C_2LP: Modelling Load Propagation and Evaluation through the Cloud Components

Rafael de Souza Mendes*, Rafael Brundo Uriarte†, Carlos Becker Westphall*
*Federal University of Santa Catarina, Florianópolis, Brazil
emails: rafael.mendes@posgrad.ufsc.br, westphal@inf.ufsc.br
†IMT Institute for Advanced Studies Lucca, Italy
email: rafael.uriarte@imtlucca.it

***Abstract*—Due to the scale and dynamism of cloud computing, there is a need for new tools and techniques for its management. This paper proposes an approach to model the load flow in cloud components using double weighted Directed Acyclic Multigraphs. Such model enables the comparison, analysis and simulation of clouds, which assist the cloud management with the evaluation of modifications in the cloud structure and configuration. The existing solutions either do not have mathematical background, which hinders the comparison and production of structural variations in cloud models, or have the mathematical background, but are limited to a specific area (e.g. energy-efficiency), which does not provide support to the dynamic nature of clouds and to the different needs of the managers. In contrast, our model has a formal mathematical background and is generic. To this aim, we present its formalisation and algorithms that support the load propagation and the states of *services*, *systems*, *third-parties* providers and resources, such as: *computing*, *storage* and *networking*. To demonstrate the applicability of our solution, we have implemented a software framework for modelling *Infrastructure as a Service*, and conducted numerical experiments with hypothetical loads.**

Keywords-Cloud Computing; Management; Simulation; Multigraph.

I. INTRODUCTION

The management of pooled resources according to high-level policies is a central requirement of the *as a service* model, as fostered by Cloud Computing (CC). Decision making in the context of clouds, where there exist many possible configurations and complex data flows, is challenging and is still an open issue in the field. In order to use the well-established approaches of *decision theory* [1] and *managerial science* [2] for the CC management, it is necessary to employ formal models to represent the *managed elements* and the flow of the service loads. Furthermore, such a model is also required for the evaluation of possible actions, and for the analysis of cloud components and their hierarchy, which are usually carried out by the management of a cloud *operation*. We highlight this lack of formal models based on our previous efforts to develop methods to CC autonomic management [3][4][5] and formalisms based on Service Level Agreement (SLA) [6].

Currently, the existing solutions which provide CC models can be classified into two main groups: general models, usually represented by the simulators; and specific models, devised for a particular field (e.g., energy saving). The former lack a mathematical formalisation that enables comparisons with variations on the modellings. The latter usually have the formal mathematical background but, since they are specific, they do not support reasoning on different management criteria and encompass only cloud elements related to the target area.

To address this gap in the literature, we analyse the domain elements and characteristics to propose the *Cloud Computing Load Propagation* (C_2LP) graph-based model, a formal schema to express the *load flow* through the cloud computing components. Considering that a *load* expresses the amount of work to process services in systems or resources, the modelling of the load flows enables cloud managers to perform several types of analysis about the cloud structure and behaviour. For example, it enables the comparison of different cloud structures, the distinction of load bottlenecks, the quantitative analysis of the load propagation and of the effects of processing a load in a cloud. In more general terms, such solution unifies heterogeneous abstraction levels of managed elements into a single model and can assist the decision-making tasks in processes, such as: *load balance*, *resource allocation*, *scale up/down* and *migrations*. Moreover, simulations performed using our model can be useful to predict the consequences of managerial decisions and external events, as well as the evolution of baseline behaviour.

More specifically, we model the basic components of CC, the *services*, *systems*, *third-party* clouds that implement services and the resources over which system are deployed: *computing*, *storage*, *networking*. Our model is based on Directed Acyclic Multigraphs. This formalism enables the manager to access consolidated analytical mathematical tools to, e.g., measure the components interdependencies, which is used to improve the availability and resource allocation. In order to demonstrate the applicability and advantages of the C_2LP model, we present a use case where our model is used to compare and evaluate different managerial configurations over several quantitative behaviour in load propagation and evaluation.

This article is organised as follows. Section II discusses the existing cloud models, the works that inspired the definition of this model and the background information necessary for the appreciation of this work. In Section III, we present an overview of the model, formalise it, the propagation algorithm, and the evaluation process. Section IV describes the implementation and the analysis performed on a use case. Finally, in Section V, we discuss the limitations and the future directions for the research.

II. RELATED WORKS

This section presents the related works that propose models to describe and simulate clouds. We have analysed them from a *cloud provider management perspective*, considering their

 ISBN: 978-1-61208-450-3

capacity to: *express general* cloud models, define *components* in distinct abstraction levels; *compare* structures; *simulate* behaviours and provide *formal* specifications with mathematical background. Table I summarises this comparison.

We grouped the proposals into two classes: *general* and *specific*. General models are usually associated with simulators and are used to evaluate numerous criteria at the same time. On the other hand, specific models are commonly associated with particular criterion evaluation, such as: performance [7], security [8][9], accounting [10][11] or energy [12].

All analysed works of the first group, i.e., CloudSim [13], GreenCloud [14], iCanCloud [15], EMUSIM [16] and MDCSim [17], are data-centre oriented, thus requiring extensions to enable the abstraction of important cloud elements, such as services and systems. One exception, the EMUSIM framework, allows the modelling of applications and, from theses models, the estimation of some performance measures but it also depends on data-centre elements, such as: *virtual machines* and *physical machines*. The limitation of the existing generic solutions to deal with abstract elements and their lack of mathematical background hinders the comparison and production of structural variations in cloud modellings, which, in turn, limits their capacity to test the performance of managerial methods.

On the other hand, the solutions in the second group, i.e., the ones specifically devised for an area, present in-depth analysis, based on robust formalisms, such as queue theory [12] [7], probability [8], fuzzy uncertainty [11] and heuristics [10]. However, these models do not fit well in integrated management methods that intend to find optimal configurations considering several criteria of distinct types. Moreover, specific optimisation models usually are sensible to structural changes, having no robustness to support the dynamic nature of the clouds.

