An Architectural Evaluation of Java TPC-W

Harold “Trey” Cain, Ravi Rajwar, Morris Marden, Mikko Lipasti

University of Wisconsin at Madison
http://www.ece.wisc.edu/~pharm

Seventh International Symposium on High Performance Computer Architecture
January 2001
Introduction

- Why do workload characterization?
  - Java: gaining widespread use in server-side middleware applications
  - Very little known about the architectural requirements server-side Java
- TPC-W: a mixed transaction processing/web serving benchmark
  - Web application middleware implemented in Java
Outline

- TPC-W Overview
- Our Java-based implementation of TPC-W
- Native Execution Results
  - Memory System Characterization
  - Collected using performance counters on an IBM RS/6000 S80 Server
  - Results for TPC-W, SPECjbb2000, SPECweb99
- Simulation Results
  - Coarse Grained Multithreading Evaluation
What is TPC-W?

- New benchmark specified by the Transaction Processing Council (in February 2000), targeting transactional web systems
  - Web Serving of static and dynamic content
  - On-line transaction processing (OLTP)
  - Some decision support (DSS)
- Models an on-line bookstore
- Consists of 14 browser/web server interactions
3-Tier Application

Web Browsing Users

TPC-W System Under Test

Web Server(s)

Database Server(s)

HPCA-7  January 2001

Cain/Rajwar/Marden/Lipasti
Web Interaction Characteristics

- Dynamic HTML required: 11/14 interactions
- DB connectivity required: 11/14 interactions
  - Query complexity varies
  - Read-only and Read/Write
- Number of images per page:
  - Varies from 3 to 9, 6 on average
- Maximum response time:
  - Varies from 3 to 20 seconds
Web Interaction Mixes

- Different web sites have different usage patterns
- TPC-W models variance using three different transaction mixes
  - Browsing Mix
    - 95% browsing, 5% ordering
  - Shopping Mix (Primary performance metric)
    - 80% browsing, 20% ordering
  - Ordering Mix (business to business)
    - 50% browsing, 50% ordering
Java Implementation of TPC-W

- All 14 TPC-W web interactions implemented as Java Servlets
- JDBC used to communicate to a database back-end (DB2)
- Did not implement
  - Secure Transactions using secure sockets layer (SSL)
  - Communication with payment gateway authority
Outline

- TPC-W Specification
- Our implementation of TPC-W
- **Native Execution Results**
  - Memory System Characterization
  - Collected using performance counters on an IBM RS/6000 S80 Server
  - TPC-W, SPECweb99, SPECjbb2000
- Simulation Results
  - Coarse Grained Multithreading Evaluation
System Parameters

- **Hardware**
  - 6 processor IBM RS/6000 S80, AIX 4.3
  - RS-64 III (Pulsar) PowerPC processors
  - 8 GB memory
  - 8 MB 4-way set associative L2 caches
  - 128 KB I-Cache, 128 KB D-Cache, 2-way set associative

- **Software:**
  - Zeus Web Server v. 3.3.7
  - Apache JServ Servlet Engine 1.0, Java 1.1.8 w/ JIT
  - DB2 Universal Database 6.1
  - Database Size: 205 MB
  - Image Set Size: 250 MB
CPU Time by Application Component

Java Servlet Engine Dominates CPU Usage
CPI Breakdown

- Most stalls due to L2 cache misses

HPCA-7  January 2001  Cain/Rajwar/Marden/Lipasti
Load misses dominate, except in DB2
Cache-to-Cache Transfers
Coherence Protocols: To E or not to E

- Removing E state would necessitate an extra bus transaction for 9%-28% of all L2 Misses.
Outline

- TPC-W Specification
- Our implementation of TPC-W
- Native Execution Results
  - Memory System Characterization
  - Collected using performance counters on an IBM RS/6000 S80 Server
  - TPC-W, SPECweb99, SPECjbb2000

- Simulation Results
  - Coarse Grained Multithreading Evaluation
Full System Simulation

- Due to the large amount of time spent in system code, full system simulation is necessary.

- SimOS-PowerPC
  - Runs modified version of AIX 4.3.1
  - System configuration occurs on real system, then a disk snapshot is created
  - Snapshot used by SimOS-PPC

- We simulate a three second snapshot of steady-state behavior
Simulated Machine Parameters

- Single-issue, in-order 500 MHZ processor
- L1 I-Cache: 128 KB, 2-way associative
- L1 D-Cache: 128 KB, 2-way associative
- L2 Cache: 8 MB, 4-way associative
- Memory: 1 GB
- Bus models the Sun Gigaplane-XB

- System configuration is considerably different from IBM S80
Coarse Grained Multithreading

- Processor contains logic for switching among several threads of execution and maintaining multiple thread contexts.

- Switch thread when:
  - Cache miss occurs in primary thread, and a suspended thread is in the ready state.
  - The primary thread is in a spin loop or the idle loop, and a suspended thread in the ready state.
  - A suspended thread has a pending interrupt or exception.
  - A suspended ready thread has not retired an instruction in the last 1000 cycles.

