Measurement of Cloud Computing Services Availability

Jakub Pavlik, Vladimir Sobeslav(✉), and Ales Komarek

Department of Information Technologies, Faculty of Informatics and Management, University of Hradec Kralove, Rokitanskeho 62, 500 03 Hradec Kralove, Czech Republic
{Jakub.pavlik,vladimir.sobeslav,ales.komarek}@uhk.cz

Abstract. Recently we are witnessing the engagement of cloud computing services such as emails, web services, mobile application, sharing data-stores and many others. Huge number of companies, customers and public institutions are considering the migration to the cloud services. The topical questions behind this effort is the efficiency and measurement of the QoS – Quality of Services of the cloud computing utilisation. This paper is focused on the problematic of measuring and monitoring service availability in Cloud Computing. It deals with the Service-Level Agreement (SLA) monitoring approaches and frameworks. Furthermore it presents a new approach of the cloud service availability monitoring from the client-centric perspective. On the basis of the client-centric approach a new solution was designed, implemented and tested on a sample cloud environment.

Keywords: QoS · SLA · Cloud Computing · Service availability · SLA monitoring

1 Introduction

Cloud Computing creates a new trend in which companies buy IT resources, platforms, and applications as a service. This approach provides multiple economic, technological and functional benefits. But, these are accompanied by new threats, problems and challenges such as security issues, quality of service definition and measuring, responsibility between related parties, service availability, etc. Cloud computing also promises to provide high quality and on-demand services. However, cloud services usually come with various levels of services and performance characteristics which complicate the chances for precise classifications [2]. As shown in Fig. 1, Cloud Computing distinguishes between three basic service models. Cloud service can be the end user software (SaaS), a platform especially used by developers (PaaS), or an infrastructure itself (IaaS) [3]. The goal of Cloud Computing services is to consolidate and optimize existing software and hardware resources and provide automated, on-demand, service-oriented solution with broad network access [4].

Success of these cloud services depends on the required functionality and other characteristics such as availability, respond time, latency, performance, timeliness, scalability, high availability, trust, security, etc. All of these characteristics can be

DOI: 10.1007/978-3-319-15392-6_19

covered by the term Quality of Cloud Service (QoCS) which comes from general QoS [5]. QoCS parameters are non-functional definition and properties of cloud service. Hence, it is difficult to assure the accurate evaluation and measuring of QoCS. On the one hand, it solves requirements such as security and trust which are very difficult to evaluate, and on the other hand it resolves the reliability, availability, and performance characteristics. Another problem is the great variety of different cloud providers, as well as the variety of cloud services. Here come the questions of how to evaluate the QoCS and how to ensure its monitoring and compliance.

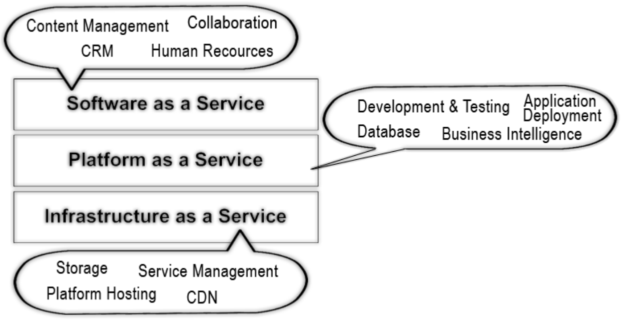

Fig. 1. Cloud services

In practice, QoCS are technical parameters of provided services which are contained and formalized into Service Level Agreements (SLA). SLAs are part of service contracts and are usually agreements between two parties (service provider and customer) which formally define the services. Service contracts use the percentage of service availability as a unit. In practice, today SLAs are not sufficiently accurate and need to be proved and measured. Furthermore, they do not provide guarantees for the availability of services. Rather, the whole process is based on customers' claims of outages or downtime incidents sent to the provider [6]. This paper is a part of a bigger project where the overall goal of the project is to define some nonfunctional aspects, using the Web Ontology Language OWL [7] and focusing on the automatic or semi-automatic generation of cloud service agreements by using ontologies including monitoring, measuring and compliance checking for SLAs. This paper defines a new approach for monitoring, measuring and compliance validation of SLA in Cloud Computing from Client-Centric point of view.