The comparison between the related works is presented schematically in Table I, where: the column "Class" specifies if a work is general or specific; "Formalism" evaluates the mathematical background that supports the models; the column "Components" presents the capacity of a model to express cloud components in different abstraction levels; the ability to compare structures is depicted in the column "Comparison"; and, "Simulation" expresses the capacity to perform simulations using the models.

Considering the gap in the existing cloud modelling techniques, our proposal intents to model the load propagation and evaluation functions over a graph to obtain expressiveness, whilst keeping the mathematical background. We opt to model the "load flow" because it is one of the most important information for managerial decisions, such as: load balance, resource allocation, scale up/down and migrations.

TABLE I: COMPARISON BETWEEN RELATED MODELS. ■ REPRESENTS A FEATURE, □ A PARTIALLY COVERED ONE AND - WHEN THE FEATURE IS NOT SUPPORTED.

Model	Class	Formalism	Components	Comparison	Simulation
CloudSim [13]	General	-	■	-	■
GreenCloud [14]	General	-	■	-	■
iCanCloud [15]	General	-	■	-	■
EMUSIM [16]	General	-	■	-	■
MDCSim [17]	General	-	■	-	■
Chang[12]	Specific	■	□	■	□
Püschel [11]	Specific	■	□	■	□
Nesmachnow [10]	Specific	■	□	■	□
Silva. [8]	Specific	■	□	□	□
Vilaplana [7]	Specific	■	□	■	□
C_2LP	General	■	■	■	□

III. MODELLING LOAD FLOW IN CLOUDS

This section discusses the main components of cloud structures and proposes a formal model based on a directed acyclic multigraph, to represent the load flow in clouds.

Subsection III-A presents the structural model and its main components. In Subsection III-B, we formally define the data structures to represent *loads*, *configurations*, *states* and *functions*. Finally, Subsection III-C discusses the computational details of the propagation of the loads and the evaluation of the states for each cloud component.

A. Modelling Clouds with C_2LP

In C_2LP, the structural arrangement of cloud elements is based in a *directed acyclic multigraph* (DAM). The nodes of the graph represent components of the model and can be of four types.

- *Computing*, *Storage* and *Networking* resources, which are the base of any cloud service. These components are always leaf nodes.
- *Systems*, which are abstractions that orchestrate resources that provide and implement services. They can be, e.g., applications and platforms. In the model, systems must be directly linked to at least one of each type of resource: computing, storage and networking. Nevertheless, these resources might be replaced by other systems or third-party services. In such cases, the relationship between the system and the element that represents the resource (i.e., another system or the third-party service) must be explicitly defined using stereotypes (virtual computing, virtual networking or virtual storage).
- *Third-Party Services*, which might represent: (i) resources to system components, when the relation is explicitly annotated with the appropriated stereotype, and (ii) entire systems which provide services and abstract the underlying layers (e.g., email services). The latter models, for example, hybrid clouds or composed services.
- *Services*, which are interfaces between the cloud and the consumers. They are invoked with specification of the consumer's needs and, in this model, they are converted into loads.

Directed edges in our model define to which elements each cloud component can transmit load. Nodes have two main processes: *propagation* of the load; and *evaluation* of the impact of the load in the node itself. Remarkably, the resources components do not propagate load and are the only nodes that actually run the assigned load, while other elements are abstract (e.g., applications, middlewares, platforms and operations systems). Moreover, we consider in the model also the configurations settings of nodes, which impact the propagation and evaluation processes.

The cloud offers services and receives requests from consumers. Requests are then propagated to other nodes using a propagation function that defines to which linked node, the division and in which form the load will be propagated. Since loads might have different forms, we model these relations enabling multiple edges between nodes, which simplifies the understanding of the model. For example, a service transmits

 ISBN: 978-1-61208-450-3

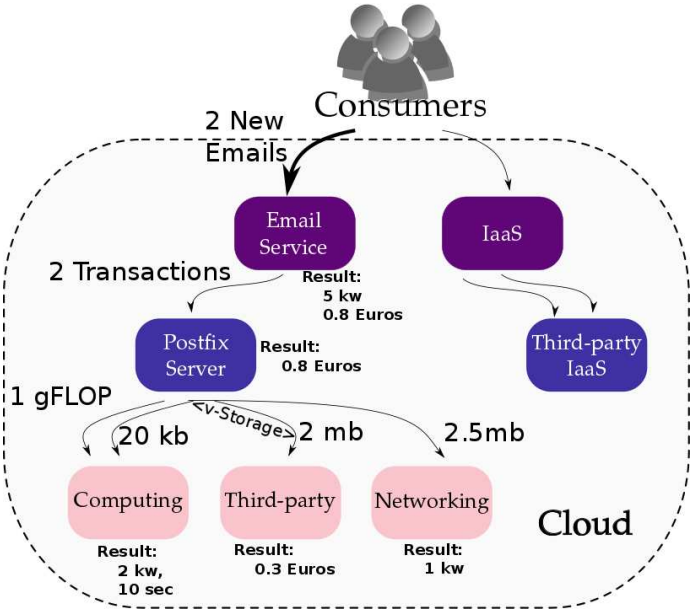

Figure 1: Example of the propagation of loads and the evaluation processes using the C_2LP model.

10 giga FLoating-point Operations Per Second (FLOPS) and 100 giga bytes of data to third-party service. This relation is modelled using two edges, one for each type of load. In case of change in the structure (e.g., a cheaper storage provider) the model can be adjusted simply by removing the storage edge between these nodes and adding it to this new third-party provider.

The evaluation process measures the impact of the load in the resources components and, consequently, on the other nodes of the structure. This process is performed in each node using an evaluation function, which receives three parameters: the configuration of the node, the load defined in the incoming edges and the evaluation of successor nodes. In the case of resource components, the evaluation function uses only the configuration and edge's loads as input.

