits the empirical distribution (collected data). This adaptation is performed by phase approximation technique (Desrochers and Al-Jaar, 1995), which computes the first and second moments of the empirical distribution, the average (μ_D) and standard deviation (σ_D), and associates to the first and second

moments of the s-transition of the abstract performance model.

- Qualitative Analysis of Refined Model represents the qualitative analysis and validation of the refined model (Murata, 1989). The INA (INA, 2015) evaluates if the refined performance model is deadlock free, for example.
- Mapping Metrics corresponds to the representation of performance criteria set on metrics through references to elements of the refined model. This work represents the response time with the SPN elements, for example.
- Quantitative Analysis of Refined Model verifies the results of refined performance model and compares with results obtained through measurements. These results should be equivalent with an acceptable error of accuracy. This paper adopts the t-paired test to quantitatively evaluate the refined performance model.

5. OPTIMIZATION MODEL

This section presents an optimization model based on GRASP metaheuristic for generating cloud infrastructures, considering different software set (e.g., cloud platforms, operating systems, databases and web servers) and hardware set (e.g., processor, physical memory and secondary memory). The optimization model is presented in Algorithm 1.

```

1: f(s):=0;
2: for i=1 to i=MaxIter do
3:  s':=GreedRandomized();
4:  if elite set f(s) has at least p elements then
5:   if s' is not feasible then
6:    Randomly select a new solution s' ∈ f(s);
7:   end if
8:   s':= ApproxLocalSearch(s');
9:   Randomly select a solution s' ∈ f(s);
10:  if elite set f(s) is full then
11:   if c(s') ≤ maxc(s) | s ∈ f(s) and s' ≠ f(s) then
12:    Replace the element most similar to s' among
all;
13:    elements with cost worst than s';
14:   end if
15:  else if s' ≠ f(s) then
16:   f(s) := f(s) ∪ s';
17:  end if
18:  else if s' is feasible and s' ≠ f(s) then
19:   f(s) := f(s) ∪ s';
20:  end if
21: end for
22: return s* = minc(s) | s ∈ f(s);

```

Algorithm 1. GRASP

The input data are the software set (SS), hardware set (HS) and the output data is an assignment vector s^*

specifying the software set assigned to each hardware set of the cloud infrastructure.

The set of elite solutions $f(s)$ for the cloud infrastructure is initialized with 0 in Line 1. The maximum number of interactions $MaxInter$ is computed from Line 2-21. The maximum number of interactions $MaxInter$ is defined by the user. During each iteration, a random and greedy solution s' is generated in Line 3. If the set of elite solutions $f(s)$ does not have at least ρ elements and s' is feasible, sufficiently different from all other solutions in the set of elite solutions $f(s)$, s' is added to the set of elite solutions in Line 19. If the set of elite solutions $f(s)$ has at least ρ elements, the steps in Lines 5-17 are computed.

The construction phase does not guarantee the generation of a feasible solution. If this phase returns a non-feasible solution, a feasible solution s' is selected randomly from the set of elite solutions $f(s)$ in Line 6.

The local search phase uses the solution s' as a start point (Line 8), resulting in a local minimum s . If the set of elite solutions $f(s)$ is complete, s' is a better solution than the worst solution and $s' \neq f(s)$, then this solution is added to the set of elite solutions $f(s)$ (Line 12). Among all elite solutions with a less cost than s , the solution s most similar to s' is selected to be removed from the set of elite solutions $f(s)$. A solution s has a lower cost than s' whenever its memory utilization, processor utilization, response time, software cost and equipment cost are lower. However, if the set of elite solutions is not complete (solutions number), s' is added to the set of elite solutions in Line 16.

The construction phase provides the assignment of different software sets to the hardware sets of the cloud infrastructure. The Algorithm 2 presents the construction phase.

```

1: SSN := 0; HSN := 0; MNSS := N; MNHS := N;
2: for SSN = 0 to SSN = MNSS do
3: Randomly generate a software set SSI  $\in$  S;
4: SS := SS  $\cup$  SSI;
5: end for
6: for HSN = 0 to HSN = MNHS do
7: Randomly generate a hardware set HSI  $\in$  HS;
8: HS := HS  $\cup$  HSI;
9: Randomly select a software set SSI  $\in$  S;
10: Assign software set SSI to hardware set HSI;
11: end for
12: return assignment  $s \in f(s)$ ;

```

Algorithm 2. Construction Phase

The software set number (SSN) and hardware set number (HSN) are initialized with 0. The maximum number of software set (MNSS) and maximum number of hardware set (MNHS) are initialized with N, in Line 1. These numbers are defined by the user. In Line 3, the software set number (SSI) is randomly generated until a maximum number MNSS. Each software set (SSI) is added to software set (SS) in Line 4. In Line 7, hardware set (HS) is randomly generated until a maximum number MNHS. Each

hardware (HSI) is added to hardware set (HS) in Line 8. In Lines 9-10, the software set (SSI) is randomly selected and assigned to each hardware set (HSI) generated.

