# Towards the user-centric analysis of the availability in IaaS

Antonio Manuel Gutiérrez-Fernández, Pablo Fernández, Manuel Resinas, Antonio Ruiz-Cortés

School of Computer Engineering
University of Seville
{amgutierrez, pablofm, resinas, aruiz}@us.es

**Abstract.** Availability is a key property in computational services and, therefore, is guaranteed by Service Level Agreements (SLAs) from the majority infrastructure services, such as virtualization (Amazon EC2, Windows Azure, Google Cloud, Joyent, Rackspace, ...) and storage (Amazon S3, Google Cloud Storage, ...). These SLAs describe availability in natural language and there are important differences in the scope and penalties that each service provides. Furthermore, descriptions use specific domain terms so they are difficult to understand by service customers. These circumstances make that availability analysis is a tedious, error-prone and time-consuming task. In this paper, we describe in detail this problem and provide a first approach to deal with these SLAs supported on current SLA analysis techniques.

**Keywords:** Service Level Agreements, Availability, IaaS, Cloud

## 1 Introduction

Nowadays, cloud services are massively used to support enterprise software infrastructure. Cloud customers outsource infrastructure services to provide their own services. Service Level Agreeemnts (SLAs) guarantee the service consumption between the different parties. Cloud customers act as service providers for third parties. A SLA for typical software provision guarantees availability periods (24x7, office time, ...) or performance (requests per second, request latency,...) and the service provider responsability on these guarantees. As external service provision bases on cloud services, customers examine cloud services SLA so their responsability relates to the infrastructure guarantee. However, main cloud providers, such Amazon, Google, Rackspace or Joyent provide a non-negotiable Service Level Agreements (SLAs) where guarantee terms semantic is far away from software service provision semantics.

These SLAs are described in a natural language so they are easily understood by customers. SLAs include terms which provider guarantees and penalties policies. These terms are mainly constraints over some quality conditions such a availability, latency or performance. The variables constrained depend on service type (computation, storage, network, database, etc).

Service availability is very important for customers and all cloud service providers offer guarantees related to availability. In this paper, we focus on analysis of guarantees over availability domain from the customer perspective. To plan infrastructure deployment, customers evaluates how the IaaSS providers gurantees match their own requirements. In spite of natural language guarantees are easily understable, manually evaluating how these guarantees apply to customer requirements is a time consuming, tedious and error-prone task, so automating this analysis has an impact on customers business plan [1].

The main infrastructure services are computing and storage resources so we focus in both kind of services. Particularly, we take SLAs where single machine availability is guaranteed. Each kind of services have different semantic about availability and customer preferences are analysed considering these differences. To afford this analysis, we model usual customer preferences in the form of Frequently Answered Questions (FAQ). First step to answer these questions, is translate natural language description to a formal language that can be computationally operated. WS-Agreement[1] is a well-known and widely used schema to describe Agreement supporting penalties and rewards terms so we used to support our approach. In a second step, as there is not existing tool or solution which automate the availability checking and penalties appliance operations, we design and develop these operations to automate answering the proposal questions.

We introduce three basic questions of interest to analyse providers guarantee for infrastructure services. These questions are:

- Q1: Given Availability guarantee, which is the maximum down time without penalties?
- Q2: Which is the penalty when a machine has been down for N minutes (in a row or in separate periods)?
- Q3: Which is the minimun time with the maximum penalty (when the guarantee limit is reached)?

In next sections, SLAs from cloud providers are described for computing (2) and storage services (3). Section 4 identifies research efforts to model SLA supporting penalties definition in WS-Agreement and design automate solutions for these questions as analysis operations.

## 2 Availability in computing services

### 2.1 Rackspace

In the Rackspace SLA, host availability is defined with the following terms:

*We guaranty the functioning of all cloud server hosts including compute, storage, and hypervisor. If a cloud server host fails, we guaranty that restoration or repair will be complete within one hour of problem identification. If we fail to meet a guaranty stated above, you will be eligible for a credit. Credits will be*

[1] `http://www.ogf.org/documents/GFD.107.pdf`

*calculated as a percentage of the fees for those Cloud Servers adversely affected by the failure for the current monthly billing period during which the failure occurred (to be applied at the end of the billing cycle), as follows: Cloud Server Hosts: 5% of the cloud server fees for each additional hour of downtime, up to 100% of the cloud server fees*

According to the SLA defined, the Rackspace availability per machine guarantee excludes the first 60 minutes and offers a 5% per every 60 minutes over the first ones. So the questions proposed can humanly be answered.

