*International Journal on Recent and Innovation Trends in Computing and Communication*
*ISSN: 2321-8169 Volume: 11 Issue: 2*
***Article Received:*** *21 December 2022* ***Revised:*** *18 January 2023* ***Accepted:*** *24 January 2023*

# Stream Applications: A Consistent Active Management Approach for Multi-Core Cache

R.Kennady[1], O.Pandithurai[2]

[1]Department of Artificial Intelligence and Data Science, Rajalakshmi Institute of Technology, Chennai, Tamilnadu

[2]Department of Computer Science and Engineering, Rajalakshmi Institute of Technology, Chennai, Tamilnadu

[1]kennady.r@ritchennai.edu.in , [2]pandics@ritchennai.edu.in

**Abstract:**

This paper proposes a novel method for managing cache consistency in multi-core systems when executing stream applications. The method involves arranging a mark cache for private data caches, which includes an optional integrality descriptor for shared reading and writing data states and shared data manipulation positions. The integrality descriptor identifies the current mode of operation for shared data in the private data cache. Additionally, the method utilizes a two-dimensional array register, referred to as the shared data manipulation position, with width N and depth M, where N distinguishes between different cache blocks and locking territories, while M corresponds to the number of cache blocks. This enables the identification of the cache capable or block corresponding to shared data during read and write operations. The proposed method offers simplicity, ease of operation, low hardware implementation cost, good extensibility, and strong configurability, ultimately improving system effectiveness.

**Keywords:** Cache consistency, Multi-core systems, Stream applications, Mark cache, Integrality descriptor, Shared data manipulation, System effectiveness.

**Introduction**:

With the proliferation of multi-core systems, efficient management of cache consistency has become crucial, particularly when executing stream applications. Stream applications often involve shared data access, which can lead to inconsistencies and synchronization issues across multiple cores. This paper presents a new method for active cache management, specifically designed to address these challenges. The method leverages a mark cache for private data caches and employs an integrality descriptor and shared data manipulation positions to facilitate efficient cache consistency management. The following sections will delve into the background, research objective, and detailed methodology of the proposed approach.

**Background**:

Along with the development of computer applications, a typical class of data-intensive applications called stream applications has become a crucial workload for multicore processors. Stream applications can be categorized into two main types: media applications, such as radio communication and image processing, which require real-time digital signal processing for audio, video, and encoding/decoding; and scientific algorithms, primarily used in high-precision science modeling, including fields like fluid mechanics, molecular dynamics, finite element analysis, and biotechnology. These stream applications exhibit high data parallelism, computational intensity, and data locality characteristics.[1,2]

Compared to traditional desktop applications, stream applications perform intensive arithmetic operations on each piece of data retrieved from internal memory. Most computations in stream applications can be parallelized at the data, thread, and task levels. However, the data access locality in stream applications is typically limited to adjacent or spanning access of long data blocks, resulting in low data reuse for producers. In multicore processors, multiple stages of intensive data computing in stream applications are decomposed and executed on different processor cores to fully utilize the resources of chip multi-core systems and achieve higher application performance.

Current commercial multi-core microprocessors mostly integrate multiple identical and powerful processor cores, such as the IBM Power7 with 8 processor cores, the Sun UltraSparcT2 chip with 8 monokernels, or the Intel ManyIntegratedCore (MIC) KNF coprocessor embedded with 32 monokernels. These multicore processors feature a basic structure (as shown in Figure 1) where the chip

*International Journal on Recent and Innovation Trends in Computing and Communication*
*ISSN: 2321-8169 Volume: 11 Issue: 2*
***Article Received:** 21 December 2022 **Revised:** 18 January 2023 **Accepted:** 24 January 2023*

integrates N single-core processors, each equipped with private data caches at different levels, including privately owned level one data caches or privately owned level one and two data caches (referred to as shared caches). The shared caches are shared among the cores through internuclear interconnections or three-level caches, aiming to improve temporal locality and reduce average memory access latency.[3,4,5]

To ensure cache coherence in multicore processors, hardware typically adopts two implementation methods: bus monitoring-based protocols or directory-based protocols. Monitoring protocols distribute the responsibility of maintaining cache coherence among the private data caches of each core. When a write operation is completed on the private data cache of one core, it must broadcast a notification to all private data caches of other cores. Each core's private data cache should be able to monitor broadcasts from other cores and respond accordingly. However, monitoring protocols have some drawbacks: the increasing cost of internuclear broadcasting with the growth of processor cores, the excessive snooping transactions on each core's private data cache, and the cancellation of cache benefits due to excessive broadcast requests. On the other hand, directory protocols maintain a shadow directory structure on the shared cache, recording the status information and update modes of each core's private data cache.[5,6] When a private data cache of one core generates an access request, the shadow directory checks the request and sends the necessary operations to the relevant cores' private data caches, invalidating the corresponding data transcriptions. While directory protocols reduce the blind broadcast requests, they suffer from increased hardware complexity as the processor check figure and cache memory capacity increase, leading to inefficiencies in algorithm implementation and scalability challenges.[9,10]