The local search phase provides a solution which is the local minimum from the solution s produced through construction phase. The Algorithm 3 presents the local search phase. This phase uses the neighborhood structure known as 1-move. In this neighborhood structure, the solution s is obtained by modification of the assignment in solution s . If this change results in a first improvement in relation to the previous solution s , it is the first fit local search. The search is repeated until a better solution occurs in the neighborhood. In this work, instead of evaluate all solutions in the neighborhood, a candidate list CLS is created with the best solutions. One of the best solutions is randomly selected and a movement is performed.

```

1: repeat
2: count := 0; CLS :=  $\emptyset$ ;
3: repeat
4:  $s' := Move(s)$ ;
5: if  $s'$  is feasible and  $cost(s') < cost(s)$  then
6: CLS := CLS  $\cup$   $s'$ ;
7: end if
8: count := count + 1;
9: until  $|count| \leq MaxCLS$  or  $count \geq MaxInter$ ;
10: if CLS  $\neq$  0 then
11: Randomly Select a solution  $s \in CLS$ ;
12: end if
13: until CLS = 0;
14: return  $s$ ;

```

Algorithm 3. Local Search Phase

The input data are the solution s , parameters $MaxCLS$ and $MaxInter$. Lines 1-13 are repeated until the local minimum production. In Line 2, the counter and candidate list (CLS) are initialized with 0. Each interaction in Lines 3-9, a movement in the neighborhood of s is performed without replacement of the previous solution through function $Move(s)$ (Line 4). If this neighbour is a better solution, it is inserted into CLS in Line 6. This procedure occur until the candidate list (CLS) becomes full or a maximum number of iterations is reached. The candidate list (CLS) size is defined by the user. In Lines 10-12, the candidate list is not empty, a solution $s \in CLS$ is randomly chosen. If the candidate list is empty, the procedure terminates returning the solution s .

6. PERFORMANCE AND COST MODELS

This section presents a SPN model and cost equations for representing cloud infrastructures and quantifying the response time, processor utilization, memory utilization, equipment cost and software cost of these infrastructures.

6.1 PERFORMANCE MODEL

The performance model is represented by a stochastic Petri net that assumes a client set performing requests to a cloud infrastructure. This model can represent the impact of the workload of different applications on response time and resources utilization of virtual machines of a cloud infrastructure.

Without loss of generality, the performance model is adopted to evaluate suitable hardware configurations for ensuring service levels with high processing and storage requirements as described in the Abstract Model Generation phase (see Section 4.1). Figure 8 depicts the conceived model, which is composed of 4 subnets: (i) client; (ii) clock; (iii) memory and (iv) processing infrastructure. Particularly, the metrics of interest is the response time, memory utilization and processor utilization.

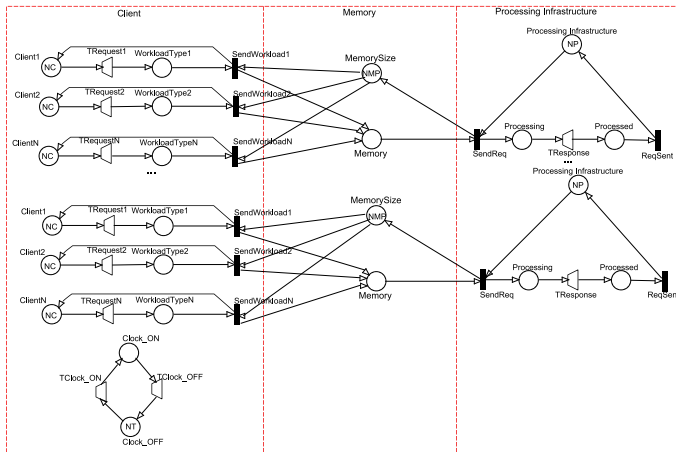


Figure 7. Performance Model

The Client Subnet represents the users requests for services offered by cloud computing and the marking NC associated to place Client defines the amount of users. The time associated to transition TRequest denotes the delay between users requests. The place WorkloadType represents different types of user requests. The weights assigned to immediate transitions SendWorkload1, SendWorkload2 and SendWorkload3 permit the selection of the types of user requests that will be sent to service hosted on the cloud with higher priority.

The Clock Subnet represents the sending frequency of users requests to the service offered by cloud computing and marking NT associated to place Clock_OFF is the number of timers that will be triggered. The delays associated to transitions TClock_ON and TClock_OFF represent the times to activate and shutdown the timers, respectively.

