

Tutorial (Elasticity on the Cloud): “Demonstration on Elasticity Control, Monitoring and Testing”

Daniel Moldovan
Distributed Systems Group,
Vienna University of Technology

d.moldovan@dsg.tuwien.ac.at
<http://www.infosys.tuwien.ac.at/staff/dmoldovan/>

Acknowledgements

Joint work with Georgiana Copil, Le Duc Hung, Hong-LinhTruong, Schahram Dustdar, and others

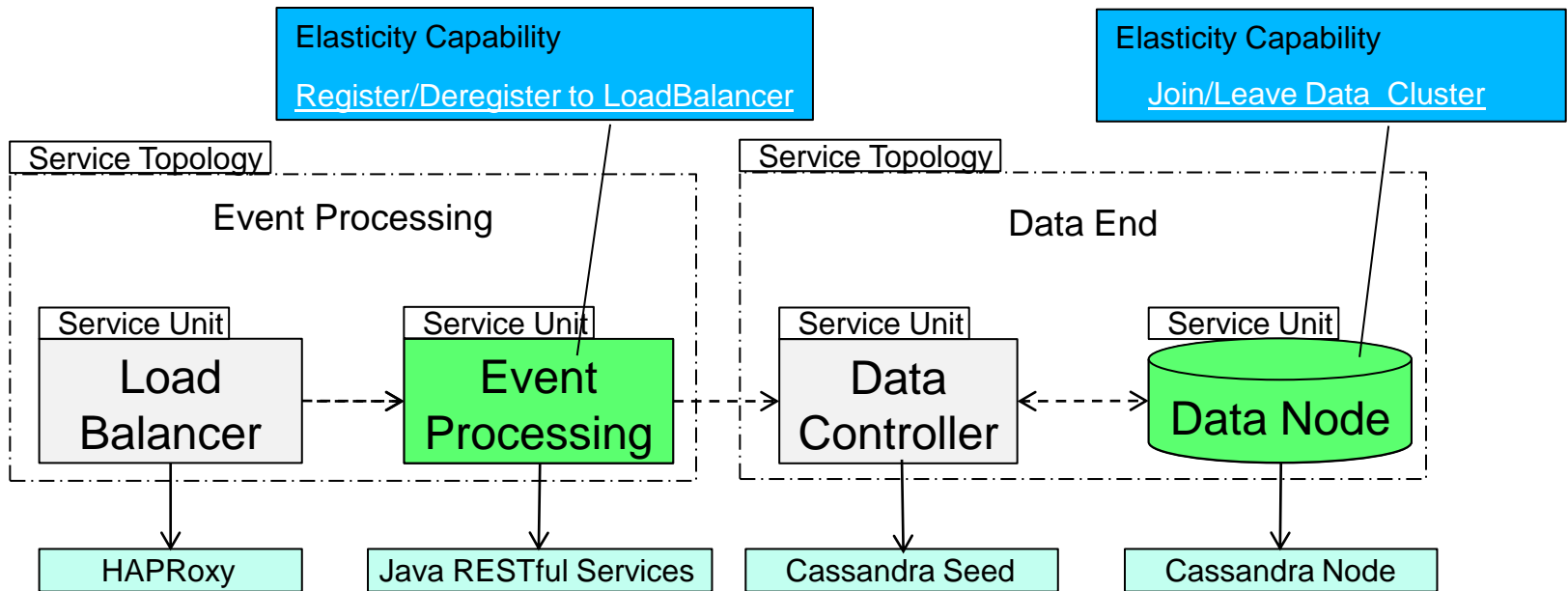


Agenda

- Elastic cloud service (M2M DaaS)
 - Elasticity capabilities
- COMOT
 - Programming
 - Domain specific language for programming elastic cloud services
 - Customization
 - Monitoring of elastic cloud services
 - Analyzing elasticity of cloud services
 - Elasticity Boundary and Pathway
 - Controlling elasticity of cloud services
 - Elasticity policy specification and enforcement

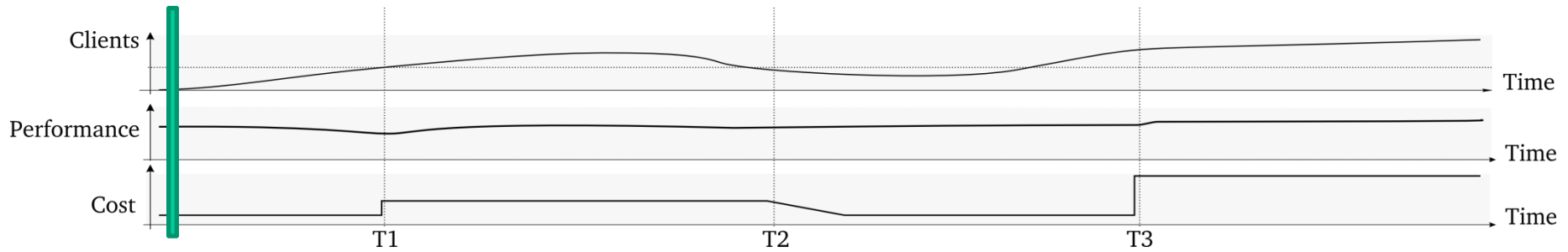
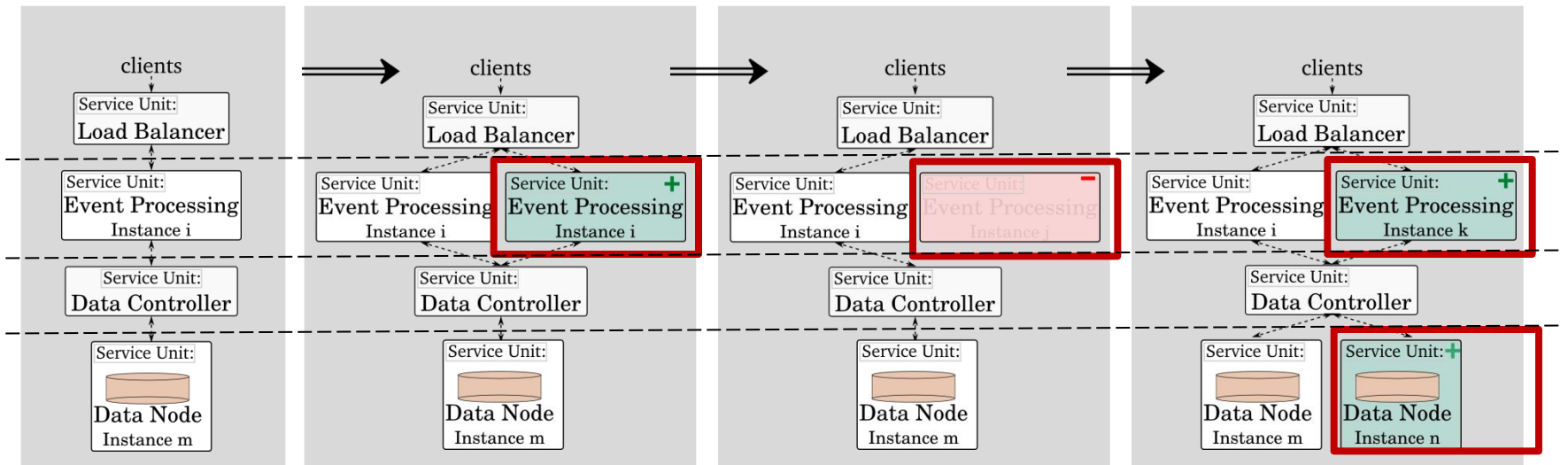
Elastic cloud service (M2M DaaS)

