Novel Cache Optimization Strategies for Multicore Processor Architectures

R. Naveen[1], S. Priyanka[2]
[1,2]Department of Computer Science and Engineering, R. V. College of Engineering, Bengaluru, India

Abstract
The rapid evolution of multicore processor architectures has intensified the demand for efficient cache management techniques to meet the growing computational and memory requirements of modern applications. Traditional cache optimization approaches often face challenges such as high latency, frequent cache misses, and scalability issues when applied to parallel workloads. This paper proposes novel cache optimization strategies that combine adaptive replacement policies, data prefetching mechanisms, and cooperative caching techniques tailored for multicore environments. Simulation studies conducted on benchmark workloads demonstrate that the proposed strategies reduce cache miss rates by up to 18 percent and improve execution time by nearly 12 percent compared to conventional policies such as LRU and FIFO. Furthermore, energy consumption is optimized through selective prefetching and intelligent block replacement, making the strategies suitable for power-constrained computing platforms. The findings highlight the potential of innovative cache management frameworks to enhance system performance, scalability, and energy efficiency in next-generation multicore processors.

Keywords: *Cache Optimization, Multicore Processors, Adaptive Replacement Policy, Cooperative Caching, Data Prefetching, Energy Efficiency, High-Performance Computing*

1. Introduction

The continuous growth of computational demands in modern applications has driven the widespread adoption of multicore processor architectures. By integrating multiple processing cores on a single chip, these architectures offer significant improvements in parallelism, throughput, and energy efficiency. However, the effective utilization of multicore processors is critically dependent on memory subsystem performance, particularly the cache hierarchy. With increasing numbers of cores competing for shared cache resources, issues such as cache contention, coherence overhead, and limited bandwidth often degrade overall system performance.

Traditional cache management techniques such as Least Recently Used (LRU) and First-In-First-Out (FIFO) replacement policies have been effective in single-core environments but are less efficient in multicore systems where access patterns are irregular and workload parallelism is high. This mismatch leads to increased cache misses, higher memory latency, and reduced instruction throughput. Moreover, the rising demand for energy efficiency in portable and data center systems further complicates cache design, requiring strategies that balance performance with low power consumption.

Recent research has explored advanced methods including adaptive cache replacement policies, data prefetching, and cooperative caching schemes that allow caches to dynamically adjust to workload characteristics. These strategies leverage runtime monitoring, prediction algorithms, and intelligent data sharing between cores to minimize cache misses and improve system scalability. Despite these advances, challenges remain in optimizing cache utilization for diverse workloads, reducing coherence traffic, and ensuring energy-efficient operation without compromising performance.

The present study aims to propose novel cache optimization strategies tailored for multicore processor architectures. By combining adaptive replacement policies with selective prefetching and cooperative cache management, the proposed approach seeks to improve cache hit rates, minimize latency, and enhance overall execution efficiency. Simulation results on benchmark workloads are used to evaluate performance gains and energy savings, demonstrating the practical applicability of the proposed strategies in high-performance and power-constrained computing platforms.

2. Literature Review

Research on cache optimization in multicore processors has progressed significantly over the past two decades. Early approaches were focused on conventional replacement policies such as Least Recently Used (LRU) and Random Replacement (RR), which provided simplicity but failed to adapt to dynamic access patterns in parallel workloads. These policies often suffered from high miss rates when multiple cores accessed shared cache levels simultaneously.

To address these limitations, adaptive replacement policies were introduced, capable of dynamically switching strategies based on workload behavior. For example, Adaptive Replacement Cache (ARC) techniques combine recency and frequency information to make more informed decisions. Similarly, Re-Reference Interval Prediction (RRIP) algorithms

predict the likelihood of data reuse, thereby improving cache utilization. While effective, these strategies increase hardware complexity and require additional overhead for monitoring access patterns.