- Q1: Maximum down time without penalties is 61 minutes.
- Q2: After N down minutes in a row,
  the penalty is 0 if $N \leq 60$minutes, 5% of monthly fee if $60 < N \leq 120$
  10% of monthly fee if $120 < N \leq 180$ ... 100% of monthly fee if $N > 1260$
- Q2: After N down minutes in separate periods, the penalty is calculated per separate period.
- Q3: As in previous question, answer depends on the distribution of offline periods. Having a distribution of N offline periods, and a 5% of penalty per 60 minutes, maximum penalty, 100%, is reached when the sum of minutes (minus 60) of every offline period is $20 \times 60$ minutes (i.e.: 1200 minutes). In the simple case, a single offline period, maximum penalty is reached in 1260 minutes.

### 2.2 Joyent

In the Joyent SLA, host availability is defined with the following terms:

*Goal: Joyents goal is to achieve 100% Availability for all customers. Remedy: Subject to exceptions, if the Availability of customers Services is less than 100%, Joyent will credit the customer 5% of the monthly fee for each 30 minutes of downtime (up to 100% of customers monthly fee for the affected server).*

So, Joyent guarantees any unavailable machine time and the penalty is 5% per each 30 unavailable minutes. The answer to proposal questions is similar to Rackspace

- Q1: There is no downtime without penalties.
- Q2: Penalty is 5% of monthly time if $0 < N <= 30$ minutes, 10% of monthly fee if $30 < N \leq 60$ ... 100% of monthly fee if 300 ¡ N
- Q3: In this case, maximum penalty is 100% and as penalty is 10% per 30 minutes, maximum is reached in $10 \times 30$ minutes (i.e.: 300 minutes).

## 3 Availability in storage services

Storage services are usually binded to computing services but as operative nature is different from computing, availability guarantees are described with different semantics.

As example scenarios for these questions, we take Google Cloud and Amazon S3 storage services.

### 3.1 Amazon S3

In the Amazon S3, storage availability guarantee is defined with two key concepts:

- Error Rate: Number of error requests divided by total number of requests in a 5 minute time period.
- Monthly Uptime Percentage (MUP): 100% minus the average of error rates in a month.

This is, monthly uptime percentage depends on the request distribution per 5 minutes periods and the failure rate. Possible penalties depends on the MUP with following rule:

- MUP is greater or equal than 99% but less than 99.9%, penalty is 10% of the credit for next payment.
- MUP is lesser than 99%, penalty is 25% of the credit for next payment.

Analysing this SLA, the answer to provided questions are:

- Q1: Maximum time without penalties depends on request distribution. Assuming 100% failure and at least one request per 5-minute period, there wouldn't be any penalty until 9 5-minute intervals (0.1% percent of monthly 5-minutes periods). So, answer to question is that maximum period without penalties is 45 minutes ($9 \times 5$), the specific requests number depends on request ratio. For example, with 1 request per minute ratio, with 45 failed requests customer could apply for penalty or with 60 requests per minute ratio, customer would have 2700 failed requets before apply penalties (i.e.: considering a distribution of A requests per minute, the maximun failing requests is $9 \times 5 \times A$ requests).
- Q2: For this question, we can consider two cases. In one hand, a distribution of consecutive 100% failure requests. In this case, the penalty depends entirely of the time period considered. Over 45 minutes, customer gets a 10% of service credit and over 450 a 25%. In the other hand, if failure requests are not in a row, penalty applied depends on the distribution of failure requests through time.
- Q3: Similarly to first question, over 1% percent of the time periods with 100% failed requests (i.e.: over 450 minutes), customers get maximum credit for next billings (i.e.: 25%). For a failure regular distribution of C failures per requests (C has to be lesser or equals than 1), the maximum penalty is applied when $0.01 \times MonthlyMinutes \div C$ minutes have passed.

