SLA-driven Management of Distributed Systems using the Common Information Model

Markus Debusmann
Fachhochschule Wiesbaden, Germany

Alexander Keller
IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
Outline

- WSLA Framework
- SLA-driven Management with CIM
  - WSLA/CIM Model
  - Architecture
  - Prototype
- Conclusion
Web Service Level Agreement (WSLA) Framework

- SLA annotates an existing Service Specification:
  - References Service Description (e.g., Web Services: WSDL)
  - Other Service Descriptions possible, e.g., for Business Processes, Messaging, IT Resources
- XML Schema based Language for SLAs,
- Runtime Architecture comprising several SLA Monitoring Services
Delegating SLA Monitoring Tasks to Third Parties

Measurement Service Providers guarantee Accuracy and Objectivity (e.g., Keynote, Matrix)
Why do we need a CIM integration?

WSLA
- Inter-Domain Management Scenario with multiple Service Providers
- Sequence: Define -> Deploy -> Measure -> Evaluate -> Notify
- “Need-to-Know” Principle realized by Splitter (SDI Format)
- Targeted at Web Services Environment

BUT: Existing Resource Instrumentations are not Web Services-based!

Need to integrate WSLA with existing management architectures (CIM):
- Q: Which WSLA Service should perform WSLA/CIM Mapping?
  A: Measurement Service
- Connect to existing Application and Resource Instrumentation
- Execute Measurements according to Schedule defined in SLA
- Based on CIM Metrics Model: Definition/Value Pattern, Late Binding
  “Definition” Classes are instantiated during Deployment
  “Value” Classes are instantiated at Runtime; triggered according to WSLA Schedule
- Precursor to OGSA CMM: Interoperability between CIM and Web Services
Terminology: SLA Parameters, Metrics, Functions

The analyzed SLAs share a common Structure:

- Involved Parties, SLA Parameters
- Metrics used as Input to compute SLA Parameters
- The Functions that define how Metrics are aggregated
- How Metrics are retrieved from Managed Resources (Measurement Directive)
WSLA/CIM Model Requirements

- Representation of SLAs defined in WSLA
- Using CIM based measurement service
  - Define relations between high-level SLA parameters and low-level resource metrics
  - Define calculation functions
  - Support schedules to retrieve and calculate metrics
- Metric definitions and instances as in CIM Metrics Schema
  - Reduce redundancy
  - Increase reusability of metric definitions
    - catalogue of metric definitions
Active CIM Providers

- CIM status quo
  - Retrieval of resource metrics is triggered by polling of manager
  - Passive resource provider
  - Simple wrapping of legacy components

- Active CIM providers
  - Retrieval of Resource Metrics initiated by CIM Provider (Schedule class)
  - Persistency for historical date provided by CIMOM
  - Less overhead
Active CIM Providers: Computing Composite Metrics

CIM request serviced by CIMOM

- CIM: getInstance()

**AveragePackets**: StatisticalCompositeMetric
  - Value: String

**SampledPackets**: TimeSeries
  - Values: String[]

**SumPackets**: ArithmeticCompositeMetric
  - Operator: Integer

**Packets1**: RawMetric
  - Value: String

**Packets2**: RawMetric
  - Value: String

Schedule:
  - EndDate: Date
  - Interval: Long
  - StartDate: Date

Trigger: call GetNewValue()

CIM: getInstance()

- do computation
- return values
- store value
- do computation
- return values
- CIM: getInstance()
Measurements triggered by WSLA_Schedule Instance

- Derived from CIM_PolicyTimePeriodCondition
- Added Property “Interval”: The sampling Interval in Milliseconds
- TimePeriod: In PCIM (RFC 3060) Format
- Schedule Provider triggers Computation of new Metric Values:
  - Bootstrap other Instance Providers
  - Needs to run continuously, automatically instantiated at Deployment Time
  - Trouble if CIMOM stops. Workaround: include enum(Schedule) in CIMOM start-up Script
Why one Provider per Class doesn’t always work

- Cyclic Dependencies: ArithmeticCompositeMetric, StatisticalCompositeMetric, TimeSeries
- Initialize TimeSeries Provider -> Resolve References to StatisticalCompositeMetric:
  - Initialize StatisticalCompositeMetric Provider -> Resolve References to TimeSeries
  - Initialize TimeSeries Provider -> LOOP
- This problem occurs when CIMOM needs to be restarted after failure (not deployment)
- Solution: Implement 3 Classes with one Provider
Prototype Implementation

- Implemented in Java using the SNIA CIMOM
- One provider per CIM class
  - Exception: joint provider for ArithmeticCompositeMetric, StatisticalCompositeMetric, and TimeSeries classes
- Minimize implementation effort by encapsulating reusable functionality in base classes
- More complex providers for
  - Schedule class
  - Metric instances
Conclusions and Outlook

Lessons learned:

- Novel approach for SLA-driven management using CIM
- Successful integration of Web Services environment and WBEM/CIM
- WSLA model can be mapped onto CIM
- Measurement service realized a CIM provider
- Active providers

Outlook:

- Proof-of-Concept Implementation used for Finalization of CIM 2.7 Metrics Model
- Full interoperability of CIM and Web Services
- Provider bootstrapping and initialization
- Candidate for standardization in CIM 2.9