Verifying the Availability of Cloud Applications

Melanie Siebenhaar, Olga Wenge, Ronny Hans, Hasan Tercan and Ralf Steinmetz
Multimedia Communications Lab (KOM), Technische Universität Darmstadt, Darmstadt, Germany

Keywords: Cloud Computing, Service Level Agreements, Verification, Monitoring, Performance, Availability.

Abstract: Cloud-based services provide a high level of flexibility and eliminate large up-front IT investments by trading capital expenditure for operational expenditure. However, performance, availability, and security still remain dominant barriers when deciding whether to move to the cloud or not. Although cloud providers already try to tackle these issues by offering SLAs and corresponding monitoring solutions, the ability of these solutions to control the performance of cloud-based services is still considered as unsatisfactory by consumers. In this paper, we present an approach for verifying availability guarantees from a consumer's perspective, since availability is one of the very few performance parameters that is considered in the SLAs of today's cloud providers. The aim of our research is to facilitate the verification of performance guarantees independently from a cloud provider, which will help to increase cloud service adoption in the future.

1 INTRODUCTION

Cloud computing promises to provide a high level of flexibility when using cloud-based services. Highly configurable computing resources are provided on-demand and with minimal management effort over the Internet (Mell and Grance, 2011) similar to utilities like electricity or water (Buyya et al., 2009). However, this also includes a shift of responsibility to the cloud provider and thus, a loss of control for the cloud consumer. In order for a cloud consumer to still maintain control, cloud providers typically offer so-called service level agreements (SLAs). Basically, a service level agreement represents a contract between a cloud provider and a cloud consumer and specifies certain quality levels a cloud provider is willing to provide (e.g., minimum values for performance parameters such as availability) and the penalties in case of violating the specified guarantees. However, this solution does not seem to be sufficient. According to a cloud market maturity study conducted by the Cloud Security Alliance and ISACA in the second quarter of 2012, there is only a low degree of confidence on consumer side, that providers effectively monitor performance against SLAs (CSA and ISACA, 2012). (Patel et al., 2009) also state that consumers may not completely trust these measurements and that cloud providers often put the burden of reporting SLA violations on their customers. Although cloud providers often implement particular monitoring solutions and provide some monitoring information to their customers, these solutions cannot be perceived to provide a sufficient and independent evidence base for reliably detecting and documenting SLA violations from a consumer's perspective. When solely relying on provider-specific monitoring solutions, cloud providers can modify some monitoring data or restrict the provided information so that it becomes very hard to prove SLA violations. This raises the question how compliance with SLAs can be verified from a consumer's perspective. Such a solution not only requires to obtain reliable data of a cloud-based service, but also requires to provide a holistic view of the end-to-end performance of a cloud-based service to consumers.

In this paper, we present such an approach for verifying the availability of cloud applications from a consumer's perspective, since availability is one of the very few performance parameters that are part of the SLAs of today's cloud providers. The remainder of the paper is structured as follows. Section 2 discusses related approaches to our work. Section 3 describes the current SLA landscape, introduces some basic information about availability and presents a taxonomy for downtimes of cloud applications. Section 4 describes our approach for availability verification and Section 5 presents the corresponding prototypical implementation as well as some experimental results. The paper closes with a conclusion and future directions in Section 6.

2 RELATED WORK

Although several monitoring approaches in the field of cloud computing have been proposed so far, only a few approaches exist which address the problem of SLA verification from a consumer's perspective.

(Chazalet, 2010) presents a generic framework that does not depend on a specific cloud service model. However, the framework focuses either on server-side or client-side monitoring.

In (Haberkorn and Trivedi, 2007), the authors present a generic approach for monitoring high-availability systems that consist of different components. Again, no client-side monitoring is considered.

A performance model based on runtime monitoring data is suggested by (Shao and Wang, 2011). Availability is calculated based on the number of successful requests. Hence, a sufficient number of requests is required in order to obtain accurate results.

