2011 First International Symposium on Network Cloud Computing and Applications

Cross-layer SLA selection for Cloud services

Yousri KOUKI
ASCOLA Research Group
EMN-INRIA, LINA
Ecole des Mines de Nantes, France
Yousri.KOUKI@inria.fr

Thomas LEDOUX
ASCOLA Research Group
EMN-INRIA, LINA
Ecole des Mines de Nantes, France
Thomas.LEDOUX@inria.fr

Remi SHARROCK
ASCOLA Research Group
EMN-INRIA, LINA
Ecole des Mines de Nantes, France
Remi.SHARROCK@inria.fr

***Abstract*—Cloud computing is a paradigm for enabling remote, on-demand access to a set of configurable computing resources as a service. The pay-per-use model enables service providers to offer their services to customers in different Quality-of-Service (QoS) levels. These QoS parameters are used to compose some bipartisan Service Level Agreement (SLA) between a service provider and a service consumer.**

A main challenge for a service provider is to manage SLAs for its service consumers, i.e. automatically determine the appropriate resources required from the lower layer in order to respect the QoS requirements of his consumers.

This paper proposes an optimization framework driven by consumer preferences to address the SLA dependencies problem across the different cloud layers as well as the need of flexibility and dynamicity required by the domain of Cloud computing. Our approach aims to select the optimal vertical business process designed by cross-layer cloud services, enforcing SLA dependencies between layers. Based on Constraint Programming (CP), our approach can take into account dynamic QoS parameters in a flexible manner to compose the best vertical business process. Experimental results demonstrate the flexibility and effectiveness of our approach.

I. Introduction

Cloud computing is an emerging paradigm that aims to streamline the on-demand provisioning of computing resources as services, providing the end-users flexible and scalable services accessible through the Internet [1]. Based on a pay-per-use service concept, one cloud service may exist in many versions and new versions of the same service appear regularly. We believe that Quality-of-Service (QoS) parameters play an important role to select the most appropriate cloud service. These QoS parameters are used to compose some bipartisan *Service Level Agreement* (SLA between two parties). An agreement is defined as a contractual support between the service provider and the service consumer on an expected QoS level.

The Cloud architecture is usually composed in several XaaS layers - including Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) - and SLAs are characterized at various levels in this hierarchy to ensure the expected QoS for different stakeholders (see Figure 1). In this scheme a *cloud layer* n is a provider for the upper layer $n+i$ (where $i > 0$) and a consumer for the lower layer $n-j$ (where $0 < j <= n$).

On top of these XaaS layers, the end-user accesses web services and may design a workflow based on Service-Oriented

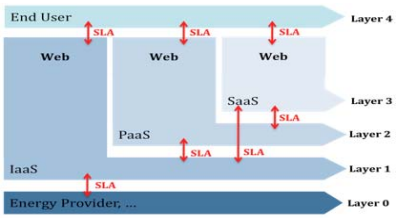

Fig. 1. SLA hierarchy

Architectures (SOA). In SOA, a business process is a collection of structured activities (services) that interact with each other in order to serve a particular goal for a particular customer. It is usually represented as an horizontal business process (see Figure 2) whereas in Cloud computing the layers are connected in a *producer-consumer* way representing a *vertical business process*. The end-to-end global SLA is defined by an aggregation of bipartisan SLAs at the different layers.

In this paper, we study the selection of the best hierarchical aggregation of SLAs in order to ensures bipartisan SLA dependencies. This issue is already addressed by QoS-aware service composition problem. Since the problem is known as NP-hard [4], it takes a significant amount of time to find optimal solutions. Several heuristics have been proposed to find semi-optimal solutions in a reasonably short time [2][3]. However, these heuristics are not flexible enough because the number of QoS parameters is fixed whereas in open environment such as Cloud computing the dynamicity has to be considered.