- 3 cycle thread switch penalty
CGMT Results

2 threads: increases throughput as much as 41%
4 threads: increases throughput as much as 60%

HPCA-7 January 2001
Cain/Rajwar/Marden/Lipasti
Conclusions

- Java servlet engine is performance critical
  - L2 cache miss stalls to unshared data are primary contributor to memory system stalls

- The exclusive state successfully reduces memory bus traffic for these commercial workloads.

- Coarse grained multithreading:
  - Decreases cache hit rates
  - Decreases branch prediction accuracy

- However, total system throughput improves due to CGMT’s memory latency tolerance.
Questions?
# Web Interaction Characteristics

<table>
<thead>
<tr>
<th>Name</th>
<th>Dynamic Html?</th>
<th>DB Complexity</th>
<th># Images</th>
<th>Max Resp Time</th>
<th>Browsing Mix</th>
<th>Shopping Mix</th>
<th>Ordering Mix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin Confirm</td>
<td>Yes</td>
<td>$O(n^4)$</td>
<td>5</td>
<td>20</td>
<td>0.09 %</td>
<td>0.09 %</td>
<td>0.11 %</td>
</tr>
<tr>
<td>Admin Request</td>
<td>Yes</td>
<td>$O(n^2)$</td>
<td>6</td>
<td>3</td>
<td>0.10 %</td>
<td>0.10 %</td>
<td>0.12 %</td>
</tr>
<tr>
<td>Best Seller</td>
<td>Yes</td>
<td>$O(n^2)$</td>
<td>9</td>
<td>5</td>
<td>11.00 %</td>
<td>5.00 %</td>
<td>0.46 %</td>
</tr>
<tr>
<td>Buy Confirm</td>
<td>Yes</td>
<td>$O(n)$</td>
<td>2</td>
<td>5</td>
<td>0.69 %</td>
<td>1.20 %</td>
<td>10.18 %</td>
</tr>
<tr>
<td>Buy Request</td>
<td>Yes</td>
<td>$O(n)$</td>
<td>3</td>
<td>3</td>
<td>0.75 %</td>
<td>2.60 %</td>
<td>12.73 %</td>
</tr>
<tr>
<td>Customer Registration</td>
<td>No</td>
<td>N/A</td>
<td>4</td>
<td>3</td>
<td>0.82 %</td>
<td>3.00 %</td>
<td>12.86 %</td>
</tr>
<tr>
<td>Home</td>
<td>Yes</td>
<td>$O(n)$</td>
<td>9</td>
<td>3</td>
<td>29.00 %</td>
<td>16.00 %</td>
<td>9.12 %</td>
</tr>
<tr>
<td>New Product</td>
<td>Yes</td>
<td>$O(n^2)$</td>
<td>9</td>
<td>5</td>
<td>11.00 %</td>
<td>5.00 %</td>
<td>0.46 %</td>
</tr>
<tr>
<td>Order Display</td>
<td>Yes</td>
<td>$O(n)$</td>
<td>2</td>
<td>3</td>
<td>0.25 %</td>
<td>0.66 %</td>
<td>0.22 %</td>
</tr>
<tr>
<td>Order Inquiry</td>
<td>No</td>
<td>N/A</td>
<td>3</td>
<td>3</td>
<td>0.30 %</td>
<td>0.75 %</td>
<td>0.25 %</td>
</tr>
<tr>
<td>Product Detail</td>
<td>Yes</td>
<td>$O(n^2)$</td>
<td>6</td>
<td>3</td>
<td>21.00 %</td>
<td>17.00 %</td>
<td>12.35 %</td>
</tr>
<tr>
<td>Search Request</td>
<td>No</td>
<td>N/A</td>
<td>9</td>
<td>3</td>
<td>12.00 %</td>
<td>20.00 %</td>
<td>14.54 %</td>
</tr>
<tr>
<td>Search Result</td>
<td>Yes</td>
<td>$O(n^2)$</td>
<td>9</td>
<td>10</td>
<td>11.00 %</td>
<td>17.00 %</td>
<td>13.08 %</td>
</tr>
<tr>
<td>Shopping Cart</td>
<td>Yes</td>
<td>$O(n)$</td>
<td>9</td>
<td>3</td>
<td>2.00 %</td>
<td>11.60 %</td>
<td>13.53 %</td>
</tr>
</tbody>
</table>
Online Bookstore

- Functionality:
  - Searching
  - Browsing
  - Shopping carts and secure purchasing
  - Rotating advertisements
  - Best seller and new product lists
  - Customer registration
  - Administrative updates
Remote Browser Emulator

- Emulates web users interacting through browsers
- Non-deterministic walk over web pages
  - Send HTTP request
  - Parse HTTP response for images and other URLs
  - Wait for think time (~7 seconds)
  - Repeat
Database Scaling

- Database size depends on two factors:
  - Number of items in bookstore inventory
  - Number of bookstore customers
- ~5MB in DB Tables per active user (like TPC-C)
- ~1 KB per item in DB tables (like TPC-D)
- Also ~25KB of static images per item
  - Images may be stored in database or standard file system