디렉토리를 이용한 캐쉬 일관성 유지 기법에서 무효화 힌트를 이용한 읽기 접근 시간 감소

(Reduction of Read Access Latency by Invalid Hint in Directory-Based Cache Coherence Scheme)

오 승 택 † 이 윤 석 †† 맹 승 렬 ††† 이 준 원 †††

(Seungtaek Oh) (Yunseok Rhee) (Seung Ryoul Maeng) (Joonwon Lee)

요 약 대규모 분산 공유메모리 다중처리기는 공유메모리 접근 지연시간이 크다는 약점을 지니고 있다. 이러한 다중처리기에서 모든 메모리 요청이 홈노드를 통해 이루어지는 디렉토리 기반의 캐쉬 일관성 유지 기법의 사용은 메모리 접근 지연시간을 더욱 크게 하는 요인으로 작용한다. 뿐만 아니라 메모리 접근 지연시간은 시스템의 규모가 커질수록 전체 성능에 중요한 요소로 작용하므로, 대규모 시스템에서 이를 줄이기 위해서 많은 연구들이 있었다. 본 논문에서는 메모리 읽기 지연시간을 줄이는 새로운 캐쉬 일관성 유지 기법을 제안한다. 제안된 기법은 무효화힌트를 이용하여 구현되었다. 무효화힌트는 어떤 노드가 전에 캐쉬블록을 무효화 시켰는가에 관한 정보이며, 메모리블록이 필요한 노드는 이 정보를 이용하여 홈노드의 도움 없이 직접 메모리 요청을 할 수 있다. 제안된 프로토콜의 성능을 측정하기 위하여 모의실험을 하였다. 모의실험 결과는 제안된 프로토콜에서 읽기 지연시간이 감소하는 것을 나타낸다.

Abstract Large scale shared memory multiprocessors have suffered from large access latency to shared memory. The large latency partially stems from a feature of directory-based cache coherence schemes which require a shared memory access to be serviced at a home node of the memory block. The home visit results in three or more hops traversal for a memory read access. The traversal becomes much longer as a system scales up. In this paper, we propose a new cache coherence scheme that reduces read access latency. The proposed scheme exploits ideas of invalid hint. Invalid hint for a cache block means which node has invalidated the cache block before. Thus a read access request can be directly sent to and serviced by the node (called owner) without help of a home node. Execution-driven simulation is employed to evaluate performance of the proposed scheme. The simulation results show that read access latency and execution time are reduced.

1. 서 론

캐쉬 메모리는 프로세서의 수행속도에 비해 현저히 느린 메모리 접근 시간을 줄이기 위해 고안된 기술이다[1]. 캐쉬 메모리를 사용하는 시스템에서는 캐쉬 실패(cache miss)에 의한 메모리 접근 시간의 지연이 전체 시스템의 성능을 저하시키는 주된 요인 중 하나로 나타나고 있다. 특히 다단계 상호연결망에 의해 구성된 대부분의 다중처리기 시스템에서는 메모리 시스템이 다수의 처리기 사이에 분산되어 구현된다. 이러한 시스템에서는 원격 노드(remote node)에 위치한 메모리의 접근을 위해 긴 네트워크 지연시간까지 감수해야 하므로 캐쉬의 역할은 더욱 중요하다.

캐쉬 메모리를 이용하는 다중처리기에서는 캐쉬 사이

† 학생회원 : 한국과학기술원 전기전산학과
stoh@camars.kaist.ac.kr
†† 비 회 원 : 한국외국어대학교 전자제어공학부 교수
rheeys@control.hufs.ac.kr
††† 종신회원 : 한국과학기술원 전기전산학과 교수
macng@cs.kaist.ac.kr
joon@kaist.ac.kr
논문접수 : 1999년 11월 17일
심사완료 : 2000년 3월 7일

의 일관성(coherency)을 유지하는 것이 중요한 문제이다. 캐쉬 메모리의 일관성이란 여러 개의 프로세싱 노드들이 하나의 메모리 블록(memory block)을 자신의 캐쉬 메모리에 가져와서 사용하고 있을 때, 메모리의 복사본(copy)들은 그 내용이 서로 일치해야 한다는 것이다[2]. 따라서 공유된 메모리에 대해 쓰기가 일어날 때는 공유된 다른 캐쉬 데이타들을 무효화(invalidation) 또는 갱신(update)시키는 작업이 필요하다[3][4]. 다중처리기에서는 단일처리기에서 나타나는 일반적인 캐쉬 실패의 원인 외에도 캐쉬 메모리 사이의 일관성(coherency)을 유지하기 위해서 발생하는 캐쉬 실패가 발생한다.

