Towards a dynamic multi-level negotiation framework in Cloud computing

Aya Omezzine[a,b,c], Saïd Tazi[a,b]
[a]Univ de Toulouse, UTI-Capitole, LAAS, F-31000 Toulouse, France
[b]CNRS, LAAS, 7 Avenue du Colonel Roche, F-31400 Toulouse, France
aya.omezzine@gmail.com, {tazi, khalil}@laas.fr

Narjes Bellamine, Ben Saoud[c], Khalil Drira[b], Gene Cooperman[b,d]
[c]RIADI, Ecole Nationale des Sciences de l'Informatique, Université de la Manouba, Manouba, Tunisia
[d]College of Computer and Info. Science Northeastern University, Boston, MA / USA
narjesbbs@gmail.com, gene@ccs.neu.edu

***Abstract*—Automated negotiation is a compelling way to reach a satisfactory agreement between two or more parties with conflicting needs. It is generally adopted in the Cloud in order to enable negotiators to establish a common agreement, written as a Service Level agreement (SLA) contract. After studying the existing cloud negotiation frameworks on the one hand and cloud-specific properties on the other hand, we select the most important modules to use in a cloud negotiation framework. Then we design new modules overcoming existing limitations in cloud negotiation. Finally, we integrate these modules in order to build a generic, multi-level and dynamic negotiation framework adapted to the Cloud, called NEGOCLOUD.**

Keywords- automated negotiation; automated re-negotiation decision model; multi-level negotiation; HPC Cloud

I. INTRODUCTION

Negotiation is the process that the different parties undertake to reach a satisfactory agreement and where each party tries to maximize its gain. Negotiation has been developed for IT interactions, without the need for human interaction. This is used, for instance, in negotiation between software agents in an e-commerce transaction. With the evolution of infrastructures and platforms leading to highly distributed environment like the Grid and Cloud environment, we need automated negotiation that handles complex interactions. Automated negotiation is based on three elements: 1) negotiation protocol; 2) negotiation objects; and 3) the decision model, which greatly influences the outcome of negotiation and must be well-defined.

Cloud computing is based on delivering software, a platform, and hardware as a standard utility (on-demand and pay-per-use). Automated negotiation in the Cloud is carried out primarily to establish the Service Level agreement (SLA) contract, which is a formal representation of values for QoS, obligations and penalties agreed on by two or more cloud negotiators with conflicting interests. On one hand the consumers want to have services satisfying not only their functional requirements but also non-functional parameters (QoS). On the other hand providers want to maximize their profits. So negotiation between consumers and providers is the best way to bridge the gap between the two parties.

The main characteristics of a cloud computing environment are: 1) a highly dynamic market (Cloud providers and consumers can enter or leave the Cloud market, consumers' needs may vary over time, and the providers' resource allocations are dynamic.); 2) on-demand service provisioning between heterogeneous parties; 3) Multi-interdependent levels (The Cloud architecture is layered. Each layer offers one type of service, which may depend on a lower layer. Also, cloud market providers may offer value-added services that depend on other providers' services [1].); and 4) an evolution toward a high performance computing (HPC) environment capable of running large-scale applications by offering on-demand computing power to resource intensive applications.

Considering the Cloud characteristics cited above, the classical negotiation models may not be suitable for many reasons. First, existing decision models in the Cloud are designed for negotiation at a single level, whereas we require multiple levels in order to allow a negotiation at one level to influence the negotiation at another level. Second, once the SLA is established, it cannot be changed and must be fulfilled, or else a penalty is paid. Since the cloud is highly dynamic, this may lead to many SLA violations. So we need a re-negotiation mechanism based on dynamic scheduling to minimize SLA violations and for efficient resource management. We aim to propose a dynamic and multi-level negotiation model to deal with the limits cited above. Our contributions are three fold:

- A generic decision model that takes into consideration not only the negotiator's own context (preference, goal, constraints, etc.) but also external contexts.
- A decision model supporting influences between interdependent negotiations.
- Negotiation at runtime through contract re-negotiation.

The aim of this position paper is: to present some existing, state-of-the-art Cloud negotiation frameworks; to compare them, and identify their limits; and to introduce a new dynamic and multi-level negotiation framework named NEGOCLOUD.

The rest of this paper is organized as follows. Section 2 presents automated negotiation basis and their use for Cloud. Section 3 reviews some state-of-the-art negotiation models in Cloud computing environments, and discusses them. Section 4 provides an overview of the NEGOCLOUD

978-1-4673-8149-9/15/$31.00 ©2015 IEEE

negotiation framewor.k. Conclusions are presented in Section 5.

