Decision-Theoretic Planning for Cloud Computing

Rafael Mendes, Rafael Weingartner, Guilherme Geronimo, Gabriel Bräscher
Alexandre Flores, Carlos Westphall, Carla Westphall
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, Brazil 88040-970
Email: {rafaeldesouzamendes,rafaelweingartner,guilherme.geronimo,gabrascher,alexandre.augusto.flores}@gmail.com,
{westphal,carlamw}@inf.ufsc.br

***Abstract*—This paper presents a mathematical model of decision planning for autonomic Cloud Computing based on the decision-theoretic planning model. It uses Markov decision process on the cloud manager to evaluate decisions and manage the Cloud environment. Also, it contributes to the state-of-art of Cloud Computing approaching the planning phase of the autonomic process with a mathematical model, considering two important factors, (1) the uncertainty of action's results and (2) the utility of the actions. Both factors are needed when dealing with complex systems as a Cloud.**

Keywords-cloud computing; decision-theoretic planning; autonomic computing; self-management

I. INTRODUCTION

The decision-theoretic planning (DTP) problems were extensively researched during the last decades. The main problem with the decision-theoretic (or probabilistic) approach for the planning phase in autonomic computing is the need to provide extensive information about the transitions between system states. However, with the arise of Cloud Computing (CC), sensor networks and other technologies that enabled the monitoring and collection of large volumes of data, the information became abundant and the recomendation of utility to solve the contradictions between rules on large rule-based polices [1], [2] must be taken seriously. On big data environments, the DTP problems no longer exist, enabling its application for the planning phase on the autonomic loop.

This work presents a model that plans actions for CC management systems using a decision-theoretic approach. It contributes to the state-of-art in CC research by:

- (i) Adapting the decision-theoretic models, which was based on *Markov Decision Process* (MDP), to use in the planning phase of the autonomic management loop;
- (ii) Introducing decision-theoretic and MDP for planning in CC;
- (iii) Applying mathematical models on a concrete decision making scenario for self-configuration of CloudStack [3].

This paper is organized as follows. Section II presents an overview of the concepts that are required to understand the proposed model. Section III presents the related works. In section IV is presented a conceptual proposal to guide the decision mechanism to CC.The Sections V and VI discusses and presents a mathematical model for the MDP approach, presenting a study case scenario of a Cloud implementation with CloudStack and Xen Cloud Platform. Finally, Section VII concludes the paper and presents future works that will improve the presented model.

II. LITERATURE REVIEW

A. Cloud Computing

After some years, the definition of CC that has grown in acceptance was created by NIST[4]:

> Cloud computing is a model for enabling ubiquitous, 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. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

Another important contribution to CC also can be found in [5, Section 3]; "Cloud Computing: the need for monitoring", where are stated some useful concepts to understand the fundamental elements of a Cloud.

As stated in [6], to deal with a complex system like a Cloud, it is necessary to be able to accurately capture its states.

Beyond the well-known CC characteristics, like on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service, etc. [4], [7], it is important to highlight the *stakeholder heterogeneity* characteristic. This characteristic is poorly defined and appears in some works like *stakeholder*, *actors* or *roles*.

In [7] the stakeholders are defined as roles:

- *Cloud Consumer*;
- *Cloud Provider*;
- *Cloud Auditor*;

 ISBN: 978-1-61208-318-6

- *Cloud Broker* ;
- *Cloud Carrier*.

Litoiu at al. [8] presents four type of stakeholders: *infrastructure providers*, *platform providers*, *application providers* and *end users*, although, it does not describe these stakeholders roles. In the same paper [8], there is a change on the terms used to present the stakeholders; the term actors is used instead of stakeholders. It places the actors in function of service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS); introducing a whole different set of actors as *layer owners*, that are: *IaaS owner*, *PaaS owner*, *SaaS owner* and *end users*.

Leimeister et al. [9] defines five actors in the *CC value network*: *Customer*, *Service providers*, *Infrastructure providers*, *Aggregate services providers (aggregators)*, *Platform provider* and *Consulting*.

In [10], it can be observed a different definition of roles on Cloud environments. The work defines these roles as *stakeholders* and present the following concepts: *Consumers*, *Providers*, *Enablers* and *Regulators*.