2 Analysis of Existing SLA Monitoring Solutions

This section focuses on SLA in terms of its monitoring and measuring, because only given availability percentage nines (e.g. 99.99%) are not enough for a quality of the cloud service. The goal is also to provide a view on the standard and current SLA monitoring approaches, frameworks or languages which is the cornerstone for designing the future of cloud availability monitoring tools. As [8] describes, the best-known projects for SLA specifications include: RBSLA, SLAng, SLA@SOI, and WSLA. Each of them are briefly specified in the following subsections.¨

2.1 RBSLA

Rule based SLA is a rule based approach to SLA representation and management which allows separating the contractual business logic from the application logic and enables automated execution and monitoring SLA's specifications. The key features of this concept are: good integration of external data or systems; ECA rules including monitoring intervals, active event monitoring (measurement) functions and executable actions [10]. The whole concept of Rule-based Service Level Management is being built with a computational model based on the ContractLog and the open source rule engine Prova [9].

2.2 SLAng

In addition to RBSLA, SLAng is a language for defining Service Level Agreements that cover needs for Quality of Service [11]. SLAng provides the format for definitions of QoS, responsibility between parties, and language appropriate for automated reasoning systems. The SLAng syntax is obviously an XML schema, specifically a combination of WSDL and BPEL. This approach is based on the Service Provision Reference Model. The nodes are architecture components and edges depict possibilities for SLA between two parties. The structure is divided into three parts: Application tier, Middle tier, and Underlying resources. Thus, the SLA classification distinguishes between Horizontal (different parties with same service) and Vertical SLA (parties on different levels of service) [11].

2.3 SLA@SOI

The SLA@SOI is the largest project related to the SLA field sponsored by leading Industrial, Academic and Research Institutes from around Europe. SLA@SOI "created a holistic view for the management of service level agreements (SLAs) and provides an SLA management framework that can be easily integrated into a service-oriented infrastructure" [12]. This approach is not only about definition of SLA, their measurement, and results, it provides a complex view on the whole area of business. It includes requirements and functions like: predictability and dependability of all components in the processes; holistic SLA management gaining the transparent IT; automated negotiation of SLA between parties.

2.4 WSLA

Last approach of existing SLA concepts is Web service SLA. Even though this is the oldest one, all other approaches mentioned above are based on it [12] [11] [8]. Considering [8], we decided that the best is to thoroughly describe this approach and build our solution on it. Software & Hardware solutions and their service availability have usually a set of specific requirements for availability, reliability, etc. For this purpose, the Web Service Level Agreement (WSLA) language was created and is used especially for the Web Services domains as SOAP, which "defines assertions of a service provider to perform a service according to agreed guarantees for IT-level and business process-level service parameters such as response time and throughput, and measures to be taken in case of deviation and failure to meet the asserted service guarantees, for example, a notification of the service customer." [13]. Result of WSLA is an XML schema forming an abstract language for implementation of the whole SLA management on both consumer and provider side. Detailed description of the SLA and its structure is not the subject of this paper.

3 Client-Centric Solution

Based on our research, cloud service availability measuring and analysis should be done from the consumer's perspective (the client-centric approach), automatically and independently of provider. The following important issues and requirements can be concluded:

- **Client-centric solution** – should be independent of the provider's infrastructure. Provider's monitoring systems or probes located in their infrastructure should not be used due to inaccuracy of results.
- **Automated solution** – fully automated or semi-automated in order to enable flexible response to changes in cloud services.
- **General and simple solution** – should be standardized and applicable for any kind of cloud service such as IaaS, PaaS, or SaaS.

All of these requirements aim to create an agent software for the consumer's machine. We propose three basic approaches or solution designs. Each of them uses BlackBox detection and monitoring systems. The detection discovers consumer requests for the cloud service and the monitoring gathers data sets (success or fail and latency). The representation of availability is evaluated by using goodness-of-fit tests analysis (probability distribution) in the next steps. As shown in Figure 2, the first solution puts the BlackBox detection and monitoring system in one location between the consumer machine and the cloud service as a kind of proxy site. The goal is to analyze direct user's request to the service to decide about availability.