II. PRESENTATION OF AUTOMATED NEGOTIATION : MAIN CONCEPTS AND THEIR USE FOR THE CLOUD

We seek to answer the central question:
"How can automated negotiation be applied in a Cloud environment?"
In this section, we begin by recalling the primary basis for automated negotiation and discuss its potential application in cloud environments.
Automated negotiation deals with three main components [2, 3]:

A. Negotiation protocols

The protocol expresses the locutions that may be exchanged between the negotiators and defines the rules of interaction; it can be also represented by a state transition diagram that expresses legal transitions. An example of a negotiation protocol is the Fipa contract net protocol. The protocol definition varies according to the multiplicity of participants:

- One-to-one (bilateral)

The protocol consists of the interaction between two participants for a specific type of service. The most commonly used bilateral protocol is Rubinstein's alternate offer protocol [4], it consists of an exchange of offers and counter offers between the two negotiators until one of them agrees on the offer received or until a deadline is reached.

- One-to-many

A consumer negotiates with many providers. In most cases, there is more than one provider that offers a service matching the user requirements. So negotiation is used as a selection mechanism to assist the consumer in choosing.

- Many-to-many (multilateral)

Many consumers are negotiating with many providers. In most cases; consumers need different types of services and are negotiating with many providers to fulfill composition of services. So, a coordination module must be considered to coordinate the one-to-many negotiations.

Three types of protocols may be used in Cloud computing. But the multilateral one is the one that is best adapted for the Cloud market environment.

B. Negotiation objects

The negotiation objects represent issues on which participants must agree. The offer exchanged between negotiators may contain one or more issues. In the case of one issue, we are facing one-issue negotiation, and in the other case multi-issue negotiation.

Also, the issues may be negotiable (values change over time during a negotiation session) or non-negotiable (with fixed values) depending on the choices of the negotiators. There are two types of issues:

- Service-specific issues: representing QoS parameters related to the service. Each type of service has specific QoS attributes. For example, for IaaS service we may negotiate CPU capacity; and memory size; for PaaS services, integration and scalability; and for SaaS services, reliability and scalability [5].
- Generic issues: attributes that match all types of services, such as price.

In a Cloud computing environment, the issues negotiated are related to the layer in which the negotiation is done (IaaS, PaaS, SaaS).

C. Negotiation decision models

The negotiation model enables negotiators to generate and evaluate offers during the negotiation process in order to realize their objectives. First, to evaluate an offer, the model is generally based on utility functions and negotiators' preferences. The utility function measures the degree of satisfaction of the negotiator and allows him or her to evaluate the offers received by his opponent and whether the negotiators' preferences offer an acceptable range of values for each issue. Second, to generate an offer, the model employs a strategy that determines at each step the counter offer that will be sent to the opponent.

There are three main approaches to automated negotiation [2]:

1) Game theory

It consists of a set of players performing strategic interactions. Each player has a set of possible actions that can be performed and a function that measures the outcome for each action taken. The aim of each player is to perform actions that maximize its outcome.
In game theory, the participants should have finite strategies (possible actions and their outcome). In fact, both the number of negotiators' choices and the number of round are finite. Also the participants should have a full knowledge of others' preferences [2]. These assumptions make a game-theoretic approach poorly adapted for a dynamic and highly distributed environment like Cloud computing; indeed the number of participants and the number of issues may be large. Further, the strategies in such a dynamic environment cannot be finite.

2) Argumentation

The argumentation approach is based on arguments exchanged in order to persuade others and change their beliefs. This approach permits the sharing of additional information about negotiators' beliefs or other circumstances. The arguments exchanged may be used to justify a proposal, to support an offer, or to explain a fact. The argumentation is based on the interaction between arguments and how to reach a consistent conclusion [6].
The main elements of an argumentation-based negotiation are 1) argument and proposal evaluation: consider both objective (quality of proof) and subjective evaluation (preference); 2) argument and proposal generation: this may done using a different approach, such as explicit rules (if-then) or a planning approach; and 3) argument and proposal selection: based on the relationships between arguments. At each step, the negotiator should select the most relevant

argument from its sets of arguments in response to the opponent's proposal [7].

The elements cited above require a high computational and communication load, and the efficiency of argumentation depends on the application. To be pertinent as a negotiation approach, the overhead must be compensated for by its potential benefits [6].

The application to Cloud computing of argumentation as a negotiation approach is a new challenge [8]. It may not be efficient in all cases.

For our research work, we have chosen to focus on a heuristic approach as being better adapted to Cloud environment. The heuristics allow for realistic assumptions in a less constrained model [2].