Figure 1 presents the modelling of a scenario, in which a cloud provides two services: an email and Infrastructure-as-a-Service (IaaS). The IaaS is provided by a third-party cloud. The email service instead, employs a system component to represent a software email server (in this case a Postfix). This component uses local computing and networking and storage from a third-party cloud. The relation (edge) between these components is annotated accordingly.

In the proposed scenario, we exemplify the load propagation with a request from consumers to send 2 new emails using the email service. These 2 emails are converted by the service component into 2 loads of type "transaction" and sent for the email server where, consequently, are converted into another types of load and propagated to the resources linked to the server.

The evaluation process of this scenario uses different metrics in each node and is marked as "result:". For example, in the service level, the load of 2 emails was measured in terms financial cost and energy necessary to complete the request.

B. Formalisation of the Model

Formally, in C_2LP model, a cloud C can be expressed as $C = (V, E, \tau^V, \sigma, \Phi, \phi, \Gamma, \gamma, \Gamma', \gamma')$, where:

- V is the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ of the multigraph, such that every item in V represents one element of the cloud and has one respective node-weight w_v, that usually is a vector of values;
- E is the set of directed edges where $E = \{e_1, e_2, \ldots, e_m\} | e = (v, v')$, that describes the ability of a source node v to transmit a load to node v', such that each e_m also has a respective edge-weight $w_{v,v'}$;
- $\tau^V : V \rightarrow T^V$ is a bijective function which maps the nodes with the respective type, where the set T^V is the set of types of nodes, such that $T^V = \{'computing', 'storage', 'networking', 'system', 'service', 'third\_party'\}$;
- $\sigma : E_{\{\rightarrow system, \rightarrow third-party\}} \rightarrow \{none, vComputing, vStorage, vNetworking\}$ is a function which maps the edges that have systems and third-party services as target with the respective stereotype, characterising the relation between the source element with the target;
- Φ represents the set of propagation functions, where $\Phi = \{f_1, f_2, \ldots, f_v\}$ and ϕ is a bijective function $\phi : V \rightarrow \Phi$ that maps each node for the respective propagation function. Each function in the set Φ is defined as $f_v : \mathbb{N}^n, \mathbb{R}^i \rightarrow \mathbb{R}^o$, where: the set $\mathbb{N}^n$ represents the space where the n-tuple for the configuration is contained; the set $\mathbb{R}^i$ represents the space where the n-tuple of incoming edge-weights is contained; and, $\mathbb{R}^o$ is the space where the n-tuple of the outgoing edge-weights is contained. To simplify the model and the algorithms, we consider that configurations are stored in the node-weight, such that w_v^{conf} represents the configuration part of the node-weight vector.
- Γ is the set of sets that contains the evaluation functions for the leaf nodes, such that there exists one function for each distinct evaluation metric (e.g., energy use, CO2 emission, ...). Then, $\Gamma = \{\Gamma_1, \Gamma_2, \ldots \Gamma_k\}$, such that $\Gamma_k = \{g_{n+1}, g_{n+2}, \ldots, g_m\}$. Each set Γ_k is related to a leaf node $v \in V_{[leaf]}$ through the bijective function $\gamma : V_{[leaf]} \rightarrow \Gamma$. Every g_{n+m} is stored in a distinct position of the node-weight vector of the respective node – representing *a partial state* of v – such that the full new state can be computed through the expression: $w'_v = (c_1, \ldots, c_n, g_{n+1}(c_1, \ldots, c_n, w_v^i), g_{n+2}(c_1, \ldots, c_n, w_v^i), \ldots, g_{n+m}(c_1, \ldots, c_n, w_v^i))$, where: $c_1, \ldots, c_n$ is the n-tuple with the configuration part of the node-weight w_v; w_v^i is the n-tuple with all incoming edge-weights $w_{*,v}$ of v; and w'_v is the new node-weight (full state) for v. The complete evaluation procedure is detailed in Figure 3;
- Γ' is the set of sets that holds the evaluation functions for non-leaf nodes. Therefore, $\Gamma' = \{\Gamma'_1, \Gamma'_2, \ldots, \Gamma'_l\}$, such that each set $\Gamma'_l = \{g'_{n+1}, g'_{n+2}, \ldots, g'_m\}$ contains the evaluation functions g'_{n+m}. Every Γ'_l is associated with a non-leaf node v through the bijective function $\gamma' : V_{non-leaf} \rightarrow \Gamma'$. Since the result of each function g'_{n+m} is stored in a distinct position of w'_v, it represents a partial state of the respective node v. A new full state of non-leaf nodes can be computed through the expression: $w'_v = (c_1, \ldots, c_n,$

1: **procedure** BREADHTFIRSTPROPAGATION(C,W^V,W^E) ▷ Requires a cloud model $C = (V, E, \tau^V, \sigma, \Phi, \phi)$, the set of node-weights $W^V | \forall v \in V \wedge \exists! w_v \in W^V$ and the set of edge-weights $W^E | \forall e_{v,v'} \in E \wedge \exists! w_{v,v'} \in W^E$.
2: $queue \leftarrow \varnothing$
3: $enqueue(*)$
4: **repeat**
5: $v \leftarrow dequeue()$
6: **for each** $u \in successorSet(v)$ **do**
7: $enqueue(u)$
8: **end for** ▷ enqueues the sucessor of each node
9: $f_v \leftarrow \phi(v)$
10: $w_v^{conf} \leftarrow configurationPart(w_v)$ ▷ gets the config. part of the node-weight (state).
11: $w_v^i \leftarrow (w_{1,v}, w_{2,v}, \ldots, w_{u,v})$ ▷ builds the incoming edge-weights in a tuple w_v^i.
12: $w_v^o \leftarrow f_v(w_v^{conf}, w_v^i)$ ▷ w_v^o contains the result of the propagation function.
13: **for each** $w_{v,u} \in w_v^o$ **do**
14: $W^E \leftarrow W^E \oplus w_{v,u}$ ▷ replaces the old value of $w_{v,u}$.
15: **end for**▷ assign the values for the outgoing edges of v.
16: **until** $queue \neq \varnothing$
return W^E
17: **end procedure**

Figure 2: Breadth-first algorithm used for the load propagation.

