Evaluation of a Microrebootable System

George Candea, Shinichi Kawamoto, Yuichi Fujiki, Greg Friedman, Armando Fox

-- Stanford University --
μRB and Crash-Only Design

• reboot-curable failures
• microreboot at the component level:
  • correct recovery
  • localized + contained recovery
  • fast + correct reintegration into running system

• crash-only design to enable μRB-ing:
  • state segregation (DB, SSM, ...)
  • componentization
  • fine grain resource reclamation
Prototype

- μRB-enabled JBoss (comps = EJBs) using MySQL and extended SSM
- RUBiS: online auction app (132K items, 1.5M bids, 100K users)
- Fault injection:  
  - null references
  - deadlocks, infinite loops
  - corruption of volatile metadata
  - resource leaks
  - Java exceptions and errors
- Client-side detection + automated recovery
Preview

• μRB-ing vs. FRB-ing on a single node
• μRB-ing complements cluster-based solns.
• Fast recovery tolerates sloppy fault det.
• Microrejuvenation averts failure at low cost
• Insignificant performance impact (1%)
The Gaw Metric

• Goodput x time is not good enough:
  • partial failure can improve metric
  • does not capture user actions

• Action-weighted goodput (Gaw):
  • user session = login ... logout (or abandon)
  • user action = op, op, ..., commit point
  • user actions are atomic

• Emulate 350 clients w/ eBay-like workload
FRB vs. $\mu$RB: Gaw

350 clients  FULL REBOOTS  requests: 56028 OK / 3916 failed

Time [minutes]

Gaw [responses/second]
FRB vs. $\mu$RB: Goodput

350 clients FULL REBOOTS requests: 56028 OK / 3916 failed

350 clients MICROREBOOTS requests: 61527 OK / 425 failed

89%

6 false positive $\mu$RBs
## Recovery Unit Size

<table>
<thead>
<tr>
<th>Component</th>
<th>Ave</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jboss restart (JVM)</td>
<td>51800</td>
<td>49000</td>
<td>54000</td>
</tr>
<tr>
<td>RUBiS restart (app)</td>
<td>11679</td>
<td>7890</td>
<td>19225</td>
</tr>
<tr>
<td>SB_CommitBid</td>
<td>286</td>
<td>237</td>
<td>520</td>
</tr>
<tr>
<td>SB_BrowseCategories</td>
<td>340</td>
<td>277</td>
<td>413</td>
</tr>
<tr>
<td>.................</td>
<td>...</td>
<td>...</td>
<td>.....</td>
</tr>
<tr>
<td>SB_SearchItemsByCategory</td>
<td>911</td>
<td>488</td>
<td>3019</td>
</tr>
<tr>
<td>IDManager</td>
<td>1059</td>
<td>663</td>
<td>1547</td>
</tr>
<tr>
<td>UserFeedback</td>
<td>1248</td>
<td>761</td>
<td>1591</td>
</tr>
<tr>
<td>BuyNow</td>
<td>1421</td>
<td>668</td>
<td>4453</td>
</tr>
<tr>
<td>User-Item</td>
<td>1828</td>
<td>876</td>
<td>4636</td>
</tr>
</tbody>
</table>

>>> order of magnitude

©2004 George Candea
Performance Recovery Time
Performance Recovery Time

- 10-sec reboot + 1-minute recovery
- 8 sec threshold for interactivity
Failure Containment

- solid line = activity
  gap = no activity

- Causes: no requests, site down, etc.
Failure Containment

Service Functionality Availability (FULL REBOOTS)

Bid/Buy/Sell
Search
Browse/View
Login/Logout

Time [minutes]

Service Functionality Availability (MICROREBOOTS)
Cluster Setup

700 concurrent clients

load balancer

JBoss node

MySQL

SSM
Failover (with SSM)

Aggregate Gaw
Failover (with SSM)

Failover loads good node w/ 700 clients

response time > 8 sec
Failover (with SSM)

FULL REBOOT requests: 73551 OK / 1556 failed

FRB good node

Failover loads good node w/ 700 clients

.response time > 8 sec

... oscillation possible
Failover (with SSM)

Failed reqs due to:
- timeout
- cluster down
- reqs in progress on recovering node
Avoiding Failover Altogether

- SSM not used
- Load bal. fails orphaned sess. right away
Avoiding Failover Altogether

95% (always uRB prior to failover?)

- SSM not used
- Load bal. fails orphaned sess. right away
Avoiding Failover Altogether

Failover is coarser than uRB-ing

recovery isolation enables partial avail.

MICROBE/NO FAILOVER
reqs: 76378 OK / 53 failed

MICROREBOOT
requests: 75923 OK / 57 failed
Lax Failure Detection

• Can uRB-ing be cheap enough to employ at the slightest hint of failure?

• 2 ways we can relax failure detection:
  -- more false positives
  -- longer FD time (think longer)

• false negatives not a problem for us
Delayed Detection/Reporting

- 0 sec w/ FRB = 53 sec w/ uRB
  --> enable more accurate diagnosis
Increased False Positive Rate

- 0% FP rate w/ FRB = 97.2% FP rate w/ uRB
  --> enable faster detection
Why Prophylactic Rebooting?
Full Rejuvenation

M_alarm = 35%
Microrejuvenation

M_{sufficient} = 80\%
Full vs. Microrejuvenation

76%

unplanned total downtime ----> planned partial downtime
## Performance Overhead

<table>
<thead>
<tr>
<th></th>
<th>JBoss 3.2.1 w/ HttpSession-RUBiS</th>
<th>uRB-JBoss w/ SSM-RUBiS</th>
<th>JBoss 3.2.1 w/ SSM-RUBiS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput [req/sec]</td>
<td>44.8</td>
<td>44.5</td>
<td>44.4</td>
</tr>
<tr>
<td>Latency [msec]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avg</td>
<td>39</td>
<td>82</td>
<td>83</td>
</tr>
<tr>
<td>Max</td>
<td>826</td>
<td>1245</td>
<td>1097</td>
</tr>
<tr>
<td>StDev</td>
<td>0.066</td>
<td>0.131</td>
<td>0.142</td>
</tr>
</tbody>
</table>

- 150 clients/node: latency=38 msec (3 -> 7 nodes)
- Human-perceptible delay: 100-200 msec
- Auction site: 41 req/sec, 33-300 msec latency
Summary of Results

- uRB-ing has many of FRB's recovery properties
- uRB is order-of-magnitude less disruptive
- always uRB prior to failing over in clusters
- cheap recovery simplifies detection (97% FP)
- rejuvenate system w/out ever shutting down
- insignificant performance cost: 1%

http://crash.stanford.edu