Autonomic Provisioning and Application Mapping on Spot Cloud Resources

Daniel J. Dubois, Giuliano Casale
Imperial College London, Department of Computing
2015 International Conference on Cloud and Autonomic Computing
Summarized by Essien Ayanam
CS 788
November 6, 2017
Outline

- Introduction
- Foundation
- System Model
- Approach
- Experiment
- Conclusion
Introduction

- Cloud computing environments provide the flexibility to decide and modify such items as the speed, the number and the lease time of virtual machines (VMs).
- Two often mentioned pricing strategies for renting VMs:
  - On-Demand Pricing
  - Spot Pricing
- Spot Pricing – resources offered at a variable price, spot price, which changes with the market condition.
Introduction

- Advantage of spot instances – Spot price tends to be lower than the on-demand price (bid price)
- Disadvantage – Shortage of resources can lead to the spot price temporarily exceeding the on-demand price (bid price)
- Resource shortage makes the decision of choosing a bid price both difficult and important
Introduction

- The study aims to help users take maximum advantage of spot instances by supporting the following decisions:
  - What type of virtual resources should be rented for a given application?
  - How to efficiently map the components of an application (e.g. web server VMs, a database VMs) to the rented resources?
  - What is the lowest bid price that still allows to fulfill quality of service requirements?
Foundation

- Proposed solution based off an industrial enterprise resource planning cloud application, SAP ERP
- SAP ERP is composed of two components: an application server and a database server
- The application must satisfy some quality requirements in terms of response time in fulfilling requests.
- Problem to solve – find the cheapest way to run the application on a spot cloud system while maintaining the quality requirements
Foundation
System Model

- Proposed model composed of two parts: application and resources
- Goal – determine the rental and allocation policies which consist in:
  - The amount of computational resources to be rented from the cloud provider
  - The mapping of the various application components to these resources
  - Bid price for each resource
System Model - Application

- Modeled as a closed queueing network $QN$ of $M$ software servers (representing the application components)
- Delay Node (representing user think time)
- K classes of requests’
- Set of constraints on the response time that was defined as Service Level Objective

---

(a) Application parameters.

- $P_{m1,m2,k}$: Probability for a request of class $k$ to visit node $m2$ after completing service at node $m1$.
- $\mu_{m,k}$: Class service rate. Number of class-$k$ requests completed at software server $m$ in a time unit.
- $\sigma_k$: Delay node service rate. Number of class-$k$ requests completed at the delay node in a time unit.
- $N_k$: Total number of users of class $k$ in the system. Each user represents a request. This parameter specifies the system workload.
- $maxMRT_k$: Maximum mean response time for class-$k$ requests.
- $maxRTP_{k,u}$: Maximum response time for the class-$k$ requests in the $u$-th percentile.
System Model - Resource

- Environment has $R + 1$ available resource types
- Type 0 – unallocated resource with zero price and rate
- Each resource is characterized by a certain rate (processing speed) and number of processors

(b) Resource parameters.

- $Y \in \mathbb{N}$: Total number of resources that can be rented.
- $T$: Rental time period.
- $A$: Minimum percentage of time in which resources are expected to be available.
- $\lambda(r)$: Nominal service rate of resources of type $r$. The value of $\lambda(r)$ is calculated as the sum of the nominal service rate of each processor of the resource, and is a measure of the total computational capacity (e.g., it may be proportional to Amazon’s ECU [1]).
- $q(r)$: Number of processors (CPUs) of resources of type $r$.
- $o(r)$: Minimum bid price for renting a single resource of type $r$ for a fixed amount of time $T$, such that the availability of the resource is at least $A$. This value can be obtained from historical traces of resource type $r$ and calculated as the minimum bid price that results in an average outbid percentage of at least $A$ for such price.
- $c(r)$: Expected price for renting a single resource of type $r$ for a fixed amount of time $T$ when bidding $o(r)$. 
System Model – Optimization Problem

\[ \begin{align*}
\min & \quad \sum_{y=1,\ldots,Y} \hat{c}_y \\
\text{s.t.} & \quad \sum_{m=1,\ldots,M} d_{m,y} \leq \lambda_y, \forall y \\
               & \quad \text{MRT}_k(D) \leq \text{maxMRT}_k, \forall k \\
               & \quad \text{RTP}_{u,k}(D) \leq \text{maxRTP}_{u,k}, \forall u, \forall k
\end{align*} \]
Approach

- Choosing the minimum computational requirements for each application component
  - Minimum computational requirements in terms of resource rates that are needed by each application component to satisfy the quality requirement
  - Amazon ECUs
Approach

