A Framework for Automatic Resource Provisioning for Private Clouds

J.O. Melendez, A. Biswas, S. Majumdar,
Dept. of Systems and Computer Engineering,
Carleton University,
Ottawa, Canada

B. Nandy
Solana Networks,
Ottawa, CANADA

M. Zaman, P. Srivastava, N. Goel
Cistel Technology, Ottawa, Canada
Outline

- Introduction
- Architecture
- Framework & Implementation
- Performance Results
- Conclusions
Introduction

- Private Cloud
  - Owned by Enterprise/Institution
  - Resources are acquired/released dynamically from a Public Cloud

- Resource Provisioning
  - How to determine the number of Resources?
  - How to change the number of resources dynamically with change in workload?
  - How to maintain a specified Grade of Service (GoS)?

- User submits requests to the cloud
  - Advance Reservation (AR) request
    - Earliest start time, execution time and deadline
  - On-Demand (OD) Request

- The broker performs matchmaking (matching requests to resources) and determines schedulability (whether request can be scheduled on resources based on parameters)

- Broker sends requests to Scheduler component for scheduling

- The broker will acquire or release additional resources from the public cloud based on the required system performance

- The broker also contains a web application with UI for submitting requests
Framework Architecture

- Resource provider specifies desired Grade of Service
  - Can be used to track various performance metrics
  - Blocking Ratio $B = \frac{\text{# of requests rejected}}{\text{Total # of requests}}$
  - ($B_{\text{spec}}$)

- GoS Monitor monitors desired performance metrics, contains logic for testing metrics that will either cause the system to acquire or release resources.

- When GoS Monitor detects that system performance is not meeting the desired level the DRPM uses the Resource Handler to acquire more resources (rule i).

- When GoS Monitor detects that system performance is meeting the desired level DRPM uses the Resource Handler to release resources (rule ii).

- Cloud API Wrapper is a component that uses the public cloud’s API for resource management.
- Administrator inputs metrics to monitor using GoS Specifier
- GoS Monitor monitors metrics
- GoS Monitor determines that system performance is not meeting desired specification and notifies Resource Handler to acquire additional resources
- GoS Monitor determines that system performance is meeting desired specification and resources can be released, notifies Resource Handler to release resources
Implementation Technology

- Java/Spring Framework
- Web Service technology
- Amazon EC2 Instances
### Key Workload Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job execution time</td>
<td>Uniform distribution [0 to 90 min]</td>
</tr>
<tr>
<td>RequestArrivals</td>
<td>Poisson process</td>
</tr>
<tr>
<td>S [used in generating earliest start time]</td>
<td>Uniform distribution [0 to 12 hours]</td>
</tr>
</tbody>
</table>

- Arrival Rate = 0.0053 requests/sec
- Shows the dynamic nature of the system for 2000 requests.
- Left Y axis: captures change in the number of resources (from 4 to 10)
- Right Y axis: capture fluctuation of B with respect to the requests.
Cost Comparison with a Static System

System subjected to two arrival rates: $\lambda_{\text{low}}$ and $\lambda_{\text{high}}$

- Cost per unit time (DPRM) = $f \cdot \text{Cost1} + (1-f) \cdot \text{Cost2}$
  - $f$: load parameter (proportion of time system is subjected to the low arrival rate)
  - Cost 1: for system with $\lambda_{\text{low}}$
  - Cost 2: for system with $\lambda_{\text{high}}$

- Static: Fixed no. of Resources so that $B < B_{\text{spec}}$
- $B_{\text{spec}} = 0.5$
- Figure 1: cost benefit provided by the DPRM-based system
- Figure 2: improvement in cost achieved by the DPRM based system over the static system for different values of $\lambda_{\text{low}}$
- $B_{\text{spec}} = 0.25$ and Mean Execution Time = 25 mins ([5 to 45 mins])
Conclusions

- Focus: Private Cloud based on resources acquired dynamically from a public cloud

- Presented a framework that automatically adjusts the number of resources based on current system load such that a given GoS is maintained

- Leads to lower average number of resources used

- Gives rise to a significantly lower cost in comparison to a static system