Elasticity Capabilities



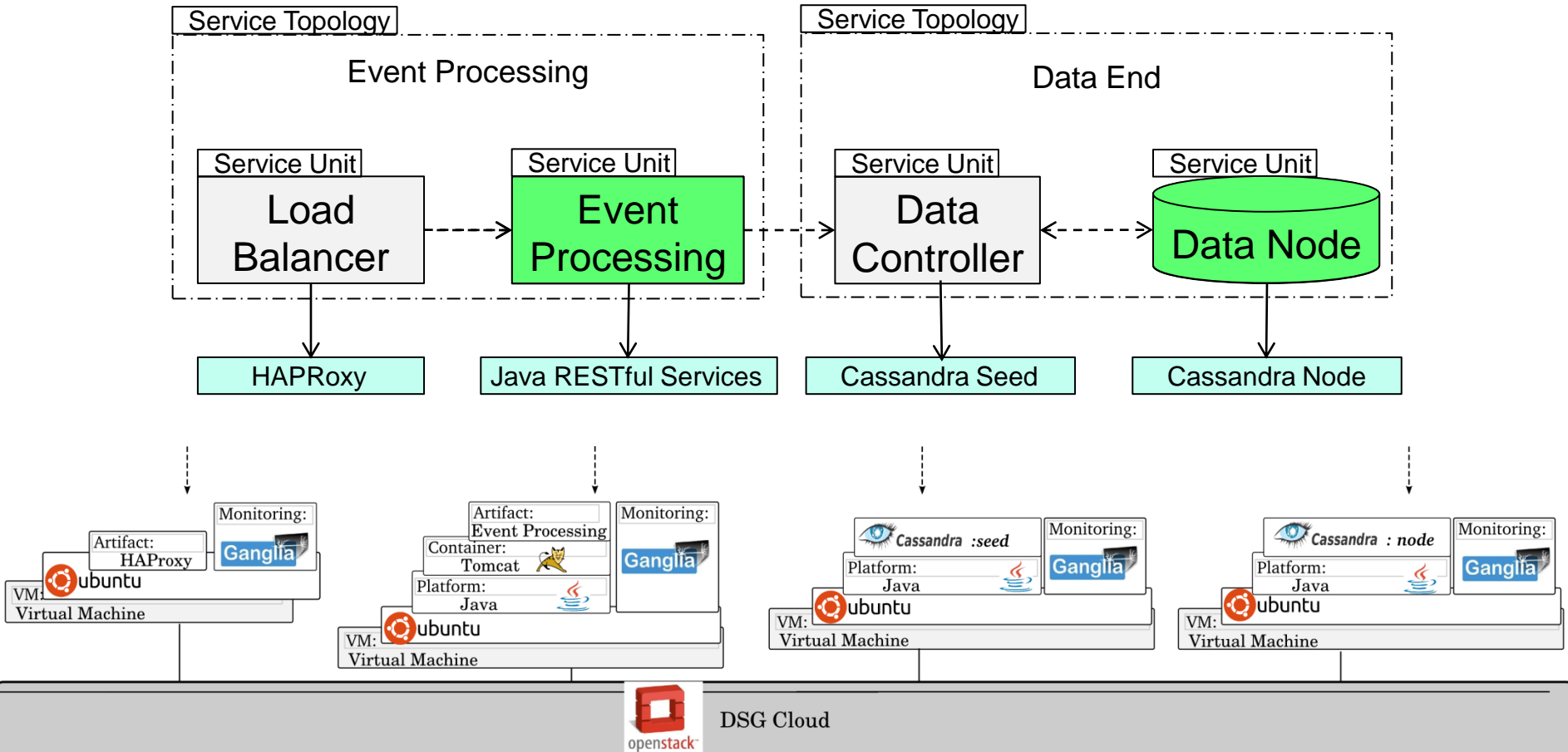
Elastic cloud service (M2M DaaS)

Elasticity Capabilities



Elastic cloud service (M2M DaaS)

Elasticity Capabilities



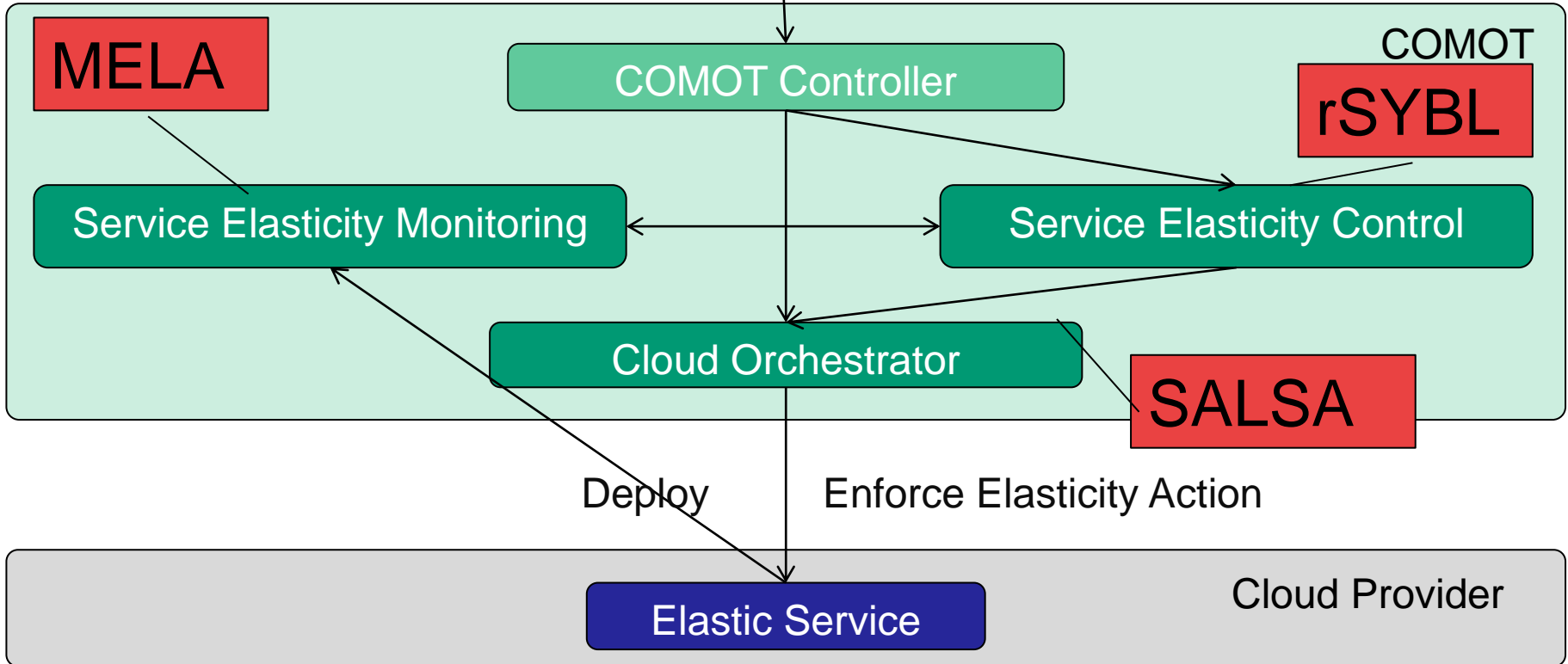
COMOT – configuration example

Domain Specific Language

```

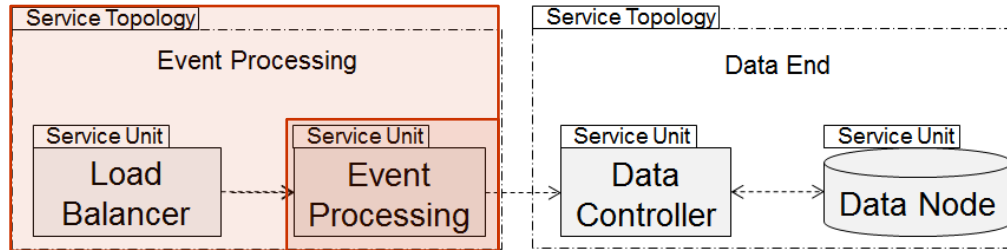
ServiceName: ElasticService = ElasticServiceConfiguration
ACTION: createService {
  serviceName: ElasticServiceConfiguration.serviceName
  hostName: ElasticServiceConfiguration.hostName
  port: ElasticServiceConfiguration.port
  ...
}
ServiceName: ElasticService = ElasticServiceConfiguration
ACTION: updateService {
  serviceName: ElasticServiceConfiguration.serviceName
  hostName: ElasticServiceConfiguration.hostName
  port: ElasticServiceConfiguration.port
  ...
}
ServiceName: ElasticService = ElasticServiceConfiguration
ACTION: deleteService {
  serviceName: ElasticServiceConfiguration.serviceName
  hostName: ElasticServiceConfiguration.hostName
  port: ElasticServiceConfiguration.port
  ...
}