(Michlmayr et al., 2009) apply client-side and server-side monitoring for SLA violation detection. It is not clear, how the authors combine the results from both monitors to determine the overall performance. Again, availability is only calculated based on the number of successful requests.

(Mastelic et al., 2012) present a generic approach for monitoring application level metrics in resource-shared cloud environments. Availability and client-side monitoring are not part of their work.

In contrast, our approach allows to verify availability from a consumer's perspective and to achieve visibility of the entire cloud service delivery chain.

3 AVAILABILITY OF CLOUD APPLICATIONS

3.1 SLA Landscape and Availability

In comparison to, e.g., Web services, cloud services exhibit a higher complexity due to the three different service models (Mell and Grance, 2011), software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS), and the fact that cloud applications on the upper SaaS layer often comprise several different components. The complexity further increases due to the utilization of virtualization so that surrounding conditions may change in the background without being noticed by consumers. Unfortunately, current cloud SLAs do not completely cover this new inherent complexity. There is often a gap between lower-level monitoring data collected by providers and higher-level guarantees provided to consumers (CSCC, 2012). Hence, besides the demand for more specific SLAs, also corresponding means for monitoring higher-level metrics of cloud-based services must be provided to consumers.

The paper at hand focuses on the availability of cloud applications, since availability is business critical and one of the very few performance guarantees that are currently offered by cloud providers (e.g., Elastic Compute Cloud (EC2) by (Amazon, 2008)).

In order to develop an availability monitoring approach, availability must be defined in a cloud computing context. (Jain, 1991) basically defines the availability of a system as "the fraction of the time the system is available to service users' requests". He further states that it is often more reasonable to use the mean uptime (MTTF), because small uptime and downtime combinations may result in high-availability values although the service cannot be delivered. Using MTTF and MTTR as the mean downtime results in the following formula:

$$availability = \frac{MTTF}{MTTF + MTTR} \tag{1}$$

Since errors on different layers or failures of components can lead to downtimes of the cloud application, different types of availability must be combined in order to measure the overall availability. Hence, the reasons for downtimes must be analyzed first in order to derive an appropriate definition.

3.2 Reasons for Downtimes

In order to successfully invoke a specific functionality, in the following denoted as service, provided by a cloud application, consumer's require working IT systems on-premise and a proper network connectivity to the cloud provider. Furthermore, due to the variety of resources involved in service delivery, many reasons for downtimes exist. In the following, we present a taxonomy for downtimes of cloud applications from a consumer's perspective (Figure 1).

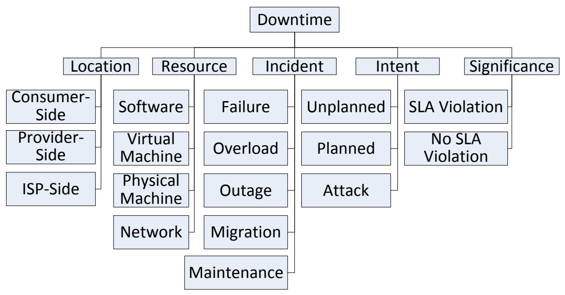

Figure 1: Taxonomy for downtimes of cloud applications.

First of all, downtimes can be distinguished according to the location, where incidents happen. Therefore, we describe their location according to the following spheres of control: consumer, Internet service provider (ISP), and cloud provider. Furthermore, different types of resources can be responsible for causing downtimes due to several incidents. Hence, we added the two categories resource and incident to our taxonomy. Besides the actual downtimes, also their pre- and postconditions must be taken into account in order to determine if an SLA violation occurred.

Basically, downtimes can happen with or without intent, and even with criminal intent when a system is under attack. Depending on the negotiated terms, all three types of intent can either be covered by SLAs or not. For example, SLAs can specify a maximum length of time for planned downtimes such as scheduled maintenance, unplanned downtimes can either refer to failures of physical machines or to emergency maintenance, and even attacks can happen due to negligence of consumers or due to an insufficient amount of implemented security mechanisms by providers. Therefore, downtimes must always be considered in conjunction with the negotiated SLAs. Moreover, our solution must be able to attribute incidents to their root cause, since cloud providers can only make guarantees with respect to their own IT systems.