캐쉬 일관성을 유지하는 대표적인 방법으로 스누핑 프로토콜(snooping protocol)과 디렉토리 프로토콜(directory protocol)이 있다. 스누핑 프로토콜은 버스를 기반으로 하는 시스템에서 사용하는 캐쉬 일관성 유지 프로토콜이다. 이는 버스의 특성상 여러 개의 캐쉬가 다른 캐쉬들의 상태를 지속적으로 관찰할 수 있으므로 가능하다. 디렉토리 프로토콜은 상호연결망 구조의 다중처리기에서 사용되며, 캐쉬에 대한 상태정보를 메모리의 데이타 블록과 함께 기록하여, 캐쉬의 일관성을 유지하는 프로토콜이다. 스누핑 프로토콜은 버스를 공유할 수 있는 프로세서의 개수에 대해 제한이 있는 반면 디렉토리 프로토콜은 이 제한을 극복할 수 있는 방법으로 대규모 다중처리기에서 많이 쓰이는 방법이다[5].

대표적인 다중처리기 시스템으로 CC-NUMA(Cache-Coherent Non-Uniform Memory Access Machines)를 들 수 있다[6][7]. CC-NUMA에서는 모든 메모리 블록들은 그 메모리 블록을 관리하는 노드가 정해져 있으며 이러한 노드를 홈노드(home node)라고 부른다. 홈노드가 하는 일은 크게 두 가지인데, 하나는 다른 노드에 대해서 메모리 블록을 제공하는 일이고, 다른 하나는 여러 캐쉬 상에 존재하는 메모리 블록의 복사본들에 대한 일관성을 책임지는 일이다. 따라서 CC-NUMA의 메모리 시스템은 비교적 구현이 쉽고 확장성이 우수하여 많은 다중처리기에서 채택하고 있다. 그러나 CC-NUMA에서는 캐쉬의 일관성 유지가 홈노드를 통해서만 관리되므로 메모리 블록에 대한 요구가 반드시 홈노드를 거쳐야 한다는 단점이 있다. 이로 인해서 캐쉬의 읽기 실패에 따른 메모리 접근 지연 시간(이하 지연 시간)은 매우 길어지며, 이 문제는 대규모 다중처리기에서 더욱 심각하다.

따라서 본 논문에서는 디렉토리 프로토콜 기반의 CC-NUMA에서 캐쉬 메모리의 읽기 실패가 일어났을 경우 지연 시간을 줄이기 위한 프로토콜(protocol)을 제안한다. 본 프로토콜에서는 지연 시간을 줄이기 위해서 무효화 힌트(invalid hint)기법을 이용한다. 무효화 힌트란 캐쉬의 일관성 유지를 위하여 자기 노드의 캐쉬 블록이 무효화 될 때 어떤 노드가 무효화 시켰는지에 관한 정보이다. 이 때 무효화 당한 지역 노드(local node)의 입장에서 무효화를 시킨 노드를 오너 가능성 노드(probable owner node)라고 한다. 지역 노드에서 무효화로 인한 캐쉬 실패가 일어났을 때 오너 가능성 노드가 여전히 유효한 메모리 블록을 가지고 있을 가능성이 있다. 그러므로 지역 노드에서 오너 가능성 노드로 메모리 블록의 요청을 보내 성공하면 홈노드를 거치지 않고 메모리 블록을 가져 올 수 있으므로 지연 시간을 줄일 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 무효화 힌트의 개념을 살펴보고, 3장에서는 무효화 힌트를 이용하여 읽기 실패의 지연시간을 줄이는 새로운 프로토콜을 제안한다. 4장에서는 제안된 프로토콜의 성능에 관한 모의실험 결과를 살펴본 후 마지막으로 5장에서 결론과 향후 연구 과제를 밝힌다.

2. 무효화 힌트

프로그램의 시간지역성(temporal locality)이란 프로그램 실행 중에 프로세서가 임의의 메모리 블록을 참조하면 일정시간 내에 같은 메모리 블록을 다시 참조할 가능성이 매우 크다는 것이다[8]. 이와 같은 성질은 다중처리기에서도 적용되어 캐쉬일관성을 유지하기 위해서 어떤 메모리 블록이 무효화될 경우 무효화된 블록들은 곧 다시 참조될 가능성이 크다. 그리고 메모리 블록을 다시 참조할 시점까지의 간격이 매우 짧다면 무효화시킨 노드가 여전히 유효한 메모리 블록을 가지고 있을 확률은 매우 높다. 이 경우에 사용된 오너 가능성 노드의 정보를 무효화 힌트라고 하며, 이를 이용해서 메모리 접근 시간을 줄이려는 노력들이 있어 왔다.

이렇게 무효화 힌트가 적용된 시스템으로는 소프트웨어 DSM과 COMA-F가 있었다. 먼저 소프트웨어 DSM 중에서 메모리의 복제와 이동이 가능한 시스템은 홈노드가 고정되어 있지 않았다. 이 경우는 각각의 메모리 블록에 대하여 책임지는 노드가 없으므로 캐쉬 실패가 일어났을 경우 해당 메모리 블록의 위치를 찾는데 어려움이 따르게 된다. 이를 손쉽게 해결하는 방법은 메모리 요청을 모든 노드에 전역방송을 하는 것이다. 그러나 이 방법은 많은 통신량을 발생시킨다. 이를 완화시키기 위해 Kai Li는 무효화 힌트를 사용하는 방법을 제안했다[9]. 무효화 힌트를 사용한 소프트웨어 DSM에서는 유

효한 메모리 블록을 가지고 있을 가능성이 높은 노드들에게만 메시지를 보냄으로써 통신량을 크게 줄일 수 있었다.