```



Controlling elasticity of cloud services

Elasticity constraints and strategies



▼<CloudService id="CloudService">



</CloudService> --

Domain Specific Language

```

ServiceUnit dataNodeUnit = SingleSoftwareUnit("DataNodeUnit")
    .deployedBy(SingleScriptArtifactTemplate("artifactID", "artifact"))
    .requires(Requirement.Variable("dataControllerIP"))
    .controlledBy(Strategy("ST1")
        .when(Constraint.MetricConstraint("ST1C01",
            new Metric("metric", "unit")).lessThan("value"))
        .then(Strategy.Action.ScaleIn));
    
```

```

OperatingSystemUnit dataNodeOS = OperatingSystemUnit("DataNodeVM")
    .providedBy(OpenstackMicro("OS_DataNode_Small")
        .withProvider("dsg@openstack")
        .addSoftwarePackage("openjdk-7-jre")
    );
    
```

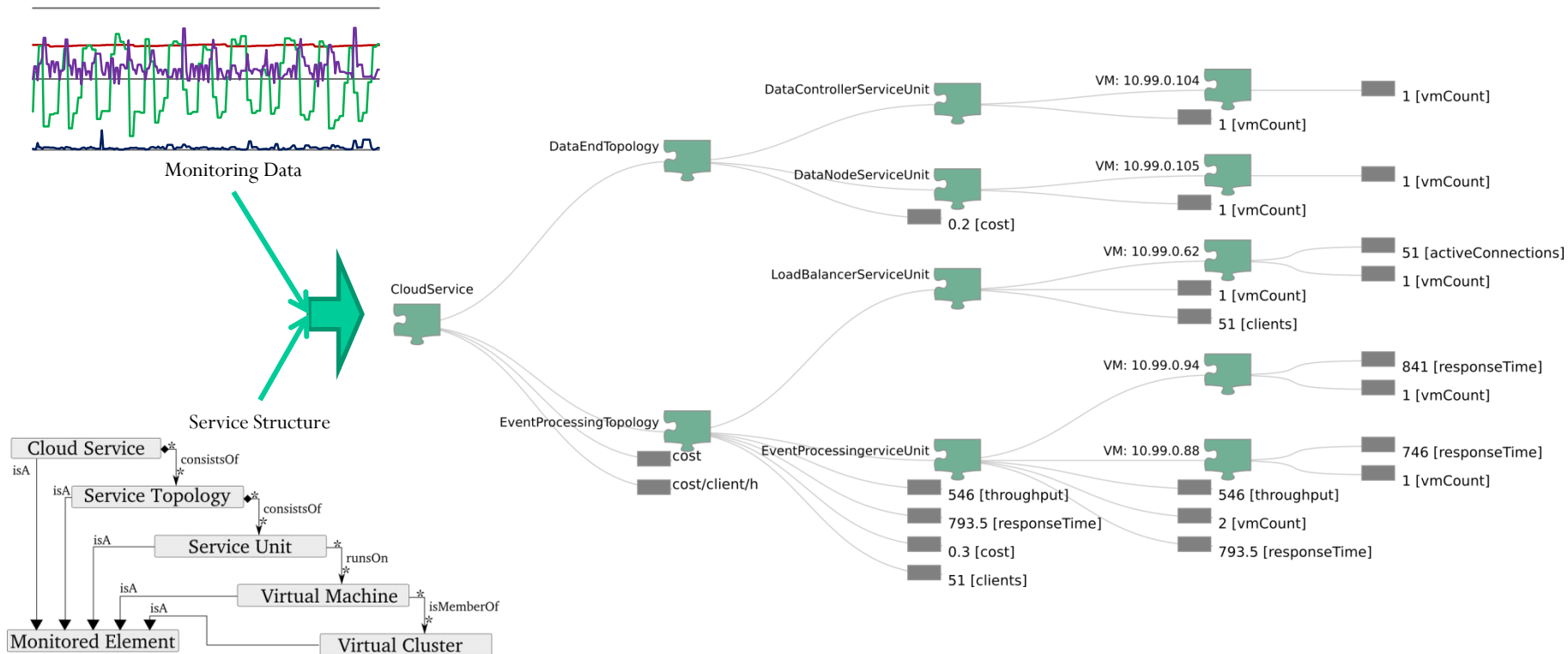
Domain Specific Language

```
ServiceTopology dataEndTopology = ServiceTopology("DataEndTopology")  
    .withServiceUnits(dataControllerUnit, dataNodeUnit,  
        dataControllerOS, dataNodeOS);
```

```
dataEndTopology.constrainedBy(  
    Constraint.MetricConstraint("DataEnd_C01",  
        new Metric("metric", "unit")).lessThan("value"));
```

Monitoring elastic cloud services

Logically structuring monitoring information



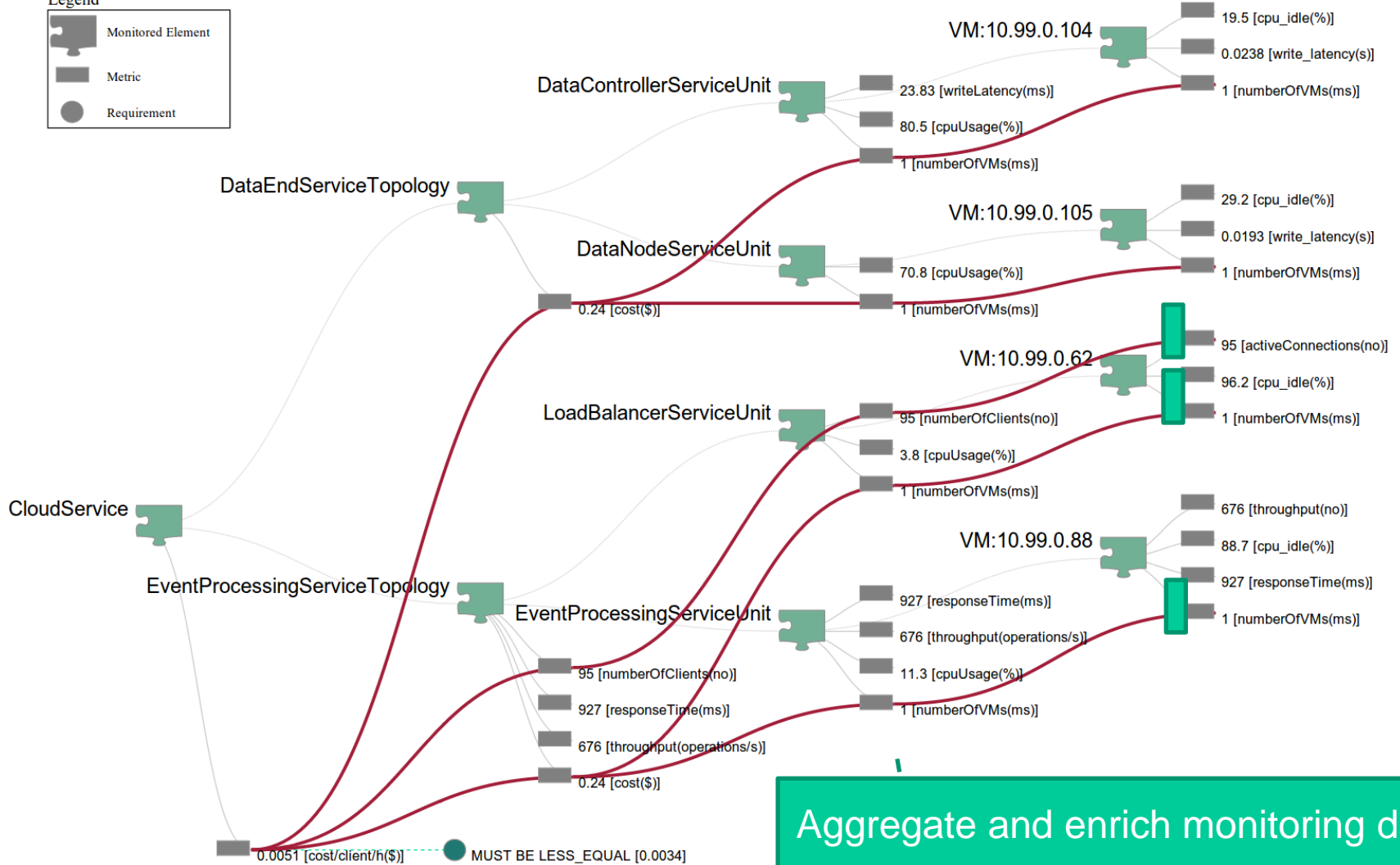
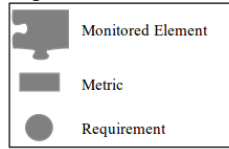
DEMO:

