Microreboot
A Technique for Cheap Recovery

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

Computer Systems Lab, Stanford University

OSDI '04
Outline

• Introduction
• Designing Microrebootable Software
• A Microrebootable Prototype
• Evaluation Framework
• Evaluation Result
• A New Approach to Failure Management
• Limitations of Recovery by Microreboot
Introduction
Introduction

• Many failures can be successfully recovered by rebooting.
• However, unexpected reboots can result in data loss and unpredictable recovery time.
• Microreboot
  – Individual rebooting of fine-grain application components.
  – Keep all important application state in specialized state stores.
  – Allow transparent call-level retires to mask a microreboot from end users.
Introduction

• Microrebooting is always attempted first, as front-line recovery.
  – May be preferable even over node failover
    • Avoids overloading non-failed nodes
    • Preserves in-memory state.
Designing Microrebootable Software
Designing Microrebootable Software

• Crash-only software
  – Programs that can be safely crashed in whole or by parts.
  – Recovery quickly every time.

• Fine-Grain components
  – Components that are as small as possible.

• State segregation
  – Keep all important state in dedicated state stores located outside the application.
    • Transactional databases.
    • Session state managers.
Designing Microrebootable Software

• Decoupling
• Retryable requests
  – Use timeouts
  – RetryAfter(t)
    • The call can be re-issued after the estimated recovery time t.

• Leases
  – Resources should be leased.
    • Improve the reliability of cleaning up after uRBs.
A Microrebootable Prototype
A Microrebootable Prototype

- The J2EE Component Framework
  - Three-tiered architecture.

- Persistence tire
  - Store long-term data in one or more databases

- Application tire
  - application

- Presentation tire
  - Stateless Web servers
A Microrebootable Prototype

- J2EE application (RUBiS) runs on J2EE application server (JBoss).
- J2EE applications consist of portable components, called EJBs, and platform-specific XML deployment descriptor files.
- One container per EJB object
  - It manages all instances of that object.
- Server-managed containers provide the application components with a rich set of services.
A Microrebootable Prototype

J2EE application

EJB

EJB

Invoked methods

WAR

Web server

JSP

Return result

End User
A Microrebootable Prototype

• Microreboot Machinery
  – Microreboot method can be applied to one or more EJB WAR components.
  – Some EJBs cannot be microrebooted individually
    • Because EJBs might maintain references to other EJBs
    • Recovery groups
A Microrebootable Prototype

• A Crash-Only Application
• E-commerce applications typically handle three types of important state
  – Long-term data that must persist for years.
    • database
  – Session data that needs to persist for the duration of a user session
    • Dedicated session state storage
  – Static presentation data.
    • Ext3FS
• eBid uses only two types of EJBs
  – Entity EJBs
  – Stateless session EJBs
A Microrebootable Prototype

- **Persistent state**
  - Maintained in MySQL database.
- **If an EJB is involved in any transactions at the time of a microreboot**
  - Aborted by the container
  - Rolled back by the database.
- **Session state**
  - FastS, an in-memory repository inside JBoss’s embedded Web server
    - Only survives uRBs.
  - SSM
    - Maintain state on separate machines.
    - Survives uRBs, JVM restarts, node reboot.
Evaluation Framework
Evaluation Framework

- Client emulator

<table>
<thead>
<tr>
<th>User operation results mostly in...</th>
<th>% of all requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only DB access (e.g., browse a category)</td>
<td>32%</td>
</tr>
<tr>
<td>Initialization/deletion of session state (e.g., login)</td>
<td>23%</td>
</tr>
<tr>
<td>Exclusively static HTML content (e.g., home page)</td>
<td>12%</td>
</tr>
<tr>
<td>Search (e.g., search for items by name)</td>
<td>12%</td>
</tr>
<tr>
<td>Session state updates (e.g., select item for bid)</td>
<td>11%</td>
</tr>
<tr>
<td>Database updates (e.g., leave seller feedback)</td>
<td>10%</td>
</tr>
</tbody>
</table>

Table 1: Client workload used in evaluating microreboot-based recovery.
Evaluation Framework

- **Failure detection**
  - Implemented in the client emulator.