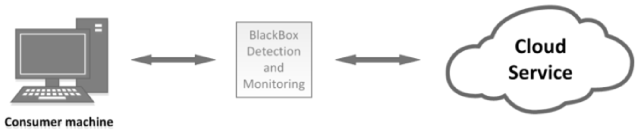

Fig. 2. Solution Approach I

The second approach places the BlackBox on a consumer side, more precisely to each consumer end machine. The consumer environment is affected by the implementation agent which detects cloud service requests and gathers data about them. Data are gathered only during consumer communication with cloud service, which actually does not provide accurate service availability, because the consumer only uses the service when he/she needs it. The goal is to provide precise SLA for the whole period of cloud service offering. Another issue is related to service unavailability caused by failure on the consumer side, e.g. internet outage on the consumer side. This means that neither of the mentioned approaches is optimal.

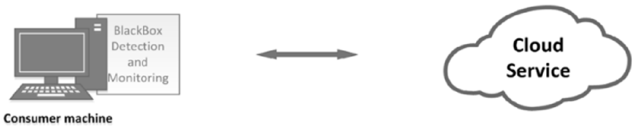

Fig. 3. Solution Approach II

The last approach tries to resolve disadvantages of the previous two approaches. This means to separate BlackBox detection and monitoring into different locations. The detection is done on the consumer's machine by an installed agent which analyzes cloud service requests and provides materials for configuration of remote monitoring. The detection can be carried out by a machine learning process together with network analysis.

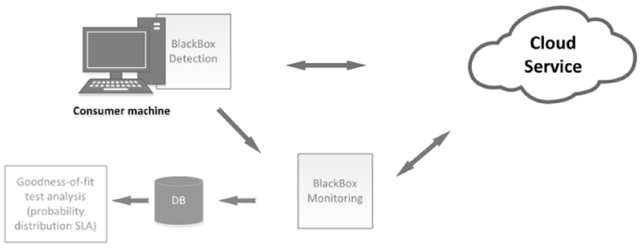

Fig. 4. Solution Approach III

4 Monitoring Solution

Generally, there are two ways how to monitor systems, applications, software, hardware, etc. - Agent-based and Agentless. An Agent-based approach uses a software lite monitoring agents installed directly inside the target monitored system, where specific data is collected and sent to the monitoring server side. In contrast, Agentless approach monitors the systems remotely from the server side, which means that the agents are installed on the monitoring server and not on the target system. An Agent-based approach has more capabilities and power than agentless monitoring solutions and allows more control of the target monitoring configuration. As Figure 4 shows, our solution should work as the Blackbox remote monitoring server. Therefore, Agentless approach is the first requirement which must be fulfilled.

In addition to the Agentless criteria, the monitoring system must be open, flexible, standard and easy to deploy withing enterprise-class. A survey was conducted (with four big customers in the Czech Republic) among IT companies and it was found that the most widely used open source tool is Nagios [14]. Nagios is the Industry Standard in IT Infrastructure Monitoring and it is provided in commercial or open source versions. Its biggest advantage is the large number of existing monitoring plugins that allow virtual monitoring of many entities. The last requirement is the web graphical interface enabling scalable real-time graphing of collected data. This function provides wide range of options to display data in time.

As mentioned above, Nagios meets the requirements of the first two paragraphs, but for graphical function a commercial license is needed and the output is more suitable for classic static infrastructure monitoring than for the purpose in question. Therefore, it was decided to use the open-source project Graphite [15] for scalable real-time graphing and replace Nagios by the open-source monitoring framework Sensu [16], which can reuse existing Nagios plugins. The reason for choosing Sensu instead of Nagios was primarily due to the absolute openness and simplicity. Figure Fig. 5 shows the resulting monitoring architecture where the two above mentioned open-source projects were joined (beyond the standard projects like Apache or RabitMQ).