4 A HYBRID APPROACH FOR AVAILABILITY VERIFICATION

We will now elaborate on how to use the knowledge about downtimes presented before in order to develop an approach for verifying the availability of cloud applications from a consumer's perspective. In this paper, we will focus on unplanned downtime only, since planned downtimes will be usually announced by cloud providers in advance and the detection of attacks is not in the scope of our work. The next section presents the requirements for such an approach.

4.1 Requirements

First of all, our approach should be able to detect all relevant downtimes precisely without affecting the overall performance and should be still applicable when the number of users and components changes. An SLA violation is considered to be relevant either if the duration of a single downtime or the aggregation of several downtimes may exceed the threshold defined in the respective SLA. Furthermore, our solution should be able to compute the overall availability. Finally, a trusted third party could generally provide the components of our monitoring approach to consumers in order to ensure reliability.

4.2 Design

Now, considering all the requirements stated above, we propose a hybrid monitoring approach that combines consumer- and cloud-side monitoring. In addition, we make use of a broker acting as a coordinating entity that collects and aggregates all data (Figure 2).

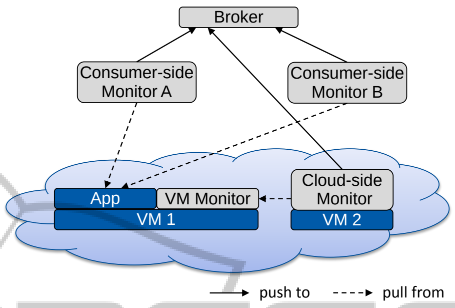

Figure 2: Overview of the monitoring framework.

Consumer-side Monitoring. The consumer-side monitor invokes predefined services (e.g., specified in the SLAs) that are essential for the proper functioning of a cloud application using a periodical pull model. The monitoring frequency can be adapted to the required resolution of an application's availability. We consider a cloud application to be unavailable if one of the essential services fails (i.e., no/incorrect response). In order to determine the overall availability, the consumer-side monitor communicates with the broker using an event-based push approach. Whenever a consumer-side monitor is started or the start or the end of a downtime is detected, a message is sent to the broker. Failures in the network could also prevent consumers from detecting all downtimes. Hence, we assume that an enterprise uses two different monitors at different network domains. Since a high monitoring frequency (e.g., in case of many consumers) will affect system performance, additional cloud-side monitoring must be considered.

Cloud-side Monitoring. For our approach we assume that a consumer has access to the VM where a cloud application is hosted. Since a monitor placed on this VM 1 would not be able to report any downtimes if the VM 1 crashes, we need an additional VM 2 within the provider's data center in order to place our cloud-side monitor. Nevertheless, we also require access to the VM 1 in order to check the status of predefined processes that are essential for running the cloud application. Therefore, a lightweight software component (VM monitor) must be installed on the VM 1. We apply a periodical pull model in order to invoke the VM monitor from the cloud-side monitor and an event-based push model to send data to the broker.

4.3 Broker and Availability Calculation

The broker maintains a downtime list for each monitor and periodically computes the overall availability. For this purpose, the broker first determines the overlap of the downtimes reported by both consumer-side monitors in order to separate cloud application downtimes from network errors and afterwards, merges the consolidated downtimes with the downtimes of the cloud-side monitor. For the latter, we assume that whenever two detected downtime intervals from consumer- and cloud-side overlap, these downtime intervals belong to the same outage. In this case, the broker decides which downtime interval better reflects the real downtime by evaluating the following conditions. Δ_T represents the difference between the reciprocals of the monitoring frequencies (i.e., $T = 1/f$) of both monitors and d_c and d_p are the durations of the downtime monitored at consumer- and provider-side, respectively.

$$(d_c - d_p)\begin{cases} > \Delta_T \rightarrow \text{ consumer-side monitor} \\ = \Delta_T \rightarrow \text{ cloud-side monitor} \\ < \Delta_T \rightarrow \text{ impossible} \end{cases} \quad (2)$$

The conditions above result from the difference in precision of both monitors. They express that whenever the difference between the durations is not caused by the inaccuracy of the consumer-side monitors (case 2), the downtime of some essential services of the application must exceed the downtime of the underlying processes monitored at cloud-side (case 1). Finally, the broker obtains a list of downtimes and calculates the overall availability based on the calculation proposed by (Haberkorn and Trivedi, 2007) as described in the following. For the calculation, we introduce a set of variables (Table 1).

