ArA: Adaptive Resource Allocation for Clouds under Burst Workloads

Jianzhe Tai, Juemin Zhang, Waleed Meleis and Ningfang Mi

Electrical and Computer Engineering Dept.
Northeastern University
{jtai, jzhang, meleis, ningfang}@ece.neu.edu
Cloud Computing

Two-level load balancing

- **Level 1**: balance the load across a set of instances of the same application
- **Level 2**: Balance the load of multiple applications among physical computers
Existing Problems (1)

- **Burstiness in computer systems**
  - Dramatically degrade the performance

- **Burstiness in Clouds**
  - Multi-remote-users
  - Not single-program-single-execution
  - application variety increases

Needs a simple yet powerful load balancer to counteract the impact of burstiness on system performance.
Existing Problems (2)

- Information query delay
  - Network delay
  - Inter-server communication delay
  - System load self-update delay

The information query delay further degrades the performance in load balancing process.
Simulation Environment

- Burst/Non-burst
- Round Robin
- Random
- Qlen, Est. QT, Act. QT ...
- Submission Delay
- Balance/Imbalance
Arrival Traces

- Non-Bursty
  - Time (1000s)
  - Num of arrivals

- Weak Bursty
  - Time (1000s)
  - Num of arrivals

- Strong Bursty
  - Time (1000s)
  - Num of arrivals

same mean
We need a new load balancer to counteract the impact of burstiness and submission delay.
Project Goal

- Adaptive resource allocation for clouds
  - Forecasts changes in user demands and system loads
  - Develop effective two-level load balancers

Our expectations

- Allow **cloud users** to experience higher quality of service
- Allow **cloud systems** to make better use of their infrastructure
ArA: A new load balancer

- Balance the load within an application
- Fixed $k = 3$ candidates

![Diagram showing load balancing with front-end dispatcher and back-end nodes with load distribution indicated by numbers 2, 5, 7, 10, 13. The load distribution is represented as a uniform distribution $\text{uniform}(1, k)$ where $k = 3$.](image-url)
Some Preliminary Results

- With fixed $k$ candidates

Tuning static $K$-value according to the dynamic user demands is difficult.
Online ArA

- Online adjust $k$ candidates

- Index of dispersion based prediction

$$I = SCV \left( 1 + 2 \sum_{k=1}^{\infty} \rho_k \right)$$

![Graph showing arrival rate over time with data points labeled.]
Index of Dispersion Based Prediction
Online ArA

- Online adjust $k$ candidates
  - Busy phase: large $k$
  - Idle phase: small $k$

$k=N$ random

$k=1$ greedy

![Graph showing arrival rate over time with different $k$ values](image-url)
Evaluation Results

- online adjust $k$ candidates
## Sensitivity Analysis

### (a) Load Balancer

<table>
<thead>
<tr>
<th>network size</th>
<th>Rand</th>
<th>ARA_OPT</th>
<th>ARA_PRED</th>
<th>Qlen</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1089.25</td>
<td>1063.39</td>
<td>1064.66</td>
<td>1101.02</td>
</tr>
<tr>
<td>16</td>
<td>1109.33</td>
<td>1056.07</td>
<td>1059.00</td>
<td>1244.32</td>
</tr>
<tr>
<td>32</td>
<td>1148.38</td>
<td>1042.79</td>
<td>1051.21</td>
<td>1751.43</td>
</tr>
</tbody>
</table>

### (b) Load Balancer

<table>
<thead>
<tr>
<th>delay time</th>
<th>Rand</th>
<th>ARA_OPT</th>
<th>ARA_PRED</th>
<th>Qlen</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>1109.33</td>
<td>1056.07</td>
<td>1059.00</td>
<td>1244.32</td>
</tr>
<tr>
<td>2s</td>
<td>1111.07</td>
<td>1057.76</td>
<td>1062.97</td>
<td>1692.26</td>
</tr>
<tr>
<td>6s</td>
<td>1110.77</td>
<td>1063.23</td>
<td>1070.57</td>
<td>3653.21</td>
</tr>
</tbody>
</table>

### (c) Load Balancer

<table>
<thead>
<tr>
<th>site load</th>
<th>Rand</th>
<th>ARA_OPT</th>
<th>ARA_PRED</th>
<th>Qlen</th>
</tr>
</thead>
<tbody>
<tr>
<td>30%</td>
<td>487.83</td>
<td>471.05</td>
<td>473.04</td>
<td>606.62</td>
</tr>
<tr>
<td>50%</td>
<td>1109.33</td>
<td>1056.07</td>
<td>1059.00</td>
<td>1244.32</td>
</tr>
<tr>
<td>80%</td>
<td>4220.09</td>
<td>3964.39</td>
<td>3968.77</td>
<td>4138.34</td>
</tr>
</tbody>
</table>
Case Study: Amazon EC2

- One High-CPU Medium Instance as load balancer
  - five EC2 compute units
- Eight Small Standard Instances as servers
  - one EC2 compute unit and 1.7GB memory
Case Study: Amazon EC2

End-to-End Response Time (second)

- Greedy
- ArA
- Random
Conclusion

• It is critical to consider the effects of burstiness and information query delay in load balancing.

• ArA shifts between randomness and greediness in the site selection based on bursty conditions.
  – Predict workload changes
  – Parameter free
  – Robust
Next Steps …

- Extended distributed load balancers

**Challenge!**

Need a global view of user demands and system loads! Consider coherency and dependency of arrival request!
Thanks

Q & A