Improving Cloud Application Resilience using Brownout-Aware Load-Balancing

Cristian Klein\textsuperscript{1}, Alessandro V. Papadopoulos\textsuperscript{2}, Manfred Dellkrantz\textsuperscript{2}, Jonas Dürango\textsuperscript{2}, Martina Maggio\textsuperscript{2}, Karl-Erik Årzén\textsuperscript{2}, Francisco Hernández-Rodriguez\textsuperscript{1}, Erik Elmroth\textsuperscript{1}

\textsuperscript{1} Umeå University, Sweden \hspace{1cm} \textsuperscript{2} Lund University, Sweden

2014-10-07, SRDS – Nara, Japan
Problem: Unexpected Events
Application in Private Cloud

- 1% reduced sale per 100ms load time*
- 20% reduced income if 0.5s longer load time**
- 25% of end-users leave if load time > 4s*

* Amazon  **Google
State-of-Practice

- **Large spare capacity**
  - May be economically impractical

- **Cloud bursting**
  - Lease capacity from a public cloud
  - Does not really solve the problem
Brownout: Idea

• Disable **optional** content
  – Minimally intrusive
• E.g. recommendations
  – 50% increase in sales *
• Challenge
  – **Maximize** optional content
  – **Avoid** high response times

Brownout: Inside a Replica

$t$ = response times

$\tau$ = target response time

$\theta$ = probability of serving optional content (dimmer)

C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén,
Brownout: building more robust cloud applications, ICSE, 2014
Brownout: Stress Test

✔ Resilient to flash crowds
✔ Resilient to capacity shortages
✖ Not resilient to hardware failures
Replication and Brownout

- **Goal:** Maximize optional content served
- **Problem:** existing load-balancers **brownout-unaware**
  - Mostly based on **response-time** or **queue-length**
Maximizing Optional Content

<table>
<thead>
<tr>
<th></th>
<th>Weight-based (periodic)</th>
<th>Queue-based (event)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brownout-unaware</td>
<td>WRR</td>
<td>SQF</td>
</tr>
<tr>
<td>Variation-based</td>
<td>PIBH</td>
<td>PIBH+</td>
</tr>
<tr>
<td>Equality-based</td>
<td>EPBH</td>
<td>EPBH+</td>
</tr>
<tr>
<td>Optimization-based</td>
<td>COBLB</td>
<td></td>
</tr>
</tbody>
</table>

Tested using **simulations**
- SQF best brownout-unaware method
- Brownout-aware methods better
- Somewhat slow to react

Tested using **experiments** (lighttpd)

J. Dürango et al. “Control-theoretical load-balancing for cloud applications with brownout”, CDC, 2014
Queue-Based Methods

- Track queue-lengths $q_i$ and dimmers $\Theta_i$
- Compute a queue offsets $u_i$
- Pick replica with lowest $q_i - u_i$

1. Variation-based (PIBH+):
   $$u_i(k+1) = (1 - \gamma) [u_i(k) + \gamma I \Delta \Theta_i(k) + \gamma I \Theta_i(k)] + \gamma q_i(k)$$

2. Equality-based (EPBH+):
   $$u_i(k+1) = u_i(k) + \gamma_e \left( \Theta_i(k) - \frac{1}{n} \sum_{j=1}^{n} \Theta_j(k) \right)$$
Evaluation

**RQ1**: How does replication benefit by adding brownout?

**RQ2**: Gains that can be made using brownout-aware load-balancing?
RQ1: Replication and Brownout
Experimental Setup

• Workload
  – Requests follow a Poisson process
  – Constant arrival rate
  – 4 seconds timeout

• Application
  – 5 replicas each in its VM
  – 4 CPU cores each (static)
  – All data in-memory
  – SQF load-balancer

• Scenario
  – Replicas subject to cascading failures
RQ1: Replication and Brownout
Results

Only replica 0 during this interval

Timeouts [req/s]

optional content ratio [%]

non-brownout

Timeouts [req/s]