3) Heuristics

The heuristic approach is based on the representation of the negotiators' proposals and counter proposals as points in the space with their outcomes. The outcomes are calculated according to the negotiators' preferences and their utility functions. The possible agreements are in the intersection of the negotiators' acceptable outcomes sets. The aim is to find sub-optimal solutions [2].

Two basic strategies may be used according to the negotiators' goals:

a) Concession:

At each step, the negotiator decreases the total utility value by decreasing the utility of individual issues. The strategy consists in determining the level of concession at each step by using either a single function or a combination of such functions. The most common functions for calculating the level of concession are proposed in [9]:

- Time-dependent: considering the time given for negotiation and modeling the fact that negotiators should concede more easily as the deadline approaches.
- Resource dependent: considering the resource remaining during negotiation.
- Behavior dependent: the negotiator imitates his or her opponent's behavior and generates a counter offer accordingly.

b) TradeOff:

At each step, one decreases some individual attributes' utility values and increases others in order to keep the total utility value unchanged. An example of a tradeoff strategy for two attributes is given in [10].

III. NEGOTIATION MODELS IN THE CLOUD

First, we present some work dealing with negotiation in the cloud. Then we identify their limitations and some challeges for cloud negotiation.

A. State-of-the-art negotiation models in Cloud computing

Negotiable SLA terms in the Cloud differ from one layer to the next, except for generic issues such as price. For each type of service, there correspond some characteristics that are the objects of negotiation. Here we will classify existing work according to the cloud layer to which it applies.

1) Negotiation at the IaaS layer

The negotiation at this level is about resources and their characteristics, and it is essentially conducted to enhance resource management. This is an important and complex task. Resource owners must efficiently manage their infrastructure in order to run user applications, to satisfy user needs, and to maximize the owners' gain. To do so, resource providers need to allocate resource dynamically, according to the current allocation and the resource demand.

The nature of the contribution of this work to the field of automated negotiation for the Cloud varies according to the negotiation objective. The authors in [11] propose a negotiation strategy based on time slot and price preferences in order to maximize resource utilization, to minimize resource fragmentation on the provider side, and to minimize cost on the consumer side. The core idea is to define a time slot utility function for the consumer and the provider, by converting high-level preferences for each time slot into mathematical parameters. This model works only if both consumer and providers can perform this conversion, a task requiring skilled users; and a task assuming that the provider can anticipate the consumer's demands, which is not so simple in a Cloud environment

The ANEKA platform presented in [12] supports negotiation for the execution of task application in parallel on many nodes. The proposed negotiation strategy consists of varying the start time according to the availability of nodes. We are interested in multi-issue negotiation in which we consider many factors for negotiation and not just the possibility of executing a job.

In [13], the authors extend Haizea (an open source resource lease manager) with an advance reservation policy supporting negotiation when the requests of users cannot be fulfilled (when available resource no longer satisfy the requirements). The authors propose two algorithms: 1) on the provider side for counter offer generation. It consists in varying at each step some of the user requirements (start time, duration, memory and CPU) in order to minimize resource fragmentation; and 2) on the consumer side. It consists in obtaining the most desirable offer by presenting greater flexibility in the requirements. The main drawback of this work is that the negotiation strategy is static for all incoming requests and considers only resource fragmentation.

In [14], the authors propose two algorithms for two different negotiation strategies (tradeoff and concession) and compare them using a "storage as a service" scenario. The algorithms are designed only for negotiation parameters having two attributes.

In [15], the authors address the problem of energy consumption in the Cloud, and try to find a balance between energy consumption (resource provided) and performance.

To do so, the authors propose a negotiation mechanism based on optimization heuristics using particle swarms. This has the advantage of higher social welfare.
The authors in [16] address negotiation between a business process owner and the infrastructure owner. This work is concerned with the mapping between business process requirements (SLA) and the IT infrastructure. The idea is to formalize both the requirements on the business process side and the capabilities of the infrastructure on the provider side, and then to compare the requirements with different configurations of technical services comprising the necessary business process realized through service replication. The negotiation is based on this comparison.
The authors in [17] present a service negotiation framework facilitating automated SLA negotiation between providers and consumers. They define a negotiation strategy for each party based on time-dependent negotiation tactics, taking into account the reliability of the provider's offer and the balancing of resource utilization.
Some work dealing with virtual machine negotiation does not consider physical resources ([11], [15], [14]). The provider generally negotiates based on supply and demand and VM characteristics, with no concern for the placement of the VM in the data center, although the distribution of virtual machines among servers highly influences highly the VM performance (response time). It is essential to find an efficient resource allocation that maintains the SLA previously established at the VM level. In this sense, and in the context of a cloud provider having many geographically distributed data centers, that the authors in [18] propose a negotiation mechanism for SLA establishment at the VM level (the same mechanism as presented in [11]) and an SLA-driven resource allocation strategy that aims to guarantee the previously established SLA (response time). The strategy considers for VM placement both the geographical location of the data centers and their workload.