다음으로 무효화 힌트가 적용된 COMA-F(COMA-Flat)는 일반적인 COMA와 같이 데이타의 이동과 복제가 가능하지만, COMA와 달리 홈노드가 정해져 있는 시스템이다[10]. 이 경우는 홈노드가 메모리 블록의 위치를 알려 줄 수 있지만 무효화 힌트를 이용하면 홈노드의 도움 없이 유효한 메모리 블록을 찾을 수 있어 기존의 COMA보다 개선된 성능을 보인다.

지금까지 소프트웨어 DSM과 COMA-F에서 무효화 힌트를 사용한 방법이 제안되었으나 CC-NUMA에서는 무효화 힌트의 사용이 제안된 적이 없었다. 그 이유는 CC-NUMA에서는 홈노드가 고정되어 있으므로 홈노드가 고정되지 않은 소프트웨어 DSM처럼 무효화 힌트의 사용이 절실하지 않았고, COMA-F보다 캐쉬 크기가 작으므로 무효화 힌트의 효과가 작을 것으로 예상되고 오히려 통신량의 증가가 성능의 손실을 가져올 수 있다는 판단에서 비롯한다. 그러나 캐쉬의 크기가 점차로 커지고 있으며, 네트웍크의 대역폭도 급속히 향상되고 있으므로 CC-NUMA에서도 무효화 힌트의 사용으로 인한 성능 개선을 기대할 수 있게 되었다.

3. 무효화 힌트를 이용한 디렉토리 프로토콜

본 장에서는 읽기 실패에 따른 지연 시간을 줄이기 위해 무효화 힌트 기법을 이용한 디렉토리 프로토콜을 소개한다. 먼저 시스템의 구성에 관련한 기본 가정을 기술하고 새로운 프로토콜의 동작과정을 상세히 설명할 것이다.

3.1 시스템 구성

- 시스템 유형 : CC-NUMA
- 캐쉬 일관성 프로토콜 유형 : 디렉토리 프로토콜
- 캐쉬 컨트롤러 : 프로그램이 가능한 전용 프로세서
- 상호연결망 유형 : 2차원 메쉬
- 라우팅 기법 : 웜홀 라우팅(wormhole routing)
- 라우팅 알고리즘 : 결정적 라우팅(deterministic routing)
- 동기화 기법 : 대기행렬 기반 록(Array-Based Queuing Lock)

디렉토리 프로토콜은 버스가 아닌 직접 상호연결망을 사용하는 다중처리기에서 일반적으로 사용되는 캐쉬 일관성 프로토콜이다. 프로그램이 가능한 전용 프로세서로 이루어진 캐쉬 컨트롤러는 복잡한 프로토콜을 쉽게 구현할 수 있다는 장점이 있어서 최근 여러 다중처리기 시스템에서 채택되고 있다[11]. 2차원 메쉬도 구조가 간단하고 확장성이 좋은 상호연결망 유형으로 알려져 있다. 웜홀 라우팅은 패킷을 전달하는데 있어서 전송 최소 단위의 플릿으로 작게 나누어 파이프라인 형태로 전송되는 방법으로 지연시간이 노드간의 거리 차에 큰 영향을 받지 않는다는 장점을 가지고 있고, 메쉬 등과 같은 구조가 간단한 상호연결망에서 주로 사용되는 라우팅 기법이다[12]. 결정적 라우팅은 시작노드와 목적노드가 정해짐과 동시에 경로가 정해지는 방법이다. 결정적 라우팅에서는 시작노드와 목적노드가 같을 경우 먼저 보내진 메시지는 먼저 도착한다는 특성을 지니고 있다. 대기행렬 기반 록 (Array-Based Queuing Lock)의 동기화는 스핀 록(Spin Lock)을 이용한 동기화와는 달리 동기화과정에서 통신량이 적고 고르게 발생하여 대규모 병렬 시스템에서 많이 사용되는 방법이다[13].

