Memory System Behavior of Java-Based Middleware

Martin Karlsson, Kevin E. Moore, Erik Hagersten and David A. Wood
Java-Based Middleware: An Important New Workload for Multiprocessor Servers

• Java-Based middleware connects Web pages to databases
• Web-based applications are deployed in 3-tier systems
  – Clients
  – Middleware (e.g. application servers)
  – Databases
• Rapid growth
• Diverse clients will increase the role of middleware
Java Middleware Benchmarks

• SPECjbb2000
  – Approximates a 3-tier system in a single application
  – Will run on any JVM without any 3rd-party software
  – Easy to install, tune and run (set up time measured in hours)

• ECperf (now SPECjAppServer2001)
  – Runs on a real 3-tier system
  – Easy to isolate the behavior of individual tiers
  – Requires expensive 3rd-party software (application server and database)
  – Difficult to install, tune and run (set up time measured in weeks)
Outline

• Background
  – 3-Tiered Systems
  – ECperf and SPECjbb2000

• Hardware monitoring experiments
  – System size scaling
  – Benchmark scaling

• Simulation Experiments
  – Cache Performance

• Design decisions
  – Shared Caches
Application Servers & 3-Tiered Systems

- 3-tiered systems are common in e-commerce and B2B applications
- Application servers provide a framework for middle-tier applications
  - Presentation
  - Business Rules
- Services include
  - Database connectivity
  - Client connectivity
  - Resource management
- Application servers often implemented in Java

![Diagram of 3-tiered systems]

Users/Customers (e-commerce)

Tier 1

Tier 2

Tier 3
ECperf

- Runs on top of existing commercial applications (Database and Application Server)
  - Adds Cost, tuning effort
  - Restricted source code
- Consists of 4 networked programs
  - Application Server
  - Database
  - Supplier Emulator
  - Driver
- Runs on multiple machines
  - Easy to isolate tiers
ECperf

- Runs on top of existing commercial applications (Database and Application Server)
  - Adds Cost, tuning effort
  - Restricted source code
- Consists of 4 networked programs
  - Application Server
  - Database
  - Supplier Emulator
  - Driver
- Runs on multiple machines
  - Easy to isolate tiers
SPECjbb2000

- Single JVM
- Database emulated by trees of Java objects
- Easy to install, tune, and run
- Available source code
- Difficult to measure behavior of individual tiers
SPECjbb2000

- Single JVM
- Database emulated by trees of Java objects
- Easy to install, tune, and run
- Available source code
- Difficult to measure behavior of individual tiers

Measurements include database and client code
Outline

• Background
  – 3-Tiered Systems
  – ECperf and SPECjbb2000

• Hardware monitoring experiments
  – System size scaling
  – Benchmark scaling

• Simulation Experiments
  – Cache Performance

• Design decisions
  – Shared Caches
Monitoring Experiments

• Hardware
  – Sun E6000 (SPECjbb2000, Application Server, Database)
    • 16, 248 MHz UltraSparc II processors
    • 2 GB RAM
    • 1 MB unified L2 cache
  – Sun Netra (Emulator, Driver)
    • 1, 500 MHz UltraSparc Ile

• Software
  – HotSpot 1.3.1 JVM
  – Solaris 8
Benchmark Settings and Alterations

- **SPECjbb2000**
  - Increased warm-up and measurement intervals
    - 60 s warm-up and 6 min measurement
  - Picked 1 value for the number of warehouses
    - \#warehouses = \#processors

- **ECperf**
  - Relaxed response time requirements

- **JVM Options**
  - Heap Size = 1424 MB
  - ISM
  - New Generation = 400 MB
Performance Scaling

![Graph showing performance scaling with varying processors, comparing SPECjbb and ECperf. The graph includes a linear trend line.](image-url)
Data Sharing

![Graph showing Cache to Cache Transfer Ratio (%) against Processors]

- **ECperf**
- **SPECjbb**

HPCA February 11, 2003  Memory System Performance of Java-Based Middleware
Memory Use vs. Scale Factor (8 p)
Scaling Effects

• Scaling System Size
  – Increased system size from 1 to 15 processors
  – High Idle times for both benchmarks on large systems
  – Contention inside the application or JVM
  – High fraction of sharing misses on large systems
  – Very few misses to main memory despite large heap
  – CPI (ECperf 2.0-2.8, SPECjbb2000 1.8-2.3)

• Benchmark Scaling
  – Increased transaction input rate and database size
    • ECperf: Orders Input Rate
    • SPECjbb2000: Warehouses
  – Affects SPECjbb2000 more than ECperf
Outline

• Background
  – 3-Tiered Systems
  – ECperf and SPECjbb2000

• Hardware monitoring experiments
  – System size scaling
  – Benchmark scaling

• Simulation Experiments
  – Cache Performance

• Design decisions
  – Shared Caches
Cache Simulations

- Experiments conducted with Virtutech Simics with an extended memory system simulator
  - 4-way set associative caches
  - 64 byte cache lines
- Cache Miss Rates
  - Uniprocessor simulations
  - Split 1-level caches
- Sharing Analysis
  - 8-processor simulations
  - Unified cache
Data Cache

![Data Cache Graph]

- Misses/1000 Instructions vs Cache Size (KB)
- **Lines and Symbols**:
  - **ECperf**
  - **SPECjbb-25**
  - **SPECjbb-10**
  - **SPECjbb-1**

---

**Figure Legend**:
- **Cache Size (KB)**: 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384
Instruction Cache

![Graph showing the relationship between cache size (KB) and misses per 1000 instructions for different benchmarks. The graph includes data for ECperf and SPECjbb with various configurations.](image-url)
Communication Distribution

- ECperf
- SPECjbb-25

- 20%, 88.5%
- 12.3%, 100%

Percent Cache-to-Cache Transfers (%)

Percent of All Cache Lines (%)
Outline

• Background
  – 3-Tiered Systems
  – ECperf and SPECjbb2000

• Hardware monitoring experiments
  – System size scaling
  – Benchmark scaling

• Simulation Experiments
  – Cache Performance

• Design decisions
  – Shared Caches
Shared Caches

• Potentially a good fit for Java-based middleware
  – High cache-to-cache transfer ratio
  – Small working sets
  – Low memory bandwidth

• Important design point for CMPs

• Experiment: Measured data miss rate for a simulated 8-processor system running each benchmark
  – All caches are 1MB
  – Varied number of caches and degree of sharing
Data Miss Rate vs. Sharing Degree

![Graph showing data miss rate vs. sharing degree](image)
Paper Summary

• Descriptions of ECperf and SPECjbb2000
• Combination of hardware monitoring and full-system simulation
  – Scalability
  – Execution time breakdown
  – I/D Cache performance
  – Input rate scaling
• Effects of garbage collection
• Data sharing analysis and shared-cache performance
• Conclusion
  – Benchmark differences can lead to opposite design conclusions
Conclusions

• Both SPECjbb2000 and ECperf
  – Have small data sets
  – High rate of sharing misses
• SPECjbb2000 approximates ECperf well except for 2 important differences
  – ECperf has a much larger instruction footprint and a higher instruction miss rate
  – The memory footprint of SPECjbb2000 is larger than that of ECperf, especially on large systems