SLA Aware Elastic Clouds

Jean Arnaud and Sara Bouchenak

Université Grenoble I
France
Sara.Bouchenak@imag.fr

ABSTRACT. Although Cloud Computing provides a means to support remote, on-demand access to a set of computing resources, its ad-hoc management for quality-of-service and SLA poses significant challenges to the performance, availability and economical costs of the cloud. This paper discusses these issues and presents early ideas to handle them. First, it introduces the SLAaaS model (SLA aware Service) that enriches the general paradigm of Cloud Computing. SLAaaS enables a systematic and transparent integration of service levels and SLA to the cloud. It is orthogonal to IaaS, PaaS and SaaS and may apply to any of them. Furthermore, the paper discusses autonomic SLA management in the cloud and presents early ideas to tackle it.

RÉSUMÉ. Le Cloud Computing est un modèle qui permet l'accès à la demande et à distance à un ensemble de ressources de calcul configurables. Cependant, la gestion ad-hoc de la qualité de service et du contrat de niveau de service (*Service Level Agreement* – SLA) dans le cloud soulève des problèmes de performance, de disponibilité, de consommation énergétique et de coûts économiques du cloud. Cet article présente un nouveau modèle de cloud appelé *SLAaaS* (*SLA aware Service*). Ce modèle permet d'intégrer la qualité de service et le contrat SLA comme éléments à part entière du cloud. Ainsi, le paradigme général de Cloud Computing sera enrichi avec le nouveau modèle SLAaaS. Ce dernier est orthogonal aux modèles IaaS, PaaS et SaaS et peut sŠappliquer à n'importe lequel dŠentre eux. L'article présente la gestion de SLA dans le cloud à travers une étude de cas menée dans un systèmes réel.

KEYWORDS : Cloud computing, SLA, Performance, Availability, Distributed systems

MOTS-CLÉS : Cloud computing, SLA, Performance, Disponibilité, Systèmes répartis

1. Introduction

Cloud Computing is a paradigm for enabling remote, on-demand access to a set of configurable computing resources. A Cloud may stand at different levels of the hardware and software stack where: (i) an Infrastructure as a Service (IaaS) cloud enables access to hardware resources such as servers and storage devices, (ii) a Platform as a Service (PaaS) cloud allows the access to software resources such as operating systems and software development environment, and (iii) a Software as a Service (SaaS) cloud is an alternative to classical software applications running locally on personal computers which are, instead, provided remotely by the cloud (e.g. messaging services, document editing services, etc.). A large number of Cloud Computing environments are proposed: (i) IaaS clouds such as Amazon S3 and AT&T Synaptic storage services [1, 2], Amazon SimpleDB and Microsoft SQL Azure database services [1, 15], or Amazon EC2 and AT&T Synaptic Compute computing services (i.e. servers) [1, 2]; (ii) PaaS clouds such as Microsoft Azure and Google AppEngine software developments environments [9, 15]; (iii) SaaS clouds such as document editing and communication services provided by Google Apps, Microsoft BPOS.

In this context where multiple clouds provide similar services (e.g. several IaaS clouds provide similar services to access computing resources), it is not easy for a consumer to compare the proposed cloud services and choose the most appropriate one for his needs. We believe that a differentiating element between Cloud Computing solutions will be the quality-of-service (QoS) and the service level agreement (SLA) guaranties provided by the cloud. Existing commercial cloud solutions include some kind of SLA, but express it with vague terms such as "small vs. large instances". Very few QoS aspects are considered in the cloud, for instance, no guaranties regarding performance are provided. Furthermore, the cloud does not automatically handle dynamic variations of cloud usage. Initiatives such as Amazon Auto Scaling help the customer to adapt the size of his cloud [4]. However, significant efforts from the customer are required for cloud capacity planning; this goes against one of the main motivations of Cloud Computing that is the ease of use of services by cloud customers. In summary, Cloud Computing faces the following open issues: (i) Need of integration of QoS and SLA requirements with the cloud; (ii) Automated dynamic elasticity of the cloud for SLA management. In order to address these issues, we call for: (i) a definition of a new cloud model that integrates SLA as part of the cloud, and (ii) an automated cloud control for building SLA-aware dynamic elastic clouds.

2. SLA-Aware Elastic Clouds

We call for a systematic and transparent integration of quality-of-service and service level agreement to the cloud. Quality-of-service (QoS) in the cloud may refer to several aspects such as performance (e.g. service response time, throughput), or availability (e.g. service abandon rate). Service level agreement (SLA) in the cloud is a contract between a cloud provider and a cloud customer. It specifies the levels of service that the cloud should provide to the customer in terms of objectives to attain for different QoS aspects. Another desirable objective is the reduction of cost of the cloud, i.e. its impact on the energy footprint and the economical costs.

