Exploring Small-Scale and Large-Scale CMP Architectures for Commercial Java Servers

R. Iyer, M. Bhat, L. Zhao, R. Illikkal, S. Makineni, M. Jones, K. Shiv, D. Newell

Intel Corporation

Outline

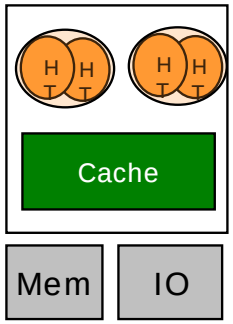
- Motivation
- CMP Overview
- Java Workload Overview
- Methodology & Evaluation Results
- Summary

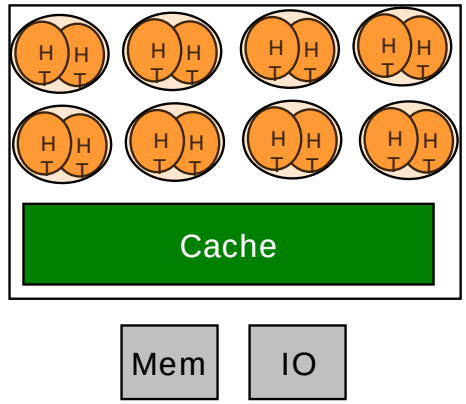
Motivation

- CMP architecture has been widely adopted
  - SCMP: a few large out-of-order cores
    - Intel Dual-core Xeon processor
  - LCMP: many small in-order cores
    - Sun Niagara, Azul
- Platform resources are different for S/LCMP
  - Cores, cache/memory architecture, memory
- How to make CMP evolve successfully

Identify critical challenges in future CMP platforms by evaluating the performance of a commercial server workload on current and future CMP platforms

CMP Overview





- SCMP: few large cores
  - Provide significant hardware features
    - out-of-order execution, complex branch prediction
  - Single-thread performance is high
- LCMP: many small cores
  - Strip out area- and power-intensive features
    - In order execution, simple branch prediction
  - Provide high throughput

SCMP vs. LCMP

- Cache scaling
  - Cache space is smaller for LCMP than for SCMP
  - Important to analyze and compare cache behavior
- Memory bandwidth requirement
  - Higher for LCMP than for SCMP
  - Memory technology trend
    - DDR bandwidth and latency trends do not scale at the same pace as the bandwidth requirements for SCMP
  - Study bandwidth requirement

Outline

- Motivation
- CMP Overview
- Java Workload Overview
- Methodology
- Evaluation Results
- Summary

SPECjbb2005 Overview

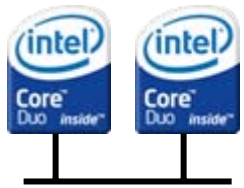
- SPEC's benchmark for evaluating the performance of server side Java applications
- Emulate a 3-tier client/server system on a single platform
  - client, business logic, database
  - The system modeled is a wholesale company, with warehouses that serve a number of districts.
- Warehouse
  - Contain 25MB of data (Java Collection objects)
  - Database tables (Java classes), data records (Java objects)
- Six types of transactions against the warehouse
  - New Order, Order Status, Payment, Delivery, Stock level, Customer Report
- Self contained and self driving
  - Generate its own data, multi-threaded operations, depend on no other package beyond the JRE
- Memory resident
  - local network I/O, no I/O to disk
- Exercise implementations of the JVM, JIT, garbage collector and threads

Outline

- Motivation
- CMP Overview
- Java Workload Overview
- Methodology & Evaluation Results
- Summary

Measurement-based Methodology

- Intel's dual-core dual-socket server platform
  - Two Intel Core 2 Duo processors
    - Each processor consists of two cores (3GHz)
    - 4M L2 cache
  - 4 FBD channels (533 MT/s) with peak bw @ 25.6GB/s



- EMON (Intel performance monitoring tool)
- Study the performance scaling characteristics
  - Number of cores, number of sockets, cache size

Measurement Results

Processor Scaling	1S1C	1S2C	2S2C	2S4C
Throughput- Scaling	1.00	1.56	1.81	2.89
CPI	1.03	1.32	1.14	1.38
Pathlength	85,942	86,163	86,104	88,368
MPI	0.0031	0.0039	0.0032	0.0039
Time Spent in GC	1.83%	2.38%	2.86%	4.28%

- 1S2C does not achieve the same performance improvement as 2S2C
  - LLC sharing (MPI increased by 23%)
- Speedup is close to perfect when going from 1S to 2S while keeping cores/socket constant

