Applying Reflective Middleware Techniques to Optimize a QoS-enabled CORBA Component Model Implementation

Nanbor Wang
Kirthika Parameswaran
Dept. of Computer Science
Washington University
{nanbor,kirthika}@cs.wustl.edu

Michael Kirchier
Siemens ZT
Munich, Germany
Michael.Kircher@mchp.siemens.de

Doug Schmidt
Dept. of Electrical and Computer Engineering
Univ. of California, Irvine
schmidt@uci.edu

June 18, 2002
Research Sponsored by NASA
Presentation Outline

- Limitations of CORBA Object Model
- CORBA Component Model overview
- Research Challenge
  - QoS-aware collocation
  - QoS-enabled CCM containers implementation
  - Dynamic linking/unlinking component implementation
- Concluding Remarks
Overview of CORBA Object Model

Limitations

• No standard way to deploy object implementations
• Limited server programming support
• Common object services are not mandatory
• Hard to extend existing interfaces
Standardized Features in CCM

- Run-time environment - component servers
- Packaging & deployment - CIF, CIDL & others
- Component configuration
- Component collaboration
Components Composition in CCM

- Components can be configured dynamically
- CCM supports built-in introspection mechanisms
  - e.g., the Navigate interface

Washington University, St. Louis
The CCM Container Programming Model

- **Components** are implemented as DLLs
- **Component servers** execute the components
- **Containers**
  - Define the component-server interfaces
  - Bridge the managed component implementation with common ORB services
  - Provide customized run-time environment of components
- **Standard interfaces for packaging & deploying components**

---

Nanbor Wang

QoS-enabled CORBA Component Model

---

Washington University, St. Louis
Key Research Challenge: Optimizing the CCM

- **QoS-aware Collocation Optimizations**
  - Components are likely collocated with each others

- **QoS-enabled Containers**
  - Separation of QoS aspects from component implementations

- **Dynamic Configuration of Component Servers**
  - Conserve & compose system resources
QoS-aware Collocation Optimizations

- Collocation optimizations must respect object’s QoS properties
  - e.g., may choose less efficient transport mechanisms to satisfy QoS requirements
- Stubs provide reflection interface for QoS requirements of object references
Collocation Mechanisms

- Criteria to select invocation mechanism: speed, QoS capability, & thread priority
- New multi-threaded shared-memory transport for co-host operations

<table>
<thead>
<tr>
<th>Throughput (calls/sec)</th>
<th>iiop</th>
<th>shmiop</th>
<th>uiop</th>
<th>thru_poa</th>
<th>direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT small seq</td>
<td>4709</td>
<td>6106</td>
<td></td>
<td>340701</td>
<td>745217</td>
</tr>
<tr>
<td>NT large seq</td>
<td>2635</td>
<td>4020</td>
<td></td>
<td>244841</td>
<td>508034</td>
</tr>
<tr>
<td>Sun small seq</td>
<td>1088</td>
<td>1207</td>
<td>1247</td>
<td>30195</td>
<td>65467</td>
</tr>
<tr>
<td>Sun large seq</td>
<td>819</td>
<td>903</td>
<td>926</td>
<td>30183</td>
<td>62520</td>
</tr>
</tbody>
</table>
QoS-enabled Containers

- Separate QoS controlling mechanisms & aspects from component implementations
  - Components or deployment information specify the QoS properties
  - QoS Property Adaptor for policy-based QoS management
- Containers provide interfaces for reflecting into the QoS requirements of components
Dynamic Configuration of Component Servers

- **Dynamic linking & unlinking** of component implementations
  - Loading strategy in deployment info
  - Components and their instances carry usage information
- **Caching** component instances
- **Applying eviction policies** to remove unused servants
- ORB features discovery
Concluding Remarks

• CCM simplifies application implementation
• We’re exploring the application of reflective principles to implement CCM middleware
  ▪ QoS-aware collocation mechanism
  ▪ Reflective container to separate QoS concerns
  ▪ Dynamic component linkage
• Current status and future work
  ▪ Prototyping CCM implementation
  ▪ Implementing, benchmarking, & optimizing QoS collocation & containers

Download the open-source TAO ORB from:
http://www.cs.wustl.edu/~schmidt/TAO.html