PROGRAMMING, DEPLOYING AND CONTROLLING ELASTIC CLOUD SERVICES

Monitoring elastic cloud services

Structuring and enriching monitoring information

Legend



Aggregate and enrich monitoring data

Demo Screenshots

Programming

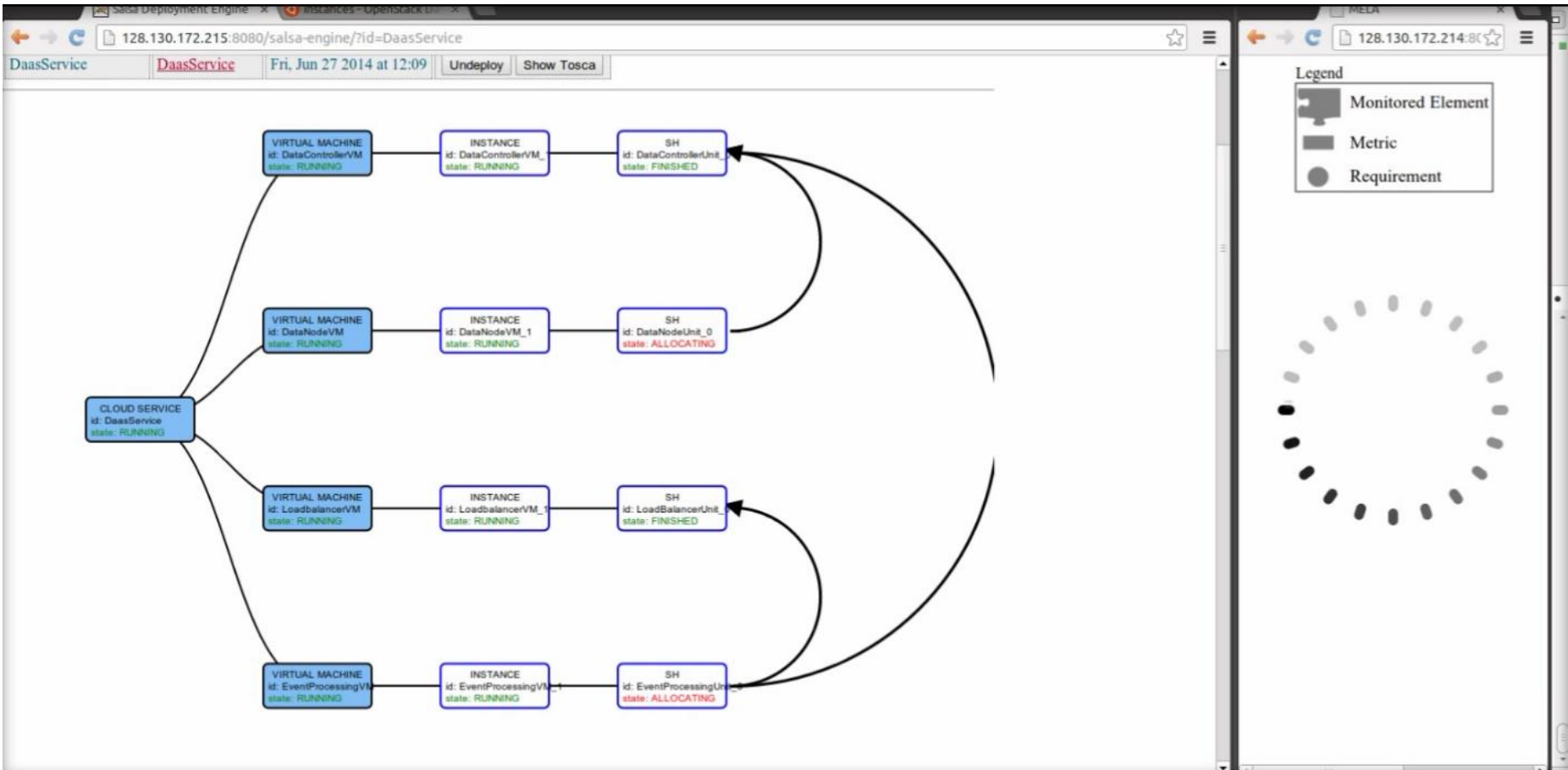
```

ProgrammingAndControllingElasticityWithCOMOT.java x
31 //specify service units in terms of software
32
33 //start with Data End, and first with Data Controller
34 ServiceUnit dataControllerUnit = SingleSoftwareUnit("DataControllerUnit")
35 //software artifacts needed for unit deployment = script to deploy Cassandra
36 .deployedBy(SingleScriptArtifactTemplate("deployDataControllerArtifact", "deployCassandraSeed.sh"))
37 //data controller exposed its IP
38 .exposes(Capability.Variable("DataController_IP_information"))
39 ;
40
41 //specify data node
42
43 ServiceUnit dataNodeUnit = SingleSoftwareUnit("DataNodeUnit")
44 .deployedBy(SingleScriptArtifactTemplate("deployDataNodeArtifact", "deployCassandraNode.sh"))
45 //data node MUST KNOW the IP of cassandra seed, to connect to it and join data cluster
46 .requires(Requirement.Variable("DataController_IP_Data_Node_Req").withName("requiringDataNodeIP"))
47 //express elasticity strategy: Scale IN Data Node when cpu usage < 40%
48 .controlledBy(Strategy("ST1"))
49 .when(Constraint.MetricConstraint("ST1C01", new Metric("cpuUsage", "%").lessThan("40"))
50 .then(Strategy.Action.ScaleIn)
51 );
52
53
54 //Describe a Data End service topology containing the previous 2 software service units
55 ServiceTopology dataEndTopology = ServiceTopology("DataEndTopology")
56 .withServiceUnits(dataControllerUnit, dataNodeUnit);
57
58 //specify constraints on the data topology
59 dataEndTopology.constrainedBy(Constraint.MetricConstraint("DataEndC01", new Metric("cpuUsage", "%").lessThan("80")));
60
61
62
63 //describe the service template which will hold more topologies
64
65 ServiceTemplate serviceTemplate = ServiceTemplate("DaasService")
66 .consistsOfTopologies(dataEndTopology)
67 //defining CONNECT_TO and HOSTED_ON relationships
68 .andRelationships(
69 //Data Controller IP send to Data Node
70 ConnectToRelation("dataNodeToDataController")
71