Letaifa et al. [11] present a definition of actors and roles in cloud computing systems as: *Vendors*, *Developers* and *End users*.

In Tan et al. [12], although the work focus is adoption (or not) of SaaS, it classifies stakeholders in three categories: *SaaS infrastructure provider*, *SaaS provider* and *SaaS consumer*.

There is no concise definition of CC stakeholders and interests. Furthermore, those distinct definitions may indicate that each Cloud implementation requires a specific stakeholder's modeling.

B. Autonomic Computing

The Autonomic Computing (AC) concept was based on the human nervous system, which regulates critical functions such as heart rate and body temperature, in the absence of a conscious brain [13]. AC systems have many common points with Expert Systems (ES) but are less generic, applied to management and control of wide computational systems, while the ES are applied in a more generic way. The AC differs from ES principally when it addresses the "action taking", that was unusual in ESs, as stated in [14].

AC systems are based on MAPE-K control cycle, that consists in *Monitor*, *Analyse*, *Plan*, *Execute* and *Knowledge* elements, Fig. 1 shows the MAPE-K life cycle.

An autonomic system, as shown in [13], to be able to perform *self-management*, must present four main abilities:

- *self-configuration* - the ability of configure itself according to high-level policies;
- *self-optimization* - the capacity of optimize its use of resource;
- *self-protection* - autonomic systems must protect itself from malicious or incorrect user behavior;
- *self-healing* - the ability of detect, diagnoses and fix problems.

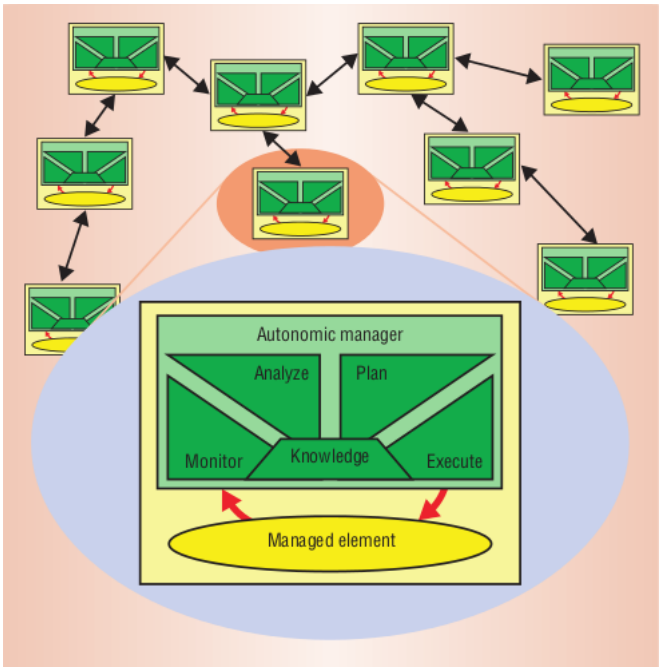

Figure 1. Structure of and autonomic agent.

In [2], the abilities were extended, adding four attributes of autonomic systems:

- *self-awareness* - the system must be aware of its internal state;
- *self-situation* - it should detect its current external operating conditions;
- *self-monitoring* - it has to detect changing circumstances;
- *self-adjustment* - it has to adapt accordingly to external or internal changes.

As stated in [15]:

> The overall goal of Autonomic Computing is the creation of self-managing systems; these are proactive, robust, adaptable and easy to use. Such objectives are achieved though self-protecting, self-configuring, self-healing and self-optimizing activities ... To achieve these objectives a system must be both self-aware and environment-aware, meaning that it must have some concept of the current state of both itself and its operating environment. It must then self-monitor to recognize any change in that state that may require modification (self-adjusting) to meet its overall self-managing goal. In more detail, this means a system having knowledge of its available resources, its components, their desired performance characteristics, their current status, and the status of inter-connections with other systems.

C. Markov Decision Process

Broadly speaking, it can be said that the planning techniques developed in the Artificial Intelligence domain are

 ISBN: 978-1-61208-318-6

concerned to obtain a course of actions which conducts the agent to a goal state or to an improvement in its condition. In deterministic planning approaches, each action leads to a single state. On the other side, the DTP is a non-deterministic way of modeling the decision taken problem where each action (or exogenous event) can lead the system state to more than one possible states with a certain probability.