Table 1: Variables for calculating the overall availability

T_s	total service time
D_s	aggregated downtime
U_s	aggregated uptime
n	number of downtime intervals
m	number of uptime intervals
$d_1, ..., d_n$	completed downtime intervals
$u_1, ..., u_m$	completed uptime intervals

The aggregated downtime D_s and uptime U_s of a cloud application can then be calculated as follows:

$$D_s = \sum_{i=1}^{n} d_i \quad and \quad U_s = T_s - D_s \quad (3)$$

Since U_s varies depending on whether the overall availability is calculated during a downtime or an uptime (Figures 3 and 4), two different cases $U_s^{'}$ and $U_s^{''}$ must be distinguished (Haberkorn and Trivedi, 2007).

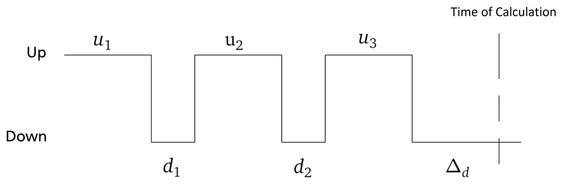

Figure 3: U_s at downtime (Haberkorn and Trivedi, 2007).

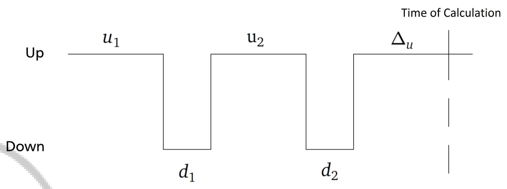

Figure 4: U_s at uptime (Haberkorn and Trivedi, 2007).

When calculating the availability during an uptime, U_s has to be calculated as follows:

$$U_s^{'} = (T_s - D_s) + \Delta_u \quad (4)$$

In contrast, when calculating the availability during a downtime, U_s can be determined as follows:

$$U_s^{''} = (T_s - D_s^{'}) = (T_s - \sum_{i=1}^{n} d_i - \Delta_d) \quad (5)$$

Finally, the overall availability can be calculated using Equation 1 (Haberkorn and Trivedi, 2007):

$$MTTF = \frac{U_s}{m} \quad and \quad MTTR = \frac{D_s}{n} \quad (6)$$

5 EXPERIMENTS

Our approach has been prototypically implemented using the (Kaltura, 2012) video platform, that we deployed on a VM in our blade center. On this VM, we also installed our VM monitor and on a second VM, we deployed our cloud-side monitor. Furthermore, we placed consumer-side monitor A as well as the broker on a local desktop computer and consumer-side monitor B on a laptop. The technical specification is shown in Table 2.

Table 2: Technical setup of the implementation.

App	Cloud Mon.	Mon. A	Mon. B
VM	VM	PC	Laptop
CentOS 5[1]	Win. 7	Win. 7	Win. 7
2×vCPU[2]	2×vCPU	4×CPU	1×CPU
2.13 GHz	2.13 GHz	2.67 GHz	2 GHz
4 GB	2 GB	4 GB	4 GB

[1]CentOS: free Linux based on Red Hat Enterprise
[2]vCPU: virtual CPU assigned to a VM

We used Java as programming language to implement all components of our monitoring framework. The communication between the different monitoring components is realized by TCP/IP sockets, except for the invocation of the cloud services provided by the Kaltura cloud application. These service invocations are based on REST and HTTP. For cloud-side monitoring, we used *httpd*, *mysql*, and *memcached* as essential processes responsible for a proper Web server functionality, database access, and memory caching. On consumer-side, we used a small sample video file that we uploaded to the Kaltura platform and that was also stored locally at consumer-side. Based on this sample video file, we periodically sent a first request to the Kaltura platform to retrieve the metadata of this video file and a second request to download the video file. In doing so, our solution verified the storage access and transmission capabilities of the platform.