In conventional cache coherence schemes with software administration, hardware requires programmable setting of lock synchronization registers by the program. When a core needs to read a shared data area, the programmer first reads the lock on this region, and memory access can proceed upon successful lock acquisition. Hardware then rejects write lock requests from the same core to maintain data consistency. Similarly, when a core intends to write to a shared data area, the programmer adds a write lock after checking if the region is already locked for reading or writing. If locked, the core waits for the lock to be released and successfully acquires it before starting the write operation, rejecting locking requests from other cores until completion. When releasing a write lock, all other cores must be notified through a broadcast, and respective private data caches must invalidate the copies of the data region, ensuring data consistency.[11] However, the extensive broadcast requests for consistency maintenance in conventional architectures present several problems: increased intercore communication bandwidth usage, resource waste due to unnecessary broadcast requests, additional time delays introduced by broadcast cancellation, and prolonged processing time for handling cancellation requests in private data caches. These challenges require special address computation structures, larger buffer structures, and private communication control modules, resulting in complex cache controllers and increased area and power consumption overheads.

Cache consistency is a well-known challenge in multi-core systems, especially in the context of stream applications. These applications typically exhibit data dependencies and require synchronization among cores to ensure correct execution. Existing cache coherence protocols have limitations when it comes to stream applications due to the dynamic nature of data access patterns. Therefore, novel approaches are needed to enhance cache consistency management and improve system effectiveness.

**Research Objective:**

The objective of this research is to develop a multi-core cache consistency active management method tailored for stream applications. The method aims to provide a simple, operationally efficient, and cost-effective solution for maintaining cache coherence in a multi-core environment. By utilizing a mark cache with integrality descriptors and shared data manipulation positions, the proposed method intends to enhance the overall system effectiveness, improve performance, and address the challenges associated with shared data access and synchronization. The proposed method involves arranging a mark cache for private data caches, which includes an optional integrality descriptor for shared reading and writing data states and shared data manipulation positions. The integrality descriptor enables the identification of the current mode of operation for shared data in the private data cache. Furthermore, a two-dimensional array register, referred to as the shared data manipulation position, is utilized, with width N and depth M. N is employed to distinguish different cache blocks and locking territories, while M corresponds to the number of cache blocks. This facilitates the identification of the cache capable or block corresponding to shared data during read and write operations. The method ensures simplicity, ease of operation, low hardware implementation cost, extensibility, and configurability, thereby contributing to improved system effectiveness.

*International Journal on Recent and Innovation Trends in Computing and Communication*
*ISSN: 2321-8169 Volume: 11 Issue: 2*
***Article Received:** 21 December 2022 **Revised:** 18 January 2023 **Accepted:** 24 January 2023*

**Research:**

The research focuses on addressing the challenges associated with cache coherence in multicore processors. The proposed solution is based on a programmable lock synchronization control register and introduces two special unlock commands: the read lock solution and the write lock solution instructions. When accessing the shared data address space, there are two types of memory access behaviors: read-only access and write access. Read lock is required for read-only access, and write lock is necessary for write access. Non-read lock can be added when writing lock is present, and only a single write lock can exist.

The read lock or write lock instructions are atomic operations performed on the lock synchronization control register, completed by normal access instructions. Once the lock operation is completed, data read or write access can commence. To terminate the shared data access, the solution read lock or solution write lock instructions are used. Simultaneously, the private data caches inspect the unlock commands to determine the validity of the shared data manipulation position. If it is valid, it is marked as invalid, and the corresponding cache performs the necessary operations, such as writing back dirty rows or clearing the significance bit.

The advantages of the research are as follows:

- The proposed active management method reduces the time required for maintaining cache coherence and allows for accurate calculations. By adopting the immediate cancellation strategy under the read-only mark release, the time for canceling read-only operations can be significantly shortened. (Table-1 Block Information)

| Physical Address | Process ID | Significant bit |
|---|---|---|
| Block Address | n | 0/1 |
| Block Address | n-1 | 1/2 |

Table-1 Block Information

- The research eliminates the need for broadcast strategies, reducing the pressure on internuclear communication. By not broadcasting the cache coherence requests within the internuclear network, the increasingly complex internuclear communication and placement-and-routing pressure are alleviated.
- Irrelevant cores no longer need to receive cache coherence requests, eliminating delays and interruptions caused by consistency request transactions from other cores, thereby improving system efficiency.
- The research reduces the complexity of programming. The controllable and predictable time delays help reduce uncertainty during programming, thus improving system availability.
- The hardware implementation of the research is cost-effective. By eliminating the need for special internuclear broadcast channels, cache inner region judgments, address comparisons, and cancellation request calculations, the hardware requirements are significantly reduced.
- The research offers good extensibility and configurability. As the number of cores increases, the method exhibits good scalability without the need for additional hardware. The configuration options include buffering of coherence requests, arrangement of coherence registers and figures, content of shared data manipulation bit registers and integrity descriptor registers, and the organizational form of cache-based coherence operations.