Cache Scaling and Frequency Scaling

Cache Scaling	1M	2M	4M
Throughput-Scaling	1.00	1.30	1.80
Pathlength	87,296	87,927	88,368
CPI	2.52	1.93	1.38
MPI	0.0087	0.0063	0.0039
HIT%	6%	4%	3%
HITM%	0%	0%	0%

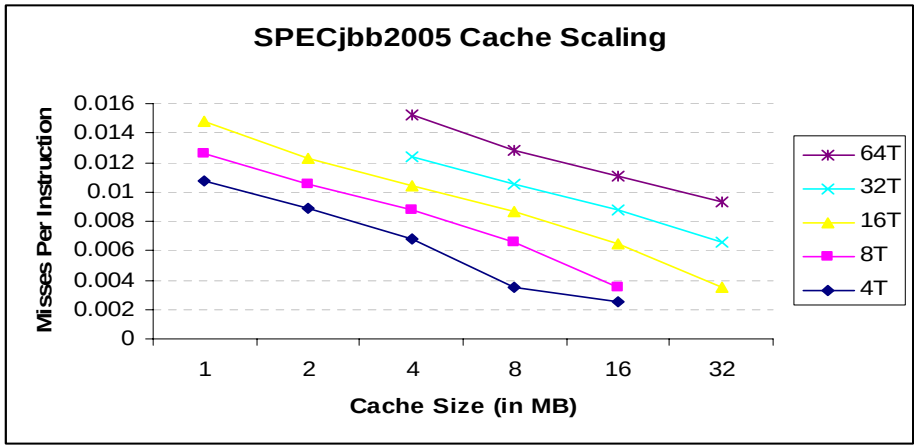
Frequency-GHz	2	2.67	3
Throughput-Scaling	1.00	1.19	1.25
Pathlength	88,126	88,061	88,368
CPI	1.16	1.29	1.38
MPI	0.0039	0.0039	0.0039

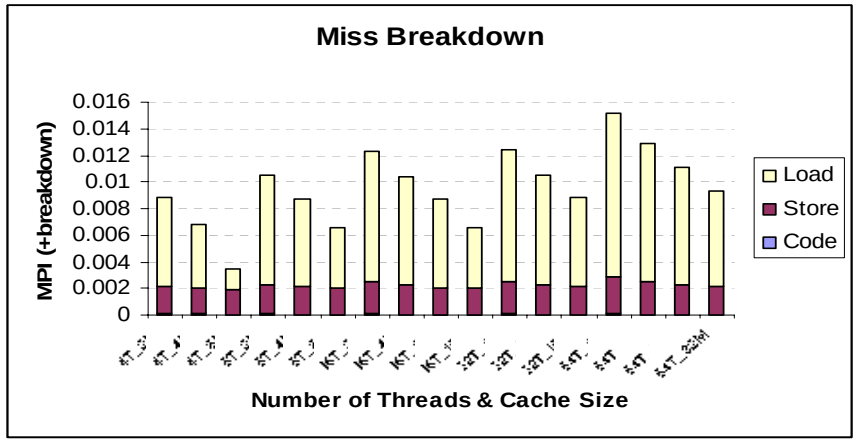
- Use 2S4C, vary the LLC size
  - ~30 to 40% improvement when double the cache size
  - Coherence / sharing behavior
    - HIT(M)%: % of misses finding line in other cache in S/E(M) state
    - HITM% is very low, HIT% decreases as we increase cache size
  - There is little sharing between SPECjbb2005 threads
- Use 2S4C, vary the frequency
  - 25% improvement when frequency is increased by 50%
  - The speedup is limited by CPI increase

Simulation-based Methodology

- Cache simulator
  - Study the impact of scaling threads and cache size
  - SCMP
    - 4/8 threads per processor socket
    - 1M ~ 16M
  - LCMP
    - 32/64 threads per processor socket
    - 4M ~ 32M
  - Focus on LLC
  - MPI: misses per instruction

