1 Introduction

rSLA is a domain specific language (DSL) for expressing and managing service level agreements (SLAs) in a cloud environment. rSLA is coded in Ruby [**?**], a dynamic language that enables rapid prototyping and application development.

The rSLA DSL is described by an alphabet and by production rules that help to extend the language. The rSLA programming library provides a runtime engine for deploying and running an rSLA service in a cloud environment.

Although the scientific literature provides plentiful results on automated management of SLAs for distributed computing [**?**, **?**, 1], cloud markets hesitate to adopt such solutions. Provisioning of cloud services is handled either manually or with software tools that do not embrace cloud service characteristics.

Cloud service management does not yet support automated and transparent solutions for the management of leased resources. Additionally, there is no established standard yet for the automatic expression and management of SLAs for cloud services.

rSLA provides a DSL library for the definition of rSLA objects and a runtime engine to create and process such objects. The DSL enables the automated generation of customized SLAs and the transparent management of cloud service compliance.

The rSLA is deployed on the IBM Bluemix platform [**?**] as a ruby web service using the sinatra gem[1]. A pilot version of the language is currently running for an IBM financial client. The monthly results from using the rSLA language to evaluate the service level compliance of resources leased by the client, showcase the rSLA DSL adequacy in managing cloud services.

How is the paper structured

-what is the problem that the language solves, motivation to solve this problem -language structure, alphabet, production rules -language runtime -current testing, future testing

2 Problem definition/ motivation

Cloud service management has not yet integrated automated and transparent solutions for controlling the provisioning of resources. Cloud customers pay for applications, however they do not have tools to verify their applications' service level values. Similarly, cloud providers do not have tools to evaluate on-demand their service level compliance and to optimally control their resource distribution.

There is no established standard for the automated expression and management of SLAs for cloud services. The research community has proposed solutions for defining SLA context [**?**, 1] and for processing such information [**?**, **?**] in a distributed computing environment. The cloud industry, however, has not adopted any methods for the automatic management and evaluation of service level values.

[1] Sinatra, `http://www.sinatrarb.com/`

SLA terms enclose data values that are retrieved from monitoring. Monitoring in this context means the systematic observation of metric values that are used by one or more services. An SLA may include numerous such values, which in turn are processed for the evaluation of service level objectives (SLOs). In service level management, an SLO verifies if SLA conditions are violated or not. A cloud provider needs to control during service runtime the value levels of countless SLOs from multiple customers. Cloud markets do not yet provide such tools.

A scheduler is used to coordinate the execution of involved processes for the systematic control of service level values. A cloud scheduler executes tasks that define the monitoring of service metrics and that perform the evaluation of service values at defined points during a service runtime. In a cloud environment, a scheduler additionally takes into account the availability of cloud resources and their distribution among service customers.

In service level management, there are functional and operational dependencies between involved entities. For example, a composite metric, as its name implies, represents the composition result from one or more base, as well as composite metric values. The values of composite metrics in an SLA are used for the definition of conditions and objectives on service levels.

Hence, a schedule configuration for measuring base metric values, may impact the evaluation schedule for one or more composite metric values. Such dependencies raise research questions on how optimal scheduling configurations can apply in a cloud environment for service measurement and evaluation tasks.

Evaluation of service levels consists from multiple computing tasks. SLOs are evaluated in scheduled intervals to determine service level compliance. An evaluation process may define a set of SLO conditions. Such conditions take the form of logical statements that are configured in the SLO definition. Logical statements may require to compare a set of composite values against one or more threshold values.

rSLA provides an SLA programming library and a service runtime engine for creating and managing SLAs in a cloud environment. The language is not intended to be used only by engineers or ruby developers. An important goal of the rSLA design is to provide a high-level, easy to use and to extend tool that is suitable either for human or machine consumption.

IBM Bluemix is a cloud-based platform to build, run and manage applications.

3 rSLA DSL

alphabet, vocabulary, language structure
production rules

3.1 rSLA language structure, alphabet

The rSLA language follows the semantic decomposition of the WSLA specification [1], where an SLA takes the form of a hierarchical tree with a single root

node and numerous uni-directional edges. In an rSLA tree, the root node represents an SLA object. Figure 1 illustrates the rSLA vocabulary as a tree of objects that the DSL implements.

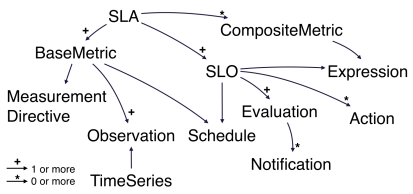

Fig. 1: rSLA DSL object diagram

```
SLA ⊃ { BaseMetric+,
    CompositeMetric∗, SLO+ }
BaseMetric ⊃ {
    MeasurementDirective,
    Observation∗, Schedule∗ }
CompositeMetric ⊃ {
    Expression∗ }
SLO ⊃ { Expression∗, Action∗,
     Evaluation∗, Schedule∗ }
```

Listing 1: rSLA vocabulary

Connections between nodes in the rSLA tree highlight nesting of SLA context. In Figure 1 nodes that are close to the root, designate SLA branches like base, composite metrics and service level objectives. Edges between nodes are uni-directed to illustrate the rSLA tree hierarchy. The edge direction points to the nested element in the hierarchical relationship. + symbols on edges indicate that the parent node may include multiple nested objects. Hence, an SLA may consist of multiple base, composite metrics as well as SLOs.

Listing 1 describes the rSLA vocabulary using set notation to highlight the nesting between language objects:

rSLA supports the creation of programming blocks that express SLA context and that are necessary to run and manage rSLAs in a cloud environment. In the rSLA language nested relationships denote inclusive associations between objects. For example, an SLA includes base and composite metrics as well as SLOs. Inclusive relationships of rSLA objects do not share same multiplicity rules (listing 1). The rSLA DSL follows the WSLA specification [1] with respect to the definition of rSLA objects and of their basic attributes.

In Figure 1 the $Notification$ and $TimeSeries$ classes do not appear in the rSLA set representation. These two classes produce objects that help with service level management operations like the statistical analysis of data coming from monitoring or automated notification reports on scheduled events of service level evaluation. Such two classes are not initially required to build and run SLA instances in a cloud environment, but may be required while one or more SLA management tasks are processed.

conclude that

3.2 rSLA language production rules

rSLA language constructs: elements have relationships/dependencies, there is nesting and management dependencies

production rules

4 rSLA editing, runtime

ready to use functions

The rSLA alphabet consists of language elements that require user input for their creation in an rSLA runtime environment. A DSL user can directly edit code-block scripts in ruby, start a cloud rSLA service and create an active SLA object. In the diagram solid black arrows indicate that user-input is required to create the equivalent objects in an rSLA running environment.

In Figure 1 language elements that require user-input represent tree branches for the creation of service level agreements. DSL user-input requires editing a ruby script to describe the attributes for any new rSLA object.

The rSLA language also contains elements, whose definition with an rSLA service requires that at least one SLA object exists (ex. base metric, slo). A DSL user can edit the context of such elements and associate them with required objects using code blocks in ruby SLA scripts.

5 rSLA deployment

evaluation examples

References

1. Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: Web Service Level Agreement (WSLA) Language Specification. Tech. rep., IBM Corporation (Jan 2003)