5.1 Setup

The experiments have been performed using monitoring intervals of 3 minutes for the consumer-side monitors and 5 seconds for the cloud-side monitor. Every 60 seconds, the broker calculates the overall availability. The timeouts at consumer-side and cloud-side for receiving responses are set to 15 seconds and 4 seconds, respectively. In order to simulate outages of the cloud application as well as network impairments, we have used the wide area network emulator (WANem, 2011). In the first two experiments, we have simulated downtimes in order to evaluate the detection rate of our framework and in the last experiment, we have simulated network impairments in order to evaluate the behaviour under real network conditions.

5.2 Short, Periodical Downtimes

In this experiment, we simulated short, periodical downtimes over a period of 15 minutes with each downtime and uptime lasting 20 seconds and 60 seconds, respectively. We have repeated this experiment for 5 times. While the cloud-monitor detected all 11 downtimes in each run, the consumer-side monitors only detected 1.8 downtimes on average. The results (Figure 5) show that our monitoring framework is basically able to detect all downtimes, but also point out the lower precision of consumer-side monitoring.
All in all, our monitoring framework only achieved a deviation of 0.826% from the real availability due to the accurate cloud-side monitoring. However, the next experiment will show that the cloud-side monitor not always delivers reliable results, which emphasizes the need for a hybrid approach.

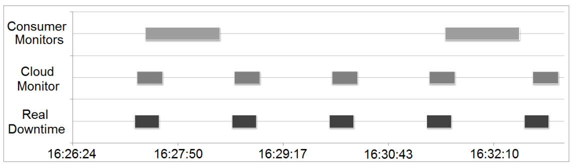

Figure 5: Difference in precision of both monitor types.

5.3 Higher Precision at Consumer-side

The second experiment simulates only a single downtime of 6 minutes during a total service time of 11 minutes. Again, this experiment has been repeated for 5 times. From the perspective of the cloud-side monitor, the failure of the cloud application is corrected after 78 seconds on average during each run. Figure 6 shows that the consumer-side monitors show a higher precision than the cloud-side monitor.

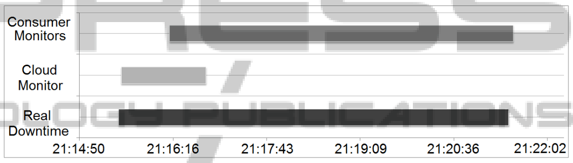

Figure 6: More precise detection by consumer monitor.

Although our framework chose the correct downtime, i.e., the result from the consumer-side monitors in order to calculate the availability, the resulting deviation of 6.03% from the real value is quite high, so that the monitoring intervals for consumer-side monitoring should be further decreased.

5.4 Network Impairments

This experiment was conducted to evaluate our monitoring framework under different network conditions. For this purpose, we applied WANem to induce several impairments into the network. As a prerequisite for our experiment, we defined six different network quality classes ranging from a network without any impairments T_0 to a network T_{1500} with a delay of 1500ms, 10% packet loss, and 10% corrupted packets. Different tests, each lasting 530 seconds and simulating a downtime of 130 seconds, were conducted at each quality level. The impairments were induced in all networks inside and outside the cloud. Although the actual availability of 75.471% of the cloud application did not change during the tests, Figure 7 shows that the difference between the actual availability and the monitored availability considerably decreases (to 56.63%) with an increasing amount of network impairments. This experiment shows that our monitoring approach is very sensitive to failures in the network, which have to be addressed in future work.

