Exploit Temporal Locality of Shared Data in SRC Enabled CMP

Haixia Wang, Dongsheng Wang, Peng Li, Jinglei Wang, and XianPing Fu

Research Institute of Information Technology,
National Laboratory for Information Science and Technology,
Tsinghua University, Beijing, 100084, P.R.China,
{hx-wang,wds}@tsinghua.edu.cn,
{p-li02,wjinglei00}@mails.tsinghua.edu.cn,fxp@dl.cn

Abstract. By run-time characteristic analysis of parallel workloads, we found that a majority of shared data accesses of parallel workload has temporal locality. Based on this characteristic, we present a sharing relation cache (SRC for short) based CMP architecture, saving recently used sharing relations to provide destination set information for following cache-to-cache miss requests. Token-SRC protocol integrates SRC into token protocol,reducing network traffic of token protocol.Simulations using SPLASH-2 benchmarks show that, a 16-core CMP system with token-SRC achieved average 15% network traffic reduction of that with token protocol.

1 Introduction

Multiple-processor systems, i.e. symmetric multiple processor systems and cluster systems are widely adopted in modern commercial and scientific computing infrastructures. Chip multiprocessor (CMP) [1–3], which integrates multiple processor cores into a single chip, is a promising technique that can efficiently exploit the inherent thread-level parallelism inside modern workloads. CMP systems share many critical design issues with traditional share-memory multiprocessor systems, especially the cache-coherence protocols.

For shared-memory multiprocessor systems, shared-bus offered a convenient solution to maintain cache-coherence with snooping mechanisms [4, 5]. Although broadcast-based protocol is simple and easy to implement, shared-bus architecture serializes all messages in the system, limiting the system scalability.

Directory-based protocols [6, 7] were proposed to solve the coherence problem in multi-processor system with unordered interconnection. They introduce a global directory keeping records of the locations and status of the cached copies. With determined destination set, network traffic of directory-based protocol will be much less than snooping protocol,which makes it applicable for large-scale shared-memory multiprocessor systems. Unfortunately, directory-based protocols suffer from long latency for cache-to-cache transfer misses.

To avoid indirections for cache-to-cache misses of directory protocol and interconnect ordering of snooping protocol, token protocol [8, 9] directly send broadcast on un-ordered interconnect, avoiding indirection for cache-to-cache miss in directory-based protocol. Unfortunately, token protocol broadcasts request to maximal destination set, which will incur heavy network traffic.

We proposed an efficient technique to reduce network traffic of broadcast-based protocol, which was called sharing relation cache (SRC) [10]. The idea came from the following characteristic that we found in the run-time characteristic analysis of parallel workloads: a majority of shared data accesses has temporal locality. SRC integrates a sharing relation cache in each core to cache directory information recently used sharing relations (that is of shared data), providing destination set information for following cache-to-cache miss requests. Different from directory, SRC keep only directory information of recently accessed shared data, not all directory information. Before issuing data requests, SRC is lookup at first. If SRC hits, requests are sent to destination set pointed by the SRC entry. Otherwise, requests are broadcast to all processor nodes in the system.

In this paper, we integrated SRC technique in token protocol to reduce network traffic. We called this protocol token-SRC protocol. Preliminary evaluation showed that running SPLASH-2 parallel benchmarks on 16-core CMP, token-SRC protocol achieved an average 15% network traffic reduction of classical token-protocol.

A related work on reducing network traffic of cache coherence protocol is destination-set prediction technique [11]. It was invented to predict destination set for directory protocol. Destination-set prediction technique provides three kinds of destination set choices for directory protocol: Owner node, in which case read request is sent to owner node directly without looking up directory; Maximal set, in which case request is broadcast to all processor nodes without looking up directory; and minimal set, in which case directory are looked up and requests are sent to minimal destination set defined by directory entry.

Destination set prediction in paper [11] yields minimal set by directory lookup. In token protocol, there is no directory so that destination set prediction can not find destination set for write requests. But in contrary, SRC technique may generate a destination set that is close to minimal set, which reduces network traffic of token protocol by avoiding broadcast.