To deal with probabilistic non-determinism, many mathematical tools must be used. A common framework used as underlying model to DTP is the MDP [16] that exposes the probabilistic relation between the system's states. Another framework is the decision theory [17] which combines the probability theory with utility theory.

In order to model the planning problem for a *stochastic dynamic system*, it is necessary to present a basic problem formulation using a MDP. This paper will model the problem according to [16], that presents the follow key elements:

- a set of decision epochs;
- a set of system state;
- a set of actions;
- a set of transition probabilities (state X action);
- a set of rewards or costs for transitions.

The problem can be mathematically expressed as $\{T, S, A_s, p_t(s'_{t+1}|s,a), r_t(s_t,a)\}$, where S is the set of states that the system can assume; A_s is the set of actions that can be taken over the system at state s; $p_t(s'_{t+1}|s_t,a)$ is the transition function that maps in time t a state s to a state s'_{t+1}, in time $t+1$, give an action a; $r_t(s_t,a)$ is function which gives the reward for the execution of an action a on state s_t.

The Fig. 2 shows a graphical representation of a MDP, where the green circles are the states, the red circles represents the actions, the arrows are the transitions between states, the numbers over the arrow are the probabilities to achieve a state, and the numbers indicated by the yellow arrows are the reward value of the transition.

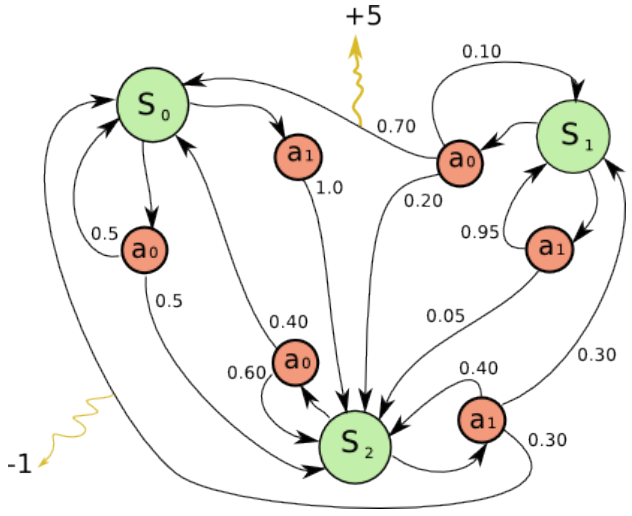

Figure 2. A graphical representation of MDP [18].

III. Related Works

Werner et al. [19] propose an integrated solution for cloud environment based on organization model of autonomous agent. It introduced concepts of rules and beliefs that in this paper can be compared with decision rules and transition functions, respectively. Despite the affinity in terms of goals and some architectural elements, [19] used neither MDP nor decision-theoretic approach and did not deepen the model to a level that would allow some kind of mathematical or logical inference.

The work in [8] presents a model that includes stakeholders, goals, sensors and actuators. It also provides an optimization and control module in the presented architecture which works in a multi-layer approach. However, it did not deepen to a level of inference and did not situate itself on AC context.

In [20], the author describe a MDP focused on the self-configuration phase. It situates the CC management on AC context, applying learning enforcement to deduce the action to be taken. The main differences between [20] and this paper is that the first one maintains the focus just in management of virtualized resource, as well as, not guiding the goals to meet stakeholder's interests.

A multi-level (physical and logical) resource manager is proposed in [1] to self-optimize on AC context. However, the paper uses utility functions regardless the uncertainty inherent to CC. It does not consider the action probabilities in this model.

An introduction of the term Autonomic Cloud Computing was made on [21], however, the work transits just as a survey and architecture proposal, without any inference model.

Sharma et al. [22] consider both utility and probability in their decision model. However, the work just provides a specific model that are based in a static set of actions to be optimized.

IV. Stakeholders, Interests and Cloud Computing

This work introduces the idea of *interests*. Interests have been implicitly referenced in many works that address the CC, like [23]. It is relevant to explain stakeholder's concerns in a way that they lead the decisions on a *cloud autonomic system*.

Sharma et al. [22] presents two approaches on decisions for dynamic provisioning: *cloud provider centric* and *customer-centric*. The paper differentiates them as follows:

> Cloud provider centric approaches attempt to maximize revenue while meeting an applications SLA in the face of fluctuating workloads, while a customer-centric approach attempts to minimize the cost of renting servers while meeting the applications SLA.