The enabling function ($\{\#Clock_ON=1\}$) assigned to immediate transitions SendWorkload1, SendWorkload2 and SendWorkload3 of the subnet Client provides the sending of different types of user requests with the frequency defined

by the delay TON_Clock associated to timed transition TClock_ON.

The Memory Subnet represents the physical memory adopted to instantiate the virtual machines on physical machines of the cloud computing. The marking NMP assigned to place MemorySize represents the total amount of primary memory allocated to instantiate of the virtual machine. Each marking on the place Memory denotes the memory utilization for a user request.

The Processing Infrastructure subnet represents the processing infrastructure adopted to instantiate the virtual machines on physical machines of the cloud computing. The marking NP associated to place Processing Infrastructure represents the total number of cores of the CPU provided by the cloud infrastructure to instantiate the virtual machines. The delay associated with timed transition TResponse represents the mean service time.

This abstract performance model is presented assuming the empirical distribution, since all timed transitions have the trapezoidal shape. This model can be refined in order to consider, for instance, polynomial-exponential distributions (Desrochers and Al-Jaar, 1995) (Section 3.2). Besides, this work adopts the following statements for estimating performance metrics: $P\{exp\}$ indicates the probability of the inner expression (exp); $E\{exp\}$ represents the mean for the expression (exp) and $exp = \#p$ denotes the number of tokens in place p; $W(T)$ represents the rate associated with the transition T.

Response time is estimated using the Little's law (Bakouch, 2011; Trivedi, 2008), according to the expression $T = N/\lambda$. T represents the mean time in the system, N indicates the average number of users in the system and λ represents the arrival rate of users to the system. $N = (E\{\#WorkloadType\} + E\{\#Memory\} + E\{\#Processing\})$ is the average number of user requests in the cloud environment and $\lambda = P\{\#Client>0\} \times W(TRequest)$ is the arrival rate of requests to cloud computing (Bakouch, 2011; Tavares et al., 2012).

Processor utilization is described by the expression $P\{\#ProcessingInfrastructure=0\}$, in which the place ProcessingInfrastructure represents the processing infrastructure of the virtual machine. This expression indicates the probability of no markings in place ProcessingInfrastructure. Memory utilization is estimated using $100 \times (E\{\#Memory\} \times QM)/QMT$, in which $E\{\#Memory\}$ represents the average number of markings in place Memory, QM is the memory amount represented by each marking in the SPN and QMT represents the total memory of the virtual machine.

6.2 COST MODEL

This section presents the equations adopted for estimating the costs of equipment and software in cloud computing.

6.2.1 EQUIPMENT COST

Equipment Cost comprises physical machines, routers, as well as switches, and it is estimated using equation bellow. Assuming N device types, EN_i is the amount of a specific device used in the IT infrastructure, and EC_i represents the cost of such item.

$$EC = \sum_{i=1}^N EN_i \times EC_i$$

Equation 12.

6.2.2 SOFTWARE COST

Software Cost is composed of client system, cloud platform, databases (DB), operating systems (OS), virtual machine monitor and web server. In this case, equation bellow is adopted, in which N is the number of distinct software components; SN_i is the amount of software of a specific type and SC_i is the respective unit cost.

$$SC = \sum_{i=1}^N SN_i \times SC_i$$

Equation 13.

7. CASE STUDY

This section presents a case study to illustrate the feasibility of the proposed methodology, representation and optimization models for cloud infrastructures planning with different hardware and software configurations. This study also evaluates the impact of these hardware and software configurations through the response time, processor utilization, memory utilization and cost. The results of this evaluation provides suggestions for cloud infrastructures that meet the performance and cost requirements.

The adopted system to evaluate if the proposed methodology, representation and optimization models can be used to cloud infrastructures planning consists of a virtual environment learning (VLE) configured on the Eucalyptus platform (Eucalyptus, 2015) (see Figure 9). The cloud environment consists of a cloud controller (CLC), a cluster controller (CC) and eleven node controllers (NC1 - NC11).

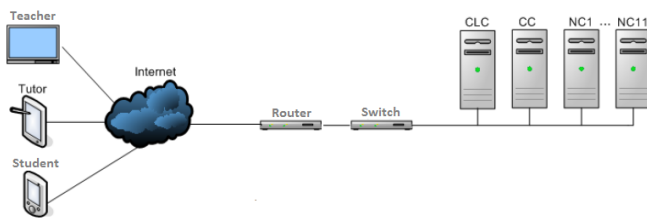


Figure 9. Moodle hosted on Eucalyptus Platform

The CLC was configured on a server, the CC also was set up on a server and NCs were configured on 11 servers.