The remainder of the paper is organized as followings. Section 2 defines the quantitative analysis method on temporal locality of shared data, describes multiprocessor simulator and parallel workloads used in this study, and analyzes experiment results on temporal locality characteristic of shared data. Section 3 addresses implementation of SRC in token protocol based system. Section 4 exhibits and analyzes the preliminary experiment results in 16-core CMP systems with directory, token and token-SRC protocols. Finally, section 5 summarizes the paper and discusses direction for future work.

2 Characteristic Analysis on Temporal Locality of Shared Data

2.1 Quantitative Analysis Method

When several processors accessed a shared data, a sharing relation was built on the shared data among those processors. The sharing relation is represented as a map ¡address, sharers¿. The address refers to the shared data, and the sharer field points out which processors own valid copies of the shared data, that is just like a directory entry.

The time interval between twice continuous accesses of a shared data is defined as a sharing relation lifespan. During each sharing relation lifespan, the number of accesses on other shared data is called the distance of the sharing relation lifespan. If the distance of a sharing relation lifespan is small, the sharing relation is reused quickly. By counting all distances of any sharing relation lifespan, we could analyze the temporal locality characteristic of shared data.

2.2 Experiment Environment and Results

We simulate a 16-core CMP system with an open source multiprocessor simulator GEMS [12], which is developed by Wisconsin Multifacet project and built on the Virtutech Simics [13] full-system functional execution-driven simulator. GEMS simulator adds timing information of memory hierarchy and interconnection on simics. GEMS extends Simics with detailed processor, memory hierarchy and interconnection network models to compute execution times, enabling detailed simulation of multiprocessor systems, including CMPs. We choose a simple shared-bus MOSI snooping protocol to build the 16-core CMP system. To ensure that hardware architecture details do not affect program behavior analysis, perfect cache with infinite capacity is used in the simulated multiprocessor.

We use SPLASH-2 [14] parallel benchmark package as workload. SPLASH-2 provides a suite of shared-memory benchmarks for parallel systems and includes a set of kernel and application components. We select a representative subset of the SPLASH-2 as workloads,four kernel workloads and four application workloads.

For each SPLASH-2 workload and each given distance size, Fig. 1 illustrates the distribution of accesses whose distances are less than the given size. Between the 10 SPLASH-2 workloads, cholesky and barnes shows slightly lower temporal locality, only about 50% distances are less than 1000. Except for cholesky and barnes, the other 8 workloads have comparatively higher temporal locality of sharing relations, especially radiosity, about 90% distances are less than 100. The high temporal locality of the 8 workloads provides high hit rate of SRC with limited storage space.

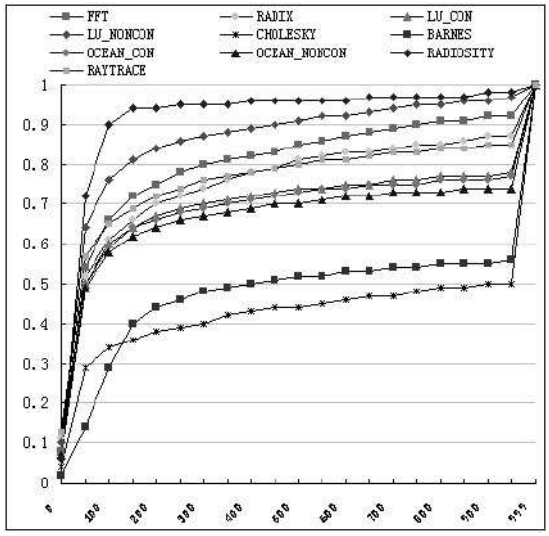

Fig. 1. Statistic of Distance Distribution

3 Token-SRC Protocol

3.1 CMP Architecture

The sketch of CMP architecture with Token-SRC protocol is shown in Fig. 2. Each processor core owns a private SRC, which records directory information of recently accessed shared data.

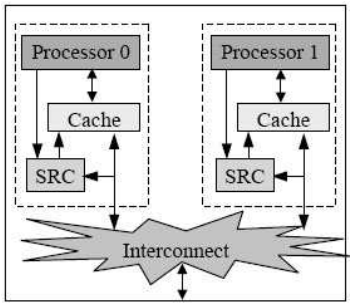

Fig. 2. Sketch of CMP architecture with Token-SRC protocol

On reading or writing, processor looks up its private caches firstly. If local cache misses, processor will send request to other processors or memory. Before issuing requests, token-SRC protocol looks up SRC for request destination set. If SRC hits, the processor will send request to destination set denoted by the SRC entry. Otherwise, the request will be broadcast to all processors.

