Performance Analysis of Cache Coherence Protocols and Interconnection Networks in Multiprocessor Systems

Erik Pahor, Jaka Škerjanc

I. Introduction

This report presents an analysis of three key aspects of multiprocessor system performance: snooping-based cache coherence protocols, the impact of false sharing on directory-based protocols, and the performance of different interconnection network topologies. Using the GEM5 simulator, we evaluated these aspects across different processor configurations (2, 4, 8, and 16 cores) using various workloads including Cholesky decomposition, PI calculation, and the STREAM benchmark.

II. Snooping-based Cache Coherence Protocol Analysis

For the snooping-based protocol, we evaluated the performance of Cholesky decomposition across different processor counts and analyzed several key metrics.

A. Results and Analysis

Table I
Snooping-based Protocol Performance Metrics

Core Count	CPI	L1D Miss Ratio	Upgrade Requests	Snoop Traffic
2	3.4359	0.0164	2,439	309,376
4	3.2925	0.0204	4,372	854,080
8	3.0917	0.0246	7,777	1,977,856
16	2.9745	0.0212	14,576	4,403,904

CPI (Cycles Per Instruction): We observe a consistent decrease in CPI as the number of processors increases, from 3.44 with 2 cores to 2.97 with 16 cores. This indicates better utilization of computational resources with higher core counts for this workload, suggesting that Cholesky decomposition parallelizes effectively.

L1D Miss Ratio: The L1 data cache miss ratio increases with the number of processors up to 8 cores (from 0.0164 to 0.0246), but then slightly decreases at 16 cores (to 0.0212). This non-linear behavior suggests that the miss ratio is not corelated with core count.

Upgrade Requests: The number of upgrade requests increases almost linearly with the number of processors, from 2,439 with 2 cores to 14,576 with 16 cores. This reflects the growing need for exclusive access to shared cache lines as more cores simultaneously process the workload.

Snoop Traffic: Snoop traffic increases substantially with processor count, growing from 309,376 with 2 cores to 4,403,904 with 16 cores. This represents a 14x increase for an 8x increase in core count, demonstrating the significant overhead of maintaining cache coherence as the system scales.

III. Impact of False Sharing on Directory-based Protocols

We analyzed the performance impact of false sharing by comparing two implementations of a PI calculation algorithm - one prone to false sharing and one optimized to avoid it.

Table II
CPI Comparison Between False Sharing and Optimized Implementations

Core Count	False Sharing CPI	Optimized CPI	Difference (%)
2	2.504	2.295	9%
4	3.001	2.296	30%
8	2.998	2.298	30%
16	3.097	2.301	34%

A. Results and Analysis

CPI Comparison:

False sharing consistently results in higher CPI across all processor counts, with the gap widening as the number of cores increases. At 16 cores, the CPI with false sharing is 34% higher than the optimized version.

Execution Time:

Table III
Execution Time Comparison

Core Count	False Sharing (cycles)	Optimized (cycles)	Difference (%)
2	13,155,904,593	12,073,162,419	9%
4	8,035,984,638	6,205,135,986	30%
8	4,193,290,845	3,281,357,025	28%
16	2,623,817,889	1,834,961,535	43%

The execution time data shows that false sharing creates a significant performance penalty, which becomes more pronounced with higher core counts. At 16 cores, the false sharing version is 43% slower than the optimized version.

Invalidations:

Table IV
Cache Invalidations Comparison

Core Count	False Sharing	Optimized	Ratio
2	500,351	327	1,530x
4	750,395	390	1,924x
8	750,528	548	1,370x
16	672,897	814	827x

The number of invalidations shows the most dramatic difference between the two versions. The false sharing implementation generates orders of magnitude more invalidations than the optimized version, highlighting how false sharing forces unnecessary coherence traffic.

Memory Access Patterns:

- False sharing implementation shows fewer loads from the M state (modified) compared to the optimized version.
- The optimized version shows more efficient cache utilization with fewer transfers between cache coherence states.
- L1_GETS and L1_GETX requests are significantly higher in the false sharing version.

Network Traffic:

- Network request control messages are 3-4 orders of magnitude higher in the false sharing version.
- Response data traffic follows a similar pattern, showing the coherence protocol overhead.
- Writeback data traffic remains relatively constant across implementations, suggesting similar actual data modification rates.

IV. Interconnection Network Performance

We evaluated three different network topologies (SimplePt2Pt, Circle and Crossbar) using the STREAM benchmark across different processor counts.

A. Results and Analysis

Network Traffic Comparison:

Table V
Request Control Traffic

Core Count	SimplePt2Pt	Circle	Crossbar
2	7,948	11,931	11,484
4	253,228	561,728	378,311
8	405,694	1,247,391	587,823
16	764,842	3,919,500	1,139,822

Table VI
Response Data Traffic

Core Count	SimplePt2Pt	Circle	Crossbar
2	4,026,888	6,484,447	6,040,032
4	4,030,964	8,381,114	6,045,455
8	4,041,748	12,190,192	6,060,095
16	4,063,940	19,874,359	6,094,745

Table VII
Writeback Data Traffic

Core Count	SimplePt2Pt	Circle	Crossbar
2	1,001,706	1,440,128	1,502,559
4	1,001,484	2,316,615	1,502,226
8	1,001,596	4,070,546	1,502,259
16	1,004,484	7,593,664	1,507,335

Analysis by Network Topology:

1) **SimplePt2Pt Network:**
   - Shows the lowest overall traffic across all core counts
   - Request control traffic increases linearly with core count
   - Response data traffic remains relatively stable regardless of core count
   - Writeback data traffic is essentially constant across different core counts
2) **Circle Network:**
   - Generates the highest traffic volumes across all metrics
   - Shows super-linear growth in all traffic types as core count increases
   - At 16 cores, request control traffic is 5.1x higher than SimplePt2Pt and 3.4x higher than Crossbar
   - Response data traffic at 16 cores is 4.9x higher than SimplePt2Pt
   - The inefficiency becomes more pronounced with scale
3) **Crossbar Network:**
   - Shows moderate traffic growth with scaling
   - Request control traffic increases at a rate between SimplePt2Pt and Circle
   - Response data and writeback traffic remain relatively stable with increasing core count
   - Provides a middle ground between the simplicity of point-to-point and the high traffic of the ring topology

V. Conclusion

Our analysis reveals several key insights about multiprocessor system design:

1) **Snooping-based Cache Coherence:**
   - Provides good performance scaling for parallel workloads like Cholesky decomposition
   - However, snoop traffic increases dramatically with core count
   - The protocol can maintain reasonable miss ratios even at higher core counts
2) **False Sharing Impact:**
   - False sharing severely degrades performance in directory-based protocols
   - The performance penalty increases with core count due to exponentially more coherence traffic
   - Optimizing code to avoid false sharing can yield performance improvements of 30-40%
   - The enormous difference in invalidation counts (up to 1900x) highlights the fundamental mechanism of the performance degradation
3) **Interconnection Networks:**
   - Network topology plays a crucial role in system scalability
   - SimplePt2Pt networks show the best traffic characteristics but may have practical implementation limitations
   - Ring topologies suffer from severe congestion at higher core counts
   - Crossbar networks provide a reasonable compromise, showing moderate traffic growth with scaling

These findings emphasize the importance of considering both software optimizations (avoiding false sharing) and hardware design choices (coherence protocols and network topologies) when building scalable multiprocessor systems. The optimal configuration depends on the specific workload characteristics, with different design choices showing distinct advantages for different application patterns.