The Moodle (Moodle, 2015) hosted on Eucalyptus platform was installed on different virtual machines of the Eucalyptus platform. These virtual machines are instantiated on servers where the NCs services are executed.

The cloud infrastructures are generated according to the proposed methodology. Every cloud infrastructure is conceived through two software configurations and a hardware set. These cloud infrastructures are selected according to the results of the response time, resource utilization and costs. In this study, the criteria are a maximum number of 10 cloud infrastructures with the response time smaller than 1.50 seconds, the processor utilization and memory utilization smaller than 95% and the total cost should be smaller than US\$ 20,000.00. These criteria are necessary for configuration of the VLE on cloud infrastructure.

The performance modeling of the generated cloud infrastructures were carried out through the measurement activities and modeling activities. The measurement activities consist of the Preparing the Environment, Workload Generation and Measurement. The Preparing the Environment concerns the two software configurations in the cloud infrastructures. The Workload Generation considers the development of the test script with the JMeter (JMeter, 2015) tool. The Measurement performs experiments and collection of performance metrics.

The preparation of the environment considers the software configuration 1 which is composed of the Eucalyptus platform (Eucalyptus, 2015), Moodle (Moodle, 2015), MySQL (MySQL, 2015), Ubuntu (Ubuntu, 2015) and Apache server (Apache, 2015) on the virtual machine of the Eucalyptus platform. This virtual machine consists of a dual-core processor, 2GB of memory and a secondary memory of 80GB. This activity also considers the software configuration 2 which differs from software configuration 1 with respect to the web server, because the Lighttpd server (Lighttpd, 2015) was configured in the virtual machine.

In the Workload Generation, a test script was developed to simulate 6 to 10 Moodle users. The users were enrolled in a course, following the standard user6 to user10 and password 12345678. This test script aims to simulate users interacting in the chat of the course. Users requesting access to the chat every 1 second.

The test script consists of six steps that are Start N Threads, UserID + SessionKey, Login on Moodle, Access the Course Page, Access the Chat and Logout on Moodle. The Start N Threads deals with the instantiation of N threads (N users). Thus, five experiments were performed for each software set configured to the cloud infrastructure. These experiments considered the instantiation of 6, 7, 8, 9 or 10 users.

In UserID + SessionKey, there is the generation of a number (i) ranging from 1 to N. This number is incremented each thread instantiated. For each thread are created a string student(i) and the password 12345678. The tuple username + password are required for user login. This step also deals

with the store of the user name, user ID and session key for each thread. In Login on Moodle, each thread performs an HTTP request to the login page of the Moodle containing the user name and password, and authenticates to the Moodle.

In Access the Course Page, each thread has access to page of the course of Performance Evaluation. Soon after, each thread is in a loop (time counter) and access to chat for 1 hour. The Access the Chat deals with the access of the user to chat of the course of Performance Evaluation and send a message. This step is carried out until the time in the loop is equal to 1 hour. The count of this time occurs independently for each thread. After one hour, each user logout on Moodle, in the Logout on Moodle.

The maximum number of users generated in this script was 10 due to processing and storage infrastructures of the cloud computing. The test script can be adapted to generate more numbers of users. Figure 10 shows the flowchart of the test script.

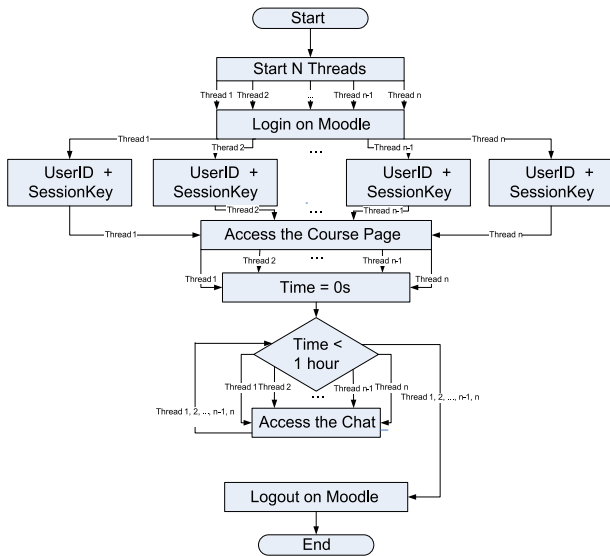


Figure 10. Flowchart of the Test Script

The measurement consists of the execution of the test script for experiments with 6 to 10 users. In each experiment are collected the response times for users requests through JMeter. The processor utilization and memory utilization of the virtual machines with the software configuration 1 and software configuration 2 are collected through the measurement script that was created with the mpstat and vmstat tools of the package sysstat (GODARD, 2015). When start the measurement script, the mpstat and vmstat tools run during a loop (time counter) with 1 hour. After this time, a storage file in text format is generated with performance metrics that were collected through the mpstat and vmstat tools. Figure 11 shows the flowchart of the measurement script.