An important optimization is to remove SRC lookup operation from critical path of memory access. SRC lookup process can be designed in parallel with normal data cache lookup process. When normal data cache completes lookup process, SRC lookup result should also be available. SRC is organized just like a normal data cache. Each entry of SRC has 3 fields: valid, tag and sharer. Sharer filed of SRC entry records identities of the processors which have valid copies of the shared data. The address lookup process of SRC is also as same as that of data cache.

3.2 Correctness Substrate

In broadcast-based protocol (such as snooping protocol, token protocol, and so on), destination sets of cache miss requests include all processor nodes, that is the maximal destination set. In directory-based protocol, destination sets of cache miss requests are minimal destination set, including one owner processor for reading request and all processors with valid cached copies for writing request.

Token protocol extended broadcast protocols from ordered network to unordered network while keeping safety property by enforcing the coherence invariant of a single writer and multiple readers. When data access conflicts, request may not be eventually satisfied(potential starvation), in which case token protocol initiates a persistent request, activates at most one persistent request and ensures all race requests be finally satisfied.

Based on the starvation solving mechanism, any definition of SRC entry is all right for token-SRC protocol. For example, if SRC presents an empty destination set to a write request, request processor will not get enough tokens, and then token protocol will reissue those requests, taking it as starvation case after timeout and solving it by issuing persistent requests.

3.3 Performance Consideration

The size of destination set decides how many messages are sent for a cache miss request. Small destination set means less messages and less network traffic. Although it is all right no matter what destination set are stored in SRC, it does degrade overall system performance when destination set of SRC are not superset of minimal destination set. In that case, token count requirement will not be satisfied and token protocol turns request into persistent request. The satisfy process of persistent requests is rather time-consuming. In token-SRC protocol design, it is better to be away from this case. Thus, for performance consideration, SRC entry should better be the superset of minimal destination set and be close to minimal destination set.

3.4 Token-SRC Protocol

To design cache controller of CMP system using Token-SRC protocol, we have to answer the following five questions: (1) Which states are used to describe a cache block? (2) What events incur cache state transition? (3) What does SRC record for each cache state? (4) How does cache state transit? (5) How is each event handled?

Firstly, we adopt MOESI protocol in token-SRC protocol design, which uses 6 states to describe a cache block. M (Modified) state means the cache block was modified. O (Owned) state means local processor owns the data block though other processor may have shared data copies. S (Shared) state means local processor has valid data copies. E (Exclusive) state means that only local processor has an exclusive data copies and not modified. I (Invalid) state means cache block in local processor is invalidated by other processors. NP (Not Present) state means that the data block does not remain in the cache, it is not a real state saved in cache block.

Secondly, we defined 4 kinds of events coming from local processor: cache read miss, cache write miss, data cache replacement, and SRC replacement. In dealing with local events, processor may generate 3 kinds of remote events: remote read request, remote write request, and remote data cache replacement. In the following cache state transition analysis, we only need to handle local events because the handling process includes remote events.

Thirdly, we give a definition on contents of SRC in Table 1. Data blocks in NP state did not exist in local cache, and SRC need not to save entries for that block. For data blocks in M or E state, reading or writing them always hit and SRC will not be searched for destination set. Reading data blocks in O or S state also hits, but writing those data blocks needs to invalidate other valid copies in the CMP system, at that time SRC can be used to denote destination set. Last, for data block in I state, read miss needs to request owner processor and write miss needs to invalidate all processors with valid cached copies. Based on definition in Table 1, SRC can helps to provide owner processor information for reading case, but no use to writing cases. Thus, writing data block in I state has to broadcast writing requests.

Fourthly, driven by each event, cache states may change. The state transition graph of token-SRC protocol is a traditional MOESI protocol.

Finally, For each cache state and each event from local processor, the event handling process consists of three continuous phases: requests sending process in local processor, requests responding process in remote processor and responses receiving process in local processor.