2) *Negotiation at the SaaS layer*

The negotiation at this level is between users who want to use applications already deployed on the Cloud and the SaaS provider. The negotiation is about software characteristics
In [19], the authors propose an automated SLA negotiation framework including different SaaS providers and a SaaS broker assisting consumers' negotiation. The broker's aim is to maximize his or her margin and to satisfy the user's requirements. The provider aims to maximize his or her revenue by accepting many requests. The strategy used for counter offer generation is time-dependent and market-dependent.

3) *More than two layers*

The authors in [20] focus on concurrent negotiation in the Cloud market across multiple levels (consumer, service provider, and resource provider). The authors propose a many-to-many protocol based on FIPA specification messages. The many-to-many protocol consists of many one-to-many protocols coordinated by a coordination entity (CE) for each participant in the negotiation. The CE is responsible for the generation of the negotiation entities that will handle the one-to-many negotiation. For decision making they use a time-dependent concession strategy.

4) *Cloud market*

Some research work considers the Cloud purely as a market, and deals with negotiation between different providers and consumers [21, 22], or brokers [23], concerning different types of services. For this, they consider only price as a negotiation issue.
In [23], the authors propose the design and development of software agents in a Cloud market with the purpose of enhancing discovery, negotiation and composition of cloud services. The Cloud market is composed of two types of markets, one to carry out negotiation between consumers and brokers using a market driven strategy (time, market factors), and a second market that is designed for interactions among brokers/providers using a utility oriented coordination strategy.
In [21], the authors focus on the resource allocation problem in a dynamic cloud market, and propose two negotiation strategies: 1) on the buyer side, based on dynamic pricing; and 2) on the seller side, by trying to maximize revenue by means of accepting or rejecting requests, or even by canceling some potential agreements.
In [22], the authors provide a design of a cloud market called "Cloud Agency" for delivering services and for resource management. The design is based on software agents having different roles in the market: each provider is represented by an agent. Provider's agents are managed by a mediator agent, a negotiator agent responsible for SLA fulfillment between the mediator and the user. Each user is represented by a client agent.

B. *Discussion and challenges*

1) *Discussion*

Most work dealing with Cloud negotiation focuses on the IaaS level, and proposes efficient strategies that can be reused. Negotiation in HPC Cloud computing is still emerging. Only Aneka [12] was tested using HPC applications. The negotiation model is a basic one: one-issue negotiation (start time) considering only the number of nodes and their availability in decision making.
There is less work dealing with negotiation at the SaaS layer. That work considers broker benefit [21], but doesn't consider the resources needed for request execution. There are different designs for the cloud market, and the work focuses only on price as a negotiation issue [23, 21].
We note that most work deals with a specific scenario implementing a specific strategy. There is no generic negotiation decision model that can match agents using different configurations.
Most work deals with single-level negotiation (IaaS, SaaS) concerning specific negotiation objects. Even in a pure cloud market, the brokers are dealing with the same negotiation object as demanded by the consumers (again a

single level). A multi-level negotiation protocol was proposed in [20]. To the best of our knowledge there in no work dealing with multi-level decision making.
Less work addressed heterogeneity among the different participants. A potential solution was proposed in [22] which consists of building a unique Cloud ontology.
Another important feature is the possibility to consider re-negotiation. This idea was presented in [22], but there is no concrete definition on how it can be used efficiently.

2) Challenges

- *Why Multi-level is crucial?*

There are two reasons justifying the multi-level aspect for use in the Cloud and the dependencies between the levels.
First, there is the layered architecture of the Cloud; it is composed of at least three layers (SaaS, PaaS, IaaS), with each layer offering one type of service (Software, platform, hardware). The service provisioning in SaaS/PaaS layer depends on the resources owned by the SaaS/PaaS provider. If the service provider doesn't maintain his own infrastructure and if the provider is renting his resources from an IaaS provider, which is the case in the most Clouds, then the negotiations for software/platform provisioning will depend on the negotiations done with the IaaS providers. In order to execute the applications required by the consumers within the specified QoS, a SaaS provider needs to negotiate for resources from the IaaS providers, since the availability of the application depends on the available resources.