$g'_{n+1}(c_1, \ldots, c_n, w_v^i, w'_{u_v})$, $g'_{n+2}(c_1, \ldots, c_n, w_v^i, w'_{u_v})$, $\ldots, g'_{n+m}(c_1, \ldots, c_n, w_v^i, w'_{u_v})$); where w'_v is the new node-weight of v, $c_1, \ldots, c_n$ is the n-tuple with the configuration part w_v^{conf} of the node-weight, w_v^i is the n-tuple with the incoming edge-weights $e_{*,v}$ of v, and w'_{u_v} is a tuple which puts together all node-weights of the successors of v (see Figure 3 for details).

The main objective of these formalisms is to specify the data structures that support a model validation, the load propagation, and elements evaluations. The details of each procedure concerned with propagation and evaluations are described in Subsection III-C.

C. Details on the Propagation and Evaluation

The load propagation consists in a top-down process that uses the *breadth-first* approach. In a breadth-first algorithm, the loads are propagated to the neighbour nodes before moving to the next level nodes. In the specific case on C_2LP the algorithm starts from the loads on the services, corresponding to the requests received from consumers.

The propagation process uses a queue with the service nodes (the roots of the graph). Then, a node v is picked from this queue and all its children are placed into the queue. Afterwards, a function $f_v = \phi(v)$ is executed to distribute the load, that is, to define all edge-weights for the outgoing edges of v. This procedure is repeated while the queue is not empty. The well defined method is detailed in Figure 2.

When the load is propagated to resources components (leaf nodes), they execute the load. This execution requires power and resources and can be evaluated in several forms. For example, *energy (kw)*, *performance*, *availability*, *accounting*, *security*, *CO_2 emissions* and other cloud specific feature units. This evaluation process takes every function $g_{n+m} \in \Gamma_k$ in order and computes each partial states, storing them into a position of the new node-weight w'_v. A finer description can be defined as: $w'_v = (w_v^{conf}, g_{n+1}(w_v^{conf}, w_v^i), \ldots, g_{n+m}(w_v^{conf}, w_v^i)$, such that w'_v represents the *a posteriori* state for the node v, w_v^{conf} are the configurations (*a priori* state) of v, w_v^i are the incoming edge-weights of v, and $g_{n+m} \in \gamma(v)$ are the evaluation functions associated with the node.

The evaluations also include the non-leaf nodes since the load also passes through them and it is useful, e.g., to understand the load distribution and to identify bottlenecks. In the case of non-leaf nodes, the evaluation requires also the evaluation results of the bottom nodes. Therefore, this process is performed from the leaves to the roots using a *depth-first* approach.

Resources evaluation occurs according to several models that convert the configuration, loads and the results of other evaluations into node-weights.

A non-leaf node receives the tuples $(config, loads, children\_states)$, and evaluates by the processing of all $g'_{n+m} \in \gamma'(v)$ functions. A representation of this process can be described as: $w'_v = (w_v^{conf}, g'_{n+1}(w_v^{conf}, w_v^i, w'_{u_v}), \ldots, g'_{n+m}(w_v^{conf}, w_v^i, w'_{u_v})$, such that w'_v represents the new node-weight (*a posteriori* state) for the node v, w_v^{conf} are the configuration part (*a priori* state) of node-weight into v, w_v^i represent the incoming edge-weights of v, w'_{u_v} are the computed node-weights of the successors of v, and $g'_{n+m} \in \gamma'(v)$ are the evaluation functions associated with the node.

The complete evaluation process is detailed in Figure 3, where a stack is used to perform a depth-first computation. The first non-visited child of a current node is placed into the stack and will be used as current node. When all children of a node are evaluated, then the node is evaluated. If the node is a leaf node the g functions are used to compute the evaluations, otherwise, the g' functions are used instead.

IV. EXPERIMENTS AND RESULTS

This section presents numerical experiments with the C_2LP model, based on a *service* modelling. These experiments serve to: *test* the applicability of the model; present a concrete *example* of modelling; and, *demonstrate* the model capacity for *quantitative behaviours generation* combining variations of propagation and evaluation functions.

To perform these experiments, we have implemented a use case using our model. This use case exemplifies the model's usage and serves to test its feasibility. The example of model's usage was made using hypothetical functions, since its objective is to prove the generation of simulations, the propagation and the evaluation. Nevertheless, our model can be used for modelling real-world clouds, provided that the propagation and evaluation functions are adjusted to the cloud instance.

As use case, we defined a *IaaS service* where consumers perform five operation: *deploy VM*, *undeploy VM*, *start VM*, *stop VM*, and *execute* tasks. To meet the demand for these services, we designed a hypothetical cloud infrastructure with which is possible to generate quantitative scenarios of propagation and evaluation – in a combinatorial fashion. Using this hypothetical infrastructure, we have tested some managerial configurations related to load distribution over the cloud elements,