The objective of this paper is to propose an optimization framework to address the SLA selection problem as well as the need of flexibility and dynamicity in the domain of Cloud computing. We propose a solution to compose the hierarchical aggregation of SLAs that ensures bipartisan SLA dependencies. The main design choice is to address both the cloud consumer and cloud provider point of view. We allow the consumer of the *cloud layer* n to formulate his preferences. Then, based on these preferences, the *cloud layer* n proposes a SLA on-the-fly while he selects the appropriate services/resources from his provider.

We rely on the flexibility of *Constraint Programming* (CP) models to address an optimization problem [5]. The use case considered is proposed by an industrial partner of the MyCloud

978-0-7695-4550-9/11 $26.00 © 2011 IEEE
DOI 10.1109/NCCA.2011.30

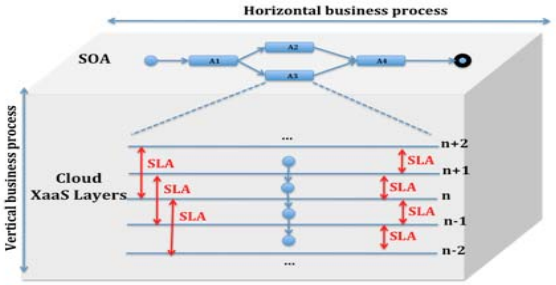

Fig. 2. SOA vs Cloud

project[1] and reflects a real usage of clouds. We show that we can (i) easily add a new QoS criterion to change the combinatorial QoS optimization problem, (ii) rapidly select the best *vertical business process*, (iii) finally propose SLAs on-the-fly.

The rest of paper is organized as follows. Section 2 shows motivating scenario and introduce a service selection model. Section 3 describes the approach of selection. Section 4 describes the experimental evaluation of our approach. Sections 5 and 6 conclude this paper with some discussion on related works.

II. Cloud service selection

This section briefly presents a motivating scenario and basic concepts and introduces our selection model.

A. Motivating scenario

In our motivational scenario, Yury is a accountant at the enterprise ABC. He needs to analyse ABC's big data. He plans to utilize Bime[2] services to accomplish these tasks. Bime is a SaaS-based solution for business intelligence accessible remotely via Internet. He would also like to propose his requirements and preferences to customize the data analysis. He finds a Bime service according to various preferences presented below:

- Preference 1: ensure the best response time for unlimited computing resources or financial cost.
- Preference 2: ensure a correct response time for a financial cost attached.

To perform large-scale distributed computing, Bime is based on Hadoop framework[3], the Amazon Simple Storage Service (S3) and the Amazon Elastic Compute Cloud (EC2). Bime seeks to provide several SLA based on response time, availability and financial cost in order to offer services with different QoS levels (e.g. for gold, silver or bronze client).

According to XaaS layers, the *cloud layer* n is represented by Bime, the consumer (*cloud layer* $n+1$) is the accountant Yury and the provider (*cloud layer* $n-2$) is Amazon[4].

[1]MyCloud. mycloud.inrialpes.fr. 2011.
[2]Bime. http://fr.bimeanalytics.com/. 2011.
[3]Hadoop. http://hadoop.apache.org/. 2011.
[4]*cloud layer* $n-1$ matches PaaS layer which is not used here.

Based on "pay only for what you use" concept, EC2[5] price varies depending on type of instance (Standard, Micro, High-Memory, etc.), operating systems (Red Hat, Windows Server, etc.) and regions (US, EU, etc.) while S3[6] depends on size of data, storage pricing, request pricing, data transfer pricing and regions. As different instances (EC2 and S3) are provided by Amazon, Bime needs to choose the appropriate instances in order to respect the SLA requirements of his consumer. Due to these considerations, a rigorous approach is needed for optimal cloud service selection. The objective of Bime (*cloud layer* n) is to help Yury (layer $n+1$: the consumer) to select the appropriate Bime service and automatically determine the resources (EC2, S3) required from Amazon (layer $n-2$: the provider) according to various SLA "Yury-Bime" presented below:

- SLA 1: ensure a response time less than 30 seconds and an availability rate of 99%, whatever the financial cost.
- SLA 2: For maximum financial cost of 0.90 €/request, ensure a response time less than 1 minute with a minimum financial cost, and an availability rate of 90%.