Second, there is the new vision of the cloud as an open cloud exchange in which cloud providers collaborate to offer value-added services [1]. Here, the provisioning for the "final service" depend on elementary services. So the negotiation for value-added service provisioning will be greatly influenced by the negotiations done for acquiring the services from which it is composed.

- *Why negotiation needs to be flexible and dynamic (monitoring and re-negotiation)?*

This challenge is motivated by the new vision of "HPC Cloud computing"; we will start by explaining this concept and then justifying the need for flexible and dynamic negotiation.
Many observers see the potential convergence between the batch and resource manager technology of high performance computing (HPC) and the Cloud computing technology [24]. HPC is dominated by CPU-intensive and parallel computing. We anticipate this convergence evolving through three generations. In the current (first) generation, a job reserves a fixed number of computers (CPUs), and the job is given exclusive use of those CPUs. This is done because in a CPU-intensive environment, there is no easy way to allocate the costs among multiple users if multiple users are sharing the same CPU. (In some variations, a job reserves a number a fixed number of CPU cores on each computer. It is given exclusive use of those CPU cores, but not the entire CPU and not the entire RAM.)
In the second generation, multiple jobs may share a CPU. This allows for better throughput. One job may be especially CPU-intensive, but use little RAM. A second job may need only one CPU core (for a highly sequential computation), but be very RAM-intensive. In this case, rules are found for allowing the jobs to coexist with low impact on each other's performance. In this case, the total throughput improves. A flexible policy (set of rules) will have to be developed in order to make it practical to run jobs from multiple users on the same CPU or the same CPU core.
However, the second generation does not allow one to allocate the costs of the multiple jobs and assign an appropriate share of the costs to each user. This may be satisfactory in computing within a departmental cloud in a university or corporation, but the approach does not scale well.
In the third generation, we anticipate a scalable approach to sharing computer resources among multiple users in a large cloud. In this setting, users will negotiate with the cloud for certain parameters related to job execution (start time, deadline, CPU, etc.) and finish by establishing an SLA for each job. To avoid SLA violation and to enhance resource management and scheduling or even to maximize revenue, the negotiation should be dynamic and flexible by communicating with both monitor and scheduler. In other words, after establishing the SLA and when the job is executing, it is important to consider the possibility of re-negotiation based on standard performance measurements given by a monitor (risk of violation). Also, it is essential to enable re-scheduling after re-negotiation. Two opportunities for this are: migrating a job to another computer with greater resources (and possibly with exclusive use of the computer); and suspending the job and restarting it later.

- *Why negotiation needs to consider heterogeneity between participants?*

Cloud computing is a heterogeneous environment. There is no standard protocol for communication between clouds nor a common semantic for expressing resources and their characteristics. So the negotiation between heterogeneous parties may lead to inconsistencies. We believe that negotiators should at least agree on the negotiation protocol and SLA terms for the semantics before beginning the negotiation. (This phase is called meta-negotiation.)

- *Why generic decision model?*

The Cloud environment is dynamic: consumers and providers can enter or leave the Cloud market, the consumers' needs may change over time, the resources are managed dynamically, etc. To make the right decision in the presence of such dynamic events, the negotiator's decision model must be generic, in order to follow the right strategy for every situation.

In the next section we will describe our negotiation framework NEGOCLOUD, which deals with the above-mentioned challenges.

IV. NEGOCLOUD

In this section, we present the architecture of NEGOCLOUD and its main modules and we show how this framework can be instantiated by presenting two scenarios.

A. Overview of NEGOCLOUD framework

The core idea is to support negotiators (final user, provider, broker) in a Cloud environment with an intelligent software agent containing the NEGOCLOUD model. So Cloud negotiation participants must be represented by software agent that can negotiate in a dynamic and flexible manner. A software agent perceives changes in the environment and then acts based on its own internal state. In our context, the changes in the environment are examples of cloud dynamism: the internal state is the agent's own preference, and the agent actions are the reactions of the negotiators.

