Architectures for Autonomic Service Management in Cloud-based Systems

E.Casalicchio  L.Silvestri
DISP
Università di Roma “Tor Vergata”

Partially founded by Amazon Web Services
Agenda

• Motivations
• Architectural components requirements
• Iaas Providers features survey
• Autonomic service management architectures
• Experimental setup and evaluation
• Lessons learned
Motivations

• How can an ASP use public IaaS providers services to dynamically control resources and satisfy SLAs?
• Do cloud providers offer really working autonomic service management features and services?
• How can an autonomic service management architecture be implemented?
• What autonomic functionalities should be implemented directly by the ASP?
From Requirements To Architectures

Autonomic Service Management Requirements

Survey on IaaS Provider Services

Architecture design
Architectural Components Required

- Performance & Workload Monitors
- Analyzer
- Planner
- Resource Manager
- Load Balancer
Cloud Providers Taxonomy

- Customization Model
- Billing Model
- Interface Type
- Load Balancing
- SLAs
- Monitoring Services
- Auto Scaling Services

Cloud IaaS Survey

<table>
<thead>
<tr>
<th>Service</th>
<th>Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Web Services</td>
<td>aws.amazon.com</td>
</tr>
<tr>
<td>AT&amp;T Synaptic</td>
<td>synaptic.att.com</td>
</tr>
<tr>
<td>CloudSigma</td>
<td>cloudsigma.com</td>
</tr>
<tr>
<td>ElasticHosts</td>
<td>elastichosts.com</td>
</tr>
<tr>
<td>FlexiScale</td>
<td>flexiant.com</td>
</tr>
<tr>
<td>GoGrid</td>
<td>gogrid.com</td>
</tr>
<tr>
<td>JoyentCloud</td>
<td>joyentcloud.com</td>
</tr>
<tr>
<td>Layeredtech</td>
<td>layeredtech.com</td>
</tr>
<tr>
<td>Locaweb</td>
<td>locaweb.com.br</td>
</tr>
<tr>
<td>Opsource</td>
<td>opsourcenet.net</td>
</tr>
<tr>
<td>Rackspace</td>
<td>rackspacecloud.com</td>
</tr>
<tr>
<td>ReliaCloud</td>
<td>reliacloud.com</td>
</tr>
<tr>
<td>RSAWEB</td>
<td>rsaweb.co.za</td>
</tr>
<tr>
<td>SliceHost</td>
<td>slicehost.com</td>
</tr>
<tr>
<td>Storm</td>
<td>stormondemand.com</td>
</tr>
<tr>
<td>Terremark</td>
<td>vcloudeprexpress.terremark.com</td>
</tr>
<tr>
<td>VPSNET</td>
<td>vps.net</td>
</tr>
</tbody>
</table>
Architectural models

- Extreme ASP control
- Full ASP control
- Partial ASP control
- Limited ASP control
Extreme ASP Control

Application Requests

- Provisioning Manager
- SLA Analyzer
- Workload Monitor
- Performance Monitor

Load Balancer

VM Allocator

Monitoring Agent

Application

VM

Application Service Provider
Full ASP Control

Application Service Provider

Load Balancer

Provisioning Manager

SLA Analyzer

Workload Monitor

Performance Monitor

VM Allocation Service

IaaS Provider

Monitoring Agent

Application

VM

VM

VM

VM
Partial ASP Control

Application requests

Application Service Provider

Provisioning Manager

SLA Analyzer

Workload Monitor

Performance Monitor

Load Balancing Service

VM Allocation Service

Performance Monitoring Service

IaaS Provider

Application

VM

VM

VM

VM
Limited ASP Control

Application requests

Auto Scaling Service

Provisioning Manager

SLA Analyzer

Workload Monitoring Service

Load Balancing Service

VM Allocation Service

Performance Monitoring Service

Application

VM

VM

VM

VM
Partial ASP-C Implementation

- Implementation based on
  - Amazon EC2
  - Our previous work on QoS-aware resource provisioning [COMNET'11 (submitted)]
Testbed setup

• Workload generation based on
  – Mediawiki Application
  – Httpperf (using Wikipedia access log)
• Testbed setup
  – 1-10 Application Servers (m1.small instances)
  – 1 DBMS (m1.large instance)
• Workload tuning
  – 4 - 20 req/sec
  – Max Util. 65% (normal working conditions)
Partial vs Limited ASP Control

**Test Workload**

- **Total Requests**

**Request Count**

- **Total Requests**
- **Request Served P-ASP-C**
- **Request Served L-ASP-C**
Partial vs Limited ASP Control

**Host Count**
- P-ASP-C
- L-ASP-C

**Latency**
- P-ASP-C
- L-ASP-C

Time (minutes)
Lessons Learned

• Problems with AWS AutoScaling
  – Unhealthy hosts not replaced
  – Elastic Load Balancer bugs
  – Request count not available if all hosts become unhealthy!

• Available performance metrics are not exhaustive

• Experimental Results: more Control on Autonomic Cycle → Better resource management