```

Demo Screenshots

Deploying



The screenshot displays the Salsa Deployment Engine interface. The main window shows a deployment graph for the service 'DaasService'. The graph consists of the following components:

- CLOUD SERVICE** (id: DaasService, state: RUNNING) - The root of the deployment.
- VIRTUAL MACHINE** (id: DataControllerVM, state: RUNNING) - Connected to the cloud service.
- INSTANCE** (id: DataControllerVM, state: RUNNING) - Connected to the DataControllerVM.
- SH** (id: DataControllerUnit, state: FINISHED) - Connected to the DataControllerVM instance.
- VIRTUAL MACHINE** (id: DataNodeVM, state: RUNNING) - Connected to the cloud service.
- INSTANCE** (id: DataNodeVM_1, state: RUNNING) - Connected to the DataNodeVM.
- SH** (id: DataNodeUnit_0, state: ALLOCATING) - Connected to the DataNodeVM instance.
- VIRTUAL MACHINE** (id: LoadbalancerVM, state: RUNNING) - Connected to the cloud service.
- INSTANCE** (id: LoadbalancerVM, state: RUNNING) - Connected to the LoadbalancerVM.
- SH** (id: LoadBalancerUnit, state: FINISHED) - Connected to the LoadbalancerVM instance.
- VIRTUAL MACHINE** (id: EventProcessingVM, state: RUNNING) - Connected to the cloud service.
- INSTANCE** (id: EventProcessingVM, state: RUNNING) - Connected to the EventProcessingVM.
- SH** (id: EventProcessingUnit, state: ALLOCATING) - Connected to the EventProcessingVM instance.

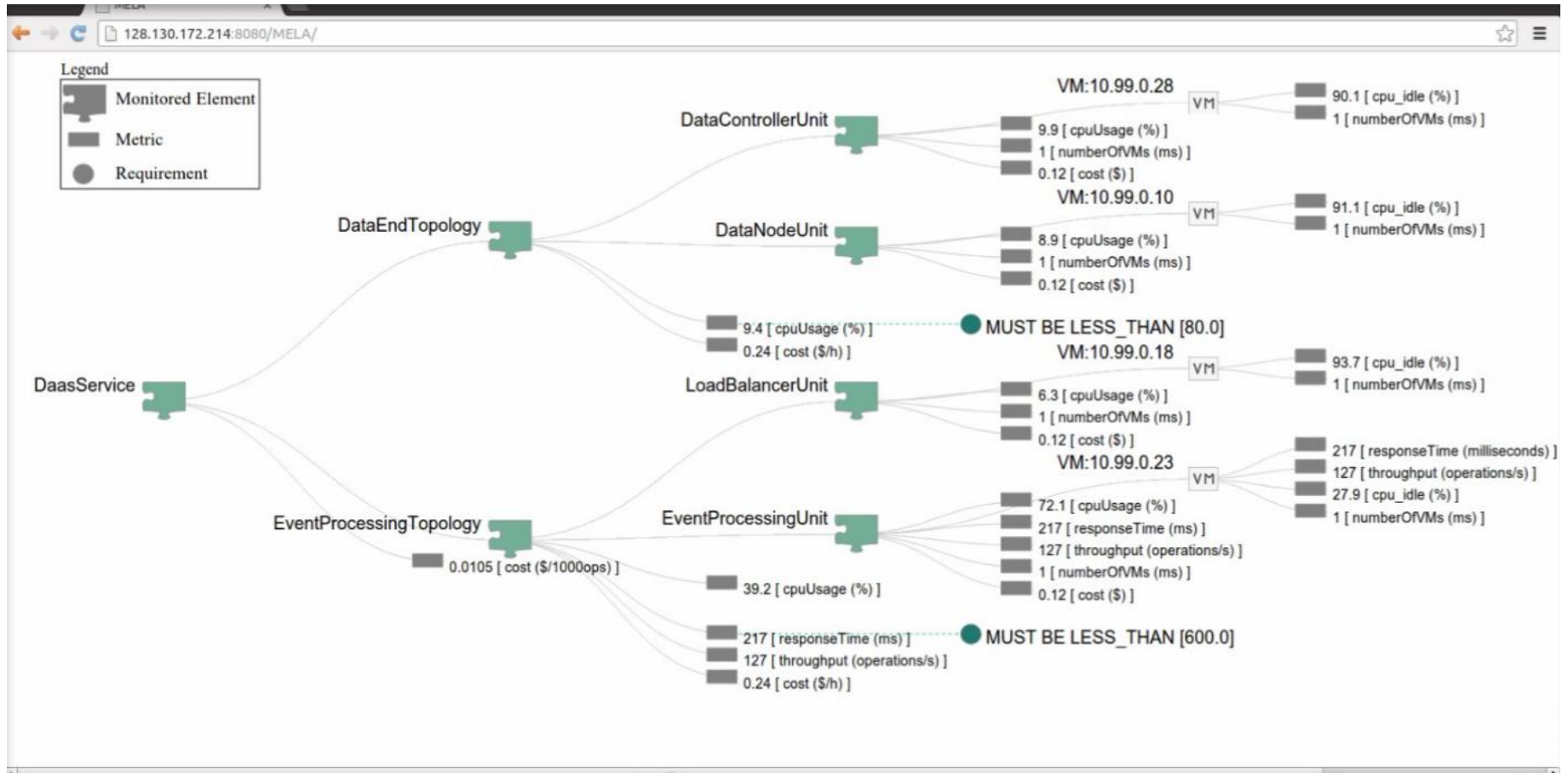
On the right side, there is a monitoring dashboard with a legend:

- Monitored Element**: Represented by a grey square.
- Metric**: Represented by a grey rectangle.
- Requirement**: Represented by a grey circle.

The dashboard shows a circular arrangement of these elements, with some elements highlighted in black, indicating their current state or requirements.