Cache Scaling for SCMP & LCMP



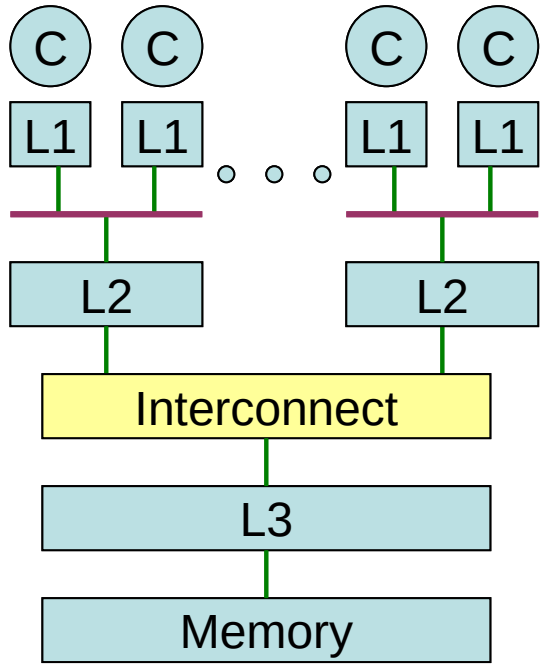


- MPI decreases constantly
- MPI stays relatively constant as the number of threads and cache size scales proportionally
- MPI breakdown
  - Code MPI is negligible
  - Write MPI is relatively constant
  - Read MPI varies as a function of cache size

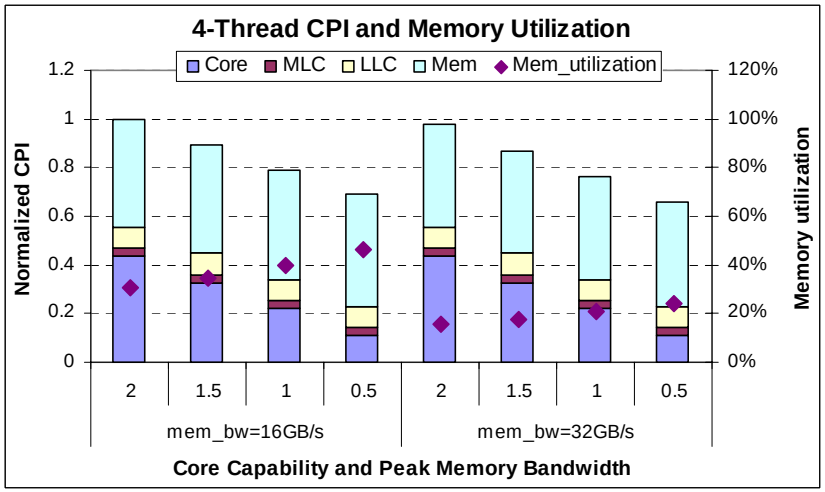
Data misses are the main contribution and scale with cache size

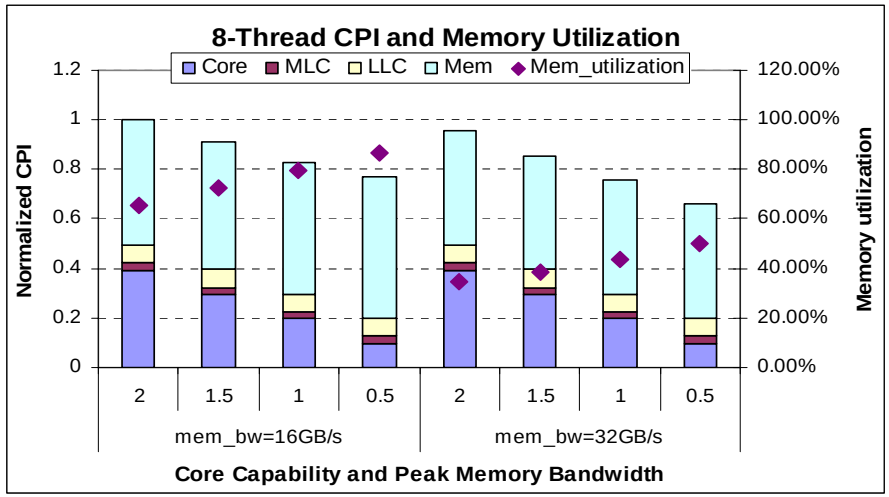
Simulation-based Methodology

- Platform simulation
  - Bandwidth and performance
  - Core model
  - Cache hierarchy
  - Interconnect model
  - Memory model
- SCMP
  - 4-thread, 8-thread, 4MB cache size
  - Core CPI: 0.85 ~ 1.5
- LCMP
  - 32-thread, 64-thread, 16MB cache size
  - Core CPI: 2 ~ 4.5