The present research aims to address the issues caused by the Cache coherence protocol used in current multi-core processors, which result in high hardware costs and delays in data consistency transactions, affecting the performance of applications. To overcome these problems, a new active management method for Cache coherence is proposed, specifically designed to meet the data locality characteristics of streaming applications and better adapt to the Cache consistency needs of producers and consumers.

In this method, each individual core's private data Cache in the multi-core processor manages its own shared data independently, without the need for broadcasting requests. This significantly reduces the hardware and communication overhead required to maintain Cache coherence, resulting in decreased delays in data consistency transactions. Additionally, this approach accelerates streaming computations and is well-suited for the development of multi-core processors.

The method involves organizing the storage structure of the current multi-core processor, specifically improving the configuration of the private data Cache for each core. The schematic diagram of a concrete Cache structure after applying this method is shown in Figure 4. The method utilizes an optional integrity descriptor that assigns a mark Cache to the shared data, indicating the state of reading and

writing operations on the data. The integrity descriptor helps identify the overall operation mode of shared data in the private data Cache. The method also introduces a shared data manipulation position, which is a two-dimensional array register with a width of N and a depth of M. N is used to differentiate between different blocks or regions that can be locked in this Cache, and M represents the number of blocks or regions in the Cache. The shared data manipulation position helps identify which Cache is responsible for reading and writing operations on the shared data.

In simpler terms, the research improves how the individual parts of a multi-core processor manage their own data. By making each core responsible for its own shared data and avoiding unnecessary communication, it reduces costs and delays in keeping the data consistent. This approach is particularly useful for streaming applications and can speed up computations. The method can be easily applied to different multi-core processors and supports their future development.

Overall, this research provides a solution to enhance cache coherence in multicore processors by introducing a programmable lock synchronization control register and optimizing coherence operations, leading to improved system performance, reduced communication overhead, and simplified programming complexity.

**Conclusion**:

In this paper, we have presented a multi-core cache consistency active management method specifically designed for stream applications. By employing a mark cache with integrality descriptors and shared data manipulation positions, the proposed method enhances cache coherence and synchronization among cores, thereby improving overall system effectiveness. The method offers simplicity, operational efficiency, low hardware implementation cost, and configurability, making it suitable for various multi-core systems. Future research can explore further optimizations and evaluate the performance benefits of the proposed approach in real-world scenarios.

**References**:

1. Ashraf, M., Huang, C., Raza, K. A., Huang, S., Yin, Y., & Wu, D. F. (2022). Dynamic cooperative cache management scheme based on social and popular data in vehicular named data network. *Wireless Communications and Mobile Computing*, *2022*.
2. Similarity caching: Theory and algorithms, M Garetto, E Leonardi, G Neglia 2020 - ieeexplore.ieee.org
3. Joint cache placement, flight trajectory, and transmission power optimization for multi-UAV assisted wireless networks, J Ji, K Zhu, D Niyato, R Wang 2020 - ieeexplore.ieee.org
4. An open privacy-preserving and scalable protocol for a network-neutrality compliant caching, D Andreoletti, C Rottondi, S Giordano 2019 - ieeexplore.ieee.org
5. A decomposition framework for optimal edge-cache leasing, J Krolikowski, A Giovanidis 2018 - ieeexplore.ieee.org
6. Caching policy toward maximal success probability and area spectral efficiency of cache-enabled HetNets, D Liu, C Yang - IEEE Transactions on Communications, 2017 - ieeexplore.ieee.org
7. Scalable cache management for ISP-operated content delivery services, D Tuncer, V Sourlas, M Charalambides, 2016 - ieeexplore.ieee.org
8. Icarus: a caching simulator for information centric networking (icn), L Saino, I Psaras, G Pavlou - SimuTools, 2014 - discovery.ucl.ac.uk
9. Caching at the edge: A green perspective for 5G networks, B Perabathini, E Baştuğ, M Kountouri 2015 - ieeexplore.ieee.org
10. A methodology for the design of self-optimizing, decentralized content-caching strategies, K Kvaternik, J Llorca, D Kilper… - IEEE/ACM Transactions …, 2015 - ieeexplore.ieee.org
11. Popularity-driven coordinated caching in named data networking, J Li, H Wu, B Liu, J Lu, Y Wang, X Wang 2012 - dl.acm.org