We introduce a new model, the SLA aware Service (SLAaaS) model, to enrich the general paradigm of Cloud Computing. SLAaaS is orthogonal to IaaS, PaaS and SaaS

clouds and may apply to any of them. With SLAaaS, a cloud clearly exhibits its service levels and the proposed SLA. This enables a consumer who looks for a cloud service to transparently compare service levels of different cloud solutions before choosing the one that is best suited for his needs.

In order to guarantee the SLA of SLAaaS clouds, automated control for dynamic cloud elasticity should be provided. This aims to meet quality-of-service requirements such as performance and availability while minimizing cloud cost (i.e. energetic impact and economical costs). In the following, we identify and discuss three complementary research directions to provide SLA-aware dynamic elastic clouds.

Online observation and monitoring of the cloud. This aims to automatically capture variations in cloud usage and workload, to detect SLA violation and to trigger cloud reconfiguration when necessary. QoS measurements may apply at different levels to provide low-level metrics for IaaS clouds or higher application-level metrics for SaaS clouds. The main issue here consists in defining scalable, accurate and non-intrusive distributed algorithms for cloud monitoring.

Modeling the cloud. A cloud has a dynamic behavior with varying and nonlinear cloud service workloads. This has a direct impact on cloud QoS. A cloud is also characterized by its actual configuration, i.e. its size (number of cloud services), its location (machines hosting cloud services), and its service parameters (configuration parameters of individual cloud services). Obviously, the cloud configuration has an impact on both cloud QoS and cloud cost. Cloud modeling aims to render the impact of cloud workload and configuration on the QoS and cost of the cloud. The challenge here is to define a model that is accurate, capable of rendering the nonlinear variation of cloud workload, and that is easy to use with real world clouds (e.g. via automated and online tuning of model parameters). Control theory modeling techniques can be effectively applied here.

Automated control of the cloud. Cloud elasticity is the ability of the cloud to change its configuration, while dynamic cloud elasticity is the ability of the cloud to be elastic while the service is online. Thus, automated cloud control aims to build a dynamic elastic cloud (i.e. a new cloud configuration) that meets QoS requirements as specified in the SLA while minimizing cloud cost (energy footprint, economical costs). To do so, the use of a cloud model allows to reason about the variations of cloud configuration and workload and their impact on cloud QoS and cost. Mathematical optimization and control theory techniques can be effectively used in this context. First, the definition of an objective function allows to precisely quantify SLA QoS requirements vs. cost of a cloud configuration. The use of mathematical optimization techniques allow to determine, for a cloud workload, an optimal configuration, i.e. a cloud configuration that maximizes the objective function by guaranteeing SLA requirements while minimizing the cost. The definition of control laws describes how to automatically change the cloud configuration to an optimal configuration with respect to the objective function.

The challenge here is multifold: (i) the definition of scalable and optimal control algorithms for the cloud, (ii) the handling of different and sometimes antagonist QoS requirements for the cloud (e.g. performance vs. availability), (iii) the monitoring of the underlying distributed system tackling scalability and accuracy of the monitored data, and (iv) the proposal of techniques for online cloud reconfiguration such as online service (un-)provisioning (i.e. cloud rescaling), online redeployment (e.g. virtual machine migration in IaaS clouds), and online service's internal parameter reconfiguration (e.g. application server parameters in PaaS clouds).

3. Case Study: SLA-Aware Multi-Tier Internet Services

Internet services usually follow the classical client-server architecture where servers provide clients with some online service (e.g. online bookstore, e-banking service, etc.). A client remotely connects to the server, sends it a request, the server processes the request and builds a response that is returned to the client before the connection is closed. We consider synchronous communication systems, that is, when the client sends its request to the server, it blocks until it receives a response. Furthermore, for scalability purposes Internet services are built as multi-tier systems. A multi-tier system consists of a series of M server tiers $T_1, T_2, ..., T_M$. Client requests flow from the front-end tier T_1 to the middle-tier and so on until reaching the back-end tier T_M. Each tier is tasked with a specific role. For instance, the front-end web tier is responsible of serving web documents, and the back-end database tier is responsible of storing non-ephemeral data. Moreover, to face high loads and provide higher service scalability, a commonly used approach is the replication of servers in a set of machines. Here, a tier consists of a set of replicated services, and client requests are dynamically balanced between replicated services.