This definition states important aspects of decision on CC management:

- different types of stakeholders have different interests over the Cloud;

- any decision method inherently carries the ability to benefit the interests of some stakeholders and harm another;
- the stakeholder's interests are not always reconcilable given certain constraints of resources and service demand;

Therefore, it is possible to postulate that an autonomic system that intends to manage a Cloud must guide its decisions in order to *maximize the total satisfaction of the stakeholders*.

The satisfaction function can have many forms. Here will be considered just a function $\sigma : S \rightarrow \mathbb{R}$, where S is the set of possible *states* of the Cloud, and $\mathbb{R}$ is the set of real numbers.

A. Interests on the Cloud

Any measurable variable may constitute the state of Cloud. The set of variables that define a state $s \in S$ can be something really big. However, looking from the view of the Cloud's stakeholders, the indicators contained in the SLAs, SLOs and in some *operative constraints* can help to reduce the sample space. Therefore, the indicators which will compose a Cloud state must be at most the intersection of variables specified in all SLAs, SLOs and operative constraints. These indicators are the *measurable interests*.

Once the measurable interest that will compose the cloud state set S were selected, it is important to refine the indicators, reducing the amount of information, avoiding redundancy, and adjusting the measures. This way, the states changes would be compatible with the speed of the decision making system.

V. A Decision-Theoretic Model to CC

To introduce a decision-theoretic model in CC, it is necessary to find the elements of MDP on the Cloud.

A. MDP for CC

The first element to be modeled is the decision epochs. The idea of applying MDP to decision making in CC induces to think in discrete time decisions (epochs). For continuous time decisions, control theory have been more adherent.

Decisions on a Cloud environment may be taken in preset time slots or triggered by events coming from the monitoring system. The time slot must be adjusted to be large enough that a selected action can be computed, executed and evaluated before a next decision can be taken. The slot also must be short enough that some important event is not missed. In any case, the decision epochs will be treated as a time instant t.

The second element is the Cloud state, presented in subsection IV-A. It can be defined as a tuple $s = (x_1, x_2, \cdots, x_n)$, where $x_1, x_2, \cdots, x_n$ are values of random variables $X_1, X_2, \cdots, X_n$ that compose the measurable interests set. The set of all possible tuples will be expressed as the set S of the Cloud's states.

The third element is a set of possible actions to be taken in each state $s \in S$, that can be represented as the set $A_s \subseteq A$, where A is the set of all possible actions to be taken over the Cloud in any state.

The fourth item to be modeled is the probability distribution $p_t(s'_{t+1}|s_t, a)$, that express the probability of the system to assume the state $s'_{t+1} \in S$ at time $t+1$ giving that was executed the action $a \in A_s$ at state s_t in time t.

At last, the fifth element is a real-valued reward function $r_t : S \times A \rightarrow \mathbb{R}$ (in another form: $r_t(s_t, a)$) that gives the reward received by the decision maker at time t for the execution of the action a in state s_t.

B. Histories, Decision Rules and Policies

Considering that at time 1 an action a_1 executed over a state s_1 will generate a state s_2 at time 2, it is plausible to say that after a time t there will exist a history $h_t = (s_1, a_1, s_2, a_2, \cdots, s_t)$. Being the H_t the set of all possible histories, that will be characterized by the Cartesian product $\{S \times A \times S \times A \times \cdots \times A \times S\} = \{S \times A\}^{t-1} \times S$, it is possible to say that the history h_t constitute an *observation* of system states and actions.

A decision rule for CC can not be memory less. Likewise, must be history dependent. Also, in a real Cloud management, the rule should be *non stationary*, in other words, itself will change over time. Therefore, the decision function to the epoch t ($d_t : H_t \rightarrow A_{s_t}$) when receiving a decision history h_t returns an action a. In another form: $d_t(h_t) \in A_{s_t}$.

The specific decision rules implemented will be described in the Section VI.

As depicted in Section II, a policy provides to decision maker, a prescription for an action selection under any possible future system state. Thus, on a Cloud where the decision rules are time and history dependent, there will a policy which will look like: $\pi = (d_1, d_2, d_3, \cdots, d_t)$.

