SLA-Driven Dynamic Orchestration and Composition for Distributed Cloud-Based Services

Alexandru-Florian Antonescu*‡, Philip Robinson†, Torsten Braun‡
*SAP (Switzerland) Inc., Regensdorf, Switzerland
†SAP (UK) Limited, Belfast, UK
‡University of Bern, Communication and Distributed Systems (CDS), Bern, Switzerland
alexandru-florian.antonescu@sap.com, philip.robinson@sap.com, braun@iam.unibe.ch

We describe a specification language and architecture for managing distributed software and mapped compute, storage and network infrastructure services dynamically, beyond the state of the art in cloud computing. This is referred to as dynamic application topology orchestration, where the mapping and configuration of distributed, interconnected, interdependent application services and infrastructure resources are dynamically adjusted, according to guarantees in Service Level Agreements (SLAs) and operational constraints.

The core concepts of our solution are application topology specification and orchestration of SLA management, provisioning, monitoring and response.

A Service Level Agreement (SLA) [1] is a contract between a consumer and a provider of a service regarding its usage and quality [2]. It defines guarantees or Quality of Service (QoS) terms under which the services are provided and the ways for checking those guarantees. The SLAs might also contain guaranteed actions, which might be used for enforcing the validity of the guaranteed states. The content of an SLA is inevitably used for concrete instantiation and configuration directives, which parameterize the provisioning of resources, deployment of software and tuning of settings to enable effective operation of the service.

We use SLAs as the basis for specifying dynamic behavior of application and infrastructure services, by using an extension of the USDL-SLA [3] vocabulary for describing the guaranteed states and actions (management rules), as well as the conditions required for automatic execution of the actions. The SLAs also contain enough information for determining the context in which the conditions are evaluated.

Figure 1 depicts the model used for defining the SLAs, which are used during the actual service deployment phase. The SLA model contains *Service Type* entities that are used for representing the linked descriptions [4] of the services. Each *Service Type* specifies one or more *Monitoring Metric*s that are used during runtime for gathering state information about the actual service instances. The *Service Type* also specifies a *Service Level Profile* that contains one or more *Service Level*s. A *Service Level* can be either a *Guaranteed State* or *Guaranteed Action*, which are used during runtime for checking the state of the services and for performing actions on them, such as scale-up and scale-down. Each *Service Level* has a *Service Level Expression* that is used as described below.

The *Guaranteed State* service level specifies a single *Service Level Expression*, which is used for checking a service state-invariant. During runtime, the expression is periodically evaluated for each service instance of the specified *Service Type* and if the guaranteed state is violated, then a log entry will be created, which can then be used for audit purposes.

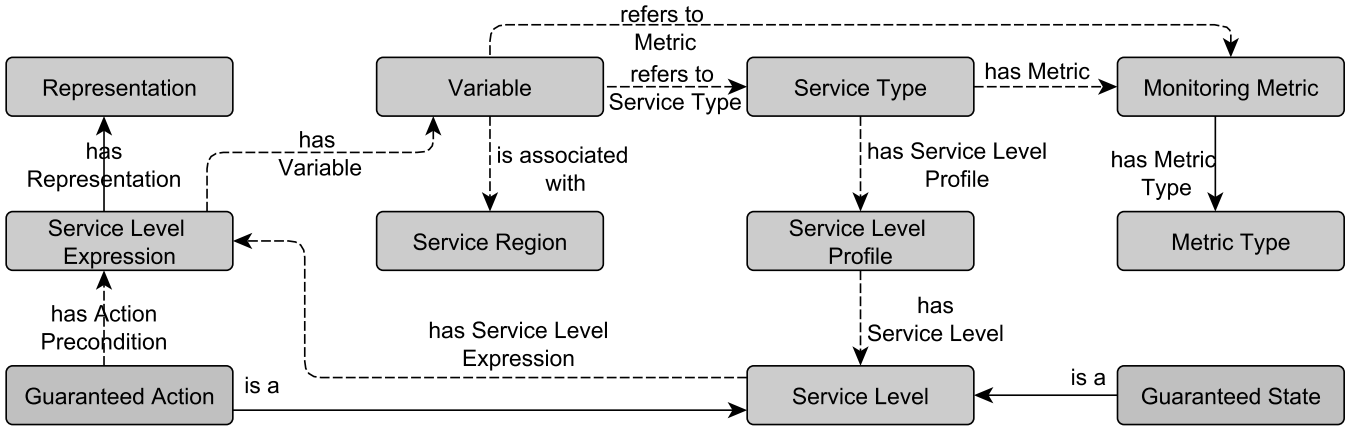

Fig. 1. Extended USDL-SLA Ontology

The *Guaranteed Action* also has a *Service Level Expression*, which contains the actual code representation of the guaranteed action, as well as another expression, which is the condition describing the state in which the application must be in order to trigger the specified action.