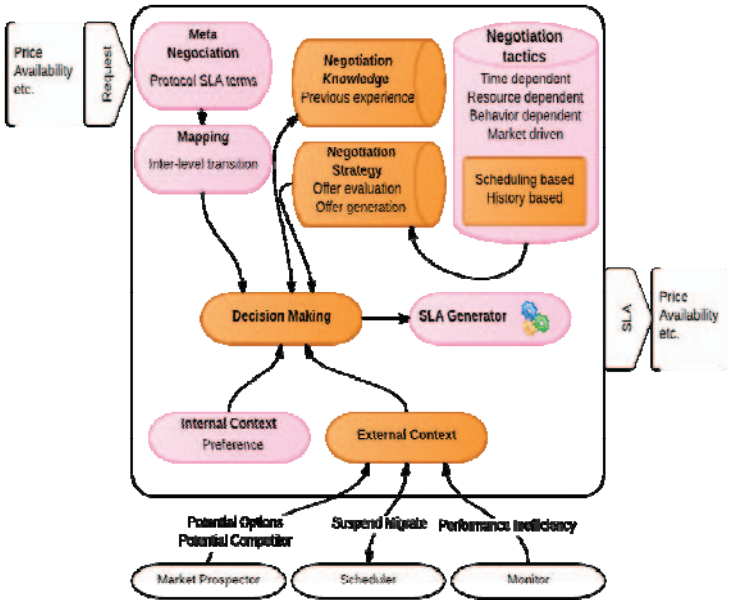

Figure 1. NEGOCLOUD framework

The Fig. 1 illustrates the main modules of the NEGOCLOUD model; each module's functionality will be described below:

1) Meta-negotiation

The meta-negotiation concept was described in [25] in order to facilitate communication among heterogeneous parties. It occurs before the start of the negotiation process, by agreeing on the negotiation protocol, security standards and SLA template that will be used during negotiation. We believe that meta-negotiation is crucial in any negotiation between heterogeneous entities, since negotiators may have different protocols and may express their SLA terms differently. So, it is essential to take meta-negotiation into consideration in our model, in order to avoid inconsistencies and reach a common understanding during negotiation.

2) Mapping between Service Request/Resource request

This consists of translating the service requirements into the necessary resources for its execution.

This mapping module is essential for each SaaS/PaaS provider. A provider receives requests stated in term of software/ platform characteristics. Realizing these requests requires corresponding resources. The mapping allows the SaaS/PaaS provider to define the resource needed for each request.

The negotiation at the levels of SaaS and PaaS depends on the requests' translation into the resources required -- even in the situation where the software/platform provider maintains its own infrastructure.

3) External context module

This module enables the exchange of information between the agent and other entities on the Cloud provider platform [12].

a) Scheduler/ dispatcher

Scheduling is the core of the resource management system. This entity must implement efficient algorithms to realize the various provider goals (maximize resource utilization, maximize revenue, minimize costs, etc.). The scheduler/dispatcher is able to schedule the requests on the VMs and it starts their execution at the start time agreed upon and based on the SLA established (SLA-based scheduling).

In NEGOCLOUD, we aim to enhance the classical scheduling through communication with the negotiation module used for decision making. In current scheduling, after the SLA establishment, there is no way to alter the execution of jobs. In our framework (after a re-negotiation), the schedule may be altered through a scheduling-based negotiation tactic. For example, his can be implemented by suspending a job and restarting it later, or y migrating it to another machine.

b) Resource Monitor

Monitoring of job execution is an important task. In the classical model, this is done by reporting an SLA violation or by measuring provider reliability for use in later negotiations [17]. In our model, we aim to use the monitoring to report standard performance measurements to the agent, which can then anticipate system degradation and so trigger a re-negotiation before SLA violation occurs. In this way, the agent avoids paying any penalties.

c) Market prospector

The market prospector estimates the number of competitors and the number of available options in the market. For this component, we will use the competition function defined in [23]. Using information from by the market prospector, the agent can decide to employ or to not employ a market-driven strategy.

For the consumer software agent, the external context module will communicate only with the market prospector.

4) Internal context module

This module contains the information related to the negotiator's preferences: utility function, the goal to pursue, the reserved and preferred value for each issues and the weights for each negotiation issue.

5) Negotiation tactics database

The negotiation module should be generic. Hence, we plan to implement all of the most commonly used negotiation tactics (time-dependent, market-dependent, resource dependent, etc.)

We intend to add to the existing negotiation tactics two new ones: 1) a scheduling-based tactic will help providers to negotiate according to the current scheduling state, and it will make re-scheduling possible even at runtime; and 2) a history-based tactic will take into account past negotiations for services on which the currently negotiated service depends (for example, in a multi-level negotiation).

6) Negotiation knowledge base

This knowledge base contains successful outcomes of past negotiations. The stored information helps in the multi-level case, where there is an interdependent relationship between the currently negotiated service and past negotiations. This is important in cases of re-negotiation, since it provides information from the original negotiation (the negotiation for establishment of an SLA).

7) Decision making

The decision-making module will choose the best strategy to follow according to the information given by both the internal and the external module. The strategy may be a combination of different tactics provided by the negotiation tactics module.

