쓰기-갱신 정책에서의 거짓 공유를 제거하는 캐쉬 일관성 프로토콜

정회원 신 창 둔*, 정 연 만**, 김 봉 기***

A Cache Coherence Protocol for Eliminating False Sharing Using Write-Update Police

Chang-Doon Shin*, Yun-Man Jung**, Bong-Gi Kim*** *Regular Members*

요 약

각 프로세서에서 전용으로 캐쉬가 연결된 버스 기반 공유 메모리 다중 프로세서 시스템에서 시스템 성능은 캐쉬 일관성 유지를 위해 발생되는 버스 트래픽에 의해 결정된다. 이러한 버스 트래픽 오버헤드 중 어떤 것은 이른바 거짓공유(false sharing)로 인해 발생되는데 거짓공유는 캐쉬 블록이 다중워드로 구성된 경우 각 프로세서가 동일 블록내의 서로 다른 워드들을 독립적으로 참조할 때 발생하는 것으로 이때 시스템은 캐쉬 블록만이 공유되고 실제 데이터는 공유되지 않았음에도 불구하고 전체 데이터가 공유된 것처럼 취급한다.

이러한 거짓공유를 제거하는 기존 방법으로 워드단위 캐쉬 일관성 프로토콜들이 제안되었다. 그러나 이러한 방법들은 쓰기-무효(write-invalidation) 프로토콜을 기반으로 하고 있고, 쓰기-갱신(write-update) 프로토콜에 기반한 캐쉬 일관성 프로토콜은 거의 전무한 실정이다. 따라서 본 연구에서는 Firefly 프로토콜을 변경하여 쓰기-갱신 정책에서 거짓 공유 제거가 가능한 효율적인 워드단위 캐쉬 일관성 프로토콜을 제안하고자 한다.

ABSTRACT

In a bus-based shared memory multiprocessor system with private caches, the performance is limited by the amount of the bus traffic in order to keep the cache coherence. Some of this traffic overhead is caused by so called false sharing when multiple processors access different words in the same cache blocks and the system treats it as if data is shared even through the cache block containing the data is actually not shared. Some word unit cache protocols were proposed to eliminate the false sharing in cache coherence protocol. But these protocols are based write-invalidate policy. In this paper we present a new cache coherence protocol to eliminate the false sharing using wtite-update cache policy.

Ⅰ. 서 론

공유메모리 다중 프로세서(shared memory multiprocessor)에서 발생하는 메모리충돌(memory contention), 통신 집중화(communication contention) 등의 문제점을 해결하기 위한 방법은 각 프로세서에 전용으로 캐쉬 메모리를 연결하는 것이다. 이러한 구조에서는 동일 메모리 블록이 여러 개의 캐쉬에 복사되어 저장될 수 있으며 이 경우 각 캐쉬간 자료의 일관성을 유지하기 위한 이른바 캐쉬 일관성 프로토콜(cache coherence protocol)이 필요하다. 기존의 캐쉬 일관성 정책 중 하드웨어적인 것으로는 스누피 프로토콜(snoopy cache protocol), 디렉토리 기법(directory scheme), 캐쉬 일관성 네트워크 구조(cache coherence network architecture) 등이 있다[1].

* 한림정보산업대학 컴퓨터응용과(cdshin@sun.hallym-c.ac.kr)
** 원주대학 전자통신과 (jymkgl@sky.wonju.ac.kr)
*** 진주산업대학교 컴퓨터공학과(bgkim@cjcc.chinju.ac.kr)
논문번호 : T00030-0822, 접수일자 : 2000년 8월 22일

이 중 공유 버스를 사용하는 시스템에서 사용할 수 있는 프로토콜의 기본이 되는 것은 스누피 프로토콜로서 크게 쓰기-무효 프로토콜과 쓰기-갱신 프로토콜로 구분된다. 두 정책 중 일반적으로 쓰기-무효 정책이 캐쉬 일관성을 유지하는 데 필요한 전반적인 오버헤드가 더 적은 것으로 알려져 있다. 기록-무효화 정책을 사용하는 것으로, Illinois, Berkeley 등이 알려져 있다. 기록-갱신 정책은 Firefly, Dragon 등이 알려져 있다.

