Brownout: Building More Robust Cloud Applications

Cristian Klein¹, Martina Maggio², Karl-Erik Årzén², Francisco Hernández-Rodriguez¹

¹Umeå University, Sweden, ²Lund University, Sweden

International Conference on Software Engineering (ICSE), 2014-06-05, Hyderabad, India
Deadline Day: Obamacare Website Down

Wal-Mart Website Crashes on Black Friday

John Oliver Crashes FCC Comments System After Net Neutrality Segment

Glasgow 2014 shut down "crashing website" after thousands of fans left frustrated in ticket rush
Cloud Computing Definition

Rapid provisioning (and release) from a shared pool of resources

• 3 stakeholders
  – Infrastructure Provider (IP): public or private
  – Service/Application Provider (SP)
  – End-user

• Infrastructure-as-a-Service (IaaS)
  – Lease CPU, memory, storage
  – Packed as a Virtual Machine (VM)
  – Allocate capacity
Problem: Unexpected Events

- 82% of end-users give up on a lost payment transaction*
- 25% of end-users leave if load time > 4s**
- 1% reduced sale per 100ms load time**
- 20% reduced income if 0.5s longer load time***

* JupiterResearch  ** 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 **non-essential** content
  - Minimally intrusive
- E.g. recommendations
  - 50% increase in sales *
- Challenge
  - **Maximize non-essential content**
  - Avoid high response times

Brownout: Design

\[ \tau = \text{target response times (e.g., 1 second)} \]
\[ t = \text{response times} \]
\[ \theta = \text{probability of serving non-essential content (dimmer)} \]
Controller Challenges

- 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
Controller Design

• Start from a simple model (k is time index)

\[ t^{k+1} = \alpha^k \cdot \Theta^k + \delta t^k \]

↑ factor to estimate  ↓ disturbance

• Adaptive PI controller

\[ \Theta^{k+1} = \Theta^k + \frac{1 - p_1}{\alpha^k} \cdot (t^k - \tau) \]

• \( p_1 \) pole (0 = reactive, 1 = stable)

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

Estimation may be 2 to 20 times wrong and response time still reaches target
Evaluation

**RQ1**: How intrusive is brownout?

**RQ2**: Does the self-adaptive application behave as predicted in theory?

**RQ3**: What pole value to choose?
RQ1: Minimally Intrusive

- **RUBiS**: eBay-like prototype
  - Added a recommender
- **RUBBoS**: Slashdot-like prototype
  - Added a recommender
  - Marked comments as non-essential
- **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><strong>165</strong></td>
<td><strong>153</strong></td>
</tr>
</tbody>
</table>
RQ2: Application Behavior

• Time-series analysis

<table>
<thead>
<tr>
<th>Number of Users</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable ~</td>
<td>constant =</td>
</tr>
<tr>
<td>constant =</td>
<td>variable ~</td>
</tr>
<tr>
<td>variable ~</td>
<td>variable ~</td>
</tr>
</tbody>
</table>
RQ2: RUBiS, flash-crowd
Non-adaptive vs. self-adaptive

non-adaptive

self-adaptive pole 0.9
RQ2: RUBiS, flash-crowd

Self-adaptive: different pole values

Controller behaves as theoretically designed
RQ3: Choosing a Pole

Revenue = $1 \times \#\text{essential} + 0.5 \times \#\text{non-essential}$
Results: Stress Test

(a) with all recommendations

(b) with no recommendations

(c) self-adaptive
Brownout Ecosystem

End-user

Client

Application

Load balancer

Replica

SP

New app

IP

Admission Controller

Resource Manager
Conclusions

• Cloud applications need to tolerate
  – **Flash-crowds** (sudden increase in users)
  – Hardware **failures**

• Brownout
  – **Minimally intrusive** self-adaptation paradigm
  – Cloud applications more **robust**

  [https://github.com/cloud-control](https://github.com/cloud-control)

• Perspectives
  – **Improve** controller
    • Event-driven, queue-length-based
  – Combine brownout with **elasticity**
Thank you for your attention!

Brownout: Building More Robust Cloud Applications

Cristian Klein\textsuperscript{1}, Martina Maggio\textsuperscript{2},
Karl-Erik Årzén\textsuperscript{2}, Francisco Hernández-Rodriguez\textsuperscript{1}

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

\[ \Theta^{k+1} = \Theta^k + \frac{1 - p_1}{\alpha^k} \cdot (t^k - \tau) \]

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{Comparison of resource usage and latency for different strategies.}
\end{figure}

(a) with all recommendations
(b) with no recommendations
(c) self-adaptive

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