As mentioned above, the cornerstones of our solution are Sensu and Graphite. The monitoring architecture (Fig. 5) contains following components:

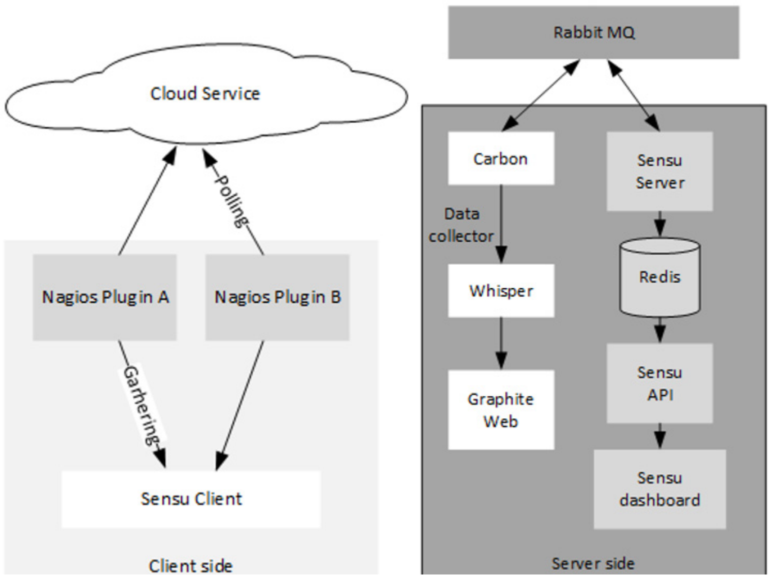

Fig. 5. Monitoring Architecture

Client side - monitoring agents located inside the target system, application, etc. These measure, poll and collect data and send them into monitoring server core. In the current case it is located on the same side (physical or virtual machine) as Server side, because the Agentless approach is used

Sensu server - is responsible for orchestrating check executions, the process of checking results, and event handling. It stores last status of check with details and launches event notifications, which are kept there until resolved. In this solution it is used primarily for scheduling execution and real-time verification of cloud service.

5 Statistical Data Analysis

The main idea of the solution is to be simple, easy to implement, and automatic or semiautomatic. Therefore, all the monitoring checks generate similar data sets. Each of them creates two data types: Availability data - discrete data that can only take certain values. In this case it is value "1" or "0", success or failure. Latency data - continuous data that can take any value within a specific range. In this case it is the response time or latency of our monitoring check given in UNIX time stamp (nanoseconds).

Availability data is almost the same for any cloud service, unlike latency for which response time can be completely different and the similarity among cloud services is almost never the same. The next difference is that the Availability data is always present (except failure of monitoring server), but the Latency data is only present if availability is "1". Otherwise, the latency is useless. This means that the latency analysis is valuable only if the cloud service is available.

We estimated types of probability distribution function using descriptive statistic and graphical methods. Descriptive statistics consists of calculating parameters such as Mean, Median, Maximum, Minimum, Standard Deviation, Variance, Summary, Skewness, and Kurtosis. This led to the finding that availability data distribution is almost always skewed towards the left, which confirms the statement that the median is greater than the mean.

After that several graphical techniques (histograms, empirical cumulative distribution function, Q-Q plot, and density plot) for data analysis were used. These can help to identify the kind of pdf to use to fit the model. They were compared with other theoretical discrete distributions, such as Poisson, Binomial, and others.

Based on the foregoing, it was concluded that the availability of data can be represented by Binomial or Poisson distributions. Both are very similar. Due to the large number of values (N), Binomial distribution looks like a Poisson with the same mean [23]. Poisson distribution is used in the case when p is very small or N is very large. The hypergeometric distribution was also considered, but it was rejected due to the fact that it describes the probability of k successes in n draws without replacement. It is a contrast to Binomial which is with replacement. The availability data is with replacement, because both values can appear more than one time.

This statement is supported by mathematical goodness of fit test using Chi-square testing. Chi-square test is based on comparing empirical frequencies with theoretical frequencies. Theoretical distribution was estimated using the maximum likelihood estimate (MLE), which is naturally implemented in R language. The chi-square test is defined for the hypothesis:

H_0: the data follows the specified distribution
H_A: the data does not follow the specified distribution
The basic calculation formula is defined as:

$$X^2 = \sum_{i=1}^{k} \frac{O_i - E_i^2}{E_i}$$

where O_i indicates the observed frequency of the number (e.g. number of 0 value) and E_i is the expected frequency of pdf. For any X^2 test, the number of degrees of freedom is given as *k-p-1*. *p* is the parameter estimated from the sample data. The hypothesis H_0 is accepted if X^2 is lower than the chi-square percent point function with degrees of freedom and a significance level of α. Each time it is assumed that the significance level is 5%. Asample plot of goodness of fit summary (Fig. 6) can be also drawn. X-axis displays the number of occurrences and Y-axis shows the squared frequencies.