쓰기-무효 캐쉬 일관성 프로토콜을 사용하는 시스템 성능은 캐쉬 실패(cache miss)로 인한 캐쉬 블록 전송, 무효화 신호(invalidation signal), 주메모리 쓰기(write)에 의해 발생하는 버스 트래픽에 의해 결정된다. 공유 주소에 대한 캐쉬 블록 쓰기 수행은 일관성 유지를 위해 버스 트래픽을 유발시킨다. 이러한 버스 트래픽 중 참공유(true sharing)에 의한 것은 필연적이나 캐쉬 블록이 다중 워드(multi-word)로 구성되기 때문에 발생하는 거짓 공유(false sharing)에 의한 것은 제거되어야 한다. 기록-무효 캐쉬 일관성 프로토콜에서 거짓 공유를 제거하기 위한 방법으로는 국내에서 워드 단위 캐쉬 일관성 프로토콜과 개선된 워드 단위 캐쉬 일관성 프로토콜이 있으며, 국외에서는 Eggers 방법이 있다[2][3][4].

그러나 거짓 공유는 쓰기-갱신 프로토콜에서도 발생할 수 있으며, 특히 한 블록 내에 워드 수가 증가할수록 통계학적 가능성은 더욱 커지게 되어, 버스 트래픽을 더욱 가중시킬 가능성이 있다. 하지만 국내외적으로 쓰기-갱신 정책에 대한 거짓 공유 제거에 대한 논의가 거의 전무한 실정이다. 따라서 본 연구에서는 Firefly 프로토콜[5]을 기반으로 거짓 공유를 제거하는 효율적인 워드단위 캐쉬 일관성 프로토콜을 제안하고자 한다.

Ⅱ. 관련 연구

쓰기-무효 캐쉬 일관성 프로토콜을 사용하는 시스템에서 거짓 공유는 블록을 공유한 프로세서들이 블록 내 서로 다른 워드들을 참조할 경우 발생한다. 거짓 공유가 발생할 경우 이를 고려하지 않는 기존의 프로토콜은 실제 데이터는 공유하지 않고 블록 자체만이 공유되었음에도 불구하고 전체 블록을 무효화시킨다. 따라서 무효화된 캐쉬 블록의 프로세서는 이후에 실제 공유하지 않고 자신만이 참조했던 데이터를 재 참조하려 할 때 사실은 새로 읽어올 필요가 없는 데도 불구하고 캐쉬 실패로 인해 해당 캐쉬 블록을 다른 프로세서나 메모리에서 재적재해(reloading) 와야 하고, 이때 추가의 버스 트래픽이 발생한다. 이렇게 거짓 공유로 인해 발생하는 캐쉬 실패는 전체 캐쉬 실패의 최대 40% 정도인 것으로 알려져 있다[2].

거짓 공유을 제거하며 캐쉬 일관성을 유지하는 기존의 방법중 워드 단위 캐쉬 일관성 프로콜(WUCP: Word Unit Cache Protocol)에 기반한 연구는 [2]와 [3]이 있다. 워드 단위 캐쉬 일관성 프로토콜은 거짓 공유를 고려하지 않은 기존의 프로토콜이 블록 단위로 각 캐쉬 블록의 일관성을 유지했던 것과 달리 블록내 각 워드에도 추가의 비트를 할당하여 워드 단위로 캐쉬 일관성을 유지하는 기법이다.