The *Service Level Expression* has a *Representation* that uses one or more *Variable*s. Each variable contains the value of a specified *Monitoring Metric*.

Dynamic topology orchestration is a 5-staged process, following the specification of a valid application topology. The orchestration process is a continuous process driven by the monitored metrics, guaranteed triggers and guaranteed actions in the topology specification. The process is terminated when the agreed lifetime of the application is expired or other guaranteed triggers are observed that lead to termination, as specified in the SLA.

1. Request Handling and Scheduling. This stage involves transformation of the application topology into multiple, service deployment requests and their service request dependencies. There is a 1-to-n mapping between services and requests. Each request states the relevant service, request type, explicit target, where it should be executed, operation to be performed, set of parameters, schedule for the request to be executed and set of post-deployment information that should be provided for subsequent requests, according to the dependencies.

2. Infrastructure Preparation. This second stage determines what infrastructure resources are required, where they are located and how they should be configured, given a set of scheduled requests. The first activity of preparation focuses on the compute and storage end-points, as these need to be activated before the properties for network paths can be configured - consider the case where IP addresses are assigned dynamically. Subsequently reservation of inter-region network capacity is done, completing the configuration of paths between relevant - the GEYSERS project[5] can be referenced for more details on path-computation. The final activity in this stage is the activation of infrastructure probes according to the infrastructure metrics specified in the topology.

3. Service Deployment. This stage is the installation of application-level assets including images, binaries, scripts and application data on infrastructure resources in regions where their associated services are mapped. Application-level probes are then activated according to the application metrics defined in the topology.

4. Service Monitoring. This stage is a continuous collection of metrics from the different infrastructure and application monitoring probes. Each probe is associated with a service metric. The metric's value is sent together with the metric identifier and the unique service identifier to the monitoring handler, where the value is recorded and a window of v values stored in memory for noise rejection. For the specified window, the minimum, maximum and average are calculated and made available for the use in the evaluation of guaranteed triggers in SLA expressions.

5. Response. This stage occurs when a guaranteed trigger is raised and a rule exists to resolve the difference between the guaranteed trigger and the guaranteed state. The resolution in the rule is an action that either returns to stage 1, creating a new request, or the invocation of a specific operation on a target infrastructure resource or service element.

We have designed an SLA-centric specification model, architecture and value model for dynamic application topology specification and orchestration. Our specification model is designed to simplify validation of selection, deployment and adaptation rules for application to infrastructure mapping. Secondly, the specification provides application architects with an application management model that is applicable at specification and operation time, separating the concerns of specification from configuration. Subsequently, given an architecture like we proposed, such a specification increases the level of infrastructure management automation, including provisioning, deployment, monitoring, problem specification and resolution, as an explicit expression of locality, scaling and adaptation constraints are included for each class of infrastructure resource: storage, computation and networking.

Acknowledgments

The work in this paper has been (partly) funded by the European Union through project GEYSERS (contract no. FP7-ICT-248657). The author would also like to thank Philip Robinson for his support and feedback.

REFERENCES

[1] W. Theilmann, J. Happe, C. Kotsokalis, A. Edmonds, K. Kearney, and J. Lambea, "A reference architecture for multi-level sla management," *Journal of Internet Engineering*, vol. 4, no. 1, 2010.

[2] A.-F. Antonescu and P. Robinson, "Towards cross stratum sla management with the geysers architecture," in *Parallel and Distributed Processing with Applications, IEEE 10th Int. Symposium on*, 2012.

[3] Leidig, T. and C. Momm, "Usdl service level agreement," http://www.linked-usdl.org/ns/usdl-sla, April 2012.

[4] C. Pedrinaci and T. Leidig, "Linked-usdl core," http://www.linked-usdl.org/ns/usdl-core, November 2011.

[5] E. Escalona, S. Peng, R. Nejabati, D. Simeonidou, J. A. Garcia-Espin, J. Ferrer, S. Figuerola, G. Landi, N. Ciulli, J. Jimenez, B. Belter, Y. Demchenko, C. de Laat, X. Chen, A. Yukan, S. Soudan, P. Vicat-Blanc, J. Buysse, M. D. Leenheer, C. Develder, A. Tzanakaki, P. Robinson, M. Brogle, and T. M. Bohnert, "Geysers: A novel architecture for virtualization and co-provisioning of dynamic optical networks and it services," in *Future Network and Mobile Summit*, 2011.