3.1. Service Level Agreement

SLA (Service Level Agreement) is a contract negotiated between clients and their service provider. It specifies service level objectives (SLOs) that the application must guarantee in the form of constraints on quality-of-service metrics, such as performance and availability. Client request latency and client request abandon rate are key metrics of interest for respectively quantifying the performance and availability of Internet services.

The latency of a client request is the necessary time for an Internet service to process that request. The average client request latency (or latency, for short) of an Internet service is denoted as ℓ. A low latency is a desirable behavior which reflects a reactive service.

The abandon rate of client requests is the ratio of requests that are rejected by an Internet service compared to the total number of requests issued by clients to that service. It is denoted as α. A low client request abandon rate (or abandon rate, for short) is a desirable behavior which reflects the level of availability of an Internet service.

Besides performance and availability, the cost of an Internet service refers to the economical and energetic costs of the service. Here, the cost ω is defined as the total number of servers that host an Internet service.

3.2. Service Configuration

The configuration κ of an Internet service is characterized in the following by a triplet $\kappa(M, AC, LC)$, where M is the fixed number of tiers of the multi-tier service, AC and LC are respectively the architectural configuration and local configuration of the Internet service that are detailed in the following.

The architectural configuration describes the distributed setting of a multi-tier Internet service in terms of the number of replicas at each tier. It is conceptualized as an array $AC < AC_1, AC_2, ..., AC_M >$, where AC_i is the number of replica servers at tier T_i of the multi-tier service.

The local configuration describes the local setting applied to servers of the multi-tier service. It is conceptualized as an array $LC < LC_1, LC_2, ..., LC_M >$. Here, LCi represents servers MPL (Multi-Programming Level) at tier T_i of the multi-tier service. The MPL is a configuration parameter of a server that fixes a limit for the maximum number of clients allowed to concurrently access the server (Ferguson, 1998). Above this

limit, incoming client requests are rejected. Thus, a client request arriving at a server either terminates successfully with a response to the client, or is rejected because of the server's MPL limit.

3.3. Service Workload

Service workload is characterized, on the one hand, by workload amount, and on the other hand, by workload mix. Workload amount is the number of clients that try to concurrently access a server; it is denoted as N. Workload mix, denoted as X, is the nature of requests made by clients and the way they interleave, e.g. read-only requests mix vs. read-write requests mix. There is no well established way to characterize the workload mix X of an Internet service.

Furthermore, service workload may vary over time, which corresponds to different client behaviors at different times. For instance, an e-mail service usually faces a higher workload amount in the morning than in the rest of the day. Workload variations have a direct impact on the quality-of-service as discussed later.

3.4. Adaptive Control of Internet Services

Both service workload and service configuration have an impact on the performance, availability and cost of services. The workload of Internet services is an exogenous input, which variation can not be controlled. Thus, to handle workload variations and provide guaranties on performance and availability, Internet services must be able to dynamically adapt their underlying configuration. Several objectives are targeted here:

– Guarantee SLA constraints in terms of service performance and availability, while minimizing the cost of the Internet service.

– Handle nonlinear behavior of Internet services taking into account both workload amount and in workload mix variations over time.

– Provide self-adaptive control of Internet services that provides online automatic reconfigurations of Internet services.

We propose MoKa, a nonlinear utility-aware control for self-adaptive Internet services. First, MoKa is based on a utility function that characterizes the optimality of the configuration of an Internet service in terms of SLA requirements for performance and availability, in conjunction with service cost. Second, a nonlinear model of Internet services is described to predict the performance, availability and cost of a service. Third, a capacity planning method is proposed to calculate the optimal configuration of the Internet service. Finally, an adaptive nonlinear control of Internet services is provided to automatically apply optimal configuration to online Internet services. MoKa is built as a feedback control of multi-tier Internet services , with three main elements: (i) online monitoring of the Internet service, (ii) adaptive control of the Internet service, and (iii) online reconfiguration of the Internet service.

Online monitoring aims at observing the Internet service and producing the necessary data in order to, on the one hand, automatically calibrate MoKa's model, and on the other hand, trigger MoKa's capacity planning and control. MoKa's model calibration is performed online and automatically. This allows rendering the dynamics of service workload mix and workload amount, without requiring human intervention and manual tuning of model parameters, which makes the model easier to use. Therefore, the controller calls the utility-aware capacity planning method to calculate the optimal configuration κ^* for the current workload amount and workload mix. That optimal configuration guarantees

the SLA performance and availability objectives while minimizing the cost of the Internet service. Finally, the new calculated configuration κ^* is applied to the Internet service. In the following, we briefly describe MoKa utility function, modeling, and capacity planning.