Compared with traditional token protocol, token-SRC protocol introduces new actions on SRC. (1) In request sending process of local processor, if SRC hits, request is sent to destination set denoted by SRC. Otherwise, request is broadcast. (2) In request responding process of remote processor, if remote processor is the owner of data block requesting, it searches its own SRC to get

Table 1. Contents of SRC for Each Cache State

Cache State	Contents of SRC
M	None
O	All processors with valid cached copies
E	None
S	All processors with valid cached copies
I	Owner processor
NP	None

sharers, sending it back together with data and tokens. Otherwise, remote processor updates its SRC according to the request type and its own cache block state. (3) In response receiving process of local processor, local processor updates its SRC according to response message type and its own cache block state. Fig. 3 shows a general event handling process.

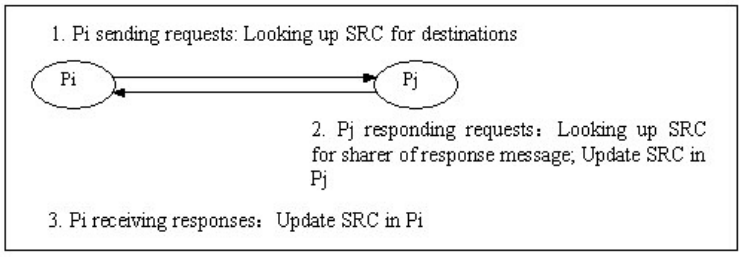

Fig. 3. General Event Handling Process of Token-SRC Protocol

4 Performance Evaluation of Token-SRC Protocol

4.1 Simulation Environment

We evaluate a 16-core SPARC CMP system using GEMS simulator. The CMP runs unmodified Solaris 8. Each processor core is a simple in-order processor with private 64KB L1 caches and 16MB L2 cache. Memory is 4GB and divided into 16 banks. We adopt 2D torus topology to interconnect 16 processor nodes, with on-chip link latency (processor-to-processor) 1 ruby cycle and out-of-chip link latency (processor-to-directory) 40 ruby cycles.

Token protocol and directory protocol has been implemented in GEMS, but we have to extend token protocol to integrate SRC. To compare directory, token and token-SRC protocol, their simulation machine use the same processor

model, same cache organization and size, same unordered 2D torus interconnection network and latency parameters, and same peripherals outside core. Their cache coherence protocols have different state transition graph but with the same MOESI cache state description.

We use SPLASH-2 benchmarks as workloads. For fast simulation, we collect statistic result of parallel execution part instead of complete execution.

4.2 Network Traffic

We compare directory protocol, token protocol and token-SRC protocol in request traffic and network traffic. Network traffic is measured by the amount of information delivered in network per time unit. We calculate the amount of information delivered in network by total network message bytes, and time unit by cache miss. The size of request message is 8B, data response is 72B (64B data with an 8B header), and data response with sharer is 76B(64B data, 4B sharer with an 8B header) in Token-SRC protocol.

Fig. 4 shows evaluation result on request message number per cache miss for three protocols. Request message includes initial request, forwarded request(directory protocol), retried request and persistent request (token and token-SRC protocol). In the statistic process, if a request is send to k destination nodes, the message number of the request is set to k no matter how many requests exactly run through interconnection network.

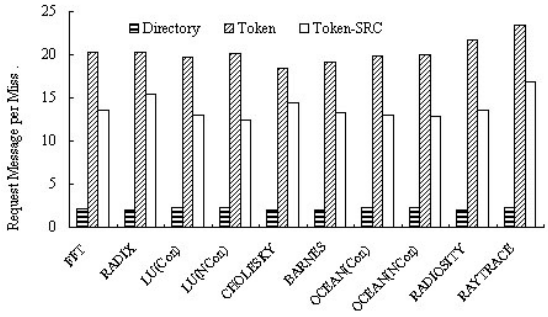

Fig. 4. Request Message per Miss(16p CMP)

As illustrated in Fig. 4, directory protocol issued 2.16 request messages per cache miss on average for 10 benchmarks. Token protocol has much higher bandwidth usage than directory protocol, issuing 20.27 request messages per cache miss on average(20.27 is greater than processor number 16, that is because the request may be reissued multiple times and turned into persistent request at last after time-out). Token-SRC protocol issues 13.85 request messages per cache miss on average. In contrast to token protocol, Token-SRC reduces request traffic by 32% in average.

Fig. 5 shows normalized network traffic per cache miss for three tokens. The network traffic of directory protocol is normalized to 1. From the figure, token protocol took about 66% more interconnection bandwidth on average than directory protocol, and token-SRC protocol used about 42% more interconnection bandwidth. Compared with token protocol, token-SRC achieved 15% network traffic reduction on average. Since 72B response message is much larger than 8B request message, it is easy to understand why token-SRC protocol issued 32% less requests than token protocol while incurred only 15% reduction in network traffic.