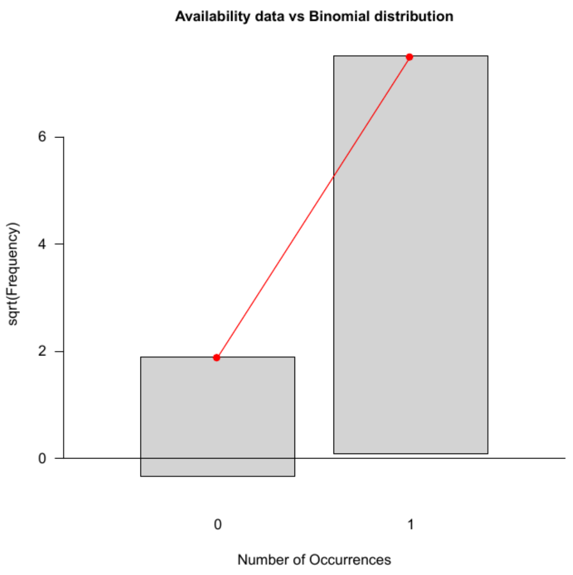

Fig. 6. Availability data vs Binomial distribution

The availability data was described by basic tests, then the pdf and estimated parameters of the model were suggested. The goodness of fit tests were performed and it was decided that the data comes from Binomial distribution. When the probability distribution is known, a specific statistical analysis and tests can be applied. The selection of these tests is based on source [24] which deals with the modeling of binary data.

Correlation or linear regression cannot be conducted to one statistic variable, which is caused by storing latency only in case of successful check. The linear dependency between statistics variables cannot be measured when the availability is zero, due the latency is missing. If H_A is rejected, then the statistical analysis of latency data is run to guarantee a defined cloud service quality. Otherwise, the quality measurement is pointless.

In general, statisticians test if data is normally distributed, because most of the specific tests (t test, z test, F test, ANOVA) assume the normality of data. That is why it was decided to replace the goodness of fit tests with normality tests. Another reason for this was the excessive existing probability distribution (Lognormal, Gamma, Weibull, Exponential, etc.), which would not allow the future automation of the entire solution, because each monitoring check may follow different distribution and it is not possible to cover all existing cases.

At first the graphical evaluation of normality Q-Q was tested. It was a scatter plot comparing the fitted and empirical distribution (availability data) in terms of the dimensional values of the variable. If the data is obtained from normal population, the points should fall approximately along the reference line. Fig. 7 shows that data is not normally distributed.

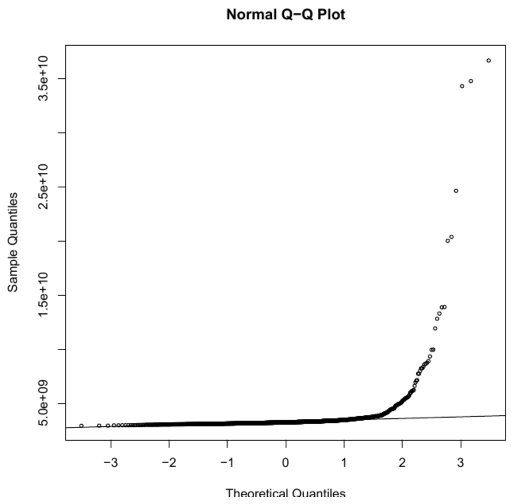

Fig. 7. Normal Q-Q Plot

6 Conclusions

The basic statistical analysis which is automated was conducted and it provided the automatic SLA compliance check for specific time period. The difference between time periods validation was not analyzed and the time series, which can be used for predictive analysis were also not considered. This procedure is simple, but meets all the requirements for the service availability and SLA. Due to this simplicity, it can be easily developed and implemented in future objectives as user request detection tools.