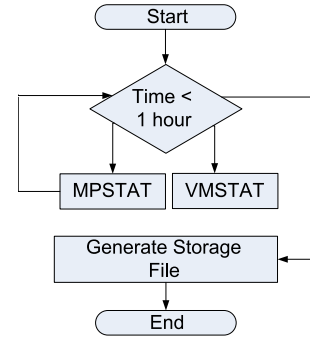


Figure 11. Flowchart of the Measurement Script

After the execution of each experiment, the virtual machine is restarted. This procedure releases the computing resources in the current test, preventing that this test disturb the next test.

Tables 1 and 2 present the response time (RT), processor utilization (PU) and memory utilization (MU), measured in relation to requests of 6 to 10 users. These requests were submitted to cloud infrastructures with the Software Configuration 1 - SC1 and Software Configuration 2 - SC2.

Table 1. Performance Measurement Result - Software Configuration 1

User Number	RT(s)	PU(%)	MU(%)
6	0.6189	92.3718	80.7475
7	0.7097	95.0311	83.5761
8	0.8110	97.1305	84.2699
9	0.9127	98.4991	86.4405
10	1.0370	99.0142	80.3260

Table 2. Performance Measurement Result - Software Configuration 2

User Number	RT(s)	PU(%)	MU(%)
6	1.2411	79.3537	80.6797
7	1.4748	80.0457	83.2397
8	1.7112	80.3726	86.2422
9	1.9762	80.1633	89.3239
10	2.2250	89.6872	91.4094

In the modeling phase, the representation of cloud infrastructures was performed based on the Refined Model Generation. The Performance Model (see Figure 8) was adopted for evaluating of response time and resource utilization of the conceived cloud infrastructures.

The refinement of the Performance Model (see Figure 8) was based on processor demand time and memory utilized for requests of 6 to 10 users to chat. The processor demand time is calculated by Service Demand Law (Menascé, 2004) according to equation below. PDT is the processor demand

time, PU indicates the processor utilization and TH represents the throughput.

$$PDT = \frac{PU}{TH}$$

Equation 14.

Table 3 and 4 present the processor demand times (PDT) and memory utilized (MUD) measured for requests of 6 to 10 users. These requests were submitted to cloud infrastructures with the Software Configuration 1 - SC1 and 2 Software Configuration - SC2.

Table 3. Results of the Processor Demand Time

User Number	PDT(s) -SC1	PDT(s) -SC2
6	0.2003	0.4201
7	0.2357	0.4864
8	0.2778	0.5555
9	0.3178	0.6343
10	0.3680	0.7023

Table 4. Results of the Memory Utilized

User Number	MUD (MB) - SC1	MUD(MB) - SC2
6	1653.7080	1652.3208
7	1711.6395	1704.7480
8	1725.8466	1766.2410
9	1770.3006	1829.3545
10	1836.7940	1872.0635

These results were analyzed to choose the most appropriate polynomial-exponential distribution to represent the time between sending requests (TSR) and processor demand times (PDT). Averages (μ_D) and standard deviations (σ_D) of these times were analyzed to calculate the inverse of the variation coefficient. Table 5 shows the results of averages (μ_D), standard deviations (σ_D) and the chosen polynomial-exponential distributions.

Table 5. Average, Standard Deviation and Polynomial-Exponential Distribution

Software Set	Metrics	μ_D (s)	σ_D (s)	Probability Distribution
SC1	TSR	0.15	0.008	HypoExponencial
SC1	PDT	0.28	0.066	HypoExponencial
SC2	TSR	0.26	0.002	HypoExponencial
SC2	PDT	0.56	0.113	HypoExponencial

After the definition of the most appropriate polynomial-exponential distribution to the time between sending requests (TSR) and processor demand times (PDT), it should be calculated the distribution parameters. As the hypoexponential distribution was chosen, the parameters μ_1 , μ_2 and γ were calculated for the different software configurations. Table 6 shows the values μ_1 , μ_2 and γ for the refined performance model of the cloud infrastructures.

Table 6. Probability Distribution Parameters

Software Configuration	Metrics	μ_1 (s)	μ_2 (s)	γ
SC1	TSR	0.000140	0.140	3
SC1	PDT	0.025000	0.250	17
SC2	TSR	0.000035	0.136	1
SC2	PDT	0.002000	0.020	24

The refined performance model was used to obtain the metrics response time, processor utilization and memory utilization considering the requests of 6 to 10 users to chat in the cloud infrastructure with different software configurations. Figures 12 and 13 present processor utilization and memory utilization measured and obtained of the refined performance model, considering the software configuration 1 and software configuration 2.