 ISBN: 978-1-61208-450-3

1: **procedure** DEPTHFIRSTEVALUATION(C,W^V,W^E) ▷ The same input described in Figure 2.
2: $\beta \leftarrow \varnothing$ ▷ initializes the set of visited nodes.
3: $stack \leftarrow \varnothing$ ▷ initializes the stack.
4: $push(*)$ ▷ starts from the hypothetical node.
5: **while** $stack \neq \varnothing$ **do**
6: $v \leftarrow peek()$ ▷ gets a node without to remove it.
7: **for each** $u \in successorSet(v)$ **do**
8: **if** $u \notin \beta$ **then**
9: $push(u)$
10: **continue while**
11: **end if**
12: **end for** ▷ if the **for** loop ends, all successors have been evaluated.
13: $w^{conf} \leftarrow configurationPart(w_v)$ ▷ gets the config. part for v.
14: $w_v^i \leftarrow (w_1, w_2, \ldots, w_{u,v})$ ▷ builds the n-tuple with the incomings of v.
15: **if** $isLeaf(v)$ **then**
16: $w'_v \leftarrow (w_v^{conf}, g_{n+1}(w_v^{conf}, w_v^i), \ldots, g_{n+m}(w_v^{conf}, w_v^i)), \forall g_{n+m} \in \gamma(v)$ ▷ computes the partial states and builds the new node-weight.
17: **else**
18: $w'_{u_v} \leftarrow (w'_{u_1}, w'_{u_2}, \ldots, w'_{u_o})$ ▷ builds the computed node-weights for all $u | \exists e_{v,u} \in E$.
19: $w'_v \leftarrow (w_v^{conf}, g'_{n+1}(w_v^{conf}, w_v^i, w'_{u_v}), \ldots, g'_{n+m}(w_v^{conf}, w_v^i, w'_{u_v})), \forall g'_{n+m} \in \gamma'(v)$ ▷ computes the partial states and builds the new node-weight.
20: **end if**
21: $W^V \leftarrow W^V \oplus w'_v$ ▷ replaces the old state of v into the node-weights.
22: **if** $v \notin \beta$ **then**
23: $\beta \leftarrow \beta \cup v$
24: **end if** ▷ puts v in the visited set if it is not there.
25: $v \leftarrow pop()$ ▷ gets and removes v from the stack.
26: **end while**
return W^V
27: **end procedure**

Figure 3: Depth-first algorithm to evaluate in specific metrics the impact of the load in each node.

in order to evaluate the average utility for all quantitative scenarios. At the end, the configurations which achieve the best average utility for all quantitative scenarios were highlighted, depicting the ability of the model to simulate configuration consequences for the purpose of selecting configurations.

A. Use Case Modelling

To deal with the consumers' loads (deploy, undeploy, start, stop and execute), the infrastructure manages: the *service interface*; systems, such as *load balancers*, *cloud managers* and *cloud platforms*; and resources, such as *servers*, *storages* and *physical networks*. All operations invoked by consumers represent an incoming load on the service interface, which is propagated to resources. In the resources the loads are evaluated to provide measures about *performance*, *availability*, *accounting*, *security* and *CO_2 emissions*. These evaluations are then executed also for systems and, at the end, for the service interfaces.

The modelling of the use case was devised considering 21 components: 1 service, 9 systems, and 11 resources. The services represent the interface with customers. In this use case, the systems are: a *load balancer*; two *cloud manager* systems; and six *cloud platforms*. Also, between the resources there are: 8 physical computing servers (6 work servers and 2 managerial), 2 storages (1 work storage and 1 managerial), and 1 physical network. A detailed list of components is presented in Appendix I.

Regarding the edges and loads, each consumer's operation is modelled as an incoming edge in a *service interface node* – with the respective loads in the edge-weights. The service node forwards the loads for a *load balancer* system, where the propagation function decides to which *cloud manager* the load will be sent, whereas the *manager servers*, the *manager storage* and the *physical network* receive the loads by its operation. In the cloud mangers, the propagation function must decide to which *cloud platform* the loads will be sent and, at the same time, generate loads for the managerial resources. The cloud platform system effectively converts its loads into simple resource loads when uses the *work server*, *work storage* and *physical network*. The complete relation of load propagation paths is presented in Appendix I, where an element at the left side of an arrow can propagate loads for an element at the right. Furthermore, a graphical representation of these tables, which depicts the graph as a whole, is also presented in Appendix I.

Besides the node and the edges, the use case model required the definition of: • 4 types of propagation functions – one for the service and tree for each type of system; • 6 types of leaf evaluation functions – two specific performance evaluations, one for computing resources and another for storage and networking; plus, four common evaluation functions (availability, accounting, security and CO_2 emissions) for each type of resource; • 5 types of non-leaf evaluations functions.

We have modelled the possible combinations to distribute the loads $\{1\text{-deployVM}, 2\text{-undeployVM}, 3\text{-startVM}, 4\text{-stopVM}, 5\text{-compute}\}$ as a partition set problem [18], resulting in 52 distinct possibilities of load propagation. Also, we introduced 2 possible configurations into each evaluation function for leaf nodes. These configurations are related to the choice of constants into the function. For example, the performance of a computing resource depends on its capacity, that can be: $a = 50GFLOPs$ or $b = 70GFLOPs$. Considering 5 distinct evaluation functions over 11 leaf nodes, we have got $(2^5)^{11} = 2^{55}$ possible distinct configurations to test.

B. Evaluations

The numerical experiments were performed running the propagation procedure, followed by the evaluation of every simulation. For each possible propagation, we tested and summarized the 2^{55} configurations for evaluation functions. Then, we analysed the average time (p, in seconds), average availability (av, in %), average accounting (ac, in currency units), average security (s, in % of risk of data exposition), and average of CO_2 emissions (c, in grammes). Each value was normalised according to the average for all propagations, tested and summarised in a global utility function, described in (1) – where the overlined variables represent the normalised values.

Such results enable cloud managers to choose the best

 ISBN: 978-1-61208-450-3

TABLE II: SUMMARY OF AVERAGE EVALUATIONS FOR EACH CONFIGURATION.

