11th IEEE International Conference on Services Computing
June 27 - July 2, 2014, Anchorage, Alaska, USA
IEEE SCC 2014

Checkpoints for service level operations

K. Stamou, V. Kantere, J.H. Morin
Institute of Services Science
University of Geneva, Switzerland

A. Longo, M. Bochicchio
Department of Eng. for Innovation
University of Salento, Italy
In a nutshell

We use checkpoints to model a service level objective (SLO) evaluation schema, where checkpoints keep track of violations during the service runtime.

Motivated by the approach of Longo et al. [1], we present checkpoints as temporal, external elements of SLA management.

In the context, the operational role of checkpoints is to provide a consistent overview of the service level status with respect to violations and missed target values.

The SLA database follows the semantics of the Web Service Level Agreement (WSLA) [2] language and is implemented as an RDF quad-store, using the AllegroGraph middleware.

Nodes and edges on the SLA graph represent unique computational resources or literal values.

Our approach demonstrates:

- the flexibility of the SLA graph schema with respect to the integration of service management elements that are external to the WSLA alphabet.

- the query expressiveness of the SLA graph with respect to the topology and representation of service resources and checkpoints.
Contents

• Service level provisioning scenario
• SLA formalization into a property graph
• SLA data complexity
• Checkpoints in SLA graphs
• SLA RDF graph
• Checkpoint integration in the RDF schema (operational design, implementation, evaluation: SPARQL statements)
• On-going work, Conclusions
A service management system operates for a cloud service provider and manages information of active SLAs, templates and log archives.

The service management system handles data operations over remote SLA repositories through a federated, virtual one.

The scenario assumes the following:

• For every SLA, the service execution time is defined as:
  \[ Serv_{exec} = \bigcup_{i=0}^{n} t_i \text{, where } i \in [0, n] \]

• At each \( t_i \), a separate service configuration is used for the management of associated SLAs.

• Service configuration with respect to a given \( t_i \):
  \[ Serv_{config} = \bigcup_{i=0}^{n} (Serv_{desc}(t_i), Obligations(t_i)) \]

• At each time interval, the activity state of SLA elements changes, according to the configuration in use.
SLA into a property graph

Use the hierarchical structure and alphabet of the Web Service Level Agreement (WSLA) specification

Transform primary WSLA language components into element sets

Denote the SLA graph alphabet as \( \text{SLA}_{\text{alphabet}} = \text{WSLA}_{\text{alphabet}} + \text{A}_{\text{custom}} \), where \( \text{A}_{\text{custom}} \) a non-finite alphabet, bounded by a custom service domain

Define an SLA as a finite, directed graph, where nodes represent SLA elements and edges directed relationships between SLA elements:

\[
\text{SLA}_g = (V, E), \quad V \supset v_i \in \text{SLA}_g \quad \text{and} \quad E \supset e_i \in \text{SLA}_g
\]

An SLA graph database can include one or more SLA subgraphs to distinguish between different SLA instances; Subgraphs can be ordered by service element communities for the SLA management operations

6/29/14
SLA data complexity

SLAs are not big data, yet they can be viewed as data sets that are comprised of volatile, temporal and static information:

- Standardization lack: SLAs consist of semi-structured information and do not follow a fixed schema for their specification.

- Heterogeneity: The service and SLA management literature lacks benchmarks that could contribute to the standardized definition and evaluation of service metrics.

- Service dependencies: Keller et al. [4] use the term dependency to define the relationship between a dependent service or application component that requires an operation performed by an antecedent component in order for the former to execute its function. In the SLA graph data schema, dependencies are represented by the edge directionality.

- Real time measurement/data operations: Data values of SLA elements need processing during the service execution. In SLA management, the purpose of service monitoring is to verify that indeed guaranteed service levels are provided.

- SLA data volatility: SLAs contain terms, whose values may change dynamically during service execution. SLA guarantee values depend on the value ranges of SLA parameters, thus are also volatile and subject to changes according to resource consumption and availability.
Checkpoints into SLA graphs

We use the alphabet of the WSLA specification to formalize SLA element relationships as following:

- **ResourceMetric (RM):** ServiceConfiguration $\rightarrow$ (definition, value)

- **CompositeMetric (CM):** LogicalExpression $\rightarrow ([RM_{1..n}]),$ where $n \in \mathbb{Z}.$

- **SLA parameter (SLApar):** LogicalExpression $\rightarrow ([RM_{0..n}], [CM_{0..m}]),$ where $n,m \in \mathbb{Z}$

- **ServiceLevelObjective (SLO):** $\bigcup_{i=0}^{n} ([SLApar_{0..n}(t_i)], [LogicalExpression_{0..m}(t_i)])$, where $n,m \in \mathbb{Z}$
Checkpoints into SLA graphs II

Checkpoints represent dynamic nodes in the SLA graph and capture the value changes of valid SLA guarantees by comparing them with their associated SLA parameter values.