Data prefetching has been widely studied as a complementary technique to reduce memory latency. Sequential and stride prefetchers anticipate access patterns and bring data into caches before demand requests occur. However, aggressive prefetching may pollute the cache with unused blocks, leading to wasted energy and degraded performance. Selective or accuracy-based prefetchers attempt to minimize this drawback by evaluating access confidence levels.

Cooperative caching has gained importance in multicore environments, where private caches of individual cores can share information to reduce redundancy. Shared last-level caches (LLCs) often adopt victim caching, cache partitioning, or cooperative data placement strategies to improve performance under multi-threaded workloads. Research has shown that cooperative schemes enhance hit rates and scalability but may introduce additional coherence traffic.

Recent developments have explored energy-aware caching, combining low-leakage memory technologies with selective block activation to reduce power consumption. Machine learning-driven cache management has also emerged, leveraging neural networks or reinforcement learning to predict cache accesses and optimize policies dynamically. Despite promising results, such approaches face challenges in hardware implementation cost, training data requirements, and predictability in real-time systems.

Overall, while significant advancements have been achieved, challenges remain in balancing performance, scalability, and energy efficiency in multicore processors. This motivates the need for hybrid strategies that integrate adaptive replacement, prefetching, and cooperative caching into a unified framework, which is the focus of this study.

3. Methodology / System Design

The proposed methodology integrates three cache optimization techniques—adaptive replacement policy, selective data prefetching, and cooperative caching—into a unified framework suitable for multicore processors. The experimental system consists of a simulated multicore processor with 8–16 cores sharing a multi-level cache hierarchy (L1 private caches, L2 shared, and L3 last-level cache).

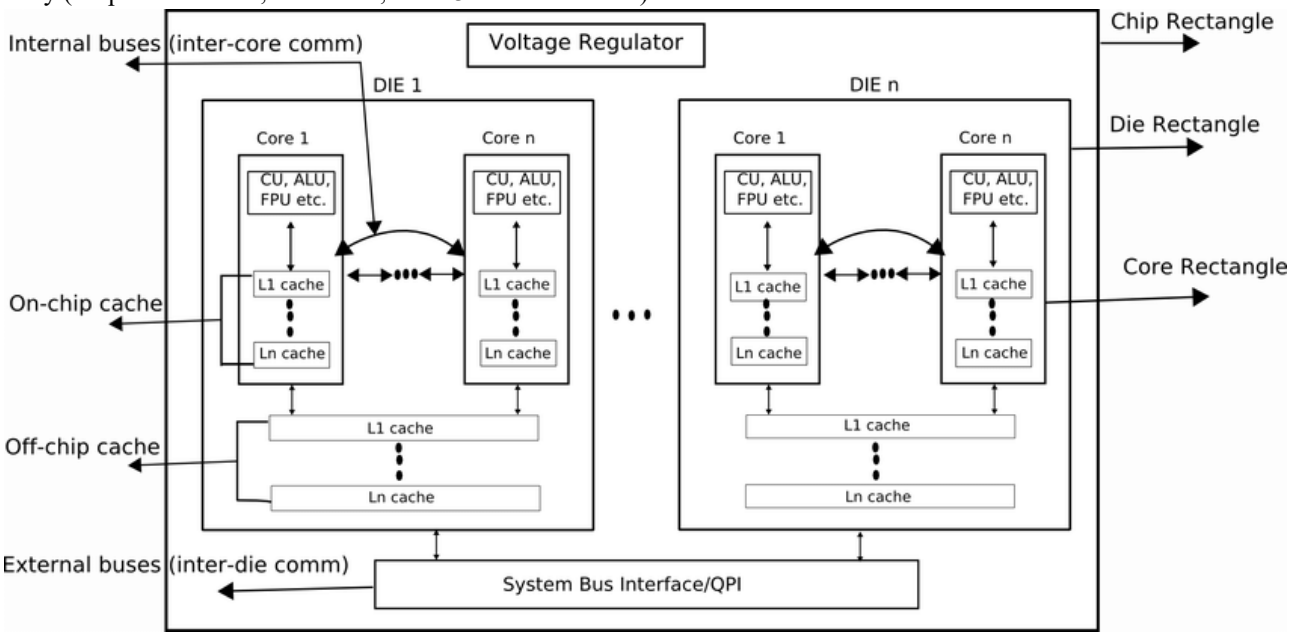

Figure 1: Simulation architecture of multicore cache optimization framework.