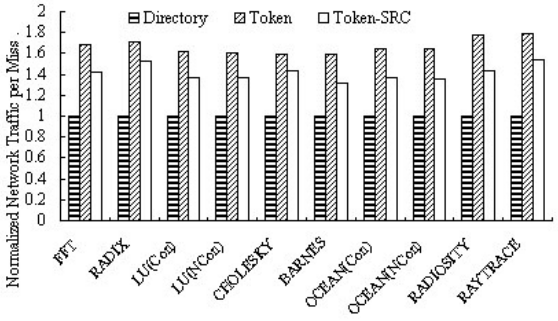

Fig. 5. Network Traffic per Miss (16p CMP)

5 Conclusions and Future Work

This paper introduced SRC into token protocol to reduce network traffic of token protocol. Evaluation based on SPLASH-2 benchmark shows that token-SRC protocol achieved 15% interconnection network traffic reduction of token protocol on average. We believe that network traffic can be reduced further if the following improvements are added to current design. (1) Evaluate how SRC organization, SRC size, data block size as well as normal cache replacement policy affects network traffic, and choose optimal policy. (2) Evaluate whether current token-SRC implementation brings more persistent requests and explore other SRC implementation optimization methods, especially for the write miss cases (3) Try speculative request issuing for broadcast case(for example, reading or writing an un-cached data block). (4) Optimize workload to improve temporal locality of shared data, achieving high SRC hit rate under limited cache space.

References

1. Hammond, L., Nayfeh, B., Olukotun K.: A single-chip multiprocessor. IEEE Computer **30** (1997) 79–85

2. Olukotun, K., Nayfeh, B., Hammond, L., Wilson, K. , Chung, K.: The case for a single-chip multiprocessor. Int'l conf. Architectural Suppport for Programming Language and Operating System (1996) 2–11
3. Hammond L., Hubbert B., Siu M., Prabhu M., Chen M., Olukotun K.: The Stanford Hydra. IEEE Micro (1996) 71–84
4. Goodman J.: Using Cache Memory to Reduce Processor-Memory Traffic. Int'l Symp. on Computer Architecture (1983) 124-131
5. Katz R., Eggers S., Wood D., Perkins C., Sheldon R.: Implementing a Cache Consistency Protocol. In 12th Int'l Symp. on Comp. Arch.(1985) 276–283
6. Tang C.: Cache Design in the Tightly Coupled Multiprocessor System: AFIPS National Computer Conference(1976) 749–753
7. Censier M., Feautier P.: A New Solution to Coherence Problems in Multicache Systems. IEEE Trans. on Computers **12** 1978 1112–1118
8. Martin M., Hill M., Wood D.: Token Coherence: Decoupling Performance and Correctness. Int'l Symp. on Computer Architecture (2003) 182–193
9. Marty M., Bingham J., Hill M., Hu A., Martin M., Wood D.: Improving Multiple-CMP Systems Using Token Coherence. Int'l Symp. on High-Perf.Computer Architecture (2005) 328–339
10. Wang H., Wang D., Li P.: SRC-based Cache Coherence Protocol in Chip Multiprocessor. Japan-China Joint Workshop on Frontier of Computer Science and Technology (2006) 60–67
11. Martin M., Harper P., Sorin D., Hill M., Wood D.: Using Destination-Set Prediction to Improve the Latency/Bandwidth Tradeoff in Shared-Memory Multiprocessors. Int'l Symp. onComputer Architecture (2003) 206–217
12. Martin M., Sorin D., Beckmann B., Marty M., Xu M., Alameldeen A., Moore K., Hill M., Wood D.: Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset. Computer Architecture News, (2005)
13. Magnusson P., Christensson M., Eskilsson J., Forsgren D., Hallberg G., Hogberg J., Larsson F., Moestedt A., Werner B.: Simics: A full system simulation platform. IEEE Computer **35** (2002) 50–58
14. Woo S., Ohara M., Torrie E., Singh J., Gupta A.: The SPLASH-2 Programs: Characterization and Methodological Considerations. Int'l Symp. on Computer Architecture (1995) 24–36