### 3.2 Google Cloud Storage

Google cloud storage SLA describe its availability guarantee, similarly to S3, around the request error rate. In this case, key concepts are:

– Error Rate: Number of error requests divided by total number of valid requests.
– Downtime Period: 10 consecutive minutes interval where Error Rate is bigger than 5% percent.
– Monthly Uptime Percentage (MUP): Total minutes in a month minus the number of the minutes in downtime periods divided by total minutes in a month.

Google offers two different SLAs to guarantee storage availability. We consider standard SLA as the concepts are similar in both of them and differences are bigger availability at bigger price. Standard SLA offers following penalty:

– MUP is greater or equal than 99% but less than 99.9%, penalty is 10% of the credit for next payment.
– MUP is greater or equal than 95% but less than 99%, penalty is 25% of the credit for next payment.
– MUP is lesser than 95%, but penalty is 50% of the credit for next payment.

So, similar to Amazon, the answers to the proposed questions for Google guarantees depend on the request distribution:

– Q1: Similarly to Amazon S3, considering 100% failure and at least one request per 10-minute period, until minute 50 (0.1% percent of monthly 10-minutes periods), there wouldn't be any penalty. The maximun number requests depend on request ratio on those 50 minutes. Considering a distribution of A requests/minute, the maximun failing requests is $50 \times A$ requests.
– Q2: To answer this question, again, as in Amazon S3, we can differentiate between the simple case, i.e.: a distribution of consecutive 100% failed requests (penalty depends exclusively on time) or a non regular failed requests distribution, where penalty depends on this distribution.
– Q3: In this question, although penalty is calculated in a similar way to Amazon S3, Google offers higher guarantees for pesimistic scenarios. If time intervals with failures get a 5% (i.e.: over 2160 minutes with 100% failed requests or, with a C failure distribution, $0.01 \times MonthlyMinutes \div Cminutes$) customer gets a 50% service credit.

## 4 Our Proposal

As far as we know, existing solutions focus on cloud monitoring or validate SLAs but no one focuses on the analysis at design time of resource availability guaranteed by cloud providers.

First step towards supporting automate analysis over availability is modelling the different guarantees proposed so that their penalties can be checked by a software component. Our proposal is based on WS-Agreement which is a standard that is widely used and that has been successfully applied in the computational domain. WS-Agreement is the prominent proposal to model SLAs and a number

of supporting tools for editing and analysing WS-Agreement documents, such our agreement management environment IDEAS[2].

### 4.1 Modelling SLA with WS-Agreement

WS-Agreement specification defines an Agreement document metamodel. This metamodel is composed of several distinct parts: An optional name, context and terms. The context section provides information about participants in the agreement (i.e., service provider and consumer) and agreement's lifetime. Terms section describe the agreement itself. There are two different kind of terms, namely service terms and guarantee terms, that can be recursively composed.

Guarantee Terms define constraints over Service Properties. They constrain the service execution that an obligated party must fulfill. These constraints are referred as Service Level Objectives (SLOs). So an SLO is an assertion over monitorable properties defined in Service Propeties. Guarantee Terms definition can be guarded by a Qualifying Condition (QC), which indicates a precondition to apply the constraint in the SLO. The valuation of the guarantee is defined using Business Value List ([2],[4]). Business Value includes the expression of importance to determine the possible penalties in a determined interval. Expressions to define SLOs and Penalties are used to operate over Agreements (i.e.: check an SLO is violated to renegotiate SLA or calculate penalties).

Using WS-Agreement document structure, natural language SLAs provided by Joyent and Rackspace are defined in a straight way (Figures 1 and 2). Syntax used in figures is iAgree, a human readable syntax proposed for WS-Agreement documents that provided a specific language to define SLO expressions. iAgree is a proposal from [3], which includes mapping from iAgree to WS-Agreement and the opposite one. Original iAgree proposal does not describe Business Value List sections so we extend it using same expression language to define Penalty expressions.

As figures depict, the Availability guaranteed by Rackspace and Joyent correspond to Service Level Objective of Unavailability lower than 60 minutes or equal to zero minutes, respectively. To simplify scenario, guarantee limitations (as planned maintenance) is not defined but it could be included as qualifying conditions for the guarantee terms. Penalties are described in Business Values for the monthly payment interval using a math expression according to the natural language definition.

### 4.2 Automate availability analysis

The analysis of agreements means extracting relevant information from these documents so it is often useful to define analysis process as operations that take a set of parameters as input, and return a result as output [3].