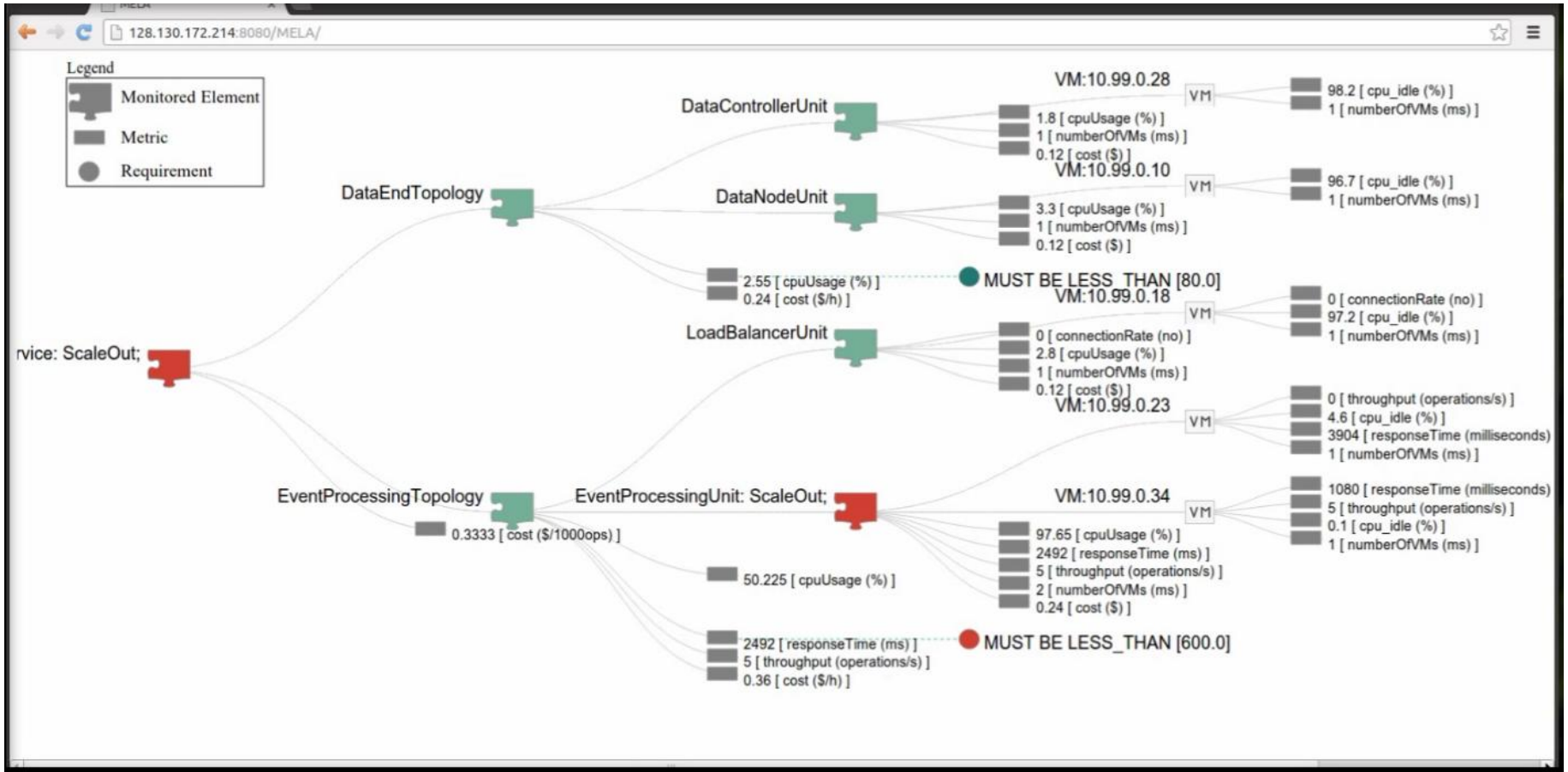
Demo Screenshots

Monitoring



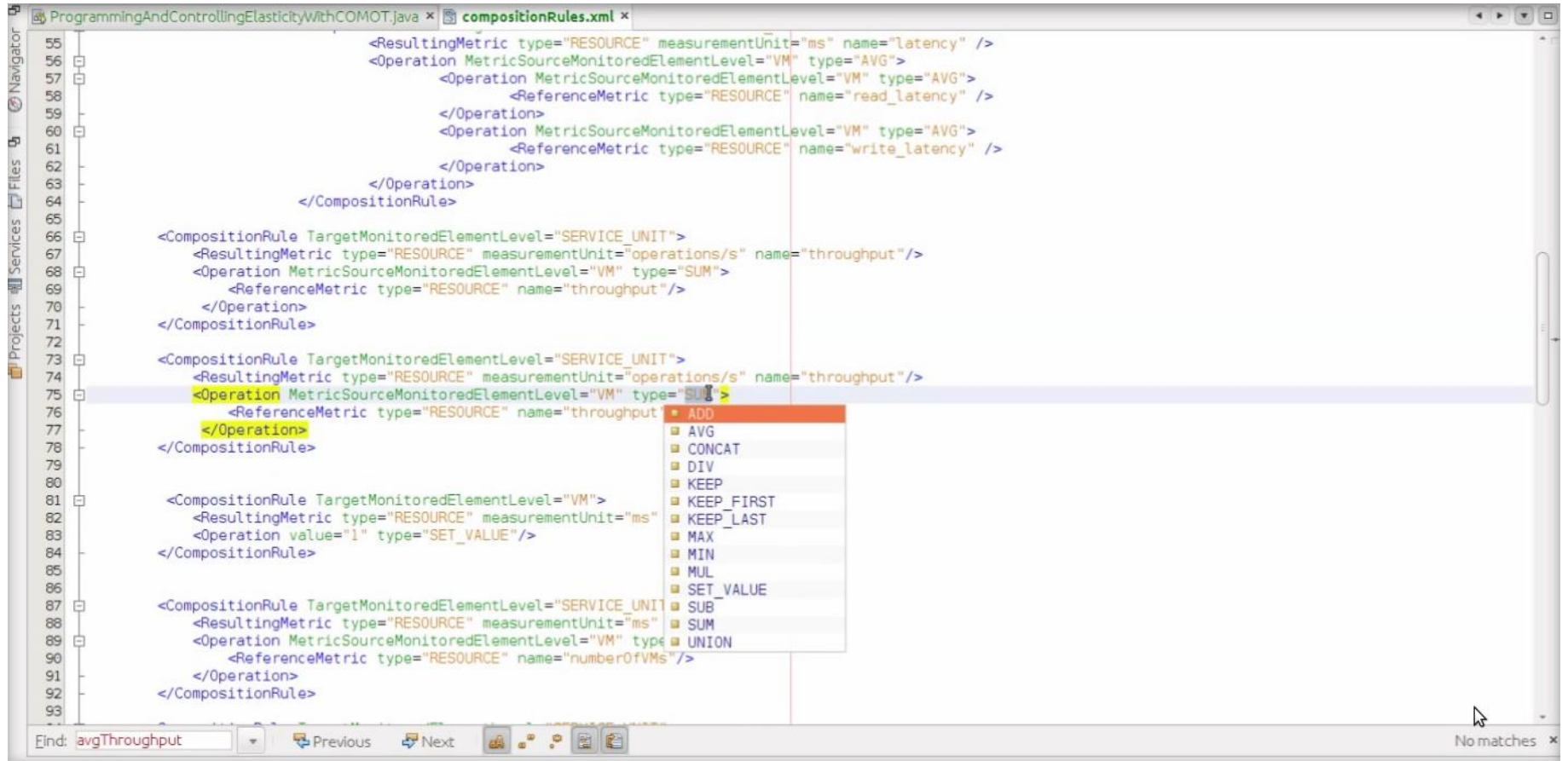
Demo Screenshots

Controlling



Demo Screenshots

Customizing metrics



```

55     <ResultingMetric type="RESOURCE" measurementUnit="ms" name="latency" />
56     <Operation MetricSourceMonitoredElementLevel="VM" type="AVG">
57         <Operation MetricSourceMonitoredElementLevel="VM" type="AVG">
58             <ReferenceMetric type="RESOURCE" name="read_latency" />
59         </Operation>
60     </Operation>
61     <Operation MetricSourceMonitoredElementLevel="VM" type="AVG">
62         <ReferenceMetric type="RESOURCE" name="write_latency" />
63     </Operation>
64 </CompositionRule>
65
66 <CompositionRule TargetMonitoredElementLevel="SERVICE_UNIT">
67     <ResultingMetric type="RESOURCE" measurementUnit="operations/s" name="throughput" />
68     <Operation MetricSourceMonitoredElementLevel="VM" type="SUM">
69         <ReferenceMetric type="RESOURCE" name="throughput" />
70     </Operation>
71 </CompositionRule>
72
73 <CompositionRule TargetMonitoredElementLevel="SERVICE_UNIT">
74     <ResultingMetric type="RESOURCE" measurementUnit="operations/s" name="throughput" />
75     <Operation MetricSourceMonitoredElementLevel="VM" type="SUM">
76         <ReferenceMetric type="RESOURCE" name="throughput" />
77     </Operation>
78 </CompositionRule>
79
80
81 <CompositionRule TargetMonitoredElementLevel="VM">
82     <ResultingMetric type="RESOURCE" measurementUnit="ms" />
83     <Operation value="1" type="SET_VALUE" />
84 </CompositionRule>
85
86
87 <CompositionRule TargetMonitoredElementLevel="SERVICE_UNIT">
88     <ResultingMetric type="RESOURCE" measurementUnit="ms" />
89     <Operation MetricSourceMonitoredElementLevel="VM" type="SUM">
90         <ReferenceMetric type="RESOURCE" name="numberOfVMs" />
91     </Operation>
92 </CompositionRule>
93

