The Performance of Component-Based Software Systems

Liam Murphy
University College Dublin
Agenda

• Introduction
• Component Based System
• Performance Problems
• Research Goals
• Research Results
What is the Performance Engineering Lab

• Joint Dublin City University and University College Dublin
• Located on both sites within Dublin
• 5 Academic staff
• 20 Post graduate research staff
• Performance related research
  • Mobile and multimedia
  • Distributed components
Components

- A component is a lego™ like piece of code
- Allow developers to concentrate on business logic
- Decouple the logic from architecture
- Distributed Components Offer
  - Security
  - Robustness
    - Persistence
    - Transactional capability
  - Scalability
Components

Reservation  Payment  Payment

ORB

May 2003
Technologies

- **CORBA**
  - Open standard created by OMG
- **.NET**
  - Developed by Microsoft
- **EJB**
  - Developed by SUN
  - J2EE technology de facto standard 65-75% enterprises (Gig 2002)
J2EE
EJB

SUN Microsystems’ definition of Enterprise JavaBeans is:

The Enterprise JavaBeans architecture is a component architecture for development and deployment of component-based disturbed business application. Application written using the EJB architecture are scalable, transactional and multi-user secure. The applications may be written once, and deployed on an server platform that supports the Enterprise JavaBeans specification.
Distributed Objects

1. Invoke Method
2. Communicate Method Invocation
3. Invoke Method
4. Communicate Result
5. Return Result
Container

- Container
  - Environment that surrounds the bean
  - Provides primary services
    - Transaction
    - Security
    - Naming
    - Concurrency
- Java Virtual Machine (JVM)
Know Your Beans

• Entity Beans
  – Hold persistent data structures used by application
  – Bean represents a logical record in DB

• Session Beans
  – Workflow or task oriented
  – Can be stateless or stateful

• Message Beans
  – Asynchronous messaging
  – Loosens coupling between sender and receiver
Performance Problems
Communication Overhead

- Communication Overhead
  - Remote Calls transport objects
    - Serialization expensive, especially for complex structures
    - Required in same JVM
    - Can use Local Interfaces (EJB 2.0)
      - Provides Direct interface to bean methods
      - Deployment and Clustering Flexibility reduced
Managing Persistence

• Container Managed Persistence
  – Vendor dependent implementation may help
    • Can result in loss of portability

• Bean Managed Persistence
  – Provides more control

• Generally use CMP
Entity Beans & DB Access

• Entity beans keep persistent state
  – 1 Bean = 1 row in the database

• Entity beans are good when
  – Few rows are selected from the DB and accessed by multiple clients
    – e.g. a person’s record and used by securityBean & BankAccountBean

• DON’T used them when
  – Need to access multiple rows
    – e.g. batch operations
Stateless vs Stateful Session Bean

• Stateless
  – EJB Containers pool stateless beans
  – Can be reused by many clients
  – Need to have client data passed in method call

• Stateful
  – Stateful session been can be used by many clients
  – Session been caches client data in memory
    • Possible I/O bottlenecks

• Trade-off
  – Multiple invocations – stateful
  – Single invocations - stateless
Example

- Work performed by Rice University
- Auction site implemented in 5 different ways
  - Servlets
  - Session Beans
    - Business logic implemented in session bean
  - EB CMP
    - Data Access implemented in a Entity Bean
  - EB BMP
  - Session Facade
    - Communication through a stateless session bean
  - EJB 2.0 Local
## Complexity