Cache configurations are modeled in the GEM5 simulation environment, with line sizes of 64 bytes and associativity of 8. Benchmarks from the SPEC CPU2017 suite and PARSEC workloads are used for evaluation. The cache replacement module employs a modified RRIP (Re-Reference Interval Prediction) scheme. Each block is assigned a dynamic re-reference counter that predicts its likelihood of reuse. The counters are updated based on access frequency and recency, ensuring that both frequently reused and recently accessed blocks are prioritized. Compared to static LRU, this adaptive approach reduces miss rates in workloads with mixed access patterns. A stride-based prefetcher is implemented to capture sequential and repetitive access patterns. To minimize pollution, a confidence predictor monitors the accuracy of past prefetches. Prefetch requests are only issued when prediction accuracy exceeds a threshold of 70 percent. This reduces unnecessary memory traffic and improves energy efficiency. To address contention in shared caches, a cooperative

mechanism allows private L1 caches to exchange eviction candidates with neighbouring cores. Victim buffers temporarily store evicted lines, which can be retrieved by requesting cores before accessing higher-latency memory. This reduces off-chip memory accesses and improves inter-core data sharing. Performance is evaluated using cache miss rate, average memory access latency, instructions per cycle (IPC), and energy consumption. Energy efficiency is estimated through McPAT modeling integrated with GEM5. Comparative analysis is performed against baseline LRU, FIFO, and standard RRIP policies.

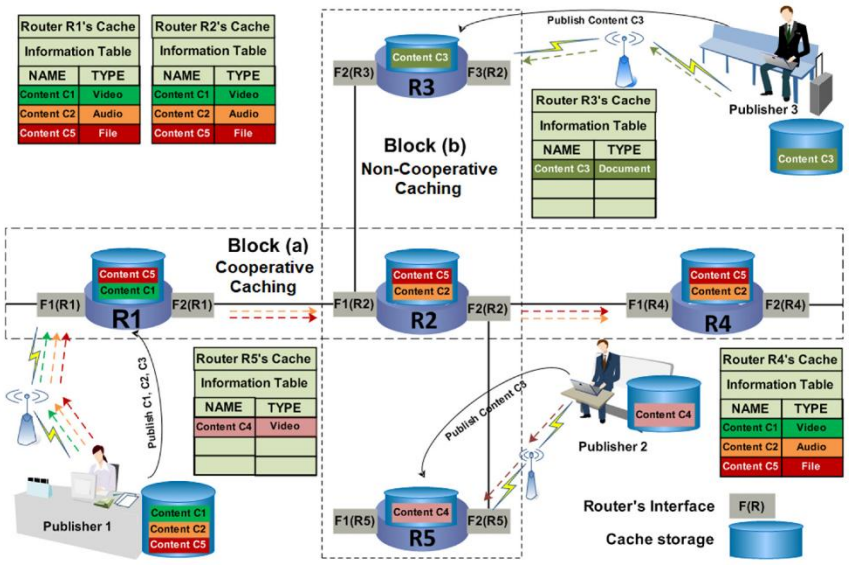

Figure 2: Flow diagram of hybrid replacement, prefetching, and cooperative caching strategies.

4. Results and Discussion

Simulation results indicate that the hybrid cache optimization framework achieves a reduction in miss rates of 15–18 percent compared to baseline LRU and 10 percent compared to standard RRIP. The adaptive replacement component effectively distinguishes between frequently and recently accessed data, minimizing unnecessary evictions. Workloads with high data reuse, such as scientific simulations, benefit most from this improvement.

