Monitoring Elastic Cloud Services

Demetris Trihinas
trihinas@cs.ucy.ac.cy

Advanced School on Service Oriented Computing (SummerSoc 2014)
30 June – 5 July, Hersonissos, Crete, Greece
Presentation Outline

• Elasticity in Cloud Computing
• Cloud Service Monitoring Challenges
• Existing Monitoring Tools and their Limitations
• JCatascopia Monitoring System
  • Architecture
  • Features
  • Evaluation
• Conclusions and Future Work
Elasticity in Cloud Computing

• Ability of a system to **expand** or **contract** its dedicated resources to meet the current demand
Cloud Monitoring Challenges

- Monitor heterogeneous types of information and resources
- Extract metrics from multiple levels of the Cloud
  - Low-level metrics (i.e. CPU usage, network traffic)
  - High-level metrics (i.e. application throughput, latency, availability)
- Metrics collected at different time granularities
Cloud Monitoring Challenges

- Operate on any Cloud platform
- Monitor Cloud services deployed across multiple Cloud platforms
- Detect configuration changes in a cloud service
  - Application topology changes (e.g. new VM added)
  - Allocated resource changes (e.g. new disk attached to VM)

Existing Monitoring Tools
Cloud Specific Monitoring Tools

Benefits

- Provide MaaS capabilities
- Fully documented
- Easy to use
- Well integrated with underlying platform

Limitations

- Commercial and proprietary which limits them to operating on specific Cloud IaaS providers
General Purpose Monitoring Tools

Benefits

- Open-source
- Robust and light-weight
- System level monitoring
- Suitable for monitoring Grids and Computing Clusters

Limitations

- Not suitable for dynamic (elastic) application topologies
Monitoring Tools with Elasticity Support

- [de Carvalho et al., INM 2011]
  - Nagios + Controller on each physical host to notify *Nagios Server* with a list of instances currently running on the system

- Lattice Monitoring Framework [Clayman et al., NOMS 2011]
  - Controller periodically requests from hypervisor list of current running VMs

**Limitations**

- Special entities required at physical level
- Depend on current hypervisor
JCatascopia Monitoring System
JCatascopia Monitoring System

✓ Open-source
✓ Multi-Layer Cloud Monitoring
✓ Platform Independent
✓ Capable of Supporting Elastic Applications
✓ Interoperable
✓ Scalable

JCatascopia Architecture
Monitoring Agents

- Light-weight monitoring instances
- Deployable on physical nodes or virtual instances
- Responsible for the metric collection process
- Aggregate and distribute collected metrics (pub/sub)
Monitoring Probes

- The actual metric collectors managed by Monitoring Agents
- JCatascopia Probe API
- Dynamically deployable to Monitoring Agents
- Filtering mechanism at Probe level
Monitoring Servers

- Receive metrics from Monitoring Agents
- process and store metrics in Monitoring Database
- Handle user metric and configuration requests
- Hierarchy of Monitoring Servers for greater scalability
JCatascopia Architecture

- JCatascopia REST API
- JCatascopia-Web User Interface
- JCatascopia Database Interface
  - Allows users to utilize their own Database solution with JCatascopia
  - Currently available: MySQL, Cassandra
Dynamic Agent Discovery and Removal

**Benefits**
- Monitoring Servers are agnostic of Agent network location
- Agents appear dynamically

**Eliminated the need to**
- Restart or reconfigure Monitoring System
- Depend on underlying hypervisor
- Require directory service with Agent locations
Metric Subscription Rule Language

- Aggregate single instance metrics
  
  \[
  \text{SUM(errorCount)}
  \]

- Generate high-level metrics at runtime
  
  \[
  \text{DBthroughput} = \text{AVG}(\text{readps} + \text{writeps})
  \]

Subscription Rule Example

Average DBthroughput from the low-level metrics readps and writeps of a database cluster comprised of \( N \) nodes:

DBthroughput = AVG(readps + writeps)
MEMBERS = [id1, ... ,idN]
ACTION = NOTIFY(<25,>75%)
Adaptive Filtering

• Simple fixed uniform range filter windows are not effective:
  • i.e. filter $currentValue$ if in window $previousValue \pm R$
  • No guarantee that any values will be filtered at all

