ELASTICITY CONTROL FOR LATENCY-INTOLERANT MOBILE EDGE APPLICATIONS

Chanh Nguyen, Cristian Klein, Erik Elmroth
Dept. Computing Science
Umeå University, Sweden
ELASTICITY IN CLOUD

• What is Elasticity?
• How does Cloud Computing Control Elasticity?
  o Re-active.
  o Pro-active.
  o Hybrid.
Most MECs applications are latency-sensitive applications.
- Limited resources with higher resource costs at the edge data centers (EDCs).
- The stochastic nature of user mobility causes resource demand fluctuated.
- Auctuation delays – allocated resources is not ready to use immediately.
ELASTICITY CONTROL IN MOBILE EDGE CLOUD
GOAL

• MECs operator’s perspective:
  o Average resource utilization at EDCs.
  o System stability.

• End-user’s perspective:
  o Average rejected rate.
Figure 1: Components of the proposed controller.
PRO-ACTIVE ELASTIC CONTROL FRAMEWORK

• **Location-aware Workload Predictor**
  o Multi-variate LSTM networks.

• **Performance Modeler**
  o Resources are abstracted at Pod modelled as a M/M/1/k FIFO queue.

• **Resource Provisioner**
  o cross-evaluating the resource requirements of EDCs in a group and determine a final number of desired resources for each EDC.

• **Group Load-balancer**
  o Weight round-robin load balancing approach.
EXPERIMENT SETTING

• Emulated MEC:
  o MEC with EDCs distributed over a metropolitan area.

• Application:
  o Extremely latency-intolerant AR application.

• Workload:
  o Real taxi mobility traces.

Figure 2: Distribution of EDCs in San Francisco.
EXPERIMENT SETTING

• Predefined Service Level Objectives:
  o Average Utilization = 80%.
  o Rejection rate = 1%.

• Controller settings:
  o Pro-active Auto Scaler.
  o Pro-active Auto Scaler + Group Load Balancer.
  o Re-active Auto Scaler: Kubernetes HPA*.

*https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/
EXPERIMENT SETTING

Figure 4: Experimental simulation.
• System and user-oriented metrics: recommend by SPEC*
  o Under-provisioning accuracy,
  o Over-provisioning accuracy,
  o Under-provisioning timeshare,
  o Over-provisioning timeshare,
  o Instability.

*Nikolas Herbst et al., Ready for rain? A view from SPEC research on the future of cloud metrics
How does the proposed pro-active controller perform when compared to the re-active controller?

<table>
<thead>
<tr>
<th>Metric</th>
<th>Pro-active AS + LB</th>
<th>Pro-active AS</th>
<th>Re-active AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_U$</td>
<td>13.6</td>
<td>41.2</td>
<td>5.4</td>
</tr>
<tr>
<td>$\theta_O$</td>
<td>14.2</td>
<td>39.5</td>
<td>305.6</td>
</tr>
<tr>
<td>$\tau_U$</td>
<td>4%</td>
<td>43%</td>
<td>5.3%</td>
</tr>
<tr>
<td>$\tau_O$</td>
<td>2.5%</td>
<td>46.7%</td>
<td>94.1%</td>
</tr>
<tr>
<td>$\nu$</td>
<td>2.44%</td>
<td>2.8%</td>
<td>3.9%</td>
</tr>
<tr>
<td>Avg. resource utilization</td>
<td>85.9%</td>
<td>80.5%</td>
<td>68.4%</td>
</tr>
<tr>
<td>Rejection rate</td>
<td>0.02%</td>
<td>0.26%</td>
<td>0.04%</td>
</tr>
<tr>
<td>total Pods</td>
<td>3154</td>
<td>4405</td>
<td>5337</td>
</tr>
<tr>
<td>Avg. Pod lifetime (minute)</td>
<td>73.3</td>
<td>35.2</td>
<td>29.6</td>
</tr>
</tbody>
</table>

Table II: The performance of the three controllers based on the elasticity metrics.
How does the proposed pro-active controller perform when compared to the re-active controller?

- a) Re-active
- b) Pro-active AS
- c) Pro-active AS + LB

Figure 5: The scaling behavior of three controllers on EDC#1.
How does the proposed pro-active controller perform when compared to the re-active controller?

Figure 6: Cumulative density of response times of the application in three elastic controller settings.
To what degree does location-awareness improve scaling behavior?

- Conduct another experiment which a group is set with different size $k$
  - $k = 1$
  - $k = 15$

Figure 7: Performance of the three studied controller configurations based on the three major elasticity metrics when the number of neighboring EDCs is varied.
What is the decision time of the elastic controller?

Figure 8: Average Decision Time of the three controllers.
What is the impact of the two predefined threshold on the controller’s scaling behavior?

Table III: The scaling behavior of the proposed controller with different predefined threshold settings.

<table>
<thead>
<tr>
<th>Targeted rejection rate[%]</th>
<th>Targeted resource utilization[%]</th>
<th>Measured resource utilization[%]</th>
<th>Measured rejection rate[%]</th>
<th>Total Pods</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>74.8</td>
<td>0</td>
<td>3812</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>80.2</td>
<td>7e-4</td>
<td>3484</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>85.9</td>
<td>0.02</td>
<td>3154</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>90.6</td>
<td>0.16</td>
<td>2890</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>95.2</td>
<td>0.8</td>
<td>2653</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>98.2</td>
<td>2.7</td>
<td>1995</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>93.5</td>
<td>0.44</td>
<td>2753</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>80</td>
<td>87.2</td>
<td>3065</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>86.3</td>
<td>0.05</td>
<td>3113</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>85.9</td>
<td>0.02</td>
<td>3154</td>
<td></td>
</tr>
</tbody>
</table>
What is the impact of the two predefined threshold on the controller’s scaling behavior?

(a) The targeted resource utilization is changed, while the targeted rejection rate is held constant at 1%.

(b) The targeted rejection rate is changed, while the targeted resource utilization is held constant at 80%.

Figure 9: The controller’s scaling behavior when varying the threshold settings.
• The correlation of workload variation in physically neighboring EDCs help improve the resource estimation.
• The Group Load-balancer further helps minimize the request rejection rate.
• The proposed controller achieves a significant better scaling behavior as compared against the state-of-the-art re-active controller.
THANK YOU

Contact for further discussion:

**Name:** Chanh Nguyen

**Email:** chanh@cs.umu.se