3.2 기존 프로토콜에서의 메모리 읽기

본 절에서는 무효화 힌트를 사용하는 프로토콜에 대하여 살펴보기에 앞서 캐쉬 무효화 기법을 사용하는 기존의 디렉토리 프로토콜에서의 메모리 읽기에 대하여 살펴본다. 디렉토리 프로토콜의 디렉토리 상태는 프로토콜의 종류에 따라 다양하게 존재할 수 있지만 기본적으로 여러 개의 노드가 하나의 메모리 블록을 공유하여 읽기만 가능한 공유상태(shared state)와 하나의 노드가 메모리 블록을 독점하여 읽기와 쓰기가 가능한 독점상태(exclusive state)로 나눌 수 있다. 독점상태에서 메모리 블록을 독점하고 있는 노드를 오너노드라고 하며, 오너노드(owner node)가 되기 위해서는 오너쉽(ownership)을 가져야만 한다. 이와 같은 디렉토리 상태를 갖는 프로토콜에서 메모리 읽기는 다음과 같이 이루어 질 수 있다(그림1참조). 지역노드는 자신의 캐쉬에 유효한 메모리 블록이 없을 경우 홈노드로 메모리 요청을 한다. 이 때 디렉토리 상태가 공유상태이면 요청 받은 메모리 블록을 지역노드에 보내준다. 반면에 디렉토리 상태가 독점상태이면 오너노드에 있는 오너쉽을 빼

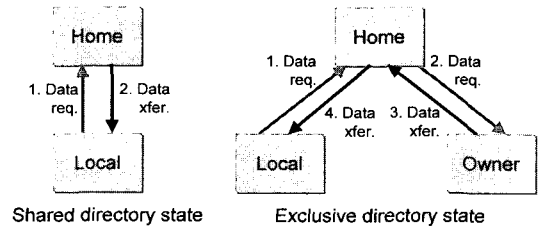

그림 1 기존 프로토콜에서의 메모리 읽기 과정

앗아서 디렉토리 상태를 공유상태로 만든 후에 해당하는 메모리 블록을 지역노드에 보내준다. 메모리 쓰기가 일어났을 경우는 디렉토리 상태를 독점상태로 만들면서, 쓰기 요청을 한 노드에게 오너쉽을 준다.

3.3 새로운 프로토콜에서의 메모리 읽기

본 절에서는 무효화 힌트를 사용하는 새로운 프로토콜에서 읽기 실패가 일어났을 경우의 메모리 읽기에 대하여 알아본다.

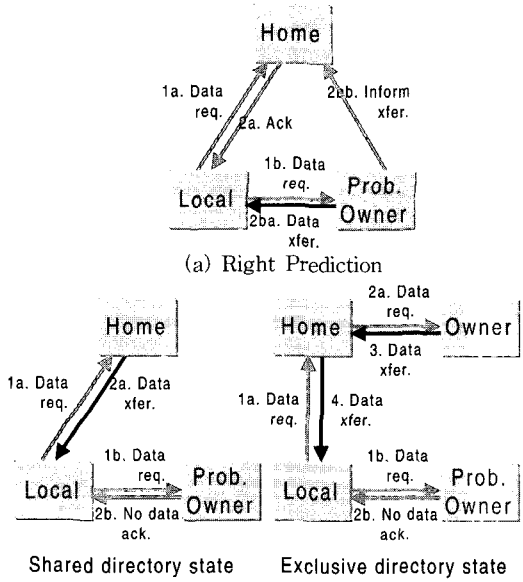

그림 2 새로운 프로토콜에서의 메모리 읽기 과정

그림2는 새로운 프로토콜에서 지역노드가 오너가능성노드를 알고 있을 경우 일어나는 메모리 읽기 과정이다. 지역노드의 캐쉬 블록이 무효화된 적이 없어서 오너가능성노드를 알 수 없을 경우와 메모리 쓰기에 대해서는 무효화 힌트를 사용하지 않는 기존의 프로토콜과 같은 동작이 일어난다.

그림2에서 볼 수 있듯이 새로운 프로토콜에서의 메모리 읽기는 오너가능성노드를 알 수 있을 경우에 오너가능성노드의 예측의 적중여부와 현재 디렉토리 상태에 따라서 세 가지로 나눌 수 있다.

3.3.1 오너가능성노드의 예측이 적중했을 때

1. 지역노드는 홈노드와 오너가능성노드에게 동시에 메모리 요청을 한다(1a, 1b). 이 때 홈노드에 보내는 메시지에는 오너가능성노드로도 메시지를 함께 보냈다는 정보를 알려준다.

2. 지역노드에서 보내온 메시지를 분석한 결과 오너노드를 바르게 예측했을 경우 홈노드의 디렉토리 상태가 공유상태이어서 유효한 메모리 블록을 가지고 있으면 메모리 블록을 지역노드로 보내주고, 디렉토리 상태가 독점상태이어서 유효한 메모리 블록을 가지고 있지 않으면 오너노드를 바르게 예측했다는 정보만 지역노드로 보낸다(2a).

3. 오너가능성노드는 바르게 예측되었으므로, 유효한 메모리 블록을 지니고 있다. 오너가능성노드는 지역노드에게 요청된 메모리 블록을 보내준다(2ba). 또한 오너가능성 노드는 지역노드에게 메모리 블록을 전달하였다는 정보를 홈노드에게 보내준다(2bb).