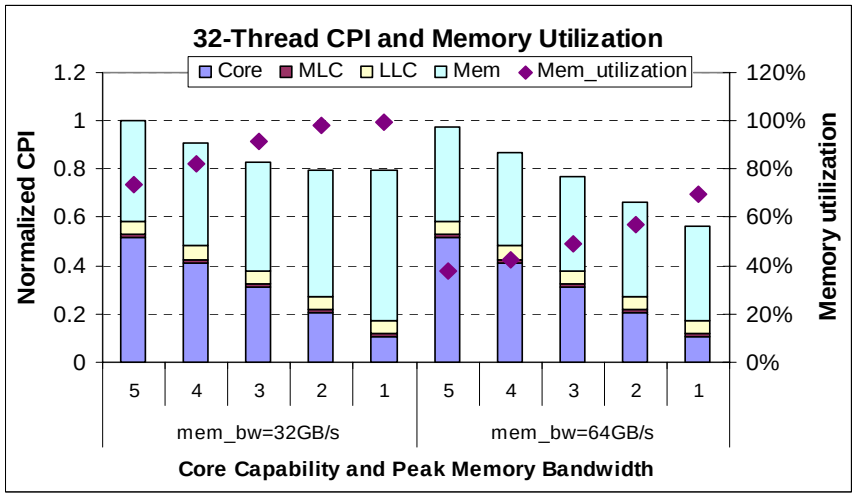
Performance for SCMP

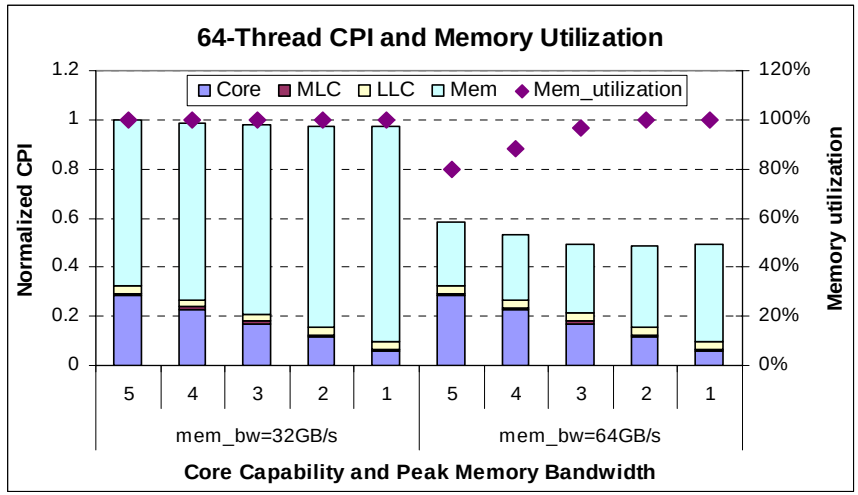




- Memory stall time is the dominant portion of CPI (45~70%)
- Memory utilization is low to high (18%~85%)
- Benefit of 2x memory bandwidth is low (<5%)
- Memory stall for SCMP is largely latency-dependent