[2]에서는 Illinois 프로토콜을 기반으로 하여 워드 단위 캐쉬 일관성 프로토콜(이하 WUCP)이 제안하여 블록의 상태를 기존 Illinois 프로토콜의 4가지 상태 즉, EX(Exclusive Unmodified), SU(Shared Unmodified), EM(Exclusive Modify), I(Invalid) 외에 거짓 공유 상태인 FS(False Shared)를 두어 5가지로 구분하였다. WUCP의 구조는 다음과 같다. 먼저 각 캐쉬 블록의 상태는 5가지 상태이고, 이 5가지 상태를 나타내기 위해 각 블록마다 3비트가 필요하다. 다음으로 캐쉬 블록내 각 워드마다 U비트와 F비트를 추가로 2비트를 두는데, 이것은 캐쉬 블록내 지역 프로세스만이 사용하고 다른 프로세스는 사용하지 않은 워드가 있을 때 거짓 공유가 발생하고 이를 고려하지 않고 바로 무효와 시킴으로 인해 불필요한 버스 트래픽이 발생한다는 점에 착안한 것이다. 즉, 해당 워드를 지역 프로세서가 참조한 경우에는 U비트를 1로 설정하고, 다른 원격 프로세서가 해당워드를 원격 참조한 경우에는 F비트를 1로 설정한다. F비트가 1이라고 해서 반드시 거짓 공유가 제거되어야 하는 것은 아니며 이 워드로 인해 거짓 공유가 제거되어야 할 가능성이 있을 뿐이다. 실제로 거짓 공유가 종료되어 제거되는 시점은 원격 프로세서에서 읽거나 쓰기를 하여 할 때이다. 이 때, 참조하려는 워드의 F비트가 1인 경우에는 무효화 신호가 발생하게 된다. 그러나 거짓 공유상태인 캐쉬 블록 내에서 U비트가 1이고 F비트가 0인 워드의 경우 이 워드는 자신만이 사용하고 있는 경우이므로 거짓 공유 발생의 가능성은 없다. 따라서 이 경우에는 무효화 신호가 발생되지 않게 되며 이렇게 함으로써 버스 트래픽을 감소할 수 있게 된다.

그러나 WUCP에도 몇 가지 문제점이 있다. 첫 번째 문제는 공유하는 블록이 모두 올바른 내용을 갖고 있는 SU 상태에서 읽기 작업이나 쓰기 작업이 일어날 때 해당 블록을 공유하는 원격 프로세서의 캐쉬 블록내 워드의 F비트를 1로 설정하는 과정을 거친다. 이것은 기존의 거짓 공유를 고려하지 않은 캐쉬 일관성 프로토콜이 쓰기 작업만을 고려하여 캐쉬 일관성을 유지한 것과 비교할 때, 거짓 공유를 감지하기 위해 추가의 버스 트래픽이 발생하는 요인이 된다. 두 번째 문제점은 기존의 거짓 공유를 고려하지 않은 캐쉬 일관성 프로토콜에 비해 추가적인 비트 오버헤드가 필요하다.

[3]는 이러한 WUCP의 문제점을 개선하기 위해 개선된 워드 단위 캐쉬 프로토콜(IWUCP: Improved Word Unit Cache Protocol)을 제안하였다. 개선된 워드 단위 캐쉬 프로토콜(이하 IWUCP)은 Write-Once 정책에 기반한 기록-무효 스누피 캐쉬 프로토콜을 확장하여 사용하였다. Write-Once는 각 블록의 상태를 블록이 올바른 내용을 갖고 있으면서 공유된 Valid 상태, 메모리와 해당 블록만이 일치하는 내용을 갖고 있는 Reserved 상태, 시스템 내에서 유일하게 올바른 내용을 갖고 있는 Dirty 상태, 블록이 무효화된 Invalid 상태와 같이 4가지로 구분된다. 여기서 Reserved 상태는 블록이 공유된 Valid 상태에서 어떤 한 블록의 내용이 최초로 갱신될 때 해당 블록을 공유한 다른 블록들을 무효화하고 메모리와 자신의 내용을 갱신하는 과정에서만 발생한다. 거짓 공유를 고려할 경우 Valid 상태에서 어떤 한 블록의 내용이 갱신된다는 것은 결국 해당 블록이 거짓 공유되고 있는 상황이고 이 경우 FS(Fasle Shared) 상태로 전이하게 되므로 Reserved 상태는 불필요하다. 따라서 IWUCP는 블록의 상태를 Valid, Dirty, Invalid, FS 상태로 구분하며 이 경우 각 블록의 상태를 나타내기 위해 각 블록당 필요한 비트는 거짓 공유를 고려하지 않은 일관성 프로토콜과 마찬가지로 2비트면 가능하다.

또한 IWUCP는 쓰기 작업이 일어났을 때에만 일관성 유지를 위해 각 워드마다 F비트만 할당된 구조를 갖는다. 따라서 IWUCP는 다른 기록-무효 프로토콜들에 비해 향상된 WUCP의 장점을 그대로 보장하며 거짓 공유를 제거하는 데 필요한 오버헤드를 줄일 수 있었다.