B. Example scenarios using NEGOCLOUD

As a first step, we choose a basic scenario to show how this architecture may be applied in a simple case, and how the generic model can be instantiated in particular environments (such as the HPC environment illustrated here).

a) The multi-level case

We focus on two levels: 1) negotiation between clients and a SaaS provider (Here we take the example of a CRM software packages provider.); and 2) negotiation between a SaaS provider and an IaaS provider.

The CRM provider needs virtual machines to run client requests, A request may contain these parameters: number of accounts, contact duration, response time, etc. The software provider negotiates with different IaaS providers for acquiring the virtual machines needed using the adopted strategy (*decision making module*) according to the provider's preferences about types of VMs (i*nternal context* in Figure 1) and according to the currently available resources (e*xternal context* in Figure 1). If the negotiation ends with success, then the SLA is generated (*SLA generator* in Figure 1). The outcome of the negotiation will be stored in the *negotiation knowledge base.* The negotiation with clients for software delivery will depend on types, price and availability of VMs (*history-based tactic).*

b) HPC: an application using dynamic negotiation

An HPC provider may be a SaaS provider delivering an HPC application as a service or it may be an IaaS provider enhanced with an HPC environment (Amazon EC2) [26]. In our scenario, we will consider an HPC cloud provider as an IaaS provider, since this is the most common case [26].

A consumer with a resource-intensive applications negotiates with an IaaS provider to execute the application. The request may contain such parameters as start time, deadline, number of nodes, price, etc. The provider strategy depends on the current scheduling (*scheduler* from Figure 1) and the market supply and demand (*market prospector* from Figure 1). The information is provided by the *external context* module of Figure 1. After agreeing on the parameters, the SLA is established (*SLA generator* from Figure 1) and execution of the application begins.

In this scenario, the *monitor* might then detect a performance inefficiency during execution of the application (e.g., due to an over-commitment of memory). If so, a re-negotiation is triggered by the *external context module* using the strategy of *scheduling-based tactics*. *T*he outcome of this re-negotiation ,might be to suspend some jobs and restart them later. This information would then be communicated to the scheduler for execution.

V. CONCLUSION

A goal of Cloud computing is to dynamically provision IT services on demand. To deliver these services to the consumer, along with the requested QOS, the Cloud provider and consumer must agree on the service level expressed by an SLA contract. The SLA terms are generally negotiated to reach a satisfactory agreement between the provider and the consumer. In this paper, we focus on the existing cloud negotiation frameworks and to what extent they can be applied in the Cloud. We note that some important Cloud computing characteristics (high dynamicity and multiple levels) are not considered in existing frameworks. This paper provides an overview of a new negotiation framework adapted to the Cloud, named NEGOCLOUD. The NEGOCLOUD framework is based on the integration of existing negotiation modules with some new ones. The new modules were designed in order to support some Cloud-specific properties.

NEGOCLOUD provides a generic negotiation model that takes into account multi-level negotiation. It supports runtime negotiation, and selects an appropriate negotiation strategy by taking into account both the internal and the external context. In future work, we intend to implement NEGOCLOUD and test it in a variety of scenarios.

ACKNOWLEDGMENT

This work has been funded by a “Chaire d’Attractivité” of the IDEX Program of the Université Féderale de Toulouse Midi-Pyrénées, Grant 2014-345.

REFERENCES

[1] A. Bestavros, O. Krieger. "Toward an Open Cloud Marketplace: Vision and First Steps", *IEEE Internet Computing*, vol.18, no. 1, pp. 72-77, Jan.-Feb. 2014.

[2] N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, M. Wooldridge, and C. Sierra. “Automated negotiation: prospects, methods and challenges”. *Intern. J. of Group Decision and Negotiation*, 10(2):199215, 2001.

[3] Nitto, E. D.; Penta, M. D.; Gambi, A.; Ripa, G. & Villani, M. L. Krämer, B. J.; Lin, K.-J. & Narasimhan, P. (ed.). "Negotiation of Service Level Agreements: An Architecture and a Search-Based Approach.", *in Proc. ICSOC*, 2007, 4749, pp. 295-306.

[4] A. Rubinstein. "Perfect equilibrium in a bargaining model". *Econometrica: Journal of the Econometric Society*, vol.50., pp.97-109, 1982.

[5] M. Alhamad, T. Dillon, E. Chang, "Conceptual SLA framework for Cloud computing," *in Proc. Digital Ecosystems and Technologies (DEST)*, 2010,pp. 606 – 610.