This research shows that the suggested client-centric approach is applicable for deployment in real environment. The questions of measuring and monitoring availability of cloud services remotely were answered and the experiments to find out, what the probability distribution data follows, were conducted. This helps to create very accurate results and moves the availability of cloud services further than just like the number of “nines” usually specified in the SLA document.

Acknowledgment. This work was supported by the project of specific research no. 2103. Faculty of Informatics and Management, University of Hradec Kralove.

References

[1] Gartner.com. Gartner it glossary - cloud computing (February 2014). http://www.gartner.com/it-glossary/cloud-computing/

[2] Wang, S., Liu, Z., Sun, Q., Zou, H., Yang, F.: Towards an accurate evaluation of quality of cloud service in service-oriented cloud computing. Journal of Intelligent Manufacturing, 1–9 (2012)

[3] Furht, B., Escalante, A.: Handbook of cloud computing. Springer, New York (2010)

[4] Mell, P., Grance, T.: The NIST definition of cloud computing. NIST Special Publication **800**(145), 7 (2011)

[5] Stantchev, V.: Performance evaluation of cloud computing offerings, pp. 187–192 (2009)

[6] Bauer, E., Adams, R.: Reliability and availability of cloud computing. Wiley-IEEE Press, Hoboken, N.J. (2012)

[7] Rady, M.: Formal Definition of Service Availability in Cloud Computing Using OWL. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) Computer Aided Systems Theory - EUROCAST 2013. Lecture Notes in Computer Science, vol. 8111, pp. 189–194. Springer, Heidelberg (2013)

[8] Nie, G.E.X., Chen, D.: Research on Service Level Agreement in Cloud Computing, pp. 39–43 (2012). 10.1007/978-3-642-28744-2_5

[9] Paschke, A.: rbsla - RBSLA: Rule Based Service Level Agreements Project. http://ibis.in.tum.de/projects/rbsla/ (accessed: March 19, 2014)

[10] Paschke, A., Bichler, M., Dietrich, J.: ContractLog: An Approach to Rule Based Monitoring and Execution of Service Level Agreements, pp. 209–217 (2005). 10.1007/11580072_19

[11] Lamanna, D.D., Skene, J., Emmerich, W.: SLAng: A Language for Defining Service Level Agreements, p. 100 (2003). http://dl.acm.org/citation.cfm?id=797134 (accessed: March 19, 2014)

[12] Wieder, P.: Service level agreements for cloud computing. Springer, New York (2011)

[13] Ludwig, H., Franck, R.: Web Service Level Agreement (WSLA) Language Specification (2003)

[14] Nagios.com.: Nagios - The Industry Standard in IT Infrastructure Monitoring and Alerting. http://www.nagios.com/ (accessed: March 19, 2014)

[15] Graphite.wikidot.com.: Graphite - Scalable Realtime Graphing - Graphite. http://graphite.wikidot.com/ (accessed: March 19, 2014)

[16] Heavy Water Operations, L. Sensu | An open source monitoring framework. http://sensuapp.org/ (accessed: March 19, 2014)

[17] Redis.io. Redis. http://redis.io/ (accessed: March 19, 2014)

[18] Rabbitmq.com. RabbitMQ - Messaging that just works. http://www.rabbitmq.com/ (accessed: March 19, 2014)

[19] Graphite.readthedocs.org. The Render URL API — Graphite 0.10.0 documentation. https://graphite.readthedocs.org/en/latest/render_api.html (accessed: March 19, 2014)
[20] Karian, Z.A., Dudewicz, E.J.: Handbook of Fitting Statistical Distributions with R. CRC Press, Boca Raton (2011)
[21] Ricci, V.: Fitting distribution with R (2005)
[22] Teaching, C.: Goodness of fit tests (2010)
[23] Collins, J.C.: Binomial Distribution: Hypothesis Testing, Confidence Intervals (CI), and Reliability with Implementation in S-PLUS (2010)
[24] Collett, D.: Modelling Binary Data. Chapman & Hall/CRC, Boca Raton (2003)
[25] Teaching, C.: Hypotesis testing: One sample test (2010)
[26] Lowry, R.: Concepts and Applications of Inferential Statistics (1998)
[27] Fee, K.: Delivering E-Learning: A Complete Strategy for Design, Application and Assessment, 4th Edn. Kogan Page, ISBN:978-0749453978 (2009)