A checkpoint node represents the result outcome of a data operation that compares the values between SLO and SLA parameters according to logical rules and returns a binary value to indicate an SLO violation.

Basic requirements for the initialization of evaluation checkpoints on the SLA graph are:

- At each time interval \( t_i \), service measurement operations take place and invoke update data operations in the SLA schema.
- The expected lifetime of a checkpoint node does not exceed the one of a \( t_i \).
- Checkpoint outgoing edges are configured by the graph topology and the volume of SLOs and SLA parameters.
SLA RDF schema

SLA resources can have an arbitrary number of attributes related to them.

RDF Predicates encapsulate relationships and dependencies.

Subgraphs (SLA_{element}+graph) allow for the direct retrieval of elements that:

- share similar properties
- are nested branches of the WSLA XML tree

Service dependencies are stated as: <Dependent_{s}, Predicate_{p}, Antecedent_{o}, Subgraph_{dependent}>, where the dependency flow is outgoing for the dependent resource and incoming for the antecedent one.

Examples:

- the Predicate "obliged" connects resources, where the Subject performs an action to retrieve values from the Object
- the "has" Predicate expresses the nesting of SLA elements according to the WSLA tree
The scope of checkpoints in the SLA graph is to control the state of active SLOs by indicating violations at each execution interval. The checkpoint data operation queries a set of SLA databases and considers the following controls:

- SLA parameters, SLOs and expressions that are used for the evaluation comparison, are active during the same $t_i$.
- Every checkpoint is associated with the values of SLA parameters, SLOs and SLO expressions that define rules for the evaluation comparison.
- The volume of data affiliated with each checkpoint node is determined by the graph size and topology.
- Comparisons are performed using numerical values.
Checkpoints into SLA RDF schema - Implementation

- 3 SLA repositories, using the AllegroGraph [6] DBMS.
- Synthetic data to populate the SLA RDF repositories by blending both SLA and service management elements.
- AllegroGraph Python programming API for schema deployment and repository configuration:
  - Session-based communication with the AllegroGraph server for the federated management of the repositories.
- Python modules generate the graph data schemas and load respective information into databases. They cover for:
  - A scheduler for the data query management and the checkpoint generation in the federated repositories and according to the provided service execution time-intervals.
  - Methods for the definition of SLA resources and of their values, where the information reaches the description of endpoint resource values to represent monitored computational resources.
for \( t_i \) in ServExec:

prefix slas: <SLAs://>

➡ SELECT ?slo, ?p, ?e
WHERE {GRAPH
  slas:IterationGraph_i
  { ?slo slas:value ?x ;
    slas:follows ?e .
  }
  ?e slas:predicate ?pred ;
  FILTER ((?pred != ?pred
    && (?x ?pred ?y)))
}

➡ INSERT DATA { GRAPH
  slas:IterationGraph_i
  { checkpoint_i
    slas:points ?slo; slas:points ?p;
    slas:points ?e .
  }
}

Snapshot of the produced checkpoint subgraph during an interval \( t_i \), query output visualization with Gruff [5]
Checkpoints into SLA RDF schema - Evaluation setup

- Initial tests with checkpoints targeted expressiveness of the SLA graph data schema and demonstrated SLA data management practices using RDF graphs:
  - deployment setting: AllegroGraph virtual image on localhost, ~100,000 triples
- Efficiency tests:
  - using AmazonWS resources, synthetic SLA data
  - increase the volume of SLA triples in each federated repository proportionally
  - re-arrange relationships between SLA elements (ex. using the Zipfian distribution)
  - measure the execution throughput of the checkpoint operation over the http layer using multi-threads to simultaneously generate checkpoints
Conclusions, On-going work

The primary SLA utility engages SLAs as measurement instruments in the service execution. SLA management requires clear and granular information flow, such that SLA data can be retrieved and processed accurately.

Checkpoints in the SLA schema provide a minimal framework for the orchestration of service level evaluation through the federated data management of remote service repositories.

The RDF standard provides additional expressiveness in the information representation and enables the identification of graph patterns for the data manipulation.

Our on-going work concentrates on efficiency tests.


5. AllegroGraph Grapher-Based Triple-Store Browser: http://franz.com/agraph/gruff/

6. AllegroGraph Federation: http://franz.com/agraph/support/learning/Federating-Basic-Stores.lhtml
Thank you!

Q&A: aikaterini.stamou@unige.ch