1. MELA Deployment

1.1 Compiling MELA

The “MELA” directory contains a MELA Maven "pom" project. The project has a set of Maven modules: MELA-AnalysisService, MELA-DataService, and MELA-Common. MELA-DataService is responsible of collecting monitoring information from existing monitoring systems, and storing it in a local monitoring repository. The MELA-AnalysisService queries monitoring data from the MELA-DataService structures, and analyzes it. MELA-Common contains the common data model used by the other two modules.

To compile the modules using Maven, from the MELA main directory, running "mvn clean package" will compile all MELA Modules. The MELA-DataService and MELA-Common compile in "jar" files, while MELA-AnalysisService compiles into a "war" file.

1.2 Configuring MELA

a. Configuring Monitoring Data Source

MELA-DataService can retrieve monitoring data from various data sources:

- Local Ganglia: monitoring data retrieved by connecting to a Ganglia running on the machine also running the MELA-DataService
- Remote Ganglia: monitoring data retrieved by issuing a Ganglia query trough SSH to a Ganglia installation running on a machine separate from the MELA-DataService
- JCatascopia
- Replay: in which monitoring data already collected is replayed to MELA.

To configure the data source, please set the appropriate options in the ”/MELA/MELA-AnalysisService/src/main/resources/config/Config.properties”. In future we plan to move this configuration to MELA- DataService, but currently it is located in MELA-AnalysisService.

Additional configurations are needed depending on the selected data source. For Local Ganglia or Remote Ganglia, the Ganglia port is requested, as Ganglia is queried by executing a “telnet localhost port”. For the “Remote Ganglia”, a SSH key is required, and the name of the OS user associated to the key, together with the IP of the machine hosting Ganglia (to which the SSH connection is opened). In case Replay is used, the monitoring sequence ID of the stored monitoring information needs to be specified.

b. Configuring Monitoring Data Collection Interval and Elasticity Analysis

Mela-AnalysisService has three configuration options: one for enabling elasticity analysis (in terms of elasticity space and pathway), and three for controlling the monitoring data access from the MELA-DataService. Please configure the appropriate properties in the ”/MELA/MELA-AnalysisService/src/main/resources/config/Config.properties” file.

1.3 Running MELA

a) The MELA-DataService must be started first, by running the MELA-DataService-1.0.jar from the maven "target" directory as a normal jar: "java -jar ./MELA-DataService-1.0.jar". This starts the data source back-end, to which the MELA-AnalysisService will connect.

b) Deploy the MELA-AnalysisService in an application server, by copying the MELA-AnalysisService-1.0.war from the "MELA-AnalysisService/target" directory. This starts the MELA RESTfull API, and MELA GUI. The MELA GUI is accessible at "applicationServerURL/MELA/" (example, for Tomcat, http://localhost:8080/MELA-AnalysisService-0.1-SNAPSHOT/). THE GUI was tested and is working on Google Chrome web browser, and does not work on Firefox. The MELA RESTful API endpoint will be "applicationServerURL /MELA-AnalysisService-0.1-SNAPSHOT/REST_WS".

2. Application Specific Configuration

MELA provides RESTful services for configuring it for a particular application, and thus all MELA configurations can be done through service calls.

2.1 Application Structure Description

MELA considers any cloud application as a service, and uses a representational model which divides Cloud Services in Service Topologies, which are composed of Service Units, which run in Virtual Machines members of Virtual Clusters.

In Listing 1 we show an application description example, which describes a CloudService having two topologies, a DataEndServiceTopology and an EventProcessingServiceTopology. The topologies are just logical concepts, and have no physical equivalent. Each topology has Service Units. For the detailed XML Schema of the application description format, check Appendix A.

The application description must be submitted to MELA issuing a "PUT" request towards the MELA "/servicedescription" service.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MonitoredElement id="CloudService" level="SERVICE">
  <MonitoredElement id="DataEndServiceTopology" level="SERVICE_TOPOLOGY">
          <MonitoredElement id="DataControllerServiceUnit" level="SERVICE_UNIT"/>
          <MonitoredElement id="DataNodeServiceUnit" level="SERVICE_UNIT"/>
  </MonitoredElement>
  <MonitoredElement id="EventProcessingServiceTopology" level="SERVICE_TOPOLOGY">
    <MonitoredElement id="LoadBalancerServiceUnit" level="SERVICE_UNIT"/>
    <MonitoredElement id="EventProcessingServiceUnit" level="SERVICE_UNIT"/>
  </MonitoredElement>
</MonitoredElement>
```

Listing 1: Application Structure Description Example

2.2 Metric Composition Rules Description

The second important configuration step is the specification of metric composition rules, issuing a "PUT" request towards the MELA "/metricscompositionrules" service. In Listing 2 we show a composition rule example, which applies a SUM operation over the "numberOfVMs" metric collected from all the Virtual Machine instances belonging to any Service unit, and creates a new metric also called "numberOfVMs" which will be associated with the Service Unit for which is computed. Full XML Schema of the composition rules description can be found in Appendix B.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CompositionRulesConfiguration>
  <MetricsCompositionRules>
      <CompositionRule TargetMonitoredElementLevel="SERVICE_UNIT">
          <ResultingMetric type="RESOURCE" measurementUnit="ms" name="numberOfVMs"/>
          <Operation MetricSourceMonitoredElementLevel="VM" type="SUM">
          <ReferenceMetric type="RESOURCE" name="numberOfVMs"/>
          </Operation>
      </CompositionRule>
  </MetricsCompositionRules>
</CompositionRulesConfiguration>
```