3.4.1. Service Utility Function

We consider an SLA of an Internet service that specifies service performance and availability constraints in the form of maximum latency ℓ_{max} and maximum abandon rate α_{max} not to exceed. Performability Preference (i.e. performance and availability preference) of an Internet service is defined as follows:

$PP(\ell, \alpha) = (\ell \leq \ell_{max}) \cdot (\alpha \leq \alpha_{max})$ (Eq. 1)

where ℓ and α are respectively the actual latency and abandon rate of the Internet service. Note that $\forall \ell, \forall \alpha, PP(\ell, \alpha) \in 0, 1$, depending on whether Eq. 1 holds or not.

Based on the performability preference and cost of an Internet service, the utility function of the service combines both criteria as follows:

$\theta(\ell, \alpha, \omega) = \frac{M \cdot PP(\ell, \alpha)}{\omega}$ (Eq. 2)

where ω is the actual cost (i.e. #servers) of the service, and M is the number of tiers of the multi-tier Internet service. M is used in Eq. 2 for normalization purposes. Here, $\forall \ell, \forall \alpha, \forall \omega, \theta(\ell, \alpha, \omega) \in [0, 1]$, since $\theta \geq M$ (at least one server per tier) and $PP(\ell, \alpha) \in \{0, 1\}$.

A high value of the utility function reflects the fact that, on the one hand, the Internet service guarantees service level objectives for performance and availability and, on the other hand, the cost underlying the service is low. In other words, an optimal configuration of an Internet service is the one that maximizes its utility function.

3.4.2. Service Modeling

The proposed analytic model predicts the latency, abandon rate and cost of an Internet service, for a given configuration κ of the Internet service, a given workload amount N and a given workload mix X. The model follows a queueing network approach, where a multi-tier system is modeled as an M/M/c/K queue. Moreover, Internet services are modeled as closed loops to reflect the synchronous communication model that underlies these services, that is a client waits for a request response before issuing another request.

3.4.3. Service Capacity Planning

The objective of the capacity planning is to calculate an optimal configuration of a multi-tier Internet service, for a given workload amount and workload mix, to fulfill the SLA in terms of latency and abandon rate constraints, and to minimize the cost of the service. Thus, an optimal configuration κ^* is a configuration that has the highest value of the utility function κ^*.

4. Related Work

The control of services to guarantee the SLA is a critical requirement for successful performance and availability management of cloud services. The management of service performance and availability is usually achieved by system administrators using ad-hoc tuning [4, 14]. However, new approaches tend to appear to ease the management of such systems. These approaches differ with regard to several criteria: tackling performance and/or availability objectives, handling service workload variations in terms of workload amount and/or workload mix, the used control techniques, and the applied control mecha-

nism (i.e. actuators). Different control mechanisms may be considered to manage service performance and availability, such as server provisioning, admission control, service differentiation, service degradation, and request scheduling [10]. In the following, we will focus on approaches using the two first techniques, namely admission control for a local configuration of the concurrency level of a server, and server provisioning for an architectural configuration of the size of a replicated distributed Internet service.

Admission control fixes the MPL concurrency level of a multi-programming system (e.g. multi-threaded servers). It has been applied to a web server [8], a database server [16], or a multi-tier system [13]. Some admission control solutions are proposed in the form of heuristics [11, 16], such as hill-climbing. These solutions have the advantage to be simple to implement; however, they provide a best-effort behavior without guarantees on the quality-of-service and SLA of the services.

Other approaches tend to provide strict guarantees on the quality-of-service, and are usually based on analytic models to characterize the system and control it. For instance, there are linear models and nonlinear models [7, 17, 18, 21], queuing theory-based models or control theory-based models [17, 7, 12], models for central systems or for distributed services [3, 19], used for providing guaranties on a unique QoS criterion or for combining multiple criteria [5, 13], applying a unique or multiple control mechanisms, i.e. actuators, [7, 16].

Other approaches control Internet services by provisioning/unprovisioning servers to the service. Autonomic provisioning of database servers is presented in [6], and server provisioning in multi-tier systems is described in [3]. While these systems are based on heuristics, other approaches tend to better characterize multi-tier applications through analytic modeling for provisioning multi-tier systems [20, 19]. However, these approaches are restricted to performance management and do not take into account service availability objectives. Furthermore, they require extensive model calibration with appropriate parameter values; and this calibration is tied to a given workload mix and must be changed each time the workload mix changes, which is not easily detectable.