Clearly, when the decision horizon is infinite (as in CC), it is impractical to compute and evaluate all decision possibilities for all future states that the Cloud may assume in time. Although it has a finite horizon for planning, the size of S and A may result in a really hard work, since the number of policies is $\{S \times A\}^{N-1} \times S$, where N is the time horizon to compute.

VI. Mathematical Modeling

A. Study Case: the Cloud

1) Cloud environment: The Cloud used to ground the experiments was a CloudStack [3], using Xen Cloud Platform [24] and Xen hypervisor [25], [26] at the bare layer.

The Cloud Lab can be summarized as follows:

- Top level, cloud management – applying Cloudstack over the infrastructure to manage not just the hardware and software pieced together, but also to provide an easy and user friendly way to create, destroy and update resources on the fly;
- Underlying structure, hypervisor level – the core structure that is responsible for running the VMs. It is based on Ubuntu server 12.10 64bits, on which was installed and configured the Xen hypervisor 64bits hypervisor version 4.1 and Xen Cloud Plataform (XCP) version

 ISBN: 978-1-61208-318-6

TABLE I. CLOUD SERVERS CONFIGURATIONS

Name	CPU	Memory	Role	Cluster
Server 1	Pentium D @ 3.4Ghz	2 GB	Storage server	-
Server 2	Core 2 E6600 @ 2.4Ghz	7GB	Processing server	PV
Server 3	Core 2 quad, Q8200 @2.3Ghz	4GB	Processing server	PV
Server 4	Xeon E5405 @ 2Ghz	8GB	Processing server	HVM
Server 5	Phenom 9650 @ 2.3Ghz	4GB	Processing server	HVM
Server 2	Phenom II 965 @ 3.4Ghz	4GB	Processing server	HVM

1.6. Since the platform has a heterogeneous server environment, this level was then subdivided into two clusters, a cluster that contains the physical hosts with hardware virtual machine capabilities (HVM) and a pure virtual cluster (PV) that contains all the servers that does not have support for HVM machines;

- Bottom level structure, storage level – at the lower level of the structure there is the storage, on which was built upon a Raid controller. The Raid controller exports 3 volumes to the storage server, a total of 3.18 Terabits using either RAID1 or RAID5, granting at least reliability against hard drive failures. The volumes exported by the RAID controller are mounted on a passive storage server as XFS file system and then exported the XFS partitions as network file system (NFS).

Table I shows the hardware configurations of each server in the Cloud, and the role that it plays.

2) Cloud Stakeholders: There are three types of stakeholders interested in the Cloud environment:

- the *cloud decision committee* – that is interested in the maximization of resource usage and the equipment aggregation of research departments by accession to Cloud, in the scope of the Federal University of Santa Catarina, Brazil;
- the *cloud managers* – which have interest in satisfying the concerns of cloud decision committee with availability assurance and energy economy;
- the *researchers* – having interests that their VMs stay available and maximize their capacity of memory, CPU and storage.

3) Cloud problems: From the stakeholders interests it is possible derive some decision problems:

1) Structure Aggregation: It is necessary to decide if a PM can integrate the cloud or not;
2) Maximize the resource usage: It is necessary to add and remove resource (CPU, memory) to idle and overloaded VMs;
3) Ensure availability: It is necessary not to lose user requests, that can be characterized as network packages.
4) Energy Economy: It is necessary to minimize the energy consumption;
5) Obtain availability and maximize the capacities of VMs: the decision needs are covered on items 3 and 2.

These decision problems have consequences when placed side-by-side. Although it is possible to provide more resource to VMs via hypervisor, the operation system will not recognize the additional resource, to add or remove them when it is necessary stop and restart the VMs. This way, to provide availability, it is critical to execute these operations in a moment with low probability of request loss.

To achieve energy economy, it is important to shutdown PMs, but, it is necessary to execute VM migrations at a moment when the request loss is not likely to happen. The aggregation of new equipments can be made at any time if there exists a good link quality between the new host and the core of the cluster where it will be added.

B. The model

To get a mathematical modeling that covers the study case, it is necessary to model four basic structures, which will be presented below.

