Simulation of SLA-Based Horizontal Scaling of Distributed Systems

Alexandru-Florian Antonescu
Universität Bern
SAP

antonescu@iam.unibe.ch
alexandru-florian.antonescu@sap.com
Simulation of SLA-Based Horizontal Scaling of Distributed Systems

Agenda

- Research Question
- SLA-Based Capacity Planning
- SLA-Driven Horizontal VM-Scaling
- Large-Scale Simulation
- Results
- Questions
Theoretical Foundation

**Research Question:** How can we efficiently horizontally-scale a cloud-distributed system given SLA constraints?

**Research Results**
- Modeling and Simulation of Concurrent Workload Processing in Cloud-Distributed Enterprise Information Systems (SIGCOMM 2014)
- SLA-Driven Simulation of Multi-Tenant Scalable Cloud-Distributed Enterprise Information Systems (ARMS-CC 2014)
- Simulation of SLA-Based VM-Scaling Algorithms for Cloud-Distributed Applications (FGCS journal)

**Evaluation**
- SLA-based large-scale simulation of a distributed application using CloudSim simulator
Distributed Enterprise Information System Architecture (recap)

- **Application**: distributed Enterprise Information System (dEIS)
- **Task**: data-intensive report generation
- **Communication** protocol: dOSGi over SOAP/CXF
SLA-Based Capacity Planning

Little’s Law: \( L = \lambda W \)

- \( L \) \text{ average system’s occupancy}
- \( \lambda \) \text{ average arrival rate}
- \( W \) \text{ average execution time}

Perform \( N \) constant-load \((L)\) experiments (with the number concurrently-processed requests constant)
### SLA-Based Capacity Planning

1. Given a SLA constraint \( \bar{W}_{svc} < W_{svc}^{\text{max}} \)
2. Perform constant-load (\( L \)) experiments (number concurrently-processed requests constant)
3. Determine optimal arrival rate (\( \lambda^{opt} \)) so that SLA constraint holds
4. Use \( \lambda^{opt} \) in a horizontal-scale VM manager

---

Simulation of SLA-Based Horizontal Scaling of Distributed Systems

---

Given: SLA constraint $W^{\text{max}}$

Determine $\lambda^{\text{opt}} (Th^{\text{max}})$

```
11 repeat every N seconds
12   if $\lambda > 0.8 \text{vm} Th^{\text{max}}$ then
13     $vm^* \leftarrow \left\lfloor \frac{\lambda}{0.8Th^{\text{max}}} \right\rfloor$;
14     if $vm^* > vm + vm^+$ then
15       out $\leftarrow vm^* - vm - vm^+$;
16   end
17 else if $\lambda < 0.3 \text{vm} Th^{\text{max}} \text{ AND time(last scaling)} < \text{cool-down}$ then
18     $vm^* \leftarrow \left\lfloor \frac{\lambda}{0.8Th^{\text{max}}} \right\rfloor$;
19     if $vm^* < vm - vm^-$ then
20       in $\leftarrow vm^* - vm - vm^-$;
21   end
22 if $Th > 0.8 \text{vm} Th^{\text{max}}$ then
23     $vm^* \leftarrow \left\lfloor \frac{Th}{0.8Th^{\text{max}}} \right\rfloor$;
24     if $vm^* > vm + vm^+ + out$ then
25       out $\leftarrow vm^* - vm - vm^+$ AND in $\leftarrow 0$;
26   end
27 end
28 perform scaling
```
SLA-Driven Horizontal VM Scaling

- Use linear regression to predict arrival rate ($\lambda$)
- Prediction horizon: N+D sec.
- Drop prediction if MAPE > $\varepsilon_{\text{max}}$
- Use prediction at $t+D$
- Use previous alg. for scale-in

### ALGORITHM 3: Predictive-Based VM-Scaling Algorithm

**Data:**
1. SLA contract with the maximum execution time $T_{\text{max}}$
2. $\mathbf{P_{\text{pred}}}$ empty
3. $d_{\text{pred}}$
4. $\theta_{\text{max}}$

**repeat** every $N$ seconds

4. $\Lambda \leftarrow \{\lambda(i) | i < t - M \}$
5. $T \leftarrow \text{sampling}(\Lambda)$
6. **if** $\mathbf{RM}$ is not NULL **then**
7. $\text{MAPE} = \text{accuracy}(\mathbf{RM}, T, \Lambda)$
8. **if** $\text{MAPE} < \varepsilon_{\text{max}}$ **then**
9. $T^* \leftarrow (t + 1, t + 2, \ldots, t + N + D)$
10. $\Lambda^* \leftarrow \text{predict}(\mathbf{RM}, T^*)$
11. $\text{Pred}[T^*] \leftarrow \left\lceil \frac{\Lambda}{0.8\theta_{\text{max}}} \right\rceil$
12. **else**
13. $\text{Pred} \leftarrow \text{empty}$
14. drop $\mathbf{RM}$
15. **end**
16. **end**
17. $\mathbf{RM} \leftarrow \text{regression}(\Lambda, T)$
18. **if** $\text{Pred}[t+D]$ exists **then**
19. $\text{out} \leftarrow \max(0, \text{Pred}[t + D] - \text{vm} - \text{vm}^+)$
20. **end**
21. **if** out = 0 AND $\text{vm}^+ = 0$ **then**
22. calculate scale-in step using another algorithm
23. **end**
Large-Scale Simulation in CloudSim

> Workload distribution over 8400 sec. (max. 28’000 rpm)

> SLAs
  - $W_{WK} < 700$ms
  - $W_{ST} < 300$ms
Results Algorithm 1

Using optimal arrival rate ($\lambda$)
Average Execution Time ($W$) was lower than the SLA threshold
Total VMs: 72 WK + 68 ST
Results Algorithm 2 (with prediction)

Max prediction error: MAPE < 2.5%
Scaling is performed with D sec. before $\lambda$ reaches the scaling threshold
Conclusions

> SLA constraints can be combined with benchmarking based on Little’s Law for optimally horizontal VM-scaling
> Prediction improves the response of the VM-scaling algorithm
> Reactive scaling can quickly become ineffective when workload is increasing fast (FGCS article)
> CloudSim can be effectively used for simulating cloud applications

> Next steps: put everything together
Questions & Answers

Thank you
### Simulation of SLA-Based Horizontal Scaling of Distributed Systems

#### Little’s Law Benchmark Results

<table>
<thead>
<tr>
<th>Concurrency</th>
<th>Execution Time (ms)</th>
<th>Arrival Rate Throughout</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WK</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>106.25317582473</td>
<td>118</td>
</tr>
<tr>
<td>2</td>
<td>136.65920341163</td>
<td>133</td>
</tr>
<tr>
<td>3</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td>4</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td>5</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td>6</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td>7</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td>8</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td>9</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td>10</td>
<td>115.38010902880</td>
<td>133</td>
</tr>
<tr>
<td><strong>ST</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.4784086832778</td>
<td>118</td>
</tr>
<tr>
<td>2</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>3</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>4</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>5</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>6</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>7</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>8</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>9</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
<tr>
<td>10</td>
<td>2.3608406820913</td>
<td>133</td>
</tr>
</tbody>
</table>
Reactive Scaling

Simulation of SLA-Based Horizontal Scaling of Distributed Systems
Simulation of SLA-Based Horizontal Scaling of Distributed Systems

Reactive Scaling

- **variable**
  - incoming
  - throughput
  - dropped

- **Time (seconds)**

- **Requests Per Minute**

- **SLA Ratio**
  - service
    - WK
    - ST