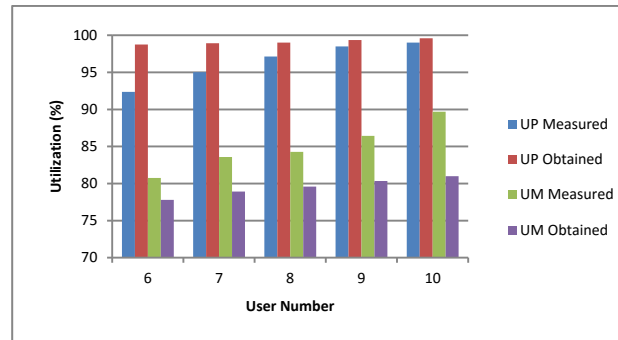


Figure 12. Resource Utilization Measured and Obtained in the Performance Model - Software Configuration 1

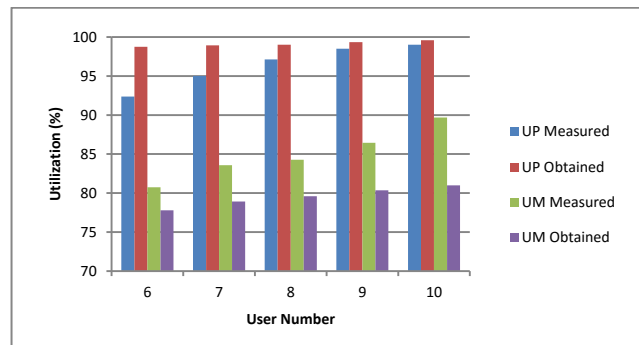


Figure 13. Resource Utilization Measured and Obtained in the Performance Model - Software Configuration 2

The t-paired test was applied to the metrics processor utilization and memory utilization measured and the metrics processor utilization and memory utilization obtained of the performance model. Considering a significance level of 5%, the t-paired test generated a confidence interval of (-4.414, 0.508) for processor utilization and a confidence interval of (-4.19, 11.96) for memory utilization. As the confidence intervals contain 0, there is no statistical evidence to reject

the hypothesis of equivalence between processor utilization and memory utilization measured and obtained of the performance model (Bakouch, 2011).

Figure 14 shows the response times measured and obtained from the refined performance model considering different software configurations. The t-paired test was applied to the metric response time measured and the metric response time obtained of the performance model. Considering a significance level of 5%, the t-paired test generated a confidence interval of (-108.0, 94.3). As the confidence interval contains 0, there is no statistical evidence to reject the hypothesis of equivalence between the response times measured and obtained of the performance model (Bakouch, 2011).

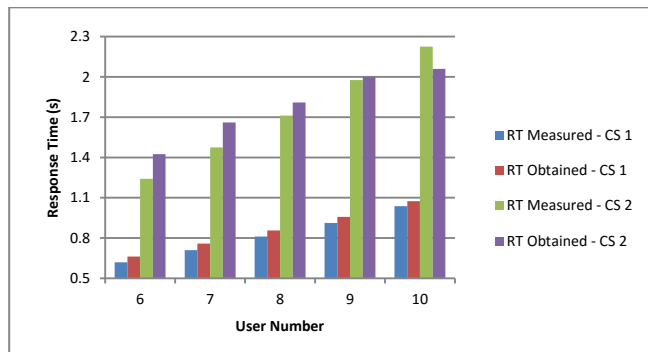


Figure 14. Response Time Measured and Obtained in the Performance Model - Software Configuration 1 and 2

The Equipment Cost Model and Software Cost Model were adopted to the representation and evaluation of the conceived cloud infrastructures. Table 7 presents the unit cost and total cost of the equipment acquisition and software licenses acquisition (Bestbuy, 2015). The unit cost of the Apache, Lighttpd, Moodle and Ubuntu is US\$ 0.00 (as they are open source). Thus, the total cost of the equipment acquisition and software licenses acquisition is US\$ 17,290.00.

Table 7. Cost Parameters

Component	Unit Cost (US\$)	Amount	Total Cost (US\$)
Cloud Platform	1,500.00	1	1,500.00
Physical Machine	500.00	13	6,500.00
Database	2,000.00	1	2,000.00
Router	3,291.00	1	3,291.00
Switch	3,999.00	1	3,999.00

The cloud infrastructures were generated by the optimization model. Table 8 shows the selected cloud infrastructures due to the results of the response time, processor utilization and memory utilization are adequate to the users requirements. These requirements are the response

time smaller than 1.50 seconds and the processor and memory utilization smaller than 95%. The first column of this table lists the chosen cloud infrastructures and the second column shows the hardware and software configurations. The other columns show the results of the response time(s) and resource utilization (%).