1) Cloud State: In order to cover stakeholders interests, there was selected a set of basic metrics to compose the Cloud states as follows:

- the PM state – on or off – $S^{PM} = \{on, \overline{on}\}$;
- the PM energy consumption – level of consumption – $S^{E} = \{low, average, high\}$;
- VM state – up or down – $S^{VM} = \{up, \overline{up}\}$;
- CPU – level of CPU utilization – $S^{cpu} = \{idle, underused, welldimensioned, overloaded\}$;
- memory – level of memory utilization – $S^{mem} = \{idle, underused, welldimensioned, overloaded\}$;
- network link utilization– level of link utilization – $S^{lu} = \{idle, underused, wellused, overloaded\}$;
- network link quality – level of link quality – $S^{lq} = \{poor, regular, good\}$;

Given the sets PM of all PMs in the Cloud, VM of all created VMs, it is possible to define the following sets to compose the Cloud state:

- $ST^{PM} = PM \times S^{PM}$, of each PM state;
- $SE^{PM} = PM \times S^{E}$, of each PM energy consumption;
- $SC^{PM} = PM \times S^{cpu}$, of PM CPU utilization;
- $SM^{PM} = PM \times S^{mem}$, of PM memory utilization;
- $LU^{PM} = PM \times S^{lu}$, of PM network connection utilization;
- $LQ^{PM} = PM \times S^{lq}$, of PM network link quality;
- $ST^{VM} = VM \times S^{PM}$, of each VM state;
- $SC^{VM} = VM \times S^{cpu}$, of VM CPU usage;
- $SM^{VM} = VM \times S^{mem}$, of VM memory utilization;
- $LU^{VM} = VM \times S^{lu}$, of VM network connection utilization;
- N^{PM}, the number of PMs in the Cloud.

 ISBN: 978-1-61208-318-6

This way, there exists a set that can cover all possible Cloud state is the set $S = ST^{PM} \times ST^{E} \times SC^{PM} \times SM^{PM} \times LU^{PM} \times LQ^{PM} \times ST^{VM} \times SC^{VM} \times SM^{VM} \times LU^{VM} \times N^{PM}$, where a state can be represented at time t as $s_t \in S$. The set of states in this example cannot be stationary, once VMs may be created or destroyed and PMs can be added or removed.

2) Actions: It is the set of base actions that can be executed on the environment that is being used to run the experiments:

- a_1 – turn on PM;
- a_2 – turn off PM;
- a_3 – turn on VM;
- a_4 – turn off VM;
- a_5 – migrate VM;
- a_6 – scale up VM;
- a_7 – scale down VM;
- a_8 – admit PM;
- a_9 – refuse PM;
- a_0 – no action;

It will be considered as different action, the actions executed on distinct resource (e.g., $a_1(pm_1)$). This way, the set of action will be larger than the nine basic actions listed above. However, the actions cannot be executed in any state of a resource. For instance, a PM in $\overline{up}$ state cannot be turned off. To get the set of all possible actions which can be performed in state s_t, there will be defined a function $\alpha_t : S \rightarrow 2^A$ that receives the state of the Cloud and returns a set of actions, where 2^A is the power set of A set. In another form $\alpha_t(s_t) = A_{s_t}$. As the Cloud state, α is not a stationary functions and can change in time.

3) Transition Function: Considering a state $s_t \supset \{ST^{PM_1} = on\}$ over which the action $a_2^{PM_1}$ is executed, it will be induced to a state $s'_{t+1} \supset \{ST^{PM_1} = off \cup LQ^{PM_1} = bad\}$ with a probability x, if there is demand to PM_1, and a state $s''_{t+1} \supset \{ST^{PM_1} = off \cup LQ^{PM_1} = \{regular, good\}\}$ with probability $1 - x$ otherwise, depending on whether the machine is on or off, and how much requests exits to its VMs.

The state and action relations must be captured by the transition function. It must provide the probability that an action a, executed on state s_t, results in a state s'_{t+1}, for each time t. In other words, $p_t(s'_{t+1}|a, s_t)$.

On a real-world CC management application, the transition functions will be the product of a series of machine learning and forecasting methods. In this paper, we will only assume that there is a non-stationary probability function.

4) Reward Function: Giving the satisfaction evaluation function described in Section IV, here presented as $v = \sigma(s_t)$, it is possible to establish a reward function that evaluates the decision maker reward to lead the system from a state s_t to a state s_{t+1}. It can be expressed as: $r_t(s_t, a, s_{t+1}) = \sigma(s_{t+1}) - \sigma(s_t)$.