• Adaptive filter window range
  • window range (R) is not static but depends on percentage of values previously filtered
JCatascopia Evaluation
Evaluation

- Validate JCatascopia functionality and performance
- Compare JCatascopia to other Monitoring Tools
  - Ganglia
  - Lattice Monitoring Framework
- Testbed
  - Different domains of Cloud applications
  - Various VM flavors
  - 3 public Cloud providers and 1 private Cloud
## Testbed

<table>
<thead>
<tr>
<th>Cloud Provider</th>
<th>VM no.</th>
<th>VM Flavor</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRNET Okeanos public Cloud</td>
<td>15</td>
<td>1GB RAM, 10GB Disk, Ubuntu Server 12.04 LTS</td>
<td>12 VMs Cassandra, 3 VMs YCSB Clients</td>
</tr>
<tr>
<td>Flexiant FlexiScale platform</td>
<td>10</td>
<td>2 VCPU, 2GB RAM, 10GB Disk, Debian 6.07 (Squeeze)</td>
<td>HASCOP</td>
</tr>
<tr>
<td>Amazon EC2</td>
<td>10</td>
<td>m1.small with CentOS 6.4 (1VCPU, 1.7GB RAM, 160GB Disk)</td>
<td>an attributed, multi-graph clustering algorithm</td>
</tr>
<tr>
<td>OpenStack Private Cloud</td>
<td>60</td>
<td>2 VCPU, 2GB RAM, 10GB Disk, Ubuntu Server 12.04 LTS</td>
<td></td>
</tr>
</tbody>
</table>

We have deployed on all VMs JCastascopia Monitoring Agents, Ganglia gmond and Lattice DataSources
# Testbed - Available Probes

<table>
<thead>
<tr>
<th>Probe</th>
<th>Metrics</th>
<th>Period (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>cpuUserUsage, cpuNiceUsage, cpuSystemUsage, cpuldle, cpuIOWait</td>
<td>10</td>
</tr>
<tr>
<td>Memory</td>
<td>memTotal, memUsed, memFree, memCache, memSwapTotal, memSwapFree</td>
<td>15</td>
</tr>
<tr>
<td>Network</td>
<td>netPacketsIN, netPacketsOUT, netBytesIN, netBytesOUT</td>
<td>20</td>
</tr>
<tr>
<td>Disk Usage</td>
<td>diskTotal, diskFree, diskUsed</td>
<td>60</td>
</tr>
<tr>
<td>Disk IO</td>
<td>readkbps, writekbps, iotime</td>
<td>40</td>
</tr>
<tr>
<td>Cassandra</td>
<td>readLatency, writeLatency</td>
<td>20</td>
</tr>
<tr>
<td>YCSB</td>
<td>clientThroughput, clientLatency</td>
<td>10</td>
</tr>
<tr>
<td>HASCOP</td>
<td>clustersPerIter, iterElapTime, centroidUpdTime, pTableUpdTime, graphUpdTime</td>
<td>20</td>
</tr>
</tbody>
</table>
Experiment 1. Elastically Adapting Cassandra Cluster

- Scale out Cassandra cluster to cope with increasing workload
- Experiment uses 15 VMs in Okeanos cluster
- Subscription Rule to notify Provisioner to add VM when scaling condition violated:

\[
\text{cpuTotalUsage} = \text{AVG}(1 - \text{cpuIdle}) \\
\text{MEMBERS} = [\text{id}_1, \ldots, \text{id}_N] \\
\text{ACTION} = \text{NOTIFY}(\geq 75\%)
\]

<table>
<thead>
<tr>
<th>VMs</th>
<th>Probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>YCSB Clients</td>
<td>YCSB</td>
</tr>
<tr>
<td>Cassandra</td>
<td>CPU, Memory, Network, DiskIO, Cassandra</td>
</tr>
</tbody>
</table>
Experiment 1. Elastically Adapting Cassandra Cluster

Monitoring Agent Runtime Impact

YCSB Agent Utilization

Cassandra Agent Utilization
Experiment 2. Monitoring a Cloud Federation Environment

- Monitor an application topology spread across multiple Clouds:
  - OpenStack (10 VMs)
  - Amazon EC2 (10 VMs)
  - Flexiant (10 VMs)