- **Recovery manager**
  - Performs simple *failure diagnosis* and *recovery*.
  - Listens on a UDP port for failure reports from the monitors.
  - Uses a simple recursive recovery policy.
  - Action-weighted throughput (Taw).
Evaluation Results
Evaluation Results

- Is Microrebooting Effective?
- Perform three types of data corruption
  - Set a value to null.
  - Set an invalid value.
    - userID larger than the maximum userID.
  - Set to a wrong value.
    - Swapping IDs between two users.
## Evaluation Results

<table>
<thead>
<tr>
<th>Injected Fault</th>
<th>Type</th>
<th>Reboot level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deadlock</td>
<td>EJB</td>
<td></td>
</tr>
<tr>
<td>Infinite loop</td>
<td>EJB</td>
<td></td>
</tr>
<tr>
<td>Application memory leak</td>
<td>EJB</td>
<td></td>
</tr>
<tr>
<td>Transient exception</td>
<td>EJB</td>
<td></td>
</tr>
<tr>
<td>Corrupt primary keys</td>
<td>set null</td>
<td>EJB</td>
</tr>
<tr>
<td></td>
<td>invalid</td>
<td>EJB</td>
</tr>
<tr>
<td></td>
<td>wrong</td>
<td>EJB</td>
</tr>
<tr>
<td>Corrupt JNDI entries</td>
<td>set null</td>
<td>EJB</td>
</tr>
<tr>
<td></td>
<td>invalid</td>
<td>EJB</td>
</tr>
<tr>
<td></td>
<td>wrong</td>
<td>EJB</td>
</tr>
<tr>
<td>Corrupt transaction method map</td>
<td>set null</td>
<td>EJB</td>
</tr>
<tr>
<td></td>
<td>invalid</td>
<td>EJB</td>
</tr>
<tr>
<td></td>
<td>wrong</td>
<td>EJB</td>
</tr>
<tr>
<td>Corrupt stateless session EJB attributes</td>
<td>set null</td>
<td>unnecessary</td>
</tr>
<tr>
<td></td>
<td>invalid</td>
<td>unnecessary</td>
</tr>
<tr>
<td></td>
<td>wrong</td>
<td>EJB+WAR</td>
</tr>
<tr>
<td>Corrupt data inside FastS</td>
<td>set null</td>
<td>WAR</td>
</tr>
<tr>
<td></td>
<td>invalid</td>
<td>WAR</td>
</tr>
<tr>
<td></td>
<td>wrong</td>
<td>WAR</td>
</tr>
<tr>
<td>Corrupt data inside SSM</td>
<td></td>
<td>WAR</td>
</tr>
<tr>
<td>Corrupt data inside MySQL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory leak outside application</td>
<td>intra-JVM</td>
<td>JVM/JBoss</td>
</tr>
<tr>
<td></td>
<td>extra-JVM</td>
<td>OS kernel</td>
</tr>
<tr>
<td>Bit flips in process memory</td>
<td></td>
<td>JVM/JBoss</td>
</tr>
<tr>
<td>Bit flips in process registers</td>
<td></td>
<td>JVM/JBoss</td>
</tr>
<tr>
<td>Bad system call return values</td>
<td></td>
<td>JVM/JBoss</td>
</tr>
</tbody>
</table>
Evaluation Results

• Is a Microreboot Better Than a Full Reboot?
## Evaluation Results

<table>
<thead>
<tr>
<th>Component name</th>
<th>μRB time (msec)</th>
<th>Crash (msec)</th>
<th>Reinit (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AboutMe</td>
<td>551</td>
<td>9</td>
<td>542</td>
</tr>
<tr>
<td>Authenticate</td>
<td>491</td>
<td>12</td>
<td>479</td>
</tr>
<tr>
<td>BrowseCategories</td>
<td>411</td>
<td>11</td>
<td>400</td>
</tr>
<tr>
<td>BrowseRegions</td>
<td>416</td>
<td>15</td>
<td>401</td>
</tr>
<tr>
<td>BuyNow*</td>
<td>471</td>
<td>9</td>
<td>462</td>
</tr>
<tr>
<td>CommitBid</td>
<td>533</td>
<td>8</td>
<td>525</td>
</tr>
<tr>
<td>CommitBuyNow</td>
<td>471</td>
<td>9</td>
<td>462</td>
</tr>
<tr>
<td>CommitUserFeedback</td>
<td>531</td>
<td>9</td>
<td>522</td>
</tr>
<tr>
<td>DoBuyNow</td>
<td>427</td>
<td>10</td>
<td>417</td>
</tr>
<tr>
<td>EntityGroup*</td>
<td>825</td>
<td>36</td>
<td>789</td>
</tr>
<tr>
<td>IdentityManager*</td>
<td>461</td>
<td>10</td>
<td>451</td>
</tr>
<tr>
<td>LeaveUserFeedback</td>
<td>484</td>
<td>10</td>
<td>474</td>
</tr>
<tr>
<td>MakeBid</td>
<td>514</td>
<td>9</td>
<td>515</td>
</tr>
<tr>
<td>OldItem*</td>
<td>529</td>
<td>10</td>
<td>519</td>
</tr>
<tr>
<td>RegisterNewItem</td>
<td>447</td>
<td>13</td>
<td>434</td>
</tr>
<tr>
<td>RegisterNewUser</td>
<td>601</td>
<td>13</td>
<td>588</td>
</tr>
<tr>
<td>SearchItemsByCategory</td>
<td>442</td>
<td>14</td>
<td>428</td>
</tr>
<tr>
<td>SearchItemsByRegion</td>
<td>572</td>
<td>8</td>
<td>564</td>
</tr>
<tr>
<td>UserFeedback*</td>
<td>483</td>
<td>11</td>
<td>472</td>
</tr>
<tr>
<td>ViewBidHistory</td>
<td>507</td>
<td>11</td>
<td>496</td>
</tr>
<tr>
<td>ViewUserInfo</td>
<td>415</td>
<td>10</td>
<td>405</td>
</tr>
<tr>
<td>ViewItem</td>
<td>446</td>
<td>10</td>
<td>436</td>
</tr>
<tr>
<td>WAR (Web component)</td>
<td>1,028</td>
<td>71</td>
<td>957</td>
</tr>
<tr>
<td>Entire eBid application</td>
<td>7,699</td>
<td>33</td>
<td>7,666</td>
</tr>
<tr>
<td>JVM/JBoss process restart</td>
<td>19,083</td>
<td>≈ 0</td>
<td>≈ 19,083</td>
</tr>
</tbody>
</table>
Evaluation Results