- Choosing the resources to rent
  - Calculate the bidding price that guarantees a minimum availability level for the resources and, based on it, decide which resources to rent.
  - Sum of the ECUs of the rented resources should be large enough to fulfill the ECU requirements of the application in step 1.
Approach

- Choosing the allocation of the application components to the resources
  - Allocate the different components into the rented resources to minimize the negative effects of allocation
  - Example of negative effect: reduction in performance due to load balancing
Approach

- Analyzing the overall system and possible scaling-up of bottlenecks
- Review performance of the system
  - Presence of multiple CPUs and load-balancing
  - Possibility of losing spot instances in case of overbid of the chosen bid price
Experiment

- Overview of the behavior of the approach when applied to queueing network models based on real data
- The paper used the public application data measurements from a real SAP ERP study
- Used historical traces of spot prices of Amazon EC2, covering a 14-month period
- Instantiated the problem using the generic non-linear solver provided by MATLAB to compare it with the proposed approach
Experiment

<table>
<thead>
<tr>
<th>Server/class</th>
<th>Service demand [ms]</th>
<th>Service rate $\mu_{m,k}$ [req/ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS dialog step</td>
<td>119.82</td>
<td>0.008346</td>
</tr>
<tr>
<td>AS update1</td>
<td>47.92</td>
<td>0.02087</td>
</tr>
<tr>
<td>AS update2</td>
<td>32.98</td>
<td>0.03032</td>
</tr>
<tr>
<td>DB dialog step</td>
<td>4.541</td>
<td>0.2202</td>
</tr>
<tr>
<td>DB update1</td>
<td>1.205</td>
<td>0.8299</td>
</tr>
<tr>
<td>DB update2</td>
<td>0.3043</td>
<td>3.286</td>
</tr>
</tbody>
</table>
## Experiment

Table II: Spot Prices of Amazon EC2

<table>
<thead>
<tr>
<th>Resource</th>
<th>Rate ECU</th>
<th>CPUs</th>
<th>Max bid to have A=90%</th>
<th>Actual prices when A=90%</th>
<th>Max bid to have A=95%</th>
<th>Actual prices when A=95%</th>
<th>Max bid to have A=99.9%</th>
<th>Actual prices when A=99.9%</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1.small</td>
<td>1</td>
<td>1</td>
<td>0.067</td>
<td>0.0653</td>
<td>0.068</td>
<td>0.0659</td>
<td>0.07</td>
<td>0.067</td>
</tr>
<tr>
<td>m1.large</td>
<td>4</td>
<td>2</td>
<td>0.266</td>
<td>0.260</td>
<td>0.271</td>
<td>0.2622</td>
<td>0.28</td>
<td>0.2672</td>
</tr>
<tr>
<td>m1.xlarge</td>
<td>8</td>
<td>4</td>
<td>0.534</td>
<td>0.5204</td>
<td>0.538</td>
<td>0.5222</td>
<td>0.559</td>
<td>0.5333</td>
</tr>
<tr>
<td>m2.xlarge</td>
<td>6.5</td>
<td>2</td>
<td>0.325</td>
<td>0.3139</td>
<td>0.329</td>
<td>0.3164</td>
<td>0.336</td>
<td>0.3203</td>
</tr>
<tr>
<td>m2.2xlarge</td>
<td>13</td>
<td>4</td>
<td>0.735</td>
<td>0.7148</td>
<td>0.737</td>
<td>0.7157</td>
<td>0.769</td>
<td>0.7337</td>
</tr>
<tr>
<td>m2.4xlarge</td>
<td>26</td>
<td>8</td>
<td>1.468</td>
<td>1.4321</td>
<td>1.47</td>
<td>1.4342</td>
<td>1.54</td>
<td>1.468</td>
</tr>
</tbody>
</table>
Experiment

Figure 11. Experiment results when varying the number of users.
Experiment

Figure 12. Experiment results when varying SLO. The SLO is a maximum limit on the mean value of the response time calculated for a rental time period $T$. 

(a) Hourly price.
(b) Execution time.
(c) QN evaluations.
Experiment

Figure 13. Experiment results when varying the availability.
Conclusion

- Paper presented a cost-aware approach to support run-time decisions for provisioning cloud resources and allocating application components.

- Approach is able to approximate a complex problem using simple greedy algorithms that are lightweight enough to be used at run-time to support pro-active and reactive system adaptation.

- Ability to predict and make decisions in light of a number of random environmental parameters such as the possibility of losing spot resources.