Performance for LCMP

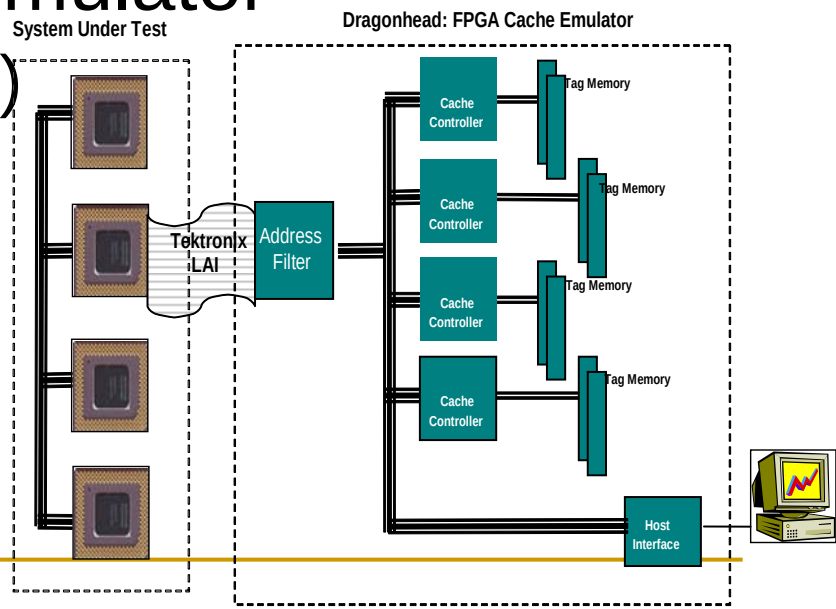




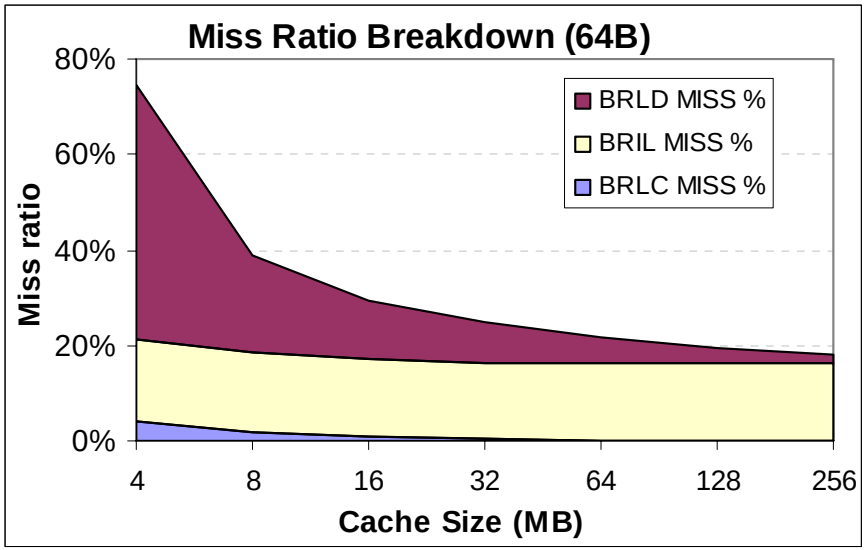
- Memory stall time is the dominant portion of CPI (40~90%)
- Memory utilization is high (40%~100%)
- Benefit of 2x memory bandwidth
  - 32-thread: low to moderate
  - 64-thread: is significant
- Memory stall time is both bandwidth and latency-dependent

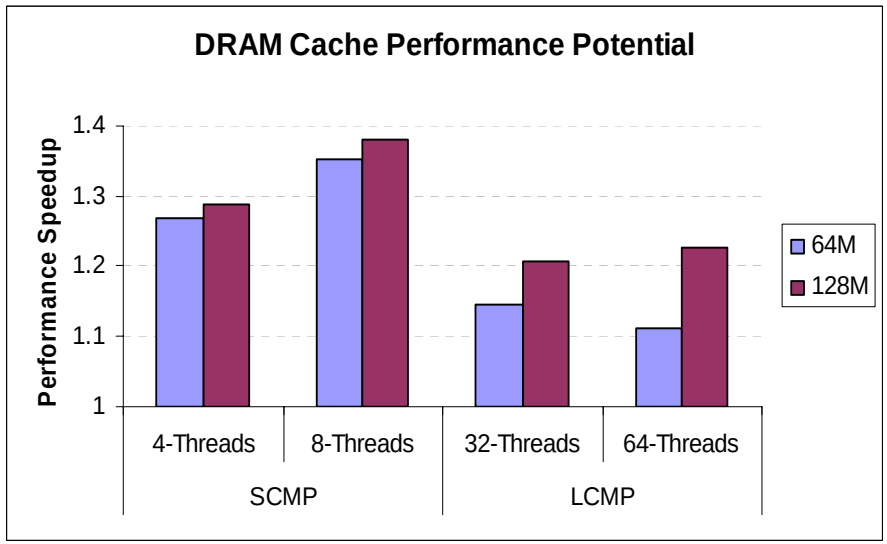
Emulation-based Methodology

- DRAM cache study
- Advantages
  - Speed and accuracy
  - Large workload coverage
- Use FPGA-based cache emulator
  - LAI (logic analyzer interface)
    - Detect memory access
    - Send to Dragonhead
  - Dragonhead
    - Emulate cache behavior



Potential for DRAM cache





- Miss rate reduces as we increase the cache size
  - Code access is the smallest component
  - Data write does not benefit from large cache
  - Data read is improved significantly
- Analytical model on DRAM cache benefits
  - 2x of bandwidth, 1/3 of latency
  - Improve the performance for SCMP significantly (35~45%)

DRAM cache is critical to LCMP by reducing memory utilization

Summary & Future Work

- Presented SCMP and LCMP performance behavior using SPECjbb2005
- SPECjbb2005 performance depends heavily on cache and memory performance
- SCMP is more memory latency sensitive, LCMP is more memory bandwidth sensitive
- DRAM cache can provide 20 to 40% performance improvement
- Future work
  - Study other java workloads
  - In-depth evaluation on DRAM cache organization and policies
  - New cache organizations

Acknowledgements

Michael Liao

Wei A Wei

Qigang Wang

Jaideep Moses

For all the help with the emulator and simulator we used in this study

Thank You