<table>
<thead>
<tr>
<th></th>
<th>Servlets</th>
<th>Beans</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Classes</td>
<td>LoC</td>
<td>Classes</td>
</tr>
<tr>
<td>Servlets-only</td>
<td>23</td>
<td>4540</td>
<td>-</td>
</tr>
<tr>
<td>Session Beans</td>
<td>22</td>
<td>2730</td>
<td>51</td>
</tr>
<tr>
<td>EB CMP</td>
<td>23</td>
<td>3980</td>
<td>40</td>
</tr>
<tr>
<td>EB BMP</td>
<td>23</td>
<td>3980</td>
<td>40</td>
</tr>
<tr>
<td>Session Facade</td>
<td>22</td>
<td>2660</td>
<td>85</td>
</tr>
<tr>
<td>EJB 2.0 Local</td>
<td>22</td>
<td>2725</td>
<td>91</td>
</tr>
</tbody>
</table>
Trade 3 Overview

6 entity beans, 2 session beans and 2 message driven beans
Trade 3 Configuration options

- EJB or direct mode (JDBC)
- Order processing immediately or asynchronous MDB
- Access mode web application $\rightarrow$ server side services: RMI, SOAP-RPC, Web Services (soon)
- Workload mix: Standard, High-Volume
Throughput

Throughput over time for HTTP Responses/s and Active Virtual Users.
Monitored values

% Disk utilization
% processor time
available memory

120-140MB
More Problems

- “the performance challenges typically revolve around excessive or inefficient accesses to backend systems such as databases and mainframe systems. “
- “Poor design does not allow optimization”
- “Excessive method calls”
- “Not designing in an architectural context”
- “No consideration for concurrent users”
- “Tuning the number of JVMs serving the clients and tuning the number of threads per JVM”
Research Goals
### Design Cycle

<table>
<thead>
<tr>
<th>System Architects</th>
<th>Performance Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developers</td>
<td>Profiling Tools</td>
</tr>
<tr>
<td>Testers</td>
<td>Performance Test</td>
</tr>
<tr>
<td>Production Support</td>
<td>Application Monitoring</td>
</tr>
<tr>
<td></td>
<td>Capacity Planning</td>
</tr>
<tr>
<td>Decide</td>
<td>Design</td>
</tr>
<tr>
<td></td>
<td>Development</td>
</tr>
<tr>
<td></td>
<td>Deploy</td>
</tr>
</tbody>
</table>

$3-4$ Billion Market
Research Goal

• $3-4 Billion market spent managing performance
• Two key Research Areas
  – Improve detection of Performance Problem
    • COMPAS
  – Improve Performance of in-production systems
    • Component Optimization
    • Load Balancing
COMPAS

Helping developers solve EJB performance problems..
Traditional Approach

Performance Test
On pre-production environment

Experience Shows
• Time is Critical
• Mistakes are costly

Decide ➔ Design ➔ Develop ➔ Deploy
Traditional Monitoring/Testing Tools

• Need Production like environments
  – Only available at end of project
  – Costly to construct & maintain
• Find Problems one at a time
  – Multiple Fix-Test-Fix cycles….
• Testing Takes Time and Money
Why is COMPAS unique

• Looks beyond the first performance problem
  – InSight Prediction
• Predicts performance for different hardware
  – ForeSight Prediction
• Expands the performance test envelope
  – ClearSight Prediction
InSight Prediction

 Helps find all the Performance Problems at once

- Synchronous Messaging
- Too Many Method Calls
- Single Threaded Code
- In-efficient Code
- DB Indexes
ForeSight Prediction

Find Performance Problems on less hardware

Content Providers

Fire Wall

LAN

Web Servers

Application Servers

Legacy System

DB Servers

May 2003
ClearSight Predication

Performance Testing Takes Time…

Scenarios

From this

To this

Volume

Leaving performance problems yet undetected.

Tests are run.
Component Optimization

- Intelligent load balancing
  - Switches load based on current and historic performance
- Self adaptive components (Autonomic computing)
  - Same functional components different performance curves
  - Detecting and switching when needed
- Container Optimization
  - Merging components for better performance
Conclusion

- Component Technology
  - Allows developers to concentrate on business logic
  - Bring technology specific performance problems
  - Removing an architectural context can create performance problems

- PEL Research
  - COMPAS Performance Prediction
  - Component optimization