Client-perceived availability

- Bid/Buy/Sell
- Browse/View
- Search
- User Account

Timeline [seconds]
Evaluation Results

• Is Microrebooting Useful in Clusters?

![Graph showing evaluation results for node failover and recovery, and relative number of failures.](image)
Evaluation Results
### Evaluation Results

- **Performance Impact**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Throughput [req/sec]</th>
<th>Average Latency [msec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>JBoss + eBid$_{FastS}$</td>
<td>72.09</td>
<td>15.02</td>
</tr>
<tr>
<td>JBoss$<em>{μRB}$ + eBid$</em>{FastS}$</td>
<td>72.42</td>
<td>16.08</td>
</tr>
<tr>
<td>JBoss + eBid$_{SSM}$</td>
<td>71.63</td>
<td>28.43</td>
</tr>
<tr>
<td>JBoss$<em>{μRB}$ + eBid$</em>{SSM}$</td>
<td>70.86</td>
<td>27.69</td>
</tr>
</tbody>
</table>
A New Approach to Failure Management
A New Approach to Failure Management

- **Alternative Failover Schemes**
  - uRB-based recovery should always be attempted first, prior to failover.
  - Redirecting those requests to other nodes will cause many requests to fail.
  - **Microfailover**
    - LB would have to be augmented with the ability to fail over only those requests that would touch the component known to be recovering.
A New Approach to Failure Management

• User-Transparent Recovery
  – Use low-level retry mechanisms to hide failure and recovery from callers.
  – Retry-After header.

<table>
<thead>
<tr>
<th>Operation / component name</th>
<th>No retry</th>
<th>Retry</th>
<th>Delay &amp; retry</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViewItem</td>
<td>23</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>BrowseCategories</td>
<td>20</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>SearchItemsByCategory</td>
<td>31</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>Authenticate</td>
<td>20</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>
A New Approach to Failure Management

- Tolerating Lax Failure Detection
A New Approach to Failure Management

• Averting Failure with Microrejuvenation
  – Resource leaks are a major problem for many large-scale Java applications.
  – Microrejuvenation
    • uRb-based rejuvenation
    • Can be as effective as a JVM restart in preventing leak-induced failures, but cheaper.
  – A list is kept sorted in descending order by released memory.
A New Approach to Failure Management
Limitations of Recovery by Microreboot
Limitations of Recovery by Microreboot

- Impact on shared state
- Interaction with external resources
- Delaying a full reboot
Key word

• Workload
  – 指的是在網路服務中，一連串的小工作。

• Crash-Only Software
  – In high level terms, a crash-only system is defined by the equations $\text{stop} = \text{crash}$ and $\text{start} = \text{recover}$.

• Non-idempotent
  – However, it is possible that a sequence of several requests is non-idempotent, even if all of the methods executed in that sequence are idempotent. (A sequence is idempotent if a single execution of the entire sequence always yields a result that is not changed by a reexecution of all, or part, of that sequence.) For example, a sequence is non-idempotent if its result depends on a value that is later modified in the same sequence.
  – A sequence that never has side effects is idempotent, by definition (provided that no concurrent operations are being executed on the same set of resources).
• JBoss
  – J2EE application server
• RUBis
  – J2EE application
• J2EE的程式由EJBs和XML組成。