상기에서 살펴본 바와 같이 거짓 공유 제거 방법은 모두 쓰기-무효 프로토콜 기반이고 국내외적으로 쓰기-갱신 프로토콜에 기반한 거짓 공유 제거 방법에 대한 논의가 전무한 실정이다. 따라서 본 연구에서는 Firefly 프로토콜을 기반으로 거짓 공유를 제거하는 효율적인 워드단위 캐쉬 일관성 프로토콜을 제안하고자 한다.

Ⅲ. 쓰기-갱신 기반 거짓공유 제거 프로토콜

3.1 쓰기-갱신 기반 거짓공유 제거 프로토콜의 구조

본 연구에서 제안할 쓰기-갱신 기반 거짓 공유 제거 프로토콜(WUWUCP : Write-Update based WUCP)은 Firefly 프로토콜을 변경하여 (그림 1)과 같은 구조를 갖는다. 기존의 Firefly의 3가지 블록 상태 Valid-exclusive, Shared, Dirty에 False-shared 상태를 추가하였다. 지역 프로세서에서의 읽기 작업과 쓰기 작업은 P-Read와 P-Write로 나타내며, 각 블록에서 블록의 상태를 나타내기 위해 2비트의 데이터를 사용하였기 때문에 하나의 상태를 추가하여도 2비트로 표현이 가능하므로 추가 오버헤드 비트가 발생하지 않는다. 또한 캐쉬 블록내의 모든 워드는 하나의 U 비트(Used bit)를 사용한다. U 비트를 사용하여 원격 프로세서의 참조로 인한 캐쉬 블록의 해당 워드를 1로 설정하며, U 비트가 설정되지 않은 같은 블록내 다른 워드들에 대하여 거짓 공유가 발생된다. 따라서 지역 프로세서의 의해 사용중인 블록을 원격 메모리로부터 새로 읽게되면 지역 프로세서에서 사용중인 워드에 대하여 U 비트 설정 명령이 공유하는 다른 프로세서로 전달된다. 만약 U 비트가 설정된 워드에 대하여 원격 프로세서가 참조하게 되면 참 공유가 되고 거짓 공유는 끝나게 된다.

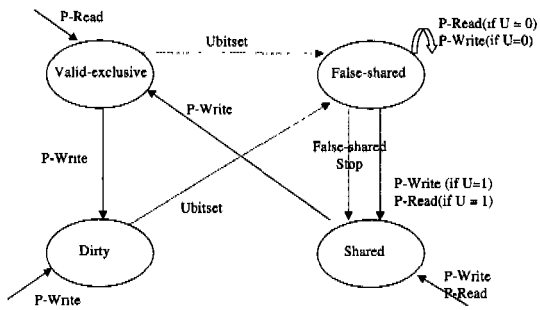


그림 1. 쓰기-갱신 기반 워드단위 캐쉬 일관성 프로토콜 상태전이도

(그림 1)에서 정의된 블록 상태를 자세히 기술하면 다음과 같다.

- Valid-exclusive : 메모리 블록과 일치하며 하나의 프로세서에 의해서 처음으로 참조되고 있는 상태이다.
- False-shared : Valid-exclusive 상태에서 한 개 이상의 또 다른 프로세서가 같은 메모리 블록을 메모리로부터 읽어온 상태로 거짓 공유 상태이다. 이때 캐쉬 제어기에 의해 U 비트가 발생되어 거짓 공유를 알려준다.
- Shared : 거짓 공유 상태에서 U 비트가 설정된 워드에 대한 참조가 일어날 때 발생하며 거짓 공유가 깨어지고 참 공유 상태가 된다.
- Dirty : Valid-exclusive 상태에서 지역 프로세서에 의해 쓰기 작업이 진행중인 상태이다. 이때 일관성 제어를 위한 동작이 요청되지 않는다.

3.2 쓰기-갱신 기반 거짓공유 제거 프로토콜의 동작

쓰기-갱신 기반 거짓공유 제거 프로토콜(이하 WUWUCP)이 거짓 공유를 제거하는 과정을 보이기 위해 다음과 같은 과정을 가정한다. (그림 2)와 같이 프로세서 1, 2, 3이 블록을 공유하고, 공유된 블록 내에는 A, B, C와 같이 세 개의 워드가 있다고 가정한다. 물론 앞에서 언급한 것처럼 각 블록에는 상태표시를 위해 2개의 비트가 있고 각 워드에는 U 비트를 위해 하나의 비트 필드가 필요하다.

