A Framework for BPMS Performance and Cost Evaluation on the Cloud

Guillaume Rosinosky, Samir Youcef, François Charoy

guillaume.rosinosky@bonitasoft.com
@grosinosky
Introduction

- Goal: find elastic methods for BPMaaS resource allocation
- CIFRE thesis (mixed Bonitasoft/INRIA Nancy)
- Dual problematic
- Bonitasoft now not very interested in allocation resource methods but the tools can be useful…
Problematic (research)

- Find elasticity methods for BPMaaS
  - Resource allocation & scheduling
  - Price minimization for defined quality
- Hypothesis
  - BPM task number throughput as a quality metric
- Need to evaluate
  - Resource size
  - BPM task throughput justification
  - Later: test load balancer coupled with algorithm
Problematic (entreprise)

- Integration tests
  - Evaluate non regression between versions
- Performances for customers
  - Sizing, cloud configuration recommendation
- Evaluate different configurations
  - On premises
  - Cloud (Amazon, but next other vendors)
- Ability to retrieve logs
Problematic

• Have an Open Source framework able to:
  • Allocate cloud resources
  • Deploy needed software on it
    • Database
    • BPMS
    • Load balancer
    • Testing tool
  • Deallocate cloud resources
• Get metrics for analysis
• Archive
Problematic
Existing work

• BPMS framework
  • Benchflow
    • Based on Faban, process centric
    • Done on premise, without orchestrator
  • Betsy
    • BPMN or BPEL compliance
• Generic frameworks
  • Commercial such as Smart CloudBench
Used tools

- Docker: containers
  - « VM without overhead » + configuration
  - Repeatable and isolated runs
  - => Database, Bonita, load balancer, injector images
- Ansible: scripts and orchestration tool
  - Automation tool for cloud provisioning, configuration management and application deployment
  - Used for cloud provisioning, resources initialization, test launches
- Jenkins: continuous integration automation server
  - Web UI used for software project builds and deployment,
  - UI, test archive and scripting
Tests

• Software
  • Bonita Benchmark tool
  • Bonita 7.2.3 community edition official Docker
  • Postgresql official Docker
• Hardware: Amazon Web Services
  • Test c4 family and m3.medium (BPMS), r3 family and m3.medium (database)
• Used process: « standard process » (20 consecutive automated tasks), launched 3000 times
• Tests: 6 consecutives tests on each configuration
Results

Cloud configuration process mean time

- 0.177 $/h$ db.m3.medium/m3.medium
- 0.223 $/h$ db.m3.medium/c4.large
- 0.399 $/h$ db.r3.large/c4.large
- 0.518 $/h$ db.r3.large/c4.xlarge
- 0.674 $/h$ db.r3.xlarge/c4.large
- 0.757 $/h$ db.r3.xlarge/c4.2xlarge
- 0.793 $/h$ db.r3.xlarge/c4.xlarge
- 1.032 $/h$ db.r3.xlarge/c4.2xlarge
- 1.587 $/h$ db.r3.2xlarge/c4.2xlarge
- 2.063 $/h$ db.r3.2xlarge/c4.4xlarge
- 3.173 $/h$ db.r3.4xlarge/c4.4xlarge
- 4.126 $/h$ db.r3.4xlarge/c4.8xlarge

Number of parallel processes vs. Process mean time
Results

<table>
<thead>
<tr>
<th>DB inst. type</th>
<th>AS inst. type</th>
<th>price</th>
<th>task TP</th>
<th>task TP per $</th>
</tr>
</thead>
<tbody>
<tr>
<td>db.m3.medium</td>
<td>m3.medium</td>
<td>0.177</td>
<td>16.400</td>
<td>92.656</td>
</tr>
<tr>
<td>db.m3.medium</td>
<td>c4.large</td>
<td>0.223</td>
<td>23.157</td>
<td>103.845</td>
</tr>
<tr>
<td><strong>db.r3.large</strong></td>
<td><strong>c4.large</strong></td>
<td><strong>0.399</strong></td>
<td><strong>55.164</strong></td>
<td><strong>138.255</strong></td>
</tr>
<tr>
<td>db.r3.large</td>
<td>c4.xlarge</td>
<td>0.518</td>
<td>58.067</td>
<td>112.100</td>
</tr>
<tr>
<td>db.r3.xlarge</td>
<td>c4.large</td>
<td>0.674</td>
<td>65.113</td>
<td>96.607</td>
</tr>
<tr>
<td>db.r3.xlarge</td>
<td>c4.xlarge</td>
<td>0.757</td>
<td>61.474</td>
<td>81.208</td>
</tr>
<tr>
<td>db.r3.xlarge</td>
<td>c4.2xlarge</td>
<td>0.839</td>
<td>83.236</td>
<td>104.963</td>
</tr>
<tr>
<td>db.r3.xlarge</td>
<td>c4.2xlarge</td>
<td>1.032</td>
<td>89.149</td>
<td>86.384</td>
</tr>
<tr>
<td>db.r3.2xlarge</td>
<td>c4.2xlarge</td>
<td>1.587</td>
<td>105.794</td>
<td>66.663</td>
</tr>
<tr>
<td>db.r3.2xlarge</td>
<td>c4.4xlarge</td>
<td>2.063</td>
<td>107.585</td>
<td>52.150</td>
</tr>
<tr>
<td>db.r3.4xlarge</td>
<td>c4.4xlarge</td>
<td>3.173</td>
<td>115.283</td>
<td>36.332</td>
</tr>
<tr>
<td>db.r3.4xlarge</td>
<td>c4.8xlarge</td>
<td>4.126</td>
<td>129.279</td>
<td>31.332</td>
</tr>
</tbody>
</table>
Conclusion

• Effective tool for performance tests
• Research part
  • used for papers, works well
  • more generalization needed
• Entreprise part:
  • used internally for performance tests,
  • used for clustered version tests
Perspectives

- Add hardware metrics
- Add other cloud providers
- Add other BPMS
- Combine with REST client for BPMS tests
- «Dockerize» orchestrator
References and links

- Tools
  - Docker : https://www.docker.com/
  - Ansible : https://www.ansible.com/
  - Jenkins : https://jenkins.io/
  - BonitaBPM : http://www.bonitasoft.com/
  - Postgresql : https://www.postgresql.org/
Thank you :) 

• Thanks to Bonitasoft, and AWS 
• Mail: guillaume.rosinosky@bonitasoft.com 
• ... questions ?
Demo

Let’s take a look here