System Support for Online Reconfiguration
Craig A. N. Soules Jonathan Appavoo Kevin Hui Dilma Da Silva
Gregory R. Ganger Orran Krieger Michael Stumm Robert W. Wisniewski
Marc Auslander Michal Ostrowski Bryan Rosenberg Jimi Xenidis
Usenix 03
Outline

- Introduction
- How Online Reconfiguration
- Implementation
- Evaluation
- Conclusion
Introduction

- Two basic mechanisms for online reconfiguration:
  - Interposition
    - Wraps an active component.
    - And extending its functionality.
Introduction

- Hot-swapping
  - Replaces an active component with a new implementation.
How Online Reconfiguration

- Component boundaries
- Quiescence
- State transfer
- External references
How Online Reconfiguration

- Four requirements.
  - Component boundaries.
    - Must have well-defined boundaries.
    - System component must be self-contained with a well-defined interface and functionality.
    - C++ object.
How Online Reconfiguration

- Quiescence
  - Blocking incoming calls to the component and waiting for active calls to complete.
  - Reader-writer lock.
- Drawbacks.
  - Overhead on each component.
  - Difficult to implement.
  - Deadlock.
- Generation count.
How Online Reconfiguration

- **State transfer**
  - Internal state can be transferred from old component to new component.
  - Transfer negotiation protocol.

- **External references**
  - System can modify all external references to a swappable component.
  - Object translation table.
Implementation

Interposition

Object Translation Table

Caller

Component

(1)

(2)

(3)
Implementation

Interposition

Object Translation Table

Interposer

Wrapper

preCall

postCall

Component

Caller

(1)

(2)

(3)

(4)

(5)

(6)
Implementation

Hot-swapping

Mediator

Thread

a

b

c

? 

New

Old
Implementation

Hot-swapping

Mediator

New

Old

Thread
d
e
b
Implementation

Hot-swapping

Mediator

Thread

d → e → f

New

Old
Evaluation

- **Basic overhead**
  - The indirection used to update external references.
- **Interposition**

<table>
<thead>
<tr>
<th>Operation</th>
<th>uSeconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attach</td>
<td>17.84</td>
</tr>
<tr>
<td>Component call</td>
<td>1.40</td>
</tr>
<tr>
<td>Detach</td>
<td>4.23</td>
</tr>
</tbody>
</table>

Memory allocation
Object initialization
Evaluation

- Hot-swapping
Evaluation

- Single v. Replicated

![Bar chart comparing performance of Single and Replicated systems]

Performance (normalized to Single):

- Postmark
- 4-way SDET
Evaluation

- Exclusive v. Shared

![Bar chart showing transactions per second for Default, Exclusive, and Small File Cache cases.](image)
Evaluation

![Graph showing the comparison between Default FCM and Adaptive FCM in terms of 1-way SDET throughput (scripts/hour) vs. Number of concurrent background streams. The graph indicates that Adaptive FCM consistently outperforms Default FCM as the number of concurrent background streams increases.]
Online reconfiguration provides an underlying mechanism for component extension and replacement through interposition and hot-swapping.