The SLA "Yury-Bime" is defined for an identified class of query, a determined data volume and its complexity. The classification of queries depends on the parameters and the number of join whereas data complexity depends on number of lines/tables in database. This classification is on-going work.

B. Basic concepts

As mentioned in section I, the cross-layered cloud services define a *vertical business process* and we address their SLA dependencies across the different layers. In this paper, for better understanding, we only consider the case with three layers: end-user, SaaS and IaaS layers. Figure 3 shows the motivating scenario with our business model. A vertical business process defines some series of abstract services mutually interacting in order to accomplish a given business goal. Each abstract cloud service encapsulates certain function (business intelligence, compute, storage, etc.). A concrete cloud service/resource (Bime, EC2, S3, etc.) implements an abstract cloud service.

A SLA is a set of Service Level Objective (SLO) that are measurable performance indicators. It is defined upon a business process as its end-to-end QoS constraints. An appropriate set of concrete services/resources may be selected so that it guarantees the fulfilment of SLA and satisfy consumers preferences. In addition, a set of service/resource instances can be deployed with different aggregation functions. Because of that, we need QoS aggregation functions to calculate each global criteria. In this paper, we limit just to sequence and parallel aggregation. QoS aggregation functions are summarized in table I.

C. Model for Selection optimization

When a QoS-aware service composition problem has several potential conflicting objectives to be optimized simultaneously,

[5]Amazon EC2. http://aws.amazon.com/fr/ec2/. 2011.
[6]Amazon S3. http://aws.amazon.com/fr/s3/. 2011.

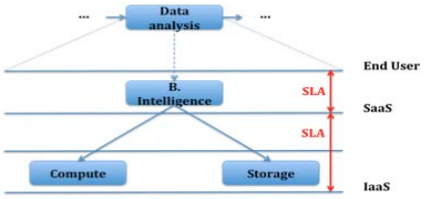

Fig. 3. Motivating scenario vertical business process

TABLE I
QOS AGGREGATION FUNCTIONS

	Unit	sequence	parallel
Availability: Av	Percentage:%	$\prod_{i=1}^{n} Av_i$	$\prod_{i=1}^{n} Av_i$
Cost: Ct	Euros: €	$\sum_{i=1}^{n} Ct_i$	$\sum_{i=1}^{n} Ct_i$
Response time: Rt	Seconds: s	$\sum_{i=1}^{n} Rt_i$	$max(Rt_i)$

there is no single optimal solution but a whole set of alternative solutions of equivalent QoS. For example, minimizing financial cost and maximizing availability are clearly conflicting and there is no single optimum to be found. Our proposition is to "externalize" the potential conflicting objectives from the selection problem by letting the cloud consumer to choose the trade-offs between different solutions. The consumer formulates his QoS preferences to the cloud provider which in return proposes different SLAs.

Concretely, we represent a SLA by three vectors (QoS parameters, QoS values and QoS weights) and we advocate an approach providing the k first solutions in order to help consumer in the negotiation phase by giving different possibilities.

For example, to represent the SLA 1, we define the following 3 vectors and k parameter: $[Rt; Av], [<= 30; >= 99], [1; 2], k = 2$. '1' in the QoS weights vector, represents the highest priority (the response time is given priority over the availability). Solutions will be ordered according to the consumer preferences shown in the QoS weights vector. The k parameter represents the number of solution to find.

To represent the SLA 2, we just need to add the third QoS criterion and update vectors values: $[Rt; Av; Ct], [<= 60; >= 90; <= 90], [1; 2; 3], k = 2$. In conclusion, our model allows us to easily add a new QoS criterion and produce new SLAs on-the-fly.

III. MULTIOBJECTIVE APPROACH

We rely on the flexibility of *Constraint Programming* (CP) models to address the optimization problem [5]. In this section we first introduce CP, then explain the objective function and finally present our approach modelling.