With the SLA provided in previous section and the iAgree modelling proposed (Figures 3, 4, 5, 6), we analyse how to define operations that answer to

[2] `http://www.isa.us.es/IDEAS`

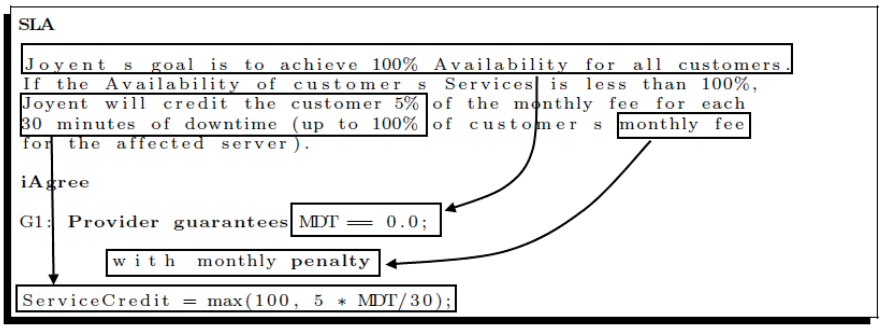


**Fig. 1.** Modelling Joyent SLA availability guarantee in iAgree

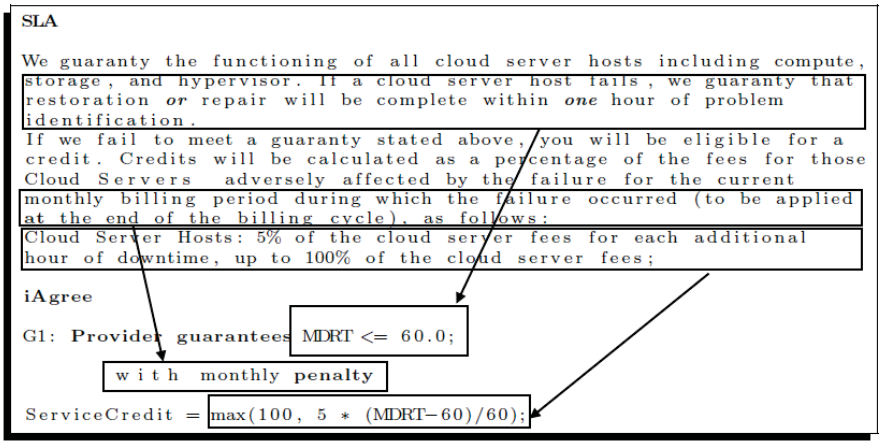


**Fig. 2.** Modelling Rackspace SLA availability guarantee in iAgree

provided questions. In spite of these operations being analysed for the example providers, we describe them for any cloud provider SLA. As SLO expressions in iAgree are defined as constraints over service properties, operating with such expression can be afford with a logic solver, such as Constraint Satisfaction Problems (CSPs). So we express the proposed questions as analysis operations.

### 4.3 Maximum failure without penalties operation

Considering the provided scenario, given availability guarantee terms, the operation to answer the question about maximum failure interval returns a time interval. This interval can be expressed as time interval or number of requests depending on service type.

To implement these operation, we consider that guarantee term expression is equivalent to penalty expression (i.e.: whenever an SLO is violated, a penalty is applied and viceversa). So the answer to this operation does not need evaluate penalty expression and compensation and can be checked with only Service Level Objective expression (SLO).

### 4.4 Applied Penalties operation

Penalty analysis depends on the failure rate. Considering the unavailability Service property (MDT in Joyent Agreement, MDRT in Rackspace Agreement or MUP in Amazon S3 and Google Storage), we get the result of this operation.

Given time input variable, constraint solution should return an expression over Penalty Value Unit. Unlike previous operation, solution to this operation depends on the availability guarantee is expressed as accumulate value or not accumulate one (i.e.: Joyent offers guarantee over any unavailable time but Rackspace only for exceding time over 60 minutes per service interruption). So, considering the simple case, this is, where failure ratio is 100% per unavailable interval, both solutions are:

- Accumulated value: Solution is calculated using the sum of every unavailable interval length.
- Non-accumulated value: Solution is the sum of solution for every single unavailable interval length

### 4.5 Minimum time with maximum penalties operation

Considering the provided scenario, given availability guarantee terms, the expected answer to the question is the minimum time period where maximum possible penalty are appliable.