3.3.2 오너가능성노드의 예측이 실패하고 공유 디렉토리 상태일 때

1. 지역노드는 홈노드와 오너가능성노드에게 동시에 메모리 요청을 한다(1a, 1b). 이 때 홈노드에 보내는 메시지에는 전과 마찬가지로 오너가능성노드에 대한 정보를 같이 보낸다.

2. 디렉토리 상태가 공유상태이므로 홈노드는 오너가능성노드의 예측과 상관없이 요청된 메모리 블록을 지역노드로 보내준다(2a).

3. 오너가능성노드는 지역노드의 예측이 잘못 되었다는 정보를 지역노드로 보내준다(2b).

3.3.3 오너가능성노드의 예측이 실패하고 독점 디렉토리 상태일 때

1. 위의 두 경우와 마찬가지로 지역노드는 홈노드와 오너가능성노드에게 메시지를 보낸다(1a, 1b).

2. 독점 디렉토리 상태이므로 홈노드에 유효한 메모리 블록이 없고 지역노드의 예측도 잘못 되었으므로, 홈노드는 오너노드에게 메모리 블록을 요청하고, 오너노드에게 메시지를 받아서 지역노드에게 보내준다(2a, 3, 4).

3. 오너가능성노드는 지역노드의 예측이 잘못 되었다는 정보를 지역노드로 보내준다(2b).

3.4 무효화 힌트 구현시 고려 사항

3.4.1 오너 가능성 노드에 대한 정보 저장

무효화 힌트를 사용하는 프로토콜에서 지역 노드는 각각의 캐쉬 블록마다 캐쉬 블록을 무효화시킨 노드에 대한 정보를 저장할 공간을 필요로 한다. 이에 따라 추가적으로 필요한 저장 공간은 노드의 개수가 n이라고 할 때, 캐쉬 블록마다 $\log_2 n$ bits이다. 이러한 크기의 저장 공간은 무시할 수 없는 크기로써 무효화 힌트를 구현하는데 큰 부담으로 작용할 수 있다. 그러나 이 정보를 저장하는데 무효화된 캐쉬 블록을 이용한다면 추가적인 저장 공간의 필요를 없앨 수 있다. 캐쉬 블록이 무

효화되면 그 안에 있는 데이타들은 더 이상 필요 없는 데이타가 되며 그 공간을 이용해서 무효화시킨 노드의 정보를 저장할 수 있다.

3.4.2 일관성 유지를 위한 메시지 요청의 순차번호 할당

일반적인 CC-NUMA는 모든 메모리 요청이 홈노드를 통해서 이루어지므로 요청된 메모리 블록의 공급이나 캐쉬의 일관성 유지에 관한 모든 관리를 홈노드 혼자 처리할 수 있었다. 그러나 무효화 힌트를 사용하는 프로토콜에서는 홈노드뿐만 아니라 오너가능성노드가 메모리 블록을 공급해 줄 수 있으므로 두 노드간의 정보가 다를 경우 메모리의 일관성이 깨지는 경우가 발생한다. 실제로 홈노드와 오너가능성노드 모두에게 읽기 요청을 할 경우 홈노드는 오너가능성노드를 오너노드로 인식하고 있지만 오너가능성노드는 이미 오너쉽을 내어준 경우가 발생할 수 있다. 이를 해결하기 위하여 오너가능성노드는 자신이 오너노드가 아닐 경우 메모리 요청을 한 노드에게 그 사실을 알려주어야 한다. 메모리 요청을 했을 때 경우에 따라서는 홈노드와 오너가능성노드가 동시에 메모리 블록을 보내오지 않는 경우도 발생할 수 있는데, 이런 때는 같은 메모리 요청을 다시 한 번 해주어야 하고, 이를 위해서 모든 메모리 요청에 대해서 고유한 순차번호(sequence number)를 할당하는 것이 필요하다.

4. 성능 평가

4.1 모의 실험 환경

본 논문에서는 제안하는 프로토콜의 성능을 비교 측정하기 위해 MINT 패키지를 이용한 모의 실험기를 사용하였다[14]. MINT는 MIPS 프로세서로 구성된 병렬 프로세서 시스템을 모의 실험하는 실행 주도(execution-driven) 모의실험기로서 프로그램을 직접 수행하며, 사용자는 모의 실험에 필요한 상호연결망의 동작과 캐쉬 일관성 유지 프로토콜을 추가하여 실험할 수 있다. 모의 실험에서는 공유 메모리 읽기/쓰기를 가로채어 메모리 요청의 지연 시간을 관찰할 수 있도록 구현되었다. 상호연결망은 8x8 메쉬로 이루어져 있으며, 연결망 내부에 웜(worm)들이 돌아다니면서 진행(progress)하거나 정지(block)하도록 구현하였으므로, 통신량에 따른 지연 시간을 정확하게 측정할 수 있다.