Criteria	Configuration			
Code	11221	11231	11232	**11212**
Time	180.59976	180.5999	180.60004	**180.59991**
Availability	0.9979606	0.99795955	0.9979587	**0.99795926**
Accounting	78.69924	78.69926	78.699234	**78.699265**
Security	0.9979606	0.99795955	0.9979587	**0.99795926**
Emissions	82848.31	82848.14	82848.51	**82848.74**
Utility	1.0526400204	1.0526410547	1.0526477776	**1.0526491889**

scenario according to the priorities of the policy or to provide input for the decision-making process, such as Markov Chains.

$$u = -(\overline{av} + \overline{s} - (\overline{p} + \overline{ac} + \overline{c})) \quad (1)$$

The best four results of the fifty two numerical experiments are presented in Table II in ascending order, where the configuration that achieves the best *average utility* is highlighted in bold. The *code* line in the table represents the propagation configuration, whereas the other lines contain the values obtained for each distinct evaluation type. The last row presents the average utility defined in Equation 1. To represent configuration we have adopted a set partition notation to express the propagation paths, such that each position in the code represents a type of load: 1-*deploy*, 2-*undeploy*, 3-*start*, 4-*stop*, and 5-*compute*. Considering that at leaves of the propagation graph there are 6 cloud platforms, a code 11212 indicates that the loads of type 1,2 and 4 were allocated on cloud platform 1, whereas the loads 3 and 5 ware allocated in the cloud platform 2.

V. CONCLUSION

Several solutions have been proposed to model clouds. However, to the best of our knowledge, none is general and has mathematical formalism at the same time, which are essential characteristics for the consolidation of analytical methods.

In this study, we have presented an approach with these characteristics to model clouds based in *Directed Acyclic Multigraph*, which has the flexibility of general models and the formalism of the specifics. Therefore, C_2LP is a flexible well-formed modelling tool to express flow of loads through the cloud components. This model supports the specification of elements in distinct abstraction levels, the generation of combinatorial variations in a use case modelling and the evaluation of the consequences of different configuration in the load propagation.

We developed a simulation software tool for the modelling of IaaS services and demonstrated the applicability of our approach through a use case. In this use case, we simulated several graph network theoretic analysis, evaluated and compared different configurations and, as a result, supplied the cloud managers with a numeric comparison of cost and benefits of each configuration. These experiments, demonstrated that this tools provides an essential support for the management of cloud.

ACKNOWLEDGEMENT

The present work was done with the support of CNPq agency, through the program Ciência sem Fronteiras (CsF), and the company Eletrosul Centrais Elétricas S.A. – in Brazil. The authors also would like to thank professor Rocco De Nicola and the group of SysMA research unit in Institute for advanced studies Lucca (IMT).

REFERENCES

[1] Itzhak Gilboa, "Theory of Decision under Uncertainty," Cambridge University Press, 2009.

[2] Cliff Ragsdale, "Modeling & Decision Analysis," Thomson, 2008.

[3] Rafael Mendes et al., "Decision-theoretic planning for cloud computing," In ICN 2014, The Thirteenth International Conference on Networks, Iaria, vol. 7, no. 3 & 4, 2014, pages 191–197.

[4] Alexandre A. Flores, Rafael de S. Mendes, Gabriel B. Bräscher, Carlos B. Westphall, and Maria E. Villareal, "Decision-theoretic model to support autonomic cloud computing," In ICN 2015, The Fourteenth International Conference on Networks, Iaria, vol. 8, no. 1 & 2, 2015, pages 218–223.

[5] Rafael Brundo Uriarte, "Supporting Autonomic Management of Clouds: Service-Level-Agreement, Cloud Monitoring and Similarity Learning," PhD thesis, IMT Lucca, 2015.

[6] Rafael Brundo Uriarte, Francesco Tiezzi, and Rocco De Nicola, "SLAC: A formal service-level-agreement language for cloud computing," In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE Computer Society, 2014, pages 419–426.

[7] Jordi Vilaplana, Francesc Solsona, and Ivan Teixidó, "A performance model for scalable cloud computing," In 13th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2015), ACS, vol. 163, 2015, pages 51–60.

[8] Paulo F Silva, Carlos B Westphall, and Carla M Westphall, "Model for cloud computing risk analysis," ICN 2015, Iaria, vol. 8, no. 1 & 2, 2015, page 152.

[9] Nada Ahmed and Ajith Abraham, "Modeling cloud computing risk assessment using machine learning," In Afro-European Conference for Industrial Advancement, Springer, 2015, pages 315–325.

[10] Sergio Nesmachnow, Santiago Iturriaga, and Bernabe Dorronsoro, "Efficient heuristics for profit optimization of virtual cloud brokers," Computational Intelligence Magazine, IEEE, vol. 10, no. 1, 2015, pages 33–43.

[11] Tim Püschel, Guido Schryen, Diana Hristova, and Dirk Neumann, "Revenue management for cloud computing providers: Decision models for service admission control under non-probabilistic uncertainty," European Journal of Operational Research, Elsevier, vol. 244, no. 2, 2015, pages 637–647.

[12] Chunling Cheng, Jun Li, and Ying Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," Tsinghua Science and Technology, IEEE, vol. 20, no. 1, 2015, pages 28–39.

[13] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Csar A. F. De Rose, and Rajkumar Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, Wiley Online Library, vol. 41, no. 1, 2011, pages 23–50.

[14] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," The Journal of Supercomputing, Springer, 2012, page 1263–1283.

[15] Alberto Núñez et al., "iCanCloud: A flexible and scalable cloud infrastructure simulator," Journal of Grid Computing, Springer, vol. 10, no. 1, 2012, pages 185–209.

[16] Rodrigo N Calheiros, Marco AS Netto, César AF De Rose, and Rajkumar Buyya, "EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," Software: Practice and Experience, Wiley Online Library, vol. 43, no. 5, 2013, pages 595–612.

[17] Seung-Hwan Lim, Bikash Sharma, Gunwoo Nam, Eun Kyoung Kim, and Chita R Das, "MDCSim: A multi-tier data center simulation, platform," In Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on, IEEE, 2009, pages 1–9.

[18] Toufik Mansour, "Combinatorics of Set Partitions," CRC Press, 2012.

 ISBN: 978-1-61208-450-3

APPENDIX I: IMPLEMENTATION DETAILS

TABLE III: THE CLOUD ELEMENTS – NODES OF THE GRAPH.