다음과 같은 일련의 시나리오에 의해 작업이 이루어진다고 하고, 이때 Firefly와 WUWUCP에서 발생하는 블록 상태와 U비트의 상태 변화를 알아본다.

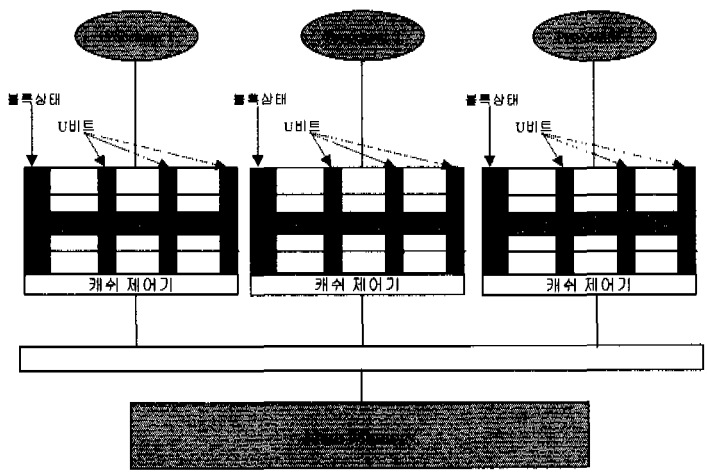


그림 2. 캐쉬 블록의 구조

최초의 상태는 아직 3개의 프로세서에서 어떤 동작도 취하지 않은 상태이다. 현재 왼쪽이 Firefly 프로세서 및 캐쉬 블록의 상태이고, 오른쪽이 WUWUCP 캐쉬 블록의 상태이다

P	상태
1	
2	
3	

P	상태	U비트
1		
2		
3		

1) Processor 1 reads word A

Firefly에서는 프로세서1에서 최초로 블록을 적재해 왔으므로 블록의 상태는 Valid-exclusive 상태가 된다. WUWUCP에서도 같은 상태가 되고 U비트는 아직 설정되지 않는다.

P	상태
1	Valid-exclusive
2	
3	

P	상태	U비트
1	Valid-exclusive	000
2		
3		

2) Processor 2 reads word B

Firefly에서는 프로세서1에서 참조하는 블록에 대하여 참조가 발생하였기 때문에 Shared 상태로 전이된다. 그러나 아직 서로 다른 워드에 대하여 공유를 하고 있으므로 WUWUCP에서는 거짓 공유 단계인 False-shared 상태로 전이된다.

P	상태
1	Shared
2	Shared
3	

P	상태	U비트
1	False-shared	010
2	False-shared	100
3		

3) Processor 3 writes word C

Firefly에서는 프로세서1, 프로세서2에 의해 Shared된 상태이므로 프로세서3까지 Shared 상태가 되고 프로세서3에 의해 변경된 블록이 쓰기-갱신 정책에 의해 공유하는 캐쉬에 전달된다. 이때 프로세서3에 의해 쓰기 작업이 계속되면 Firefly에서는 다른 2개의 캐쉬를 쓰기-갱신을 계속해야 하며 버스에 많은 트래픽이 유발된다. 하지만 WUWUCP에서는 프로세서3이 계속 지역 쓰기 작업이 진행되는 동안 거짓 공유 상태를 유지하고 있다가 거짓 공유가 끝나는 시점에서 갱신이 전달된다.

P	상태
1	Shared
2	Shared
3	Shared

P	상태	U비트
1	False-shared	011
2	False-shared	101
3	False-shared	110

4) Processor 2 writes word B

Firefly에서는 모든 프로세서가 Shared된 상태이므로 하나의 프로세서에서 쓰기 작업이 발생하면 공유하는 다른 프로세서의 캐쉬에 변경된 내용이 갱신된다. 그러나 WUWUCP에서는 U 비트에 대한 변화가 없이 지역 프로세서에 의한 쓰기 작업(Write-hit)만 발생한다.

P	상태
1	Shared
2	Shared
3	Shared

P	상태	U비트
1	False-shared	011
2	False-shared	101
3	False-shared	110

5) Processor 2 writes word A

U 비트가 설정된 워드에 대한 지역 프로세서의 쓰기 작업은 블록 상태를 거짓 공유 상태에서 해제되어 참 공유 상태로(Shared) 전이된다. 이때부터 Shared-exclusive 상태가 될 때까지 Firefly와 같은 동작을 유지한다.