A. Constraint programming

Optimization problems can be addressed by several techniques such as Linear Programming (LP). The most obvious difference between CP and LP is the modeling of the problem. CP accepts any type of relations to formulate constraints consisting of linear inequalities and logical constraints. In our approach, we chose CP since it is also flexible and can easily take into account new constraints without modifying the initial algorithm.

CP allows solving combinatorial problems modelled by a Constraint Satisfaction Problem (CSP). Formally, a CSP is defined by a triplet (X, D, C):

- Variables: $X = (X_1, X_2, ..., X_n)$ is the set of variables of the problem.
- Domains: D is a function which associates to each variable X_i its domain $D(X_i)$.
- Constraints: $C = (C_1, C_2, ..., C_m)$ is the set of constraints.

The idea is to model and solve a problem by exploring the constraints that can fully characterize the problem, hence any solution of the problem must satisfy all of the stated constraints. We are going to solve a selection problem where unknowns are concrete services/resources and variables are QoS criteria (e.g., X_1 : Response time of Bime gold service) knowing domains from business process (e.g., $D(X_1) = \{3, 4, 5, 6\} seconds$) and constraints from SLA (e.g., C_1 : Response time less than 30 seconds). In case of optimization problem, we also define an objective function.

B. Objective function

We introduce an objective function (equation 1) for SLA-aware cloud service selection. With this function, we can find the optimal concrete service/resource to realize a *vertical business process* and satisfies the SLA.

$$o = min \sum_{i=0}^{n-1} \alpha_i w_{Ni} q_{Ni} \tag{1}$$

Where n is the number of criteria, α_i is the objective of criterion i (maximize or minimize), w_{Ni} is weight of criterion i and q_{Ni} is global normalized value of criterion i. To calculate the global value of each criterion, we use functions defined in table I.

The α_i is defined by equation 2.

$$\alpha_i = \begin{cases} +1 & \text{if } minimize(q_{Ni}) \\ -1 & \text{if } maximize(q_{Ni}) \end{cases} \tag{2}$$

The w_{Ni} is represented by weight equation (equation 3):

$$w_{Ni} = \begin{cases} \frac{w_{n-i}}{w_T} & \text{if } w_T \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

Where $w_T = \sum_{i=1}^{n} w_i$ and w_i is the weight of QoS criterion q_i .

When dealing with multiple criteria decision-making, it is necessary to normalise the criteria due to numerical stability. Variable normalization requires taking values that span a specific range and representing them in another range. The standard method is to normalize variables to [0,1]. Then, the simplest method to normalize values is the linear scaling transform (equation 4 and 5) for positive (decreasing dimensions)

Fig. 4. QoS values

and negative (increasing dimensions) quality criteria.

$$q_{N\,i} = \begin{cases} \frac{q_i - q_m}{q_M - q_m} & \text{if } q_M - q_m \neq 0 \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

$$q_{N\,i} = \begin{cases} \frac{q_M - q_i}{q_M - q_m} & \text{if } q_M - q_m \neq 0 \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

where q_i is the global value of criterion i, q_m = min (q_i) and q_M = max (q_i).

C. Approach modelling

In this section, we see in a few steps, how to model and to solve a selection problem using CP where inputs are a business process, QoS parameters, QoS values, QoS weights and k parameter. Our approach returns the set of solutions that satisfies constraints.

As we want to solve our problem with CP, we need to create a constraint programming model. Variables and constraints are associated to it. Values of variables are taken from a domain which is defined by a set of values from business process and SLA. The constraints define relations to be satisfied between variables and QoS values. We limit the number of solution to k. The last step is to define the search strategies and solve the objective function.

IV. Evaluation

This section describes the experimental evaluation of the proposed approach. The implementation is described before presenting the results of the evaluation.