본 논문의 모의실험을 위해서 사용한 응용 프로그램들은 SPLASH-2의 FFT, Radix, Barnes이다[15]. FFT는 전체 프로그램이 6개의 배리어(barrier)로 이루어지며, 이로 인해 발생하는 통신량이 많은 프로그램이다. Radix는 공유 변수에 대한 읽기와 쓰기의 비중이 특히 높은 프로그램으로 알려져 있다. Barnes는 전체 프로그램 실행시간 중에서 공유 변수를 읽고 쓰는데 걸리는 시간의 비중은 상대적으로 낮지만, 록 변수(lock variable)가 아닌 데이타 변수에 대한 공유가 많아서 무효화 힌트를 사용하기 알맞은 프로그램이다.

4.2 무효화 힌트의 사용빈도와 적중률

캐쉬 일관성 유지를 위해서 무효화된 메모리 블록을 다시 읽어오는 경우에 메모리 블록을 무효화시킨 노드가 여전히 유효한 메모리 블록을 가지고 있을 확률이 크다는 것이 무효화 힌트의 제안 동기이다. 따라서 무효화 힌트의 사용이 메모리 읽기 접근시간의 향상을 가져오기 위해서는 무효화 힌트의 사용빈도와 무효화 힌트의 적중률이 높아야만 한다. 무효화 힌트의 사용빈도란 다른 노드로 요청한 메모리 읽기 중에서 무효화 힌트를 사용한 비율이며, 무효화 힌트의 적중률이란 무효화 힌트를 사용한 메모리 요청 중에서 무효화 힌트가 적중하여 오너가능성노드에 원하는 데이타가 있는 비율이다.

표 1 무효화 힌트의 사용빈도와 적중률

application	usage rate	hit rate
FFT	17.2%	88.6%
RADIX	37.6%	94.9%
BARNES	41.3%	93.8%

표1은 실험에 사용된 각 프로그램이 보이는 무효화 힌트의 사용빈도와 적중률을 나타낸다. 무효화 힌트의 사용빈도는 FFT가 17%로 비교적 낮게 나타났으며, Radix와 Barnes가 약 40%로 높게 나타났다. 무효화 힌트가 사용되기 위해서는 '메모리 읽기 - 다른 노드에 의한 무효화 - 메모리 읽기'의 데이타 접근 유형이 나타나야 하며, 무효화 힌트의 사용빈도가 높다는 것은 이러한 데이타 접근 유형이 많다는 것을 의미한다. 무효화 힌트의 적중률은 표1에서 알 수 있듯이 대부분의 프로그램에서 90%에 가까운 수치를 나타낸다. 따라서 무효화 힌트를 사용한 디렉토리 프로토콜은 프로그램의 실행 시간을 효과적으로 줄일 것으로 예측할 수 있다.

4.3 통신량 증가의 비교

무효화 힌트 기법의 사용은 통신량의 증가를 가져오며, 통신량의 증가는 디렉토리 상태와 무효화 힌트의 적중여부에 따라 달라진다. 또한 통신량의 증가를 단순화시켜서 메시지 수에 비례한다고 생각하면, 공유 상태에서 예측이 적중했을 경우 최고 2.5배까지 증가한다. 하지만 전체적인 통신량의 증가는 그다지 많이 늘어나지

않는다. 그 이유는 다음과 같다. 통신량은 크게 메모리 읽기 통신량과 메모리 쓰기 통신량으로 나눌 수 있고, 메모리 쓰기 통신량은 무효화 힌트의 영향을 받지 않는다. 메모리 읽기에 있어서도 무효화 힌트를 사용할 때만 통신량이 증가하므로 전체적인 통신량의 증가는 크지 않다.

표 2 통신량의 증가

applications	increase of traffic
FFT	1.2%
RADIX	4.1%
BARNES	16.5%

표2를 보면 대부분의 프로그램에서 무효화 힌트를 사용했을 때 통신량이 증가한 것을 볼 수 있다. 통신량의 증가는 대부분 메모리 읽기 통신량의 증가로 인한 것이다. FFT의 통신량의 증가가 크지 않은 것은 앞에서도 살펴보았듯이 무효화 힌트의 사용빈도가 낮기 때문이다. 또한 무효화 힌트의 사용빈도가 비슷한 Radix와 Barnes를 비교할 때 Radix의 통신량의 증가가 크지 않은 것은 Radix가 메모리 읽기보다 쓰기에서 통신량을 많이 발생시키기 때문이다.