5. Conclusion

This paper describes ideas for a systematic integration of SLA to the cloud through the definition of the SLAaaS cloud model, and research directions for an automated cloud control for building SLA-aware dynamic elastic clouds. The paper also presented the MoKa case study, a system for adaptive control of Internet services to guarantee performance and availability objectives and to minimize cost. The contribution of MoKa is multifold. First, a utility function is defined to quantify the performance, availability and cost of distributed Internet services. Second, a utility-aware capacity planning method is developed; given SLA performance and availability constraints, it calculates a configuration of the Internet service that guarantees the constraints while minimizing the cost of the service. Third, a queuing theory-based analytic model of multi-tier Internet services is proposed; the model accurately predicts service performance, availability and cost, and is used as a basis of the capacity planning. Finally, an adaptive control of online Internet services is proposed in the form of a feedback control loop that automatically detects workload mix and workload amount variation, and reconfigures the service with its optimal configuration.

6. Acknowledgements

This work has been partially supported by the French National Agency (ANR) in the frame of its Technological Research ARPEGE program. (MyCloud project)

References

[1] Amazon. Amazon Web Services. http://aws.amazon.com/.

[2] AT&T. AT&T Synaptic Storage as a Service. https://www.synaptic.att.com/.

[3] S. Bouchenak, N. De Palma, D. Hagimont, and C. Taton. Autonomic Management of Clustered Applications. In *IEEE International Conference on Cluster Computing (Cluster 2006)*, Barcelona, Spain, September 2006.

[4] Martin Brown. Optimizing Apache Server Performance, February 2008. http://www.serverwatch.com/tutorials/article.php/3436911.

[5] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin Vahdat, and Ronald P. Doyle. Managing energy and server resources in hosting centres. In *Symposium on Operating Systems Principles*, pages 103–116, 2001.

[6] J. Chen, G. Soundararajan, and C. Amza. Autonomic provisioning of backend databases in dynamic content web servers. In *The 3rd IEEE International Conference on Autonomic Computing (ICAC 2006)*, Dublin, Ireland, June 2006.

[7] Yixin Diao, N. Gandhi, J.L. Hellerstein, S. Parekh, and D.M. Tilbury. Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. *Network Operations and Management Symposium*, 2002.

[8] Sameh Elnikety, Erich Nahum, John Tracey, and Willy Zwaenepoel. A Method for Transparent Admission Control and Request Scheduling in E-Commerce Web Sites. In *13th international conference on World Wide Web*, New York, NY, May 2004.

[9] Google. Google App Engine. http://code.google.com/intl/fr/appengine/.

[10] Jordi Guitart, Jordi Torres, and Eduard Ayguadé. A survey on performance management for internet applications. *Concurr. Comput. : Pract. Exper.*, 22(1):68–106, 2010.

[11] Hans-Ulrich Heiss and Roger Wagner. Adaptive Load Control in Transaction Processing Systems. In *17th International Conference on Very Large Data Bases*, San Francisco, CA, 1991.

[12] Luc Malrait, Sara Bouchenak, and Nicolas Marchand. Fluid Modeling and Control for Server System Performance and Availability. In *39th Annual IEEE International Conference on Dependable Systems and Networks (DSN 2009)*, Estoril, Lisbon, Portugal, June 2009.

[13] D. A. Menascé, D. Barbara, and R. Dodge. Preserving QoS of E-Commerce Sites Through Self-Tuning: A Performance Model Approach. In *ACM Conference on Electronic Commerce*, Tampa, FL, October 2001.

[14] Microsoft. Optimizing Database Performance. http://msdn.microsoft.com/en-us/library/aa273605(SQL.80).aspx.

[15] Microsoft. Windows Azure Platform. http://www.microsoft.com/windowsazure/.

[16] J.M. Milan-Franco, Ricardo Jimenez-Peris, Marta Patino-Martinez, and Bettina Kemme. Adaptive middleware for data replication. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 175–194, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[17] S. Parekh and N. Gandhi and J. Hellerstein and D. Tilbury and T. Jayram and J. Bigus. Using Control Theory to Achieve Service Level Objectives In Performance Management. *Real-Time Syst.*, 23(1-2):127–141, 2002.

[18] D. Tipper and M.K. Sundareshan. Numerical methods for modeling computer networks under nonstationary conditions. *IEEE Journal on Selected Areas in Communications*, 8(9):1682–1695, December 1990.

[19] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. Analytic modeling of multitier internet applications. *ACM Transactions on the Web (ACM TWEB)*, 1(1):2, 2007.

[20] D. Villela, P. Pradhan, and D. Rubenstein. Provisioning servers in the application tier for e-commerce systems. *ACM Trans. Interet Technol.*, 7(1):7, 2007.

[21] Wei-Ping Wang, D. Tipper, and S. Banerjee. A simple approximation for modeling nonstationary queues. *IEEE INFOCOM*, March 1996.