Optional Content Ratio [%]

Time [s]
### RQ1: Replication and Brownout

More Results

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Metric</th>
<th>Non-brownout</th>
<th>Brownout</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 cores</td>
<td>Requests served</td>
<td>31.2%</td>
<td>99.3%</td>
</tr>
<tr>
<td>200 requests/s</td>
<td>With optional content</td>
<td>31.2%</td>
<td>81.0%</td>
</tr>
<tr>
<td>2 cores</td>
<td>Requests served</td>
<td>31.6%</td>
<td>99.3%</td>
</tr>
<tr>
<td>100 requests/s</td>
<td>With optional content</td>
<td>31.6%</td>
<td>82.0%</td>
</tr>
<tr>
<td>heterogeneous</td>
<td>Requests served</td>
<td>68.8%</td>
<td>99.5%</td>
</tr>
<tr>
<td>166 requests/s</td>
<td>With optional content</td>
<td>68.8%</td>
<td>90.2%</td>
</tr>
</tbody>
</table>
RQ2: Brownout-Aware Load-Balancing
Experimental Setup

• Same workload
• Application
  – 5 replicas each in its VM
  – Heterogeneous number of cores
• Scenario
  – Steady capacity shortage
• Statistically significant results
  – 30 experiments for each algorithm
  – T-test
RQ2: Brownout-Aware Load-Balancing
Results: 2 x 1 core, 3 x 8 cores

PIBH: 5.34% more
EPBH: 4.52% more
optional content than SQF

95th percentile response time well below target
RQ2: Brownout-Aware Load-Balancing

Results: 3 x 1 core, 2 x 8 cores

- PIBH: 5.17% more optional content than SQF
- EPBH: 3.13% more optional content than SQF

No significant increase in 95th percentile response time.
Conclusions and Perspectives

• Brownout
  – Minimally intrusive method
  – Applications resilient to capacity shortage

• Replication and brownout
  – More resilient to cascading failures

• PIBH: 5% more optional content than SQF
  
  https://github.com/cloud-control

• Perspectives
  – Combine brownout with elasticity
Thank you for your attention!
ご清聴ありがとうございました！

Cristian Klein

Improving Cloud Service Resilience using Brownout-Aware Load-Balancing

resources [% CPU]

Latency [s]

Optional Content Ratio [%]

https://github.com/cloud-control
References


Appendix
Replica Controller (1)

- Need to adapt to changes
  - Number of users
  - Available capacity
- Not all requests take the same time
  - E.g., cached in memory, disk
- Need to reject disturbances
  - E.g., NTP daemon firing up, cron jobs
Replica Controller (2)

- Start from a simple model
  \[ t^{k+1} = \alpha^k \cdot \Theta^k + \delta t^k \]

- Adaptive PI controller
  \[ \Theta^{k+1} = \Theta^k + \frac{1 - p_1}{\alpha} \cdot e^{k+1} \]

- \( \alpha \) estimated using RLS
Robustness to Model Uncertainties

\[ \alpha = \tilde{\alpha} \cdot \Delta \alpha \]
Cloud Architecture

End-user

Client

Application

SP

Load balancer

Replica

New app

IP

Admission Controller

Resource Manager
Solution: Replication

- **Hide** infrastructure failures
- Scale beyond a physical machine
Brownout: Applicability

- **RUBiS**: eBay-like prototype
  - Added a recommender
- **RUBBoS**: Slashdot-like prototype
  - Added a recommender
  - Marked comments as optional
- **Effort in lines of code:**

<table>
<thead>
<tr>
<th>Modification</th>
<th>RUBiS</th>
<th>RUBBoS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommender</td>
<td>37</td>
<td>22</td>
</tr>
<tr>
<td>Dimmer</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>Reporting response time to controller</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Controller</td>
<td>120</td>
<td>120</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>165</td>
<td>153</td>
</tr>
</tbody>
</table>