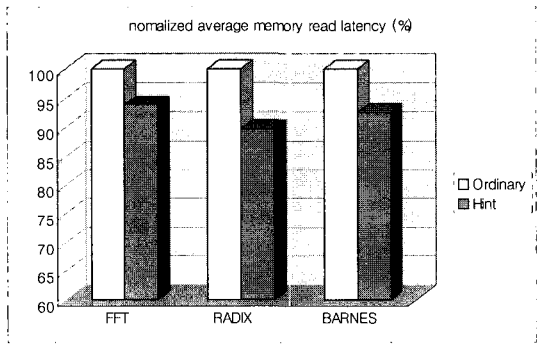

그림 3 메모리 읽기의 평균 지연시간

4.4 메모리 읽기 지연시간

그림3은 무효화 힌트를 사용했을 때 메모리 읽기 지연 시간의 감소를 나타낸다. 무효화 힌트를 사용했을 때 FFT의 지연시간 감소가 가장 작은 것은 앞에서도 살펴보았듯이 FFT에서는 무효화힌트의 사용빈도와 적중률이 낮기 때문이다. 이 두 가지 수치가 비슷한 Radix와 Barnes를 비교해보면 Radix가 10.2%의 읽기 지연시간 감소를 나타냈으며, Barnes는 7.5%의 지연시간 감소를 나타냈다. Barnes의 성능개선이 Radix보다 좋지 않은 이유는 단위시간동안 발생하는 통신량이 Barnes가 많기 때문에 제한된 네트웍 대역폭에서 통신량의 증가가 성능개선의 부담으로 작용했기 때문이다. 다시 말해서 Radix는 전체적인 통신량이 많지 않기 때문에 무효화 힌트의 사용으로 인한 통신량의 증가가 성능에 크게 영향을 미치지 않는데 비해서 Barnes는 통신량이 많은 프로그램이기 때문에 무효화 힌트로 인하여 통신량이 더욱 늘어나면 그로 인한 성능저하가 발생하게 된다.

4.5 메모리 접근에 관한 총 소요 시간의 비교

그림4는 각 프로그램에서 프로그램의 연산 작업에 필요한 시간을 제외한 메모리 접근(읽기와 쓰기)에 걸리는 시간을 나타낸다. 메모리 접근에 소요되는 시간을 나타내는 그래프의 대체적인 모양은 메모리 읽기 지연시간을 나타내는 그래프와 크게 다르지 않다. 다만 메모리 접근 시간의 성능개선은 메모리 읽기 지연 시간의 성능개선보다 떨어지는 것을 볼 수 있다. 그 이유는 무효화 힌트를 사용에 따른 통신량의 증가로 쓰기에 대한 지연시간이 더 커지기 때문이다.

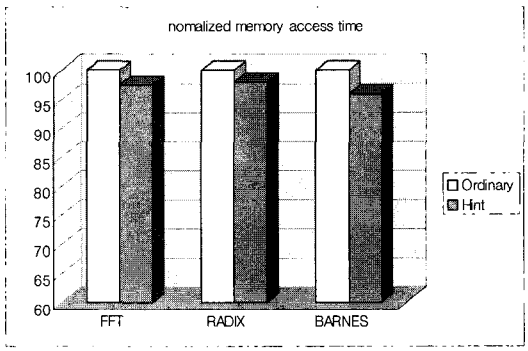

그림 4 메모리 접근의 총 소요시간

5. 결 론

분산 공유 메모리를 가지는 병렬처리 시스템에서 공유메모리에 대한 읽기는 큰 지연시간을 필요로 한다. 또한 이러한 시스템에서는 캐쉬 일관성 유지 프로토콜에 따라 유효한 메모리 블록을 얻기 위해 원격 노드에 요청하는 횟수와 한 번 요청 시에 거쳐가야 하는 노드의 수가 달라지게 되며, 이는 메모리 읽기 지연시간에 영향을 미친다. 따라서 캐쉬 일관성 유지 프로토콜은 메모리 읽기 지연 시간을 줄일 수 있도록 설계되어야 한다.

본 논문에서는 무효화 힌트를 사용하여 메모리 읽기

지연 시간을 줄일 수 있는 캐쉬 일관성 유지 프로토콜을 제안하고 성능을 평가하였다. 무효화 힌트 기법은 캐쉬 일관성 유지를 위하여 자신의 캐쉬가 무효화되었을 때 무효화시킨 노드의 정보를 이용하여 읽기 지연 시간을 줄이는 방법이다. FFT, Radix, Barnes를 이용하여 모의실험을 하였을 때 무효화 힌트를 이용한 프로토콜은 6.2% ~ 10.2% 의 읽기 지연시간 감소를 나타냈으며, 무효화 힌트의 사용빈도와 적중률이 크고 프로그램의 통신량이 작을수록 읽기 지연시간은 크게 줄어든다. 본 논문에서는 사용한 프로그램들은 서로 다른 특성을 지닌 프로그램들로서 이에 따른 성능 측정 결과는 다양한 환경에서 무효화 힌트의 사용이 효율적으로 동작한다는 주장을 뒷받침해 준다. 특히 Barnes는 SPLASH-2 중에서 프로세서간의 통신이 고정된 유형을 갖지 않는 프로그램으로 알려져 있으며[15], Barnes의 실험결과도 좋게 나온 것으로 미루어 볼 때 다른 복잡한 통신 환경에서도 좋은 결과가 나올 것이라고 조심스럽게 예측할 수 있다.