[6] N. Maudet, S. Parsons and I. Rahwan. "Argumentation in Multi-Agent Systems: Context and Recent Developments". *in Proc. AAMAS International Workshop on Argumentation in Multi-Agent Systems (ArgMAS),* 2007, pp. 35-52

[7] I. Rahwan, Sarvapali D. Ramchurn, Nicholas R. Jennings, P. Mcburney, Simon Parsons, and Liz Sonenberg. "Argumentation-based Negotiation", *Knowl. Eng. Rev.,* vol.18, pp. 343-375, Dec. 2003.

[8] S. Heras, F. de la Prieta, S. Rodríguez and J. Bajo and Botti, J. Vicente and V. Julián. "The Role of Argumentation on the Future Internet: Reaching agreements on Clouds". *in Proc. CEUR Workshop*. 2012.

[9] P. Faratin and C. Sierra and Nick R. Jennings, "Negotiation decision functions for autonomous agents", *International Journal of Robotics and Autonomous Systems,* vol. 24, pp.3-4, 1998.

[10] Yan, Jun and Kowalczyk, Ryszard and Lin, Jian and Chhetri, Mohan B. and Goh, Suk Keong and Zhang, Jianying, " Autonomous Service Level Agreement Negotiation for Service Composition Provision", *Future Gener. Comput. Syst.,* vol.23, pp.748-759, July 2007.

[11] K. Mong Sim, "A Price- and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations*", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS*, vol. 42, no. 3, June 2012.

[12] R. Buyya, C. Shin Yeo, S. Venugopal, J. Broberg, I. Brandic. "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility". *Future Gener. Comput. Syst.,* 25(6):599616, June 2009.

[13] J. Akhani.; Chaudhary, S. & Somani, G., "Negotiation for resource allocation in IaaS cloud.", *in R. K. Shyamasundar & Lokendra Shastri*, 2011 , pp. 15.

[14] X. Zheng and Martin, Patrick and Brohman, Kathryn. "Cloud Service Negotiation: Concession vs. Tradeoff Approaches". *In Proc. the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012*). 2012

[15] G. Copil ,Moldovan, I. Salomie, C. Tudor, I. Anghel, D. Borza. « Cloud SLA Negotiation for Energy Saving – A Particle Swarm Optimization Approach". *In Proc. 8th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP2011)*, 2012.

[16] V. Stantchev, C. Schröpfer. "Negotiating and enforcing QoS and SLAs in grid and Cloud computing.", *in Proc. Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing*, pp. 25-35., 2009

[17] A. Vahid Dastjerdi, R. Buyya. "An Autonomous Reliability-Aware Negotiation Strategy for Cloud Computing Environments". *In Proc. of the 2012 12th IEEE/ACM International Symposium on Cluster*, Cloud and Grid Computing (ccgrid 2012). 2012.

[18] S. Seokho, J. Gihun, J. Sung Chan, "An SLA-based Cloud Computing That Facilitates Resource Allocation in the Distributed Data Centers of a Cloud Provider", *J. Supercomput.,vol.64,pp. 606-637, May 2013.*

[19] Wu, Linlin and Garg, Saurabh Kumar and Buyya, Rajkumar and Chen, Chao and Versteeg, Steven, "Automated SLA Negotiation Framework for Cloud Computing.", *In Proc. CCGRID, 2013, pp. 235-244.*

[20] Siebenhaar, M.; Nguyen, T. A. B.; Lampe, U.; Schuller, D. & Steinmetz, R. (2011), "Concurrent Negotiations in Cloud-Based Systems., in Kurt Vanmechelen"; *In Proc. Jörn Altmann & Omer F. Rana,* pp. 17-31, 2011.

[21] Bo An and Lesser, Victor and Irwin, David and Zink, Michael. "Automated negotiation with decommitment for dynamic resource allocation in Cloud computing". *In Proc. International Conference on Autonomous Agents and Multiagent Systems*. 2010.

[22] S. Venticinque, R. Aversa, B. Di Martino. « A Cloud Agency for SLA negotiation and management". *In Proc. conference on Parallel processing.* 2011

[23] K. Mong Sim. "Towards Complex Negotiation for Cloud Economy." *Advances in Grid and Pervasive Computing,* 2010, pp. 395-406.

[24] http://people.rennes.inria.fr/Adrien.Lebre/VTDC/vtdc15.html [Feb 28, 2015].

[25] I. Brandic, D. Music, S. Dustdar. "Service mediation and negotiation bootstrapping as first achievements towards self-adaptable grid and cloud services". *In Proc. 6th International Conference Industry Session on Grids Meets Autonomic Computing*, 2009, pp. 18.

[26] Church, P., Wong, A., Brock, M., & Goscinski, A; "Toward exposing and accessing HPC applications in a SaaS cloud". *In Proc. ICWS,* 2012, pp. 692-699.