A. Implementation

Our approach implementation uses Choco[7] as constraint solver. Choco is a Java library for CSP and CP. It is built on an event-based propagation mechanism with backtrackable structures. The QoS values for each concrete service/resource are presented in an XML file provided by the *cloud layer* n. Furthermore, we implemented an XML parser using the open source SAX API to parse the XML file (see Figure 4) in order to extract the QoS values.

[7]Choco. http://www.emn.fr/z-info/choco-solver/. 2011.

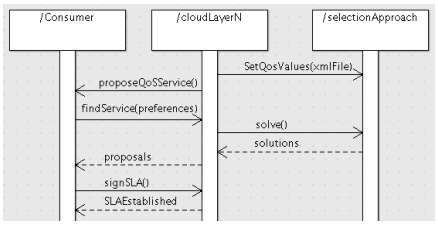

Fig. 5. Selection interactions

QoS values are the results of calibration between QoS criterion in different layers. The calibration is performed by the *cloud layer* n (Bime in our case) via tests. If the *cloud layer* n decide to propose a new QoS criterion - for particular service - to its customers, it must calculate these QoS value for different resources/services that used from other parties. A *mapping function* $f_{\text{QoS}}()$ of QoS criteria relationship between different layers will be proposed in our future work. In our use case, based on the type of consumer query, data volume and its complexity, f_{QoS} calculates the response time for any instance (EC2 or S3). For example : $f_{\text{QoS}}(QueryClass = simple, DataVolume = 1TB; DataComplexity = simple)(EC2largeInstance) = 10s$.

The used test system consists of a Mac OS X 10.6.7, 2.7 GHz Intel Core i7, 4GB 1333 MHz DDR3 memory, and 3Mb L2 Cache.

B. Solutions

For experimentation values, we used values that are summarized in motivating scenario and the XML QoS file (Figure 4). A solution is a set of concrete services. Our approach determines which concrete services to use in order to ensure bipartisan SLA dependencies. According to the value of k parameter, our approach provides possible proposals. For example, we find two proposals for SLA 2 :

- Proposal A: Business Intelligence: GoldBime, Compute: EC2 Large Instance and Storage: S3 EU.
- Proposal B: Business Intelligence: BronzeBime, Compute: EC2 Extra large Instance and Storage: S3 EU.

On the one hand, Bime provides to Yury the possible solution (GoldBime, BronzeBime) with more details about all QoS parameters. On the other hand, the approach defines the appropriate instance (EC2, S3) to use in order to provide the service.

The sequence diagram in Figure 5 shows interactions between the customer Yury, the *cloud layer* n (Bime) and our prototype. At the end of selection interactions, the consumer chooses one proposal then a negotiation phase can be started. If both parties (Yury and Bime) agree, the SLA is signed.

Conceptually, our approach goes through a search tree recursively in depth-first order. It runs backtracking until computing k feasible solutions or until proving infeasibility. CP reduces computing times by rewriting [6]: propagation and labeling. Constraint propagation may prove that the problem has no solution by reducing a domain to a empty set.

C. Flexibility

Our approach is flexible enough for both the consumer and the provider points of view. Based on CP, our prototype handles the various consumer requirements, which are traditionally not considered. It can handle dynamically different types of side constraints and deal with any combinations of them. On the other hand, our selection model support changeable QoS criteria thus Bime can update in easy manner the QoS criteria exposed to their consumer. To add new QoS criterion dynamically such as the importance of geolocating Bime data in the Cloud, for example, it is simply sufficient to update the QoS XML file.

About implementation, thanks to CP, a constraint can be easily implemented in one line of code: the optimization algorithm is not modified.

D. Scalability

Our approach capability to cope and perform under an increase of the number of concrete service/resource than the number of SLA criteria are represented by Figure 6. Our approach is able to maintain its level of performance when tested in different configurations.

For experimentation values, we used values that are summarized in motivating scenario. In Figure 6, we show how the execution time vary when the number of concrete service/resource increases for the two SLA examples:

- SLA 1 (response time and availability) and
- SLA 2 (response time, availability and financial cost).