Table 8. Chosen Cloud Infrastructures

Solution	Configurations	Response Time (s)	Processor Utilization (%)	Memory Utilization (%)
1	SC1, 4 cores, 4 GB	1.05	92.72	77.52
2	SC1, 4 cores, 2 GB	1.23	94.86	93.27
3	SC1, 8 cores, 2 GB	0.85	82.57	91.46
4	SC1, 8 cores, 4 GB	0.72	80.52	76.91
5	SC1, 4 cores, 8 GB	0.93	90.37	70.00
6	SC1, 8 cores, 8 GB	0.62	78.56	66.60
7	SC2, 8 cores, 2 GB	1.45	78.15	81.30
8	SC2, 8 cores, 4 GB	1.36	76.66	71.60
9	SC2, 4 cores, 8 GB	1.68	84.71	61.51
10	SC2, 8 cores, 8 GB	1.12	65.61	59.31

The designer can adopt one of these 10 cloud infrastructures with the software configuration 1 and software configuration 2, since these infrastructures meet the performance and cost requirements. These 10 cloud infrastructures have the total cost of US\$ 17,290.00, but the solution 6 is the optimal, since it has the lowest response time, processor utilization and memory utilization.

The aim of this case study was reached because the proposed methodology, representation models and optimization model allows the generation, modeling and evaluation of cloud infrastructures with different hardware and software configurations. In addition, this study also evaluated the effect of various hardware and software configurations in the response time, resource utilization and cost of the cloud infrastructures.

8. CONCLUSIONS

This work proposed a methodology, representation models and an optimization model for cloud infrastructure planning. The optimization model provided cloud infrastructures, considering the suitable software and hardware sets. Next, the cloud infrastructures were evaluated using a SPN model. These evaluation provided

the response time, processor utilization and memory utilization of these infrastructures. The cloud infrastructures also were evaluated by two equations that estimate the software licenses and equipment cost. The metrics results were used by the optimization model to suggest cloud infrastructures according to performance and cost requirements.

This work also provided a script of workload generation to simulate Moodle users interacting in a chat and a measurement script to collect performance metrics.

A case study based on cloud infrastructures planning for virtual learning environment was presented in order to illustrate the feasibility of the proposed methodology, representation model and optimization model.

As future work, we intend to evaluate the performance of other cloud platforms, such as Nimbus, Open Nebula and Open Stack. We also intend to extend this study to evaluate the impact of failure events in the performance of cloud infrastructures.