P	상태
1	Shared
2	Shared
3	Shared

P	상태	U비트
1	Shared	000
2	Shared	000
3	Shared	000

Ⅳ. 프로토콜 검증

본 논문에서 제안한 프로토콜을 Firefly와 비교하여 검증하기 위해서는 캐쉬 일관성을 유지하기 위해 발생되는 버스 트래픽 양과 메모리 오버헤드 양을 추정해야 한다. 따라서 그 양을 정확히 비교하기 위해서는 시뮬레이션 시험을 통하여 여러 상황에서 비교해야 하지만 다른 추정 가능한 방법들을 이용하여 프로토콜을 검증하려고 한다.

4.1 버스 트래픽 비교

본 논문에서 제안한 방법과 Firefly 방법의 버스 트래픽을 상태전이도를 통해 비교하면 <표 1>과 같다. <표 1>에서 살펴 본 바에 의하면 Firefly 방법은 거짓 공유에 의한 버스 트래픽이 추가로 발생하지만 본 연구 방법에서도 U 비트 설정에 필요한 트래픽이 필요하며 또한 버스 신호를 추가로 설계해야 하는 부담이 있다. 3.2절에서 언급된 것처럼 거

표 1. 프로토콜간 버스 트래픽 발생 종류

프로토콜	블록상태	작업종류	버스트래픽발생	비고
Firefly	Valid-Exclusive(VE)	Read-Blk	○	Shared 상태로 전이
		P-Read	×	
		P-Write	×	D 상태로 전이
	Shared (S)	Read-Blk	○	
		P-Read	×	
		P-Write	○	캐쉬 블록 전송
	Dirty (D)	Read-Blk	○	S 상태로 전이
		P-Read	×	
		P-Write	×	지역적 갱신
본 연구	Valid-Exclusive (VE)	Read-Blk	○	U비트 1로 설정 방송, FS 상태로 전이
		P-Read	×	
		P-Write	×	D 상태로 전이
	False-shared (FS)	Read-Blk	○	U 비트 1로 설정
		P-Read	X	U 비트 0 일때
			X	U 비트 1 이면 FS 상태를 종료하고 S로 전이
		P-Write	X	U 비트 0 이면 지역적 갱신
			○	U 비트 1 이면 FS 상태를 종료하고 S로 전이
	Shared (S)	Read-Blk	○	
		P-Read	×	
		P-Write	○	캐쉬 블록 전송
	Dirty (D)	Read-Blk	○	U비트 1로 설정, FS 상태로 전이
		P-Read	×	
		P-Write	×	지역적 갱신

짓 공유된 상태에서 지역 쓰기가 많은 발생하는 경우에는 거짓 공유에 의한 버스 트래픽이 굉장히 많아져 Firefly 방법보다 본 연구 방법이 U 비트에 대한 오버헤드를 고려하더라도 성능이 좋아진다. 그러나 참 공유가 많이 발생하는 경우에는 오히려 본 연구 방법이 성능이 떨어지게 된다.

4.2 메모리 오버헤드

본 논문에서 제안한 방법에서 블록 상태를 표시하기 위한 추가 비트 메모리 오버헤드는 없다. 즉 기존의 방법과 같이 2 비트를 사용하여 블록 상태를 나타낸다. 그러나 각 블록내의 워드마다 U 비트 표시를 위한 1 비트의 메모리가 필요하다. 각 프로세서마다 한 개의 캐쉬가 연결되었다고 하고 캐쉬의 수를 N, 캐쉬내의 블록 수를 B, 블록 내 워드 수를 W 라 할 때, 필요한 비트 오버헤드를 계산하면 <표2>와 같다. 캐쉬 블록 크기가 증가하면 비트 오버헤드가 줄어들게 되며 워드 수가 증가하면 오버헤드 비율이 증가하게 된다. Firefly에 비해 블록내의 워드 수가 증가하면 비트 오버헤드는 증가하지만 워드 수의 증가에 따라 또한 거지 공유의 비율은 높아지게 되어서 쓰기-갱신에 의한 버스내의 트래픽 요구량은 본 논문에서 제시한 방법보다 상당하다. 따라서 본 논문에서 제시한 방법은 상대적으로 메모리 오버헤드는 적고 버스 트래픽 요구량은 많이 감소시켜 효율적인 캐쉬 일관성 프로토콜이라 할 수 있다.