The execution time for both SLA example is slightly different. This can be explained by rewritting constraints strategy [6]. Overall, the performance results show that our approach is perfectly usable by Bime for service selection for a very large configuration, which is typically the case in Cloud environment.

V. Related work

A number of service selection approaches have been proposed recently [7][3][8][9][10]. However, these previous approaches have not considered user preferences in their QoS models. In contrast, our approach define weight values for each objective using user preferences. We present a flexible selection model that support SLAs on-the-fly.

In addition, computation time of service selection approaches becomes larger with the increase of cloud services. Real-time optimal cloud service selection becomes more and more difficult. Thanks to CP that reduces computation time by rewriting constraints [6], our approach is competitive for a large number of cloud services.

Finally, as the cloud services are composed together hierarchically in a *producer-consumer* manner, our approach focuses on the role of *cloud layer* n to ensure the selection at any level. Therefore, our approach is generic since with a simple "shift" (e.g., from n to $n-2$), our model can be used by Amazon (*cloud layer* n) to help their client (the consumer, e.g. Bime) to select the appropriate Amazon service and automatically determine the energy resources required for energy provider

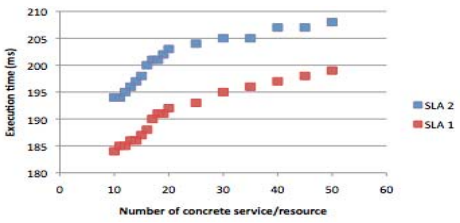

Fig. 6. Approach scalability

(the provider) according to various SLA "consumer-Amazon".

VI. Conclusion and future work

This paper proposes an innovative approach to address the SLA-aware cloud service selection problem: a constraint modelling approach that support dynamic QoS parameters in a flexible manner. Our selection model is driven by consumer preferences in order to compose the best *vertical business process*. Besides, it is a generic approach because it focuses on the role *cloud layer* n for any layer in the cloud stack.

Several extensions are planned as future work in particular define a mapping between a *vertical business process* and CP implementation using a transformation model. We are currently designing a SLA description language to improve the implementation of our selection method. Finally, we plan to identify strategies at run-time for managing requests/responses and services/resources to respect the SLAs.

Acknowledgement

This work is supported by the MyCloud project (ANR-10-SEGI-010).

References

[1] M. Hogan and al., NIST Cloud Computing Standards Roadmap, Version 1.0. 2011.
[2] H. Wada, P. Champrasert, J. Suzuki, and K. Oba. Multiobjective Optimization of SLA-aware Service Composition. IEEE Workshop on Methodologies for Non-functional Properties in Services Computing, 2008.
[3] N.B Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, V. Issarny. QoS-aware service composition in dynamic service oriented environments. Middleware Conference, 2009.
[4] G. Canfora, M. D. Penta, R. Esposito, and M. L. Villani. An Approach for QoS-aware Service Composition based on Genetic Algorithms. In Genetic and Evolutionary Computation Conference, 2005.
[5] R.Francesca, P. V. Beek and T. Walsh. Handbook of Constraint Programming (Foundations of Artificial Intelligence), 2006.
[6] P. V. Hentenryck and T. Le Provost. Incremental search in Constraint Logic Programming. New Generation Computing ,1991.
[7] T. Yu, Y. Zhang, K.-J. Lin: Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints, ACM Transactions on the Web, 2007.
[8] Kofler, Kevin and Haq, Irfan ul and Schikuta, Erich. A Parallel Branch and Bound Algorithm for Workflow QoS Optimization. Proceedings of the 2009 International Conference on Parallel Processing, 2009.
[9] H. Wada , J. Suzuki and K. Oba. Queuing Theoretic and Evolutionary Deployment Optimization with Probabilistic SLAs for Service Oriented Clouds, Proceedings of the 2009 Congress on Services - I, 2009.
[10] S. Wang and al. Cloud Model for Service Selection, IEEE INFOCOM 2011 Workshop on Cloud Computing, 2011.