CS - computing service	**CP21** - platform 21	**WS12** - work server 12
LB - load balancer	**CP22** - platform 22	**WS13** - work server 13
CM1 - cloud manager 1	**CP23** - platform 23	**WS21** - work server 21
CM2 - cloud manager 2	**MS1** - manager server 1	**WS22** - work server 22
CP11 - platform 11	**MS2** - manager server 2	**WS23** - work server 23
CP12 - platform 12	**MSTO** - manager storage	**WSTO** - work storage
CP13 - platform 13	**WS11** - work server 11	**PN** - physical network

TABLE IV: THE LOAD PROPAGATION RELATIONS – EDGES OF THE GRAPH.

$\xrightarrow{5}$CS	CM1$\xrightarrow{5}$CP11	CP11 $\rightarrow$ WS11	CP21 $\rightarrow$ PN
CS$\xrightarrow{5}$LB	CM1$\xrightarrow{5}$CP12	CP11$\rightarrow$PN	CP21$\rightarrow$WSTO
LB$\xrightarrow{5}$CM1	CM1$\xrightarrow{5}$CP13	CP11$\rightarrow$WSTO	CP22$\rightarrow$W22
LB$\xrightarrow{5}$CM2	CM1 $\rightarrow$ PN	CP12$\rightarrow$WS12	CP22 $\rightarrow$ PN
LB $\rightarrow$ MS1	CM2 $\rightarrow$ MS2	CP12 $\rightarrow$ PN	CP22 $\rightarrow$ WSTO
LB $\rightarrow$ MS2	CM2 $\rightarrow$ MSTO	CP12 $\rightarrow$ WSTO	CP23 $\rightarrow$ W23
LB $\rightarrow$ WSTO	CM2 $\xrightarrow{5}$ CP21	CP13$\rightarrow$ W13	CP23 $\rightarrow$ PN
LB $\rightarrow$ PN	CM2 $\xrightarrow{5}$ CP22	CP13$\rightarrow$ PN	CP23 $\rightarrow$ WSTO
CM1 $\rightarrow$ MS1	CM2 $\xrightarrow{5}$ CP23	CP13 $\rightarrow$ WSTO	
CM1 $\rightarrow$ MSTO	CM2 $\rightarrow$ PN	CP21 $\rightarrow$ W21	

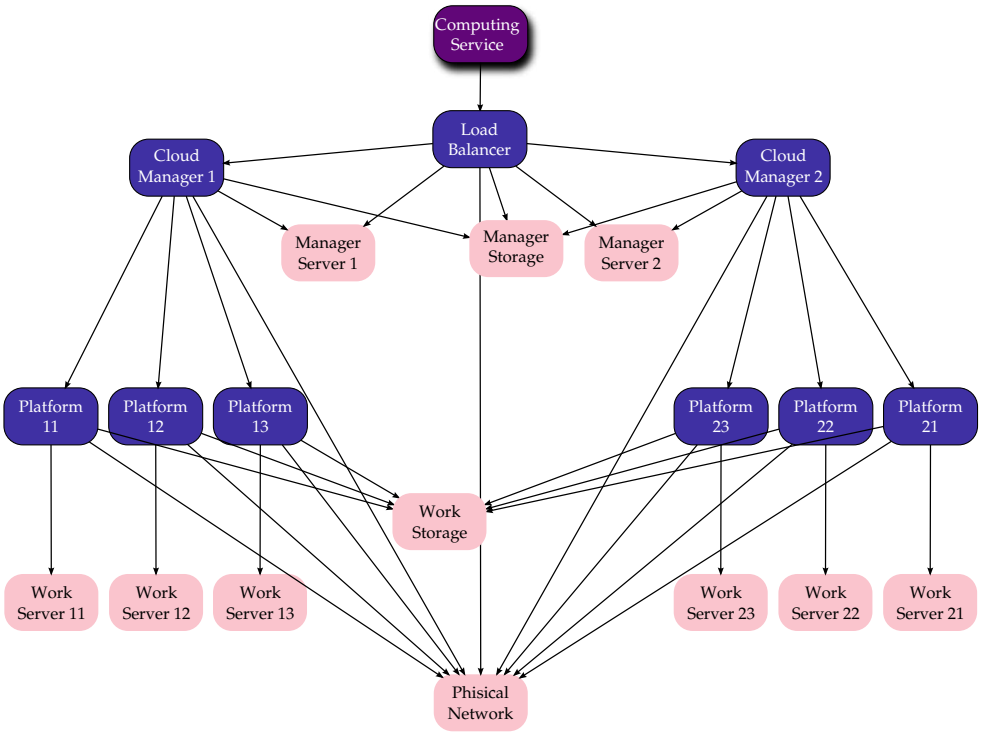

Figure 4: Graphical representation of structural arrangement for the modelling use case.

TABLE V: PROPAGATION FUNCTIONS.