- Compare JCatascopia, Ganglia and Lattice runtime footprint

- Compare JCatascopia and Ganglia network utilization

<table>
<thead>
<tr>
<th>VMs</th>
<th>Probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASCOP</td>
<td>CPU, Memory, DiskUsage, HASCOP</td>
</tr>
</tbody>
</table>
When in need of application-level monitoring, for a small runtime overhead, JCatascopia can reduce monitoring network traffic and consequently monitoring cost.
Experiment 3. JCatascopia Scalability Evaluation

- Experiment uses the 60 VMs on OpenStack private Cloud to scale a HASCOP cluster
- 1 Monitoring Server for 60 Agents
- Subscription Rule:

\[
\text{hascopIterElapsedTime} = \text{AVG}(\text{iterElapTime}) \\
\text{MEMBERS} = [\text{id1}, \ldots, \text{idN}] \\
\text{ACTION} = \text{NOTIFY(ALL)}
\]

<table>
<thead>
<tr>
<th>VMs</th>
<th>Probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASCOP</td>
<td>CPU, Memory, DiskUsage, HASCOP</td>
</tr>
</tbody>
</table>
Scalability Evaluation

Archiving time grows linearly

\[ f(x) = 5.11x \cdot 60.79, \quad R^2 = 0.9829 \]
Experiment 3. JCatascopia Scalability Evaluation

New Setup

- 2 Intermediate Monitoring Servers which aggregate metrics from underlying Agents
- 1 root Monitoring Server

<table>
<thead>
<tr>
<th>VMs</th>
<th>Probes</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASCOP</td>
<td>CPU, Memory, DiskUsage, HASCOP</td>
</tr>
</tbody>
</table>

SummerSoc 2014, Crete, Greece, 1 July 2014
When archiving time is high, we can redirect monitoring metric traffic through Intermediate Monitoring Servers, allowing the monitoring system to scale.
Conclusions

• Experiments on public and private Cloud platforms show that JCatascopia is:
  • capable of supporting automated elasticity controllers
  • able to adapt in a fully automatic manner when elasticity actions are enforced
  • open-source, interoperable, scalable and has a low runtime footprint
Future Work

• Further pursue adaptive filtering

• Enhance Probes with adaptive sampling
  • Adjust sampling rate when stable phases are detected

• Create Monitoring Toolkit for PaaS Cloud applications

• Provide Monitoring as a Service to Cloud consumers
Acknowledgements

www.celarcloud.eu

co-funded by the European Commission

JCatascopia

https://github.com/CELAR/cloud-ms
Thank you!
BACKUP SLIDES
JCatascopio Architecture
Dynamic Agent Removal

- Heartbeat monitoring to detect when Agents:
  - Removed due to scaling down elasticity actions
  - Temporary unavailable (network connectivity issues)
Monitoring Agents

 JCatascopia MS Agent

 CONNETS to tcp://SERVER_IP:4242

 Metric Processor
 Process metrics, Aggregation, create message

 Aggregator

 Distributor

metrics

Controller

Ping MS Server, send metadata of available metrics

Server Connector

Remote Requests/Responses to/from MS Server

Controller

Connects to tcp://SERVER_IP:4245

BINDS to tcp://AGENT_IP:4243

Configure Probe parameter(s)

Probes Agent Components

Host & VM Level

Cassandra

My Custom App

Configure Probe parameter(s)

CPU Probe
Memory Probe
Network Probe
Disk I/O Probe
Process Probe
Cassandra Probe
Tomcat Probe
Custom App Probe
Latency Probe
Throughput Probe
Monitoring Servers

JCatascopia MS Server

MS Agent
MS Agent
MS Agent
MS Agent

Agent Listener

Metric Queue

Processor

DB Handler

Local DB

CleanUp Daemon

HeartBeat Monitor

Remove

Add

Agent Map
Metric Map
Sub Map

Subscription Scheduler

Retrieve and update

Aggregator

ReDistributor

If Server is an intermediate then forward metrics

Control Queue

Control Listener

User

Decision Module

Subscribe to aggregated metrics

BINDS to tcp://SERVER_IP:4242

BINDS to tcp://SERVER_IP:4245

CONNECTS to tcp://NEXT_HOP_IP:4242