표 2. 각 기법에 따른 비트 오버헤드

기법	비트 오버헤드
Firefly	2NB
WUWUCP	2NB + WBN

V. 결 론

본 연구에서는 Firefly 프로토콜을 변경하여 쓰기-갱신 정책에서 거짓 공유 제거가 가능한 효율적인 워드단위 캐쉬 일관성 프로토콜을 제안하였다. 기존의 Firefly의 3가지 블럭 상태 Valid-exclusive, Shared, Dirty에 False-shared 상태를 추가하였으며, 캐쉬 블럭내의 모든 워드는 하나의 U 비트(Used bit)를 사용하여 거짓 공유를 위한 태그 비트로 사용하였다.

프로토콜 동작 상태와 거짓 공유를 제거하는 과정을 보임으로써 쓰기-갱신 정책을 사용하는 스누피 캐쉬 프로토콜의 일관성을 확인하였다. 거짓 공유로 인한 쓰기-갱신 트래픽이 많은 경우 본 연구에서 제안한 방법이 효과적이라는 가능성을 입증하였다. 이러한 경우에는 기존의 스누피 공유버스 구조의 확장성을 더욱 넓힐 수 있는 방법 중에 하나라고 생각된다.

그러나 시뮬레이션 모델을 통한 충분한 시험이 부족하여 전체적인 성능 비교가 미흡했으며, 메모리 오버헤드 또한 다중 워드 구성에 따라 성능에 달라질 수 있다. 향후 이러한 점을 보완하여 쓰기-갱신 정책에서 거짓 공유 제거 가능성에 대한 밑거름으로 사용하고자 한다.

참 고 문 헌

[1] Per Stenstrom, "A Survey of Cache Coherence Schemes for Multiprocessors," Computer, Vol.23, No.6, pp.12-24, Jun, 1990.

[2] 정인범, 이준원, 박승규, "거짓 공유를 제거하기 위한 캐쉬 프로토콜", 정보과학회 논문지(A) 제23권 제 6호, pp.639 ~ 649, 1996.6.

[3] 이은희, 김주균, "거짓 공유를 제거하는 효율적 워드 단위 캐쉬 일관성 프로토콜", 정보과학회 논문지(A) 제25권 제 6호, pp.616 ~ 625, 1998.6.

[4] Eggers, S.J. and Jeremiassen. T.E., "Elminating False Sharing," Proceedings of the 1991 International Conference on Parallel Proceeding, pp.377~381, Agu.1991.

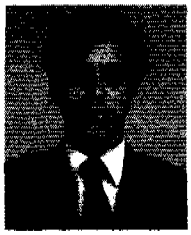
[5] Thacker, C. and Stewart L.C., "Firefly : A Multiprocessor Workstation," IEEE Transactions on Computer, Vol37, No 8, pp.909 ~ 920, Aug. 1988.

신 창 둔(Chang-Doon Shin) 정회원

1987년 2월 : 숭실대학교 전자계산학과 졸업(공학사)
1989년 2월 : 숭실대학교 대학원 전자계산학과 졸업 (공학석사)
1998년 9월~현재 : 숭실대학교 대학원 박사과정 재학중
1989년 1월~1997년 2월 : 한국전자통신연구원 선임

연구원
1997년 3월~현재 : 한림정보산업대학 컴퓨터응용과 조교수
<주관심 분야> 전자지불시스템, 디지털 워터마크, 무선서비스 응용

정 연 만(Yun-Man Jung) 정회원



한국통신학회 논문지 제 25권 제 6T호 참조
현재 : 원주대학 전자통신과 부교수

김 봉 기(Bong-Gi Kim) 정회원

1987년 2월 : 숭실대학교 전자계산학과(공학사)
1989년 2월 : 숭실대학교 전자계산학과(공학석사)
1999년 2월 : 숭실대학교 전자계산학과(공학박사)
1994년 3월~1999년 2월 : 한림정보산업대학 컴퓨터응용과 교수
1999년 3월~현재 : 진주산업대학교 컴퓨터공학과 교수
<주관심 분야> 내용기반검색, 전자상거래, 웹데이터베이스