To implement these operation, we consider that penalty expression and solve for maximum penalties. For instance, in Joyent solving the single penalty expression:

$$P = min(100, 5 \times T \div 30)$$

```
Template RackspaceSLA version 1
Provider Rackspace as Responder;
...
Guarantee Terms
  G1: Provider guarantees MRT <= 60.0
    with monthly penalty
    ServiceCredit = 5 * (MRT - 60) / 60
...
```

**Fig. 3.** Computing SLA provided by Rackspace *iAgree*

```
Template JoyentSLA version 1
Provider Joyent as Responder;
...
Guarantee Terms
  G1: Provider guarantees MDT == 0.0
    with monthly penalty
    of ServiceCredit = 5 * MDT / 30
...
```

**Fig. 4.** Computing SLA provided by Joyent *iAgree*

```
Template AmazonS3SLA version 1
Provider Amazon as Responder;
...
Guarantee Terms
  G1: Provider guarantees MUP <= 99.9%
    with monthly penalty
    of ServiceCredit = 10 if MUP >= 99% AND MUP < 99.9%
    of ServiceCredit = 25 if MUP < 99%
...
```

**Fig. 5.** Storage SLA provided by Amazon S3 *iAgree*

```
Template GoogleCloudSLA version 1
Provider Google as Responder;
...
Guarantee Terms
  G1: Provider guarantees MUP <= 99.9%
    with monthly penalty
    of ServiceCredit = 10 if MUP >= 99% && MUP < 99.9%
    of ServiceCredit = 25 if MUP >= 95% && MUP < 99%
    of ServiceCredit = 50 if MUP < 95%
...
```

**Fig. 6.** Storage SLA provided by Google Cloud *iAgree*

Being maximum Penalty = 100%,

$$MinimunTime = 100 \times 30 \div 5 = 20 \times 30 = 600 minutes.$$

In Amazon S3, similarly, Being Maximun Penalty = 25%, it is reached when unavailability = 1%. With a 100% failed requests ratio: 1% is 86,4 5-minutes intervals or 432 minutes. For lower failed requests ratio, time to get maximun penalty would be inverse to the failure ratio:

1 / 3 failed request ratio would take $3 \times 86,4$ intervals to reach maximun penalty.

## 5 Conclusions and Future Work

The contribution of this paper is based on automating availability analysis procedure which can be applied to any provider with availability guarantees. Availability guarantees are not expressed in a homogenous semantic in the different infrastructure providers. Guarante Terms refered to availability usually express penalties. These penalties easily reflects different proposals from the provider goals, so it is advisable extend common validation criteria in SLA and its analysers and compilers to detect related errors.

As questions about availability are not usually straight to answer and, in all cases, are tedious and error-prone, automating these questions support terms analysis. Expressing these questions as analysis operations over SLA simplifies infrastructure solutions design and fasts the development of concept proof and time to market.

This proposal focuses on analyse availability SLA guarantees in major infrastructure providers. However, early analysis detects availability comprises a wide range of semantics depending on the cloud provider and kind of services. Therefore, this work can be extended to define validation criteria in providers where availability concept has a wider scope and greater complexity or different domains such as Platform as a Service (PaaS) or Software as a Service (SaaS). Furthermore, analysis operations over availability are designed to check SLAs guarantees at service design and planning stage but it is not reviewed how this operations can apply to execution and monitoring phases.

## References


1. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: Sybl: An extensible language for controlling elasticity in cloud applications. In: CCGRID. pp. 112–119 (2013)
2. Ludwig, H., Ludwig, H.: Ws-agreement concepts and use agreement-based service-oriented architectures. Tech. rep. (2006)
3. Müller, C.: On the Automated Analysis of WS-Agreement Documents. Applications to the Processes of Creating and Monitoring Agreements. International dissertation, Universidad de Sevilla (2013)
4. Rana, O., Warnier, M., Quillinan, T., Brazier, F., Cojocarasu, D.: Managing violations in service level agreements. In: Grid Middleware and Services, pp. 349–358. Springer US (2008), http://dx.doi.org/10.1007/978-0-387-78446-5_23