Model Based Control for Multi-Cloud Applications

May 18th, 2013

Authors:
Marco Miglierina
Giovanni P. Gibilisco
Danilo Ardagna
Elisabetta Di Nitto

Speaker:
Giovanni P. Gibilisco
Cloud and elasticity

 Courtesy of Microsoft
Quality of service on the Cloud

• No native mechanisms to guarantee the Quality of Service required by specific application domains
• Claims: 99.95% of availability (Amazon, Azure)
• Actual observations$^1$:
  • From users’ perspective:

<table>
<thead>
<tr>
<th>Provider</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2 EU</td>
<td>96.32%</td>
</tr>
<tr>
<td>Google App Engine</td>
<td>93.05%</td>
</tr>
<tr>
<td>Windows Azure</td>
<td>95.39%</td>
</tr>
</tbody>
</table>

• Outages: Amazon$^2$ (Apr 2011), Google$^3$ (May 2011), Azure$^4$ (Feb 2012)

Goal

- High availability is usually obtained by **replication** of critical components
- Solution: exploit two or more Clouds as replication method (**multi-Cloud application**)
Goal

- High availability is usually obtained by replication of critical components.
- Solution: exploit two or more Clouds as replication method (multi-Cloud application).
Goal

- High availability is usually obtained by **replication** of critical components
- Solution: exploit two or more Clouds as replication method (multi-Cloud application)
Our solution

Multi-Cloud Model

Modeling Tool

Design Time

Controller

Actuator

Run Time

Monitors

Multi-Cloud Model

Controller

Actuator

Design Time

Run Time
Modeling multi-Cloud applications

- Physical Node
- DTMC Node
- Logical Node

- Limited Throughput
- Unlimited Throughput

- Autoscaling
- Fixed

1: Load Balancer
2: Cloud 1 Access
3: Cloud 1 Failure
4: Autoscaling Group 1
5: Inhouse Server Access
6: Inhouse Server
7: Cloud 2 Failure
8: Success State

- c0
- 1-c0
- a2
- 1-a2
- a4
- 1-a4
- a5
- 1-a5
- a6
- 1-a6
Modeling multi-Cloud applications

Type: Autoscaling  
Number of VMs: 4  
Max VM SR: 100 reqs/s  
Avg CPU load: 80%  
Cost per VM: 0.3 $/h

Type: Fixed  
Number of VMs: 1  
Max VM SR: 100 reqs/s  
Avg CPU load: 80%  
Cost per VM: 0 $/h
Our solution

Multi-Cloud Model

Design Time

Modeling Tool

Run Time

Controller

Monitors

Actuator
Building the controller

- **Objectives**
  - Guarantee the required availability
  - Minimize costs
Building the controller

- Objectives
  - Guarantee the required availability
  - Minimize costs
The Autoscaling Controller

- **Reference**: average CPU usage \( u \)
- **Control variable**: number of VMs \( n \)
- **Monitored data** at the node (over a time window):
  - Arrival Rate \( AR \)
  - VM Max Service Rate \( sr \)

\[
\bar{n} = \frac{AR}{sr \cdot u}
\]

- Desired Number of VMs

\[
n(k + 1) = \alpha n(k) + (1 - \alpha)\bar{n}
\]

- Number of VMs at the next step
- Convergence factor in \((0,1)\)

- Current number of VMs
The Load Balancer Controller

- **Reference:** System availability – \( v \)
- **Control variable:** traffic distribution probabilities – \( c_i \)
- **Monitored data** (over time window):
  - Incoming requests to node \( i \) - \( IN_i \)
  - Successful requests to node \( i \) – \( OUT_i \)
  - VM Max Service Rate – \( sr \)

\[
v(k+1) - \hat{s}(k+1|k) \leq \beta \cdot \max(0, v(k) - s(k))
\]

- Estimated system availability
- Measured system availability
- Convergence factor in \((0,1)\)
- Arrival Rate – \( AR \)
- VM cost per second
The Load Balancer Controller

\[ \nu(k+1) - \hat{s}(k+1|k) \leq \beta \cdot \max(0, \nu(k) - s(k)) \]
The Load Balancer Controller

• Solution is chosen so to minimize an objective function $J(c_i)$
  • It is built so to allow cost minimization by preferring the most convenient Cloud, and to discourage nodes overloading
Evaluation

- **Objective**
  - Test how the controller is able to **track** the reference system availability
  - Test how the controller reacts to **sudden changes** in the environment, such as Cloud outages or performance degradations

- **Experiment setup**
  - For the evaluation we used Matlab
  - We implemented our controller
  - The environment and the different scenarios were simulated
  - One of the tested scenarios are now presented
Evaluation

Results

System Availability

Time

Availability

Measured
Reference
Evaluation
Results

Cloud 2 has higher availability.

Availability reference goes from 90% to 50%.

Availability reference goes from 70% to 90%.
Conclusions

• We defined an **adaptive approach** able to guarantee **availability requirements**, managing cloud to cloud migration and in-cloud autoscaling policies, **minimizing costs**

• The **controller** is able to track the reference system availability and to react to changes in the environment
Future Work

• Analyze convergence parameters (α and β) and CPU reference (u) setting, *studying optimality* of this choice

• The approach should provide *more realistic* descriptions and features of the current Cloud offer (e.g. pay by the hour)

• The proposed approach should be tested on *real Cloud infrastructures*
Any question?