When a decision maker recognizes the Cloud in state s_t and evaluates the impact of action a over that state, it does not have the resultant state s_{t+1}. This way, it is necessary to introduce the uncertainty, as seen on (1).

$$r_t(s_t, a) = \sum_{s_{t+1} \in S} r(s_t, a, s_{t+1}).p_t(s_{t+1}|s, a) \quad (1)$$

It is assumed that each stakeholder $k \in K$ has a set of interests $I_k \subseteq I$, each $i_k \in I_k$ having a weight w_{i_k}. Therefore, if there exists a function $\sigma^{I_k} : S \rightarrow \mathbb{R}$ that evaluates the Cloud state according to the interest i of stakeholder k, it is possible to propose a weighted interest function σ^k as presented in (2).

$$\sigma^k = \sum_{i_k \in I^k} w_{i_k} \cdot \sigma^{I_k}(s_t) \quad (2)$$

The σ function presented above can be obtained by the sum of all weighted interests, as presented by (3).

$$\sigma = \sum_{k \in k} \sigma^k(s_t) \quad (3)$$

The expected reward is computed as presented in (1).

C. Decision rules

It is simple to elaborate a decision rule for one MDP epoch. All is needed is the selection of the action that has the best expected reward (like in (1)). This way, the decision rule to $t = 1$ must return the set of the best actions as shown in (4).

$$A_t^* = \arg\max_{a \in A} \{ \sum_{s_{t+1} \in S} r(s_t, a, s_{t+1}).p_t(s_{t+1}|s_t, a) \} \quad (4)$$

Nevertheless, to compute the expected reward for a policy π_t^N, where t is the actual time and N is the horizon of the policy, it is necessary to sum the product of all rewards that achieve a final state and the probability to achieve each state, like in (5).

$$r(\pi_t^N) = \sum_{h_t^N \in H_t^N} (\sigma(s_N) - \sigma(s_t)) \cdot p(s_N) \quad (5)$$

The $p(s_N)$ can be computed from the sum of all probabilities of histories $p(h_t^N)$ that start on s_t and finishes on s_N. Considering $X_N(h_t^N) = s_N$ as a random variable that returns the state N of a history, it is possible observe in (6) how to compute a state probability.

$$p(s_x) = \sum_{h_t^N, X_N = s_x} p(h_t^N) \quad (6)$$

The probability of a history can be calculated by (7).

$$p(h_t^N) = p(s_{t+1}|s_t, a_t) \times \prod_{i=t+1}^{N-1} p(s_{i+1}|s_i, a_i) \quad (7)$$

 ISBN: 978-1-61208-318-6

VII. Conclusion and Future Works

This article has presented a decision-theoretic modeling to decision making for CC management in an AC context, using the MDP as a mathematical framework.

This paper contributed to the state-of-the-art in CC research in the sense that it tackles the phase *planning* of an autonomic cycle with a mathematical model which takes into consideration the uncertainty of action resulting in complex systems such as a CC management systems.

For future work, the following steps will be considered:

- A big data model to feed the transition function created with the monitoring data bases;
- Extend the CloudStack to implement the model on its resource manager and perform experiments to observe the performance of the model in taking decisions;
- Analyze a meta-management model to optimize the autonomic planner;
- Research methods of action discovery and learning.

References

[1] W. Walsh, and G. Tesauro, and J. Kephart, and R. Das, "Utility functions in autonomic systems", in: Autonomic Computing, 2004. Proceedings. International Conference on, May 2004, pp. 70–77.

[2] S. Dobson, and R. Sterritt, and P. Nixon, and M. Hinchey, "Fulfilling the vision of autonomic computing", Computer, vol. 43, no. 1, Jan. 2010, pp. 35–41.

[3] Apache Foundation. Apache CloudStack, 2013 retrieved in September 2013 from http://cloudstack.apache.org/

[4] P. Mell and T. Grance, The NIST Definition of Cloud Computing, Tech. rep., National Institute of Standards and Technology, Information Technology Laboratory, Jul. 2009.