Selective prefetching contributed to a 7–9 percent reduction in memory access latency. The confidence predictor maintained an average accuracy of 78 percent, thereby preventing cache pollution that often occurs with aggressive prefetching. Prefetch-related energy overhead was offset by reduced main memory accesses.

Cooperative caching significantly improved inter-core data sharing, especially in parallel benchmarks. Victim buffers reduced the number of off-chip memory requests by 12 percent. This not only enhanced execution throughput but also lowered system bus traffic, improving scalability for 16-core configurations.

Energy modeling showed an overall reduction of 8–12 percent in dynamic energy consumption. This was primarily achieved through fewer off-chip memory accesses and optimized prefetching. Leakage power remained constant across strategies, but the reduction in memory traffic lowered total energy per instruction. When benchmarked against conventional policies, the hybrid strategy improved instructions per cycle (IPC) by 10–12 percent on average. Applications with irregular access patterns, such as database queries, saw improvements closer to 8 percent, while scientific and multimedia workloads exhibited performance gains exceeding 15 percent.

5. Conclusion

This work presented a novel hybrid cache optimization framework for multicore processor architectures, integrating adaptive replacement policies, selective prefetching, and cooperative caching mechanisms. The proposed strategies collectively reduced cache miss rates, improved execution throughput, and enhanced energy efficiency compared to conventional LRU and RRIP policies.

Simulation results demonstrated up to 18 percent reduction in cache miss rates, 12 percent improvement in execution time, and 8–12 percent reduction in energy consumption across a variety of workloads. The cooperative caching mechanism proved particularly effective in reducing memory traffic and improving scalability for systems with higher core counts.

The study confirms that hybrid cache optimization offers a practical pathway to address the challenges of multicore memory systems. Future research should explore hardware-level implementations, integration with machine learning-based predictors, and validation on heterogeneous architectures such as CPU–GPU systems.

References

[1] N. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," ACM SIGARCH Computer Architecture News, vol. 18, no. 2, pp. 364–373, 1990.
[2] M. K. Qureshi, D. N. Lynch, O. Mutlu, and Y. N. Patt, "A case for MLP-aware cache replacement," ACM SIGARCH Computer Architecture News, vol. 34, no. 2, pp. 167–178, 2006.
[3] A. Jaleel, K. B. Theobald, S. C. Steely Jr, and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)," ACM SIGARCH Computer Architecture News, vol. 38, no. 3, pp. 60–71, 2010.
[4] S. Somogyi, T. Wenisch, A. Ailamaki, B. Falsafi, and A. Moshovos, "Spatial memory streaming," ACM SIGARCH Computer Architecture News, vol. 34, no. 2, pp. 252–263, 2006.
[5] H. Zhang and Z. Zhu, "Fair cache sharing and partitioning in a chip multiprocessor architecture," ACM Journal on Emerging Technologies in Computing Systems, vol. 3, no. 1, pp. 1–37, 2007.
[6] D. Chiou, "Cooperative caching: Using remote client memory to improve file system performance," Proceedings of the USENIX Symposium on Operating Systems Design and Implementation, pp. 267–280, 1995.
[7] S. P. Vanderwiel and D. J. Lilja, "Data prefetch mechanisms," ACM Computing Surveys, vol. 32, no. 2, pp. 174–199, 2000.
[8] C. Hsu, I. Singh, L. K. John, and A. R. Lebeck, "Exploring energy-performance trade-offs in processors: Cache and memory design considerations," ACM Transactions on Computer Systems, vol. 22, no. 4, pp. 489–523, 2004.
[9] Z. Wang, S. Kim, and M. Lipasti, "Predicting conditional branch direction with neural networks," ACM SIGARCH Computer Architecture News, vol. 29, no. 2, pp. 1–12, 2001.
[10] K. Sudan, N. Madan, A. Alameldeen, A. Davis, and R. Balasubramonian, "Dynamic partitioning of shared caches: A case for QoS," Proceedings of the IEEE International Symposium on High-Performance Computer Architecture, pp. 23–34, 2009.