```

Context menu options:

- ADD
- AVG
- CONCAT
- DIV
- KEEP
- KEEP_FIRST
- KEEP_LAST
- MAX
- MIN
- MUL
- SET_VALUE
- SUB
- SUM
- UNION

Find: avgThroughput | Previous | Next | No matches

Demo Screenshots

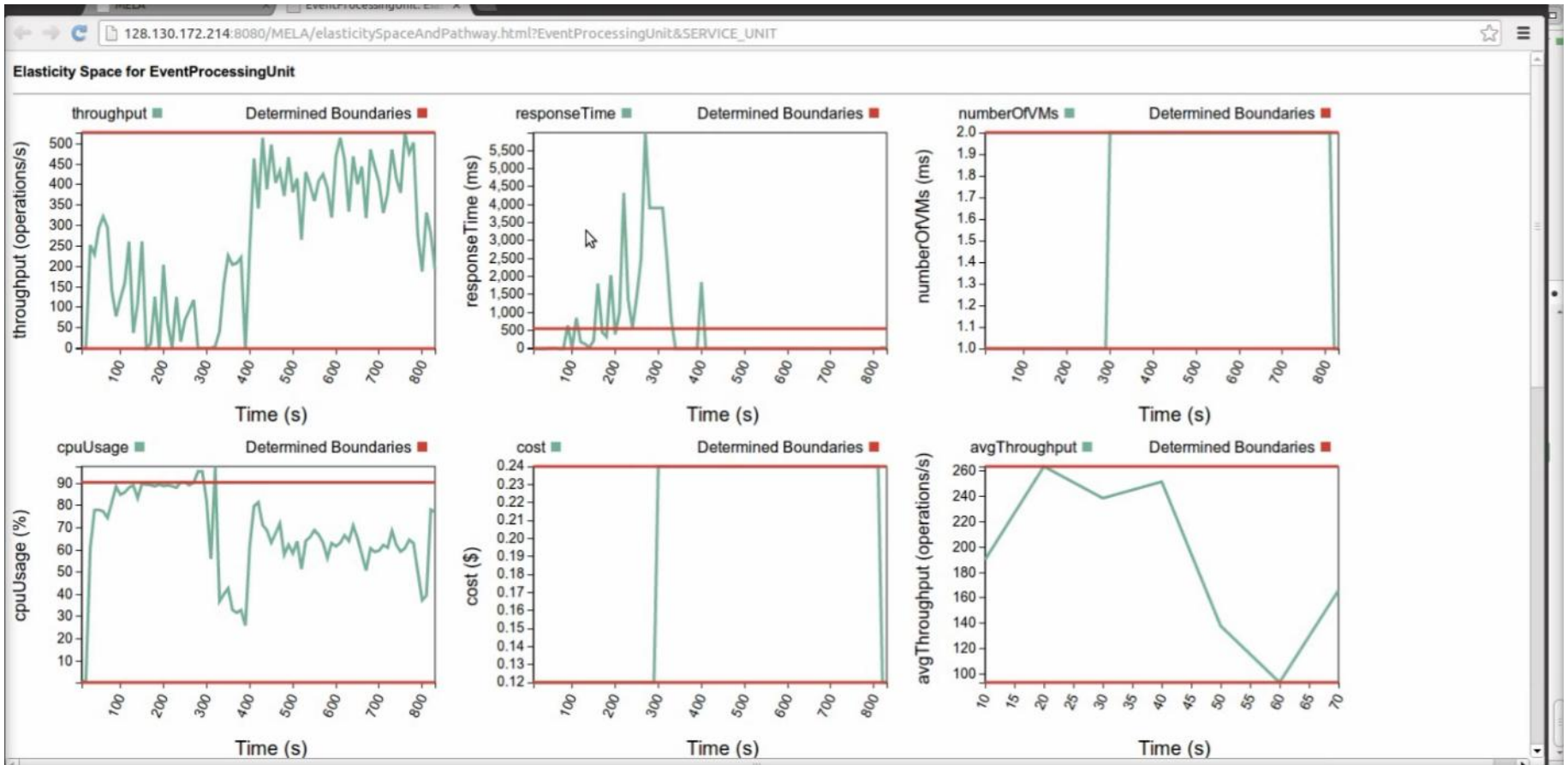
Customizing elasticity requirements

```

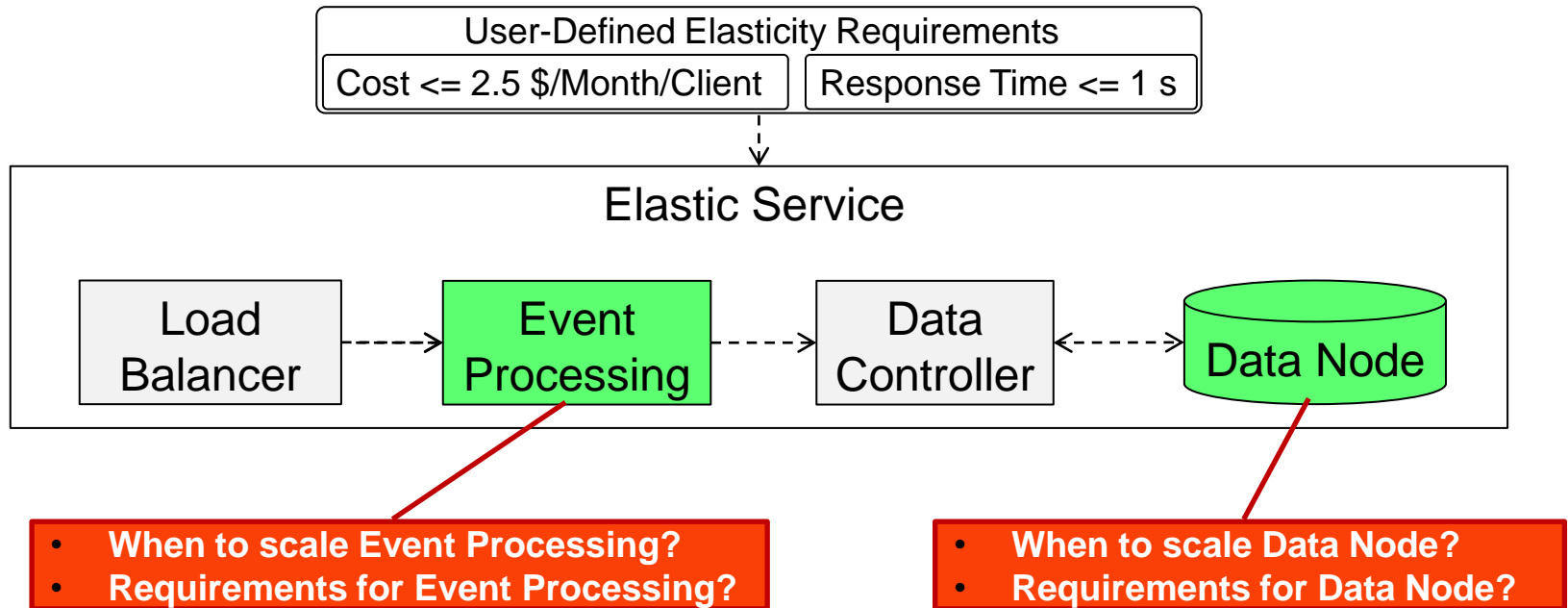
85 //add the service units belonging to the event processing topology
86 ServiceUnit eventProcessingUnit = SingleSoftwareUnit("EventProcessingUnit")
87   .deployedBy(SingleScriptArtifactTemplate("deployEventProcessingArtifact", "deployEventProcessing.sh"))
88   //event processing must register in Load Balancer, so it needs the IP
89   .requires(Requirement.Variable("EventProcessingUnit_LoadBalancer_IP_Req"))
90   //event processing also needs to query the Data Controller to access data
91   .requires(Requirement.Variable("EventProcessingUnit_DataController_IP_Req"))
92   //scale IN if throughput < 200 and responseTime < 200
93   .controlledBy(Strategy("ST2"))
94     .when(Constraint.MetricConstraint("ST2C01", new Metric("responseTime", "ms")).lessThan("200"))
95     .and(Constraint.MetricConstraint("ST2C02", new Metric("throughput", "operations/s")).lessThan("200"))
96     .then(Strategy.Action.ScaleIn)
97   );
98
99 //add the service units belonging to the event processing topology
100 ServiceUnit loadbalancerUnit = SingleSoftwareUnit("LoadBalancerUnit")
101   //load balancer must provide IP
102   .exposes(Capability.Variable("LoadBalancer_IP_information"))
103   .deployedBy(SingleScriptArtifactTemplate("deployLoadBalancerArtifact", "deployLoadBalancer.sh"));
104
105 //Describe a Data End service topology containing the previous 2 software service units
106 ServiceTopology dataEndTopology = ServiceTopology("DataEndTopology")
107   .withServiceUnits(dataControllerUnit, dataNodeUnit //add also OS units to topology
108     , dataControllerVM, dataNodeVM
109   );
110
111 //specify constraints on the data topology
112 //thus, the CPU usage of all Service Unit instances of the data end Topology must be below 80%
113 dataEndTopology.constrainedBy(Constraint.MetricConstraint("DataEndC01", new Metric("cpuUsage", "%")).lessThan("80"));
114
115 //define event processing unit topology
116 ServiceTopology eventProcessingTopology = ServiceTopology("EventProcessingTopology")
117   .withServiceUnits(loadbalancerUnit, eventProcessingUnit //add vm types to topology
118     , loadbalancerVM, eventProcessingVM
119   );
120
121 eventProcessingTopology.constrainedBy(Constraint.MetricConstraint("C02", new Metric("responseTime", "ms")).lessThan("600"));
122
123 //describe the service template which will hold more topologies
124 ServiceTemplate serviceTemplate = ServiceTemplate("DaasService")
125   .consistsOfTopologies(dataEndTopology)
  