[5] G. Aceto, and A. Botta, and W. de Donato, and A. Pescapè, "Cloud monitoring: A survey", Computer Networks, vol. 57, issue 9, Jun. 2013, pp. 2093–2115.

[6] A. Viratanapanu, and A. Hamid, and Y. Kawahara, and T. Asami, "On demand fine grain resource monitoring system for server consolidation", Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services, 2010 ITU-T, Dec. 2010, pp. 1–8.

[7] R. B. Bohn, and J. Messina, and F. Liu, and J. Tong, and J. Mao, "Nist cloud computing reference architecture", in: Services (SERVICES), 2011 IEEE World Congress on, July 2011, pp. 594–596.

[8] M. Litoiu, and M. Woodside, and J. Wong, and J. Ng, and G. Iszlai, "A business driven cloud optimization architecture", in: In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), 2010, pp. 380–385.

[9] S. Leimeister, and M. Bhm, and C. Riedl, and H. Krcmar, "The business perspective of cloud computing: Actors, roles and value networks", in: In Proceedings of the 7th international conference on Economics of grids, clouds, systems, and services (GECON'10), 2010, pp. 129–140.

[10] S. Marston, and Z. Li, and S. Bandyopadhyay, and J. Zhang, and A. Ghalsasi, "Cloud computing - The business perspective". Decis. Support Syst. vol. 51, no. 1, Apr. 2011, pp. 176–189.

[11] A. B. Letaifa, and A. Haji, and M. Jebalia, and S. Tabbane, "State of the Art and Research Challenges of new services architecture technologies: Virtualization, SOA and Cloud Computing", International Journal of Grid & Distributed Computing, vol. 3, issue 4, Dec. 2010, pp. 69–88.

[12] C. Tan, and K. Liu, and L. Sun, "A design of evaluation method for saas in cloud computing", Journal of Industrial Engineering and Management, vol. 6, no. 1, Feb. 2013, pp. 50–72.

[13] J. Kephart and D. Chess, "The vision of autonomic computing", Computer, vol. 36, no. 1, Jan. 2003, pp. 41–50.

[14] S. Gutirrez and J. Branch, "A comparison between expert systems and autonomic computing plus mobile agent approaches for fault management", DYNA, vol. 78, no. 168, Aug. 2011, pp 173–180.

[15] R. Sterritt and D. Bustard, "Autonomic computing - a means of achieving dependability?", in: Engineering of Computer-Based Systems, 2003. Proceedings. 10th IEEE International Conference and Workshop on the, Apr. 2003, pp. 247–251.

[16] M. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley Series in Probability and Statistics, Wiley, 2009.

[17] B. Lindgren, Elements of decision theory, Macmillan, 1971.

[18] Wikipedia, Markov Decision Process, 2013 retrieved in September 2013 from http://en.wikipedia.org/wiki/Markov_decision_process

[19] J. Werner, and G. Geronimo, and C. B. Westphall, and F. L. Koch, and R. Freitas, C. M. Westphall, "Environment, services and network management for green clouds", CLEI Electron. J. vol. 15, no. 2, Aug. 2012, pp 2–2.

[20] J. Rao, Autonomic Management of Virtualized Resources in Cloud Computing, Ph.D. thesis, Wayne State University, Jan. 2011.

[21] R. Buyya, and R. Calheiros, and X. Li, "Autonomic cloud computing: Open challenges and architectural elements", in: Emerging Applications of Information Technology (EAIT), 2012 Third International Conference on, Sep. 2012, pp. 3–10.

[22] U. Sharma, Elastic resource management in cloud computing platforms, Ph.D. thesis, University of Massachusetts, May 2013.

[23] D. Durkee, "Why Cloud Computing Will Never Be Free", Queue Journal vol. 8 no. 4, Apr. 2010, pp. 20-29.

[24] The Linux Foundation, Xen Cloud Platform, June 2013 retrieved in September 2013 from http://www.xenproject.org/downloads/xen-cloud-platform-archives.html

[25] The Linux Foundation, Xen Hypervisor, 2013 retrieved in September 2013 from http://www.xenproject.org/users/virtualization.html

[26] P. Barham et al., "Xen and the Art of Virtualization", in: Proceedings of the nineteenth ACM symposium on Operating systems principles, Oct. 2003, pp. 164–177.

 ISBN: 978-1-61208-318-6