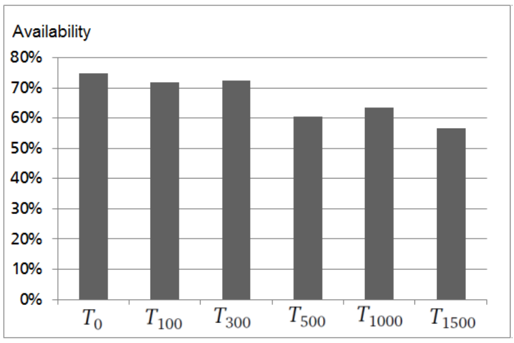

Figure 7: Precision decreases with network impairments.

6 CONCLUSIONS AND OUTLOOK

In this paper, we introduced a taxonomy for downtimes of cloud applications and presented a hybrid approach for verifying the availability of cloud applications from a consumer's perspective. The approach combines consumer- and cloud-side monitoring in order to be able to attribute downtimes to their root cause and to distinguish SLA violations from other types of downtimes. Our framework periodically provides an updated overall availability to consumers. The results of our experiments reveal that a hybrid approach is indeed required in order to allow for a precise calculation of the overall availability. However, the experiments also reveal that in case of network impairments, the detection accuracy decreases. Hence, we will develop approaches to increase the robustness of our monitoring framework in future work. Furthermore, we will conduct experiments in real cloud environments and we will explore how to determine an appropriate ratio between the monitoring frequencies on consumer- and cloud-side.

ACKNOWLEDGEMENTS

The work presented in this paper was partially funded by the German Federal Ministry of Education and Research (BMBF) under grant no. "01|C10S05" in the context of the Software-Cluster project SWINNG (www.software-cluster.org). In addition, this work is supported in part by E-Finance Lab Frankfurt am Main e.V. (http://www.efinancelab.com). The authors assume responsibility for the content.

REFERENCES

Amazon (2008). Amazon EC2 Service Level Agreement. http://aws.amazon.com/ec2-sla/, [last access: 3 December 2012].

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, 25(6):599–616.

Chazalet, A. (2010). Service Level Checking in the Cloud Computing Context. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing*, pages 297–304.

CSA and ISACA (2012). Cloud Computing Market Maturity. Study Results. Cloud Security Alliance and ISACA. http://www.isaca.org/Knowledge-Center/Research/Documents/2012-Cloud-Computing-Market-Maturity-Study-Results.pdf, [last access: 28 November 2012].

CSCC (2012). Practical Guide to Cloud Service Level Agreements. Cloud Standards Customer Council. http://www.cloudstandardscustomercouncil.org/04102012.htm, [last access: 30 November 2012].

Haberkorn, M. and Trivedi, K. (2007). Availability Monitor for a Software Based System. In *Proceedings of the 10th IEEE High Assurance Systems Engineering Symposium (HASE 2007)*, pages 321–328.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. Wiley.

Kaltura (2012). Kaltura Inc. http://corp.kaltura.com/, [last access: 3 December 2012].

Mastelic, T., Emeakaroha, V. C., Maurer, M., and Brandic, I. (2012). M4Cloud - Generic Application Level Monitoring for Resource-shared Cloud Environments. In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012)*, pages 522–532.

Mell, P. and Grance, T. (2011). The NIST Definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, [Last access: 28 November 2012].

Michlmayr, A., Rosenberg, F., Leitner, P., and Dustdar, S. (2009). Comprehensive QoS Monitoring of Web Services and Event-Based SLA Violation Detection. In *Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing (MW-SOC 2009)*, pages 1–6.

Patel, P., Ranabahu, A., and Sheth, A. (2009). Service Level Agreement in Cloud Computing. Technical report, Knoesis Center, Wright State University, USA.

Shao, J. and Wang, Q. (2011). A Performance Guarantee Approach for Cloud Applications Based on Monitoring. In *Proceedings of the 35th Annual Computer Software and Applications Conference Workshops (COMPSACW 2011)*, pages 25–30.

WANem (2011). The Wide Area Network Emulator. Performance Engineering Research Centre, TATA Consultancy Services. http://wanem.sourceforge.net/, [last access: 5 December 2012].