Listing 2: Metric Composition Rules Description Example

2.3 Application Structure Update after Scaling Action

As MELA is designed to support controllers of elastic services, it does not know when a new Virtual Machine has been allocated/deallocated for a particular Service Unit. To keep MELA up to date and ensure it collects monitoring information also for the newly added Virtual Machines, the application structure needs to be kept updated by the MELA user, by submitting to MELA an updated application description after each scaling in/out action. The updated description must contain the Virtual Machine and Virtual Cluster instances which are currently active, and is to be submitted issuing a "POST" request towards the MELA "/servicedescription" service.

3. Appendix A – Cloud Application Description XML Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="MonitoredElement" type="monitoredElement"/>

 <xs:element name="MonitoredElementLevel" type="monitoredElementLevel"/>

 <xs:complexType name="monitoredElement">
  <xs:sequence>
   <xs:element ref="MonitoredElement" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
```

```
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="level" type="monitoredElementLevel" use="required"/>
 </xs:complexType>

 <xs:simpleType name="monitoredElementLevel">
  <xs:restriction base="xs:string">
   <xs:enumeration value="SERVICE"/>
   <xs:enumeration value="SERVICE_TOPOLOGY"/>
   <xs:enumeration value="SERVICE_UNIT"/>
   <xs:enumeration value="VM"/>
   <xs:enumeration value="VIRTUAL_CLUSTER"/>
  </xs:restriction>
 </xs:simpleType>
</xs:schema>
```

4. Appendix B – Composition Rules Description XML Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="CompositionOperationType" type="compositionOperationType"/>

 <xs:element name="CompositionRule" type="compositionRule"/>

 <xs:element name="CompositionRulesConfiguration" type="compositionRulesConfiguration"/>

 <xs:element name="Metric" nillable="true" type="xs:anyType"/>

 <xs:element name="MetricsCompositionRules" type="compositionRulesBlock"/>

 <xs:element name="MonitoredElementLevel" type="monitoredElementLevel"/>

 <xs:element name="Operation" type="compositionOperation"/>

 <xs:complexType name="compositionRulesConfiguration">
  <xs:sequence>
   <xs:element ref="MetricsCompositionRules" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="TargetServiceID" type="xs:string" use="required"/>
 </xs:complexType>

 <xs:complexType name="compositionRulesBlock">
  <xs:sequence>
   <xs:element ref="CompositionRule" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
```

```
 <xs:complexType name="compositionRule">
  <xs:sequence>
   <xs:element name="TargetMonitoredElementID" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
   <xs:element name="ResultingMetric" type="metric"/>
   <xs:element ref="Operation"/>
  </xs:sequence>
  <xs:attribute name="TargetMonitoredElementLevel" type="monitoredElementLevel"
use="required"/>
 </xs:complexType>

 <xs:complexType name="metric">
  <xs:sequence/>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="measurementUnit" type="xs:string" use="required"/>
  <xs:attribute name="type" type="metricType" use="required"/>
 </xs:complexType>

 <xs:complexType name="compositionOperation">
  <xs:sequence>
   <xs:element name="ReferenceMetric" type="metric" minOccurs="0"/>
   <xs:element name="SourceMonitoredElementID" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
   <xs:element ref="Operation" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="type" type="compositionOperationType" use="required"/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="MetricSourceMonitoredElementLevel" type="monitoredElementLevel"
use="required"/>
 </xs:complexType>

 <xs:simpleType name="monitoredElementLevel">
  <xs:restriction base="xs:string">
   <xs:enumeration value="SERVICE"/>
   <xs:enumeration value="SERVICE_TOPOLOGY"/>
   <xs:enumeration value="SERVICE_UNIT"/>
   <xs:enumeration value="VM"/>
   <xs:enumeration value="VIRTUAL_CLUSTER"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="metricType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="RESOURCE"/>
   <xs:enumeration value="COST"/>
   <xs:enumeration value="QUALITY"/>
  </xs:restriction>
```

```
  </xs:simpleType>

  <xs:simpleType name="compositionOperationType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="SUM"/>
      <xs:enumeration value="MAX"/>
      <xs:enumeration value="MIN"/>
      <xs:enumeration value="AVG"/>
      <xs:enumeration value="DIV"/>
      <xs:enumeration value="ADD"/>
      <xs:enumeration value="SUB"/>
      <xs:enumeration value="MUL"/>
      <xs:enumeration value="CONCAT"/>
      <xs:enumeration value="UNION"/>
      <xs:enumeration value="KEEP"/>
      <xs:enumeration value="KEEP_LAST"/>
      <xs:enumeration value="KEEP_FIRST"/>
      <xs:enumeration value="SET_VALUE"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```