앞으로의 연구과제는 보다 고성능의 네트웍크와 보다 많은 프로세싱 노드를 갖춘 컴퓨팅 환경과 보다 많은 응용프로그램 상에서 무효화 힌트 기법의 성능을 측정하는 것과 무효화 힌트 기법을 실제 환경에서 구현하여 정확한 성능 평가와 함께 실용적인 시스템을 구축하는 것이다.

참 고 문 헌

[1] A.Smith, "Cache Memories," *ACM Computing Surveys*, pp. 473-530, 1982.

[2] L.M.Censier and P.Feautrier, "A New Solution to Cache Problems in Multicache Systems," *IEEE Transactions on Computers*, pp. 1112-1118, Dec. 1978.

[3] R.H.Katz, S.J.Eggers, D.A.Wood, C.L.Perkins, and R.G.Sheldon, "Implementating a Cache Coherence Protocols," in *Proceedings of the International Symposium on Computer Architecture*, pp. 276-283, Jun. 1985.

[4] H.Nilsson and P.Stenstorm, "An Adaptive Update-Based Cache Coherence Protocol for Reduction of Miss Rate and Traffic," in *Proceedings of Parallel Architectures and Languages Europe*, Jun. 1994.

[5] D.D.Corso, M.Kirrman, and J.Nicoud, *Micro-computer Buses and Links*. Academic Press, 1986

[6] D.E.Lenoski, J.P.Laudon, K.Gharachorllo, A Gupta, and J.L.Hennessy, "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor," in *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pp. 148-159, 1990.

[7] A.Agarwal, B.H.Lim, D.Kranz, and J.Kubiatowicz, "APRIL: A Processor Architecture for Multiprocessing," in *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pp. 104-114, May. 1990.

[8] M.Wolf and M.Lam, "A Data Locality Optimizing Algorithm," in *Proceedings of the ACM SIGPLAN '91 Conference on Proceeding Language Design and Implementation*, pp. 30-44, Jun. 1991.

[9] K.Li and P.Hudak, "memory Coherence in Shared Virtual Memory Systems," *ACM Transactions on Computer Systems*, pp. 321-359, Nov. 1989.

[10] A.Gupta, T.Joe, and P.Stenström, "Performance Limitations of Cache-Coherence NUMA and Hierarchical COMA Multiprocessors and the Flat-COMA Solution," Tech. Rep. CSL-TR-92-524, Stanford University, 1992.

[11] J.Kuskin, D.Ofelt, M.Heinrich, J.Heinrich, R.Simoni, K.Gharachorloo, J.Chapin, D.Nakahira, J.Baxter, M.Horowitz, A.Gupta, M.Rosemblum, and J.Hennessy, "The Stanford FLASH Multiprocessor," in *Proceedings of the 21th International Symposium on Computer Architecture*, pp. 302-313, Apr. 1994.

[12] X.Lin, P.K.Mckinley, and L.M.Li, "Deadlock-Free Multicast Wormhole Routing in 2-D Mesh Multiprocessor," *IEEE Trans. on Parallel and Distributed Systems*, pp. 793-804, Aug. 1994.

[13] T.E.Anderson, "The Performance of Spin Lock Alternatives for Shared-Memory Multiprocessors," *IEEE Trans. on Parallel and Distributed Systems*, pp. 6-16, Jan. 1990.

[14] J.E.Veensta and R.J.Fowler tech. rep., Rochester University, Jun. 1993.

[15] S.Cameron, M.Ohara, E.Torrie, J.P.Singh, and A.Gupta, "The SPLASH-2 Programs: Characterization and methodological Considerations," in *Proceedings of the 22th Annual International Symposium on Computer Architecture*, pp. 24-36, Jun. 1995.

오 승 택

1974년 10월 19일생. 1997년 한국과학기술원 전산학과 졸업. 1999년 한국과학기술원 전산학 석사. 1999년 3월 ~ 현재 한국과학기술원 전기전산학과 박사과정. 관심분야는 병렬 컴퓨터 구조, 운영체제 등임.

이 윤 석
1988년 서울대학교 자연과학대학 계산통계학과 이학사. 1995년 한국과학기술원 정보 및 통신공학과 공학석사. 1999년 한국과학기술원 전산학과 공학박사. 1988년 ~ 1994년 시스템공학연구소 연구원. 1999년 ~ 현재 한국외국어대학교 전자제어공학부 조교수. 관심분야는 병렬 컴퓨터 구조, 분산 운영체제 등임.

맹 승 렬
1977년 서울대학교 공과대학 전자공학과 졸업. 1979년 한국과학기술원 전산학과 석사학위 취득. 1984년 한국과학기술원 전산학과 박사학위 취득. 1984년 ~ 현재 한국과학기술원 전산학과 정교수. 1988년 ~ 1989년 펜실바니아대학 교환교수. 1994년 ~ 1995년 텍사스대학 교환교수. 관심분야는 parallel computer architecture, dataflow machines, vision architecture, multimedia임

이 준 원
정보과학회논문지 : 시스템 및 이론
제 27 권 제 1 호 참조