9. REFERENCES

- Apache. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 8, 2016, from <http://www.apache.org/>.
- Bakouch, Hassan S. (2011). Probability, Markov chains, queues, and simulation, Volume 38, Number 8, Pages 1746-1746, Taylor & Francis, 2011.
- Bestbuy. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 8, 2016, from <http://www.bestbuy.com/>.
- Callou, G., Ferreira, J., Maciel, P., Tutsch, D. and Souza, R. (2014). An Integrated Modeling Approach to Evaluate and Optimize Data Center Sustainability, Dependability and Cost, Multidisciplinary Digital Publishing Institute, Energies, Volume 7, Number 1, Pages 238-277, 2014.
- Chee, B. J. S. and Franklin Jr, C. (2009). Cloud computing: technologies and strategies of the ubiquitous data center, CRC.
- Desrochers, A. A. and Al-Jaar, R. Y. (1995). Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis, IEEE Press.
- Eucalyptus. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 3, 2016, from <https://www.eucalyptus.com/eucalyptus-cloud/iaas>.
- Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures, Journal of global optimization, Springer, Volume 6, Number 2, Pages 109-133.
- Ferreira, J., Callou, G. and Maciel, P. (2013). A power load distribution algorithm to optimize data center electrical flow, Multidisciplinary Digital Publishing Institute, Energies, Volume 6, Number 7, Pages 3422-3443, 2013.
- Ghosh, R., Naik, V. K. and Trivedi, K. S. (2011). Power-performance trade-offs in IaaS cloud: A scalable analytic approach, Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops, IEEE Computer Society, Pages 152-157.
- Ghoshal, D., Canon, R. S. and Ramakrishnan, L. (2010). Understanding I/O Performance of Virtualized Cloud Environments, Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing, Miami, FL, IEEE Computer Society, Pages 51-58.
- GODARD, S. Sysstat. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 28, 2016, from <http://sebastien.godard.pagesperso-orange.fr/>.
- INA. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 30, 2016, from <http://www2.informatik.hu-berlin.de/~starke/ina.html>.
- Iosup, A., Ostermann, S., Yigitbasi, N., Prodan, R., Fahringer, T. and Epema, D. (2010). Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. IEEE Transactions on Parallel and Distributed Systems, Pages 1-16.
- Jain, Raj. (1991). The art of computer systems performance analysis, Volume 182, John Wiley & Sons Chichester.
- JMeter. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 15, 2016, from <http://jmeter.apache.org/>.
- Li, Xinhui, Li, Ying, Liu, Tiancheng, Qiu, Jie and Wang, Fengchun. (2009). The Method and Tool of Cost Analysis for Cloud Computing, Proceedings of the 2009 IEEE International Conference on Cloud Computing, IEEE Computer Society, Pages 93-100.
- Lighttpd. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 4, 2016, from <http://www.lighttpd.net/>.
- Marsan, M. A., Balbo, G., Conte, G., Donatelli, S. and Franceschinis, G. (1998). Modelling with Generalized Stochastic Petri Nets, ACM SIGMETRICS Performance Evaluation Review, ACM Press New York, NY, USA, volume 26, number 2.
- Martens, Benedikt, Walterbusch, Marc and Teuteberg, Frank. (2012). Costing of Cloud Computing Services: A Total Cost of Ownership Approach, Proceedings of the 2012 45th Hawaii International Conference on System Sciences, IEEE Computer Society, Pages 1563-1572.
- Menasce, D. A., Almeida, V. A. F., Dowdy, L. W. and Dowdy, L. (2004). Performance by design: computer capacity planning by example, Prentice Hall.
- Moodle. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 13, 2016, from <http://www.moodle.org.br/>.
- Murata, T. (1989). Petri Nets: Properties, Analysis and Applications, Proc. IEEE, number 4, volume 77, pages 541-580.
- MySQL. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 28, 2016, from <http://www.mysql.com/>.
- NIST. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 18, 2016, from <http://www.nist.gov/itl/cloud/>.
- Ostermann, S., Iosup, A., Yigitbasi, N., Prodan R., Fahringer, T. and Epema, D. (2010). A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing, Pages 115-131, Institute for Computer Science, Social-Informatics and Telecommunications Engineering 2010.
- Shafer, J. (2010). I/O virtualization bottlenecks in cloud computing today, Proceedings of the 2nd conference on I/O virtualization, USENIX Association.
- Sousa, E. Lins, F. Tavares, E. and Maciel P., 2014 IEEE 7th International Conference on Cloud Computing, IEEE, Pages 546 - 553, 2014.
- Trivedi, K.. (2002). Probability and Statistics with Reliability, Queueing and Computer Science Applications, Wiley Interscience Publication, Edition 2.

Trivedi, Kishor S. (2008). Probability & statistics with reliability, queuing and computer science applications, John Wiley & Sons.

Ubuntu. (2016). Electronic source for references without author nor publication time. (n.d.). Retrieved April 18, 2016, from <http://www.ubuntu.com/>.

Velte, A. T., Velte, T. J., Elsenpeter, R. C. and Babcock, C. (2010). Cloud computing: a practical approach, McGraw-Hill.

Xiong, K. and Perros, H. (2009). Service performance and analysis in cloud computing, Services-I, 2009 World Conference on IEEE, Pages 693-700.

from Federal University of Pernambuco, respectively. He was a faculty member with the Department of Electrical Engineering of Pernambuco University from 1989 to 2003. Since 2001 he has been a member of the Informatics Center at Federal University of Pernambuco, where he is currently Associate Professor.

Authors



Erica Sousa is graduate at Electronic Engineering from Pernambuco University (UPE/ Brazil) and received her M.Sc and Ph.D. degrees in the Federal University of Pernambuco (UFPE/Brazil). She has experience in computer science, and is involved in cloud computing, virtual learning environment, electronic funds transfer systems, stochastic models, performance model, dependability model, and performability analysis. Currently, she is assistant professor at Federal Rural University of Pernambuco.



Fernando Lins is graduated in Computer Engineering at Pernambuco University and received his MSc and PhD. degrees in the Federal University of Pernambuco. Currently, he is Associate Professor at Federal Rural University of Pernambuco. He has several interests in Computer Science, focusing on the following subjects: cloud computing, business process, web service composition, web services, security performance evaluation.



Eduardo Tavares graduated in Computer Science in 2002 and he received his MSc, as well as PhD. degrees in Computer Science from Federal University of Pernambuco (UFPE) in 2006 and 2010, respectively. Since 2011, he has been a member of Center for Informatics - UFPE, in which he is currently Associate Professor. His research interests include performance and dependability evaluation, Petri Nets, formal models and real-time systems.



Paulo Maciel graduated in Electronic Engineering in 1987 and received his MSc and PhD. degrees in Electronic Engineering and Computer Science