```

Demo Screenshots

Elasticity Space and Boundaries



Analyzing elasticity of cloud services



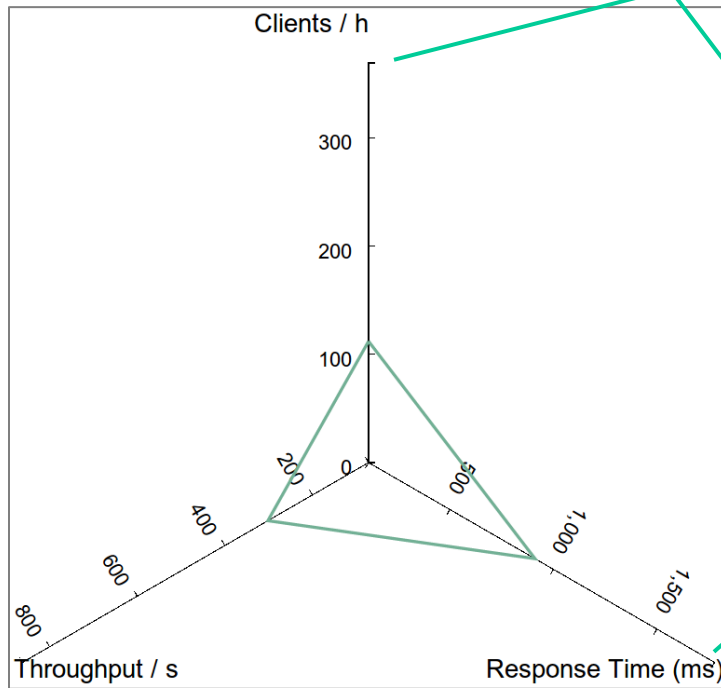
Analyzing elasticity of cloud services

Elasticity Space and Boundary

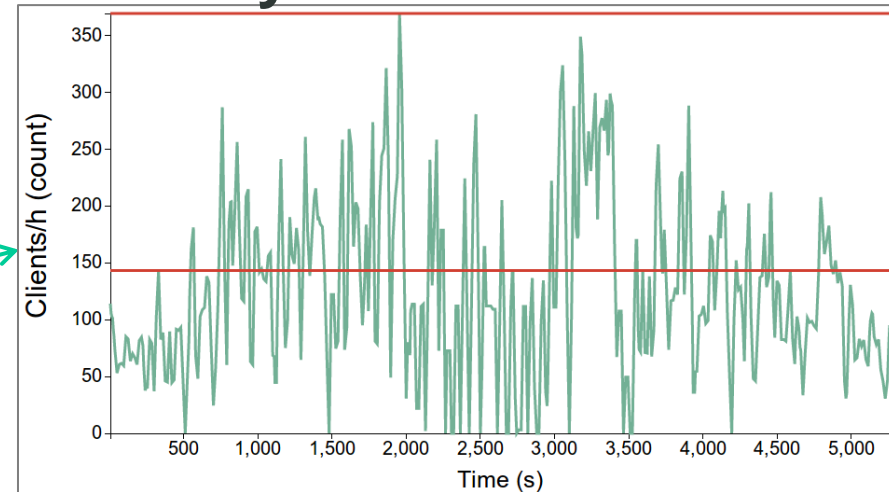
User-Defined Elasticity Requirements

Cost \leq 2.5 \$/Month/Client

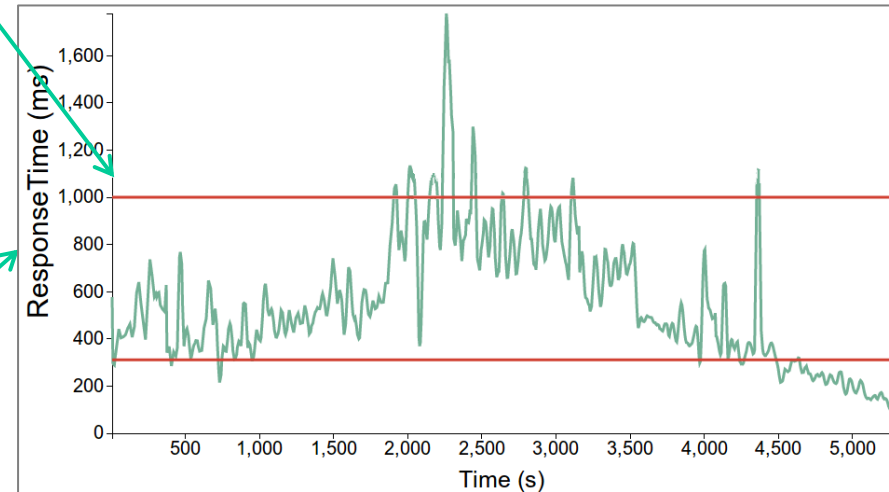
Response Time \leq 1 s



Elasticity Space Snapshot



Elasticity Space "Clients/h" Dimension



Elasticity Space "Response Time" Dimension

Conclusions

- Elastic cloud services
 - Elasticity capabilities
- COMOT
 - Domain specific language for programming elastic cloud services
 - Monitoring of elastic cloud services
 - Analyzing elasticity of cloud services
 - Controlling elasticity of cloud services

The Vienna Elastic Computing Model (<http://dsg.tuwien.ac.at/research/viecom>)

Distributed Systems Group (<http://dsg.tuwien.ac.at/>)

Vienna University of Technology (<http://www.tuwien.ac.at/>)

Work partially supported by the **European Commission** in terms of the **CELAR FP7** project (<http://www.celarccloud.eu/>)