Types	Declarations	Definitions
service	$(w_1, \cdots, w_5) \xmapsto{f^{CS}} (w'_1, \cdots, w'_5)$.	w_n is the weight for $\xrightarrow{n} CS$). w'_n is the weight for $(CS \xrightarrow{n} LB)$. $w'_n = w_n \vert \forall w'_n \in f^{CS}$.
balancer	$(c_1, \ldots, c_5, w_1, \cdots, w_5) \xmapsto{f^{LB}}$. $(w'_1, \ldots, w'_{14})$	$c_n \in \{CM1, CM2\}$, are the configurations which represent the targets of each load $w_n \vert 1 \leqslant n \leqslant 5$. $$w'_n = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CM1 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert\, 1 \leqslant n \leqslant 5$$. $$w'_{n+5} = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CM2 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert\, 1 \leqslant n \leqslant 5$$. $w'_{1 \geqslant n \geqslant 5}$, are the weights in the edges LB$\xrightarrow{5}$CM1. $w'_{6 \geqslant n \geqslant 10}$, are the weights in the edges LB$\xrightarrow{5}$CM2. $w'_{11} = 1Gflop$, is the a constant computing load in LB$\rightarrow$MS1. $w'_{12} = 1Gflop$, is the a constant computing load in LB$\rightarrow$MS2. $w'_{13} = 50GB$, is the a constant storage load in LB$\rightarrow$MSTO. $w'_{14} = w_1 + 40$, is the load over LB$\rightarrow$PN, such that w_1 is the VM image size in *GB*, comes from *deploy VM* operation, and 40 is a constant value in *GB* for the another operations.
cloud manager	$(c_1, \ldots, c_5, w_1, \cdots, w_5) \xmapsto{f^{CMn}}$ $(w'_1, \ldots, w'_{18})$	$c_n \in \{CPm1, CPm2, CPm3\}$, are the configurations which represent the targets of each load $w_n \vert 1 \leqslant n \leqslant 5$. $$w'_n = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CPm1 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert\, 1 \leqslant n \leqslant 5$$. $$w'_{n+5} = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CPm2 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert\, 1 \leqslant n \leqslant 5$$. $$w'_{n+10} = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CPm3 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert\, 1 \leqslant n \leqslant 5$$. $w'_{16} = 1Gflop$, is the a constant computing load in CMn$\rightarrow$MSn. $w'_{17} = 50GB$, is the a constant storage load in CMn$\rightarrow$MSTO. $w'_{18} = w_1 + 40$, is the load over CMn$\rightarrow$PN, such that w_1 is the VM image size in *GB*, comes from *deploy VM* operation, and 40 is a constant value in *GB* for the another operations.
cloud platform	$(w_1, \cdots, w_5) \xmapsto{f^{CPnn}} (w'_1, w'_2, w'_3)$.	$w_1, \cdots, w_5$, are the main loads come from the service, associatively, w_1 – deploy VM, w_2 – undeploy VM, w_3 – start VM, w_4 – stop VM, and w_5 – compute tasks. w'_1, w'_2 and w'_3 are, respectively, the edge-weight for the arcs CPnn$\rightarrow$WSnn, CPnn$\rightarrow$WSTO and CPnn$\rightarrow$PN, where: $w'_1 = w_1 - w_2 + w_3 - w_4 + w_5$; $w'_2 = w_1 - w_2 + 1MB$; $w'_3 = w_1 + w_3 - w_4 + 1MB$.

TABLE VI: EVALUATION FUNCTIONS FOR LEAF NODES.

Types	Functions
computing specific functions	*performance* (*duration*): $d(load) = \frac{load}{capacity}$, where $load$ is expressed in GFlop, $capacity$ is a constant of 70GFLOPs and d is the total time to resolve the load. *energy increment* (kWh): $energy_{increment}(load)$ here is considered a linear function which returns the amount of energy necessary to process the load above the average consumption of standby state. For computing have been considered $0.001kW$ per GFLOP.
storage and network specific functions	*performance* (*duration*): $d(load) = \frac{load}{capacity}$, where $load$ is expressed in GByte, $capacity$ is a constant of 1GBps and d is the total time to resolve the load. For the networking resources this concept is intuitively associated with the network throughput, however, for storage is necessary to explain that the performance refers to throughput of the data bus. *energy increment* (kW): $energy_{increment}(load)$ for data transmission is assumed as linear, and was here considered $0.001kW$ per GB transferred.
common functions	*availability*: $av(load) = 1 - p_{fault}(d(load))$, where p_{fault} is the probability which a fault occurs during the load processing. Here will be considered a linear naive probability, such that $p_{fault}(d) = d \times 0.01$. *accounting*: $ac(load) = price_{energy} \times energy_{total}$, where $price_{energy}$ is a constant of $0.38US\$/kW$ or $0.58US\$/kW$, depending on node configuration; and $energy_{total} = energy_{increment}(load) + energy_{average}(d(load))$, such that $energy_{average}(d(load)) = d(load) \times 0.1kW$ is the shared energy spent by the cloud by time slot, and $energy_{increment}(load)$ is the increment of energy result of resource usage. *security* (*risk of data exposition*): $s(load) = 1 - p_{exposure}(load)$, where $p_{exposure}(load)$ is the probability that the load processing results in data exposure and $s(load)$ is the trustability of the operation. The $p_{exposure}(load)$ is calculated as 0.001 for each second of operation. *CO_2 emission*: $c = energy_{total} \times 400$, where $energy_{total}$ was defined in the accounting evaluation function and 400 is a constant which represents the grammes of CO_2 per kW.

TABLE VII: EVALUATION FUNCTIONS FOR NON-LEAF NODES.

Types	Declarations	Definitions
performance	maximum duration of loads sent for successor nodes.	$p_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = max(w'_1[p], \ldots, w'_n[p])$, where p_v represents the total time to process the incoming loads, and $w'_n[p]$ represents the specific part of in the node-weight of n successor nodes, regards to the duration to process the loads sent by the node v.
availability	the product of the availability of successor nodes according to the sent loads.	$av_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \prod w'_n[av]$, where av_v represents the total availability of a node v according its dependencies, and $w'_n[av]$ represents the availability part in node-weights of the successors of v, related to the loads sent.
accounting	the sum of costs relative to the sent loads for successor nodes.	$ac_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \sum w'_n[ac]$, where ac_v is the total cost related to v and regards to the loads processed in the successors, and $w'_n[ac]$ is the accounting part of the successors' node-weight.
security	the product of security (regards to data exposition) of successor nodes according to the sent loads.	$s_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \prod w'_n[s]$, where s_v represents the total security measure of a node v, and $w'_n[s]$ represents the security measure part in node-weights of the successors of v, related to the loads sent.
CO_2 emission	the sum of total emissions relative to the loads sent to the successor nodes.	$c_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \sum w'_n[ac]$, where c_v is the total CO_2 emission associated with a node v, and $w'_n[ac]$ is the node-weight part associated with the emissions caused by the loads sent from v.

 ISBN: 978-1-61208-450-3