코달링 구조의 CC-NUMA 시스템을 위한 원격 캐쉬 교체 정책

(A Remote Cache Replacement Policy for the Chordal Ring Based CC-NUMA System)

김 수 한 † 김 인 석 † 김 봉 준 † 장 성 태 ††
(Soo-Han Kim) (In-Suk Kim) (Bong-Joon Kim) (Seong-Tae Jhang)

요 약 Chordal Ring 구조의 CC-NUMA 시스템은 그 구조적 특징 때문에 지역 노드와 원격 노드사이의 트랜잭션 전송을 위해 지나가는 링크의 수가 많게 된다. 그러나, 이러한 트랜잭션이 코달 링보다 링 링크로 몰리는 경향은 링 링크의 트래픽 증가와 응답 지연 시간의 증가를 유발하게 되면서 Chordal Ring 구조의 CC-NUMA 시스템의 성능을 하락시킨다. 이러한 문제를 극복하기 위해서 본 논문에서는 원격 캐쉬를 교체할 경우에 지역 노드와 원격 노드사이에 데이타 전송을 위하여 지나가는 총 링크 수와 링 링크의 수를 고려한 새로운 원격 캐쉬 교체 정책을 제안한다. 본 논문에서 제안하는 원격 캐쉬 교체 정책은 Chordal Ring 구조의 CC-NUMA 시스템의 특징을 반영하였기 때문에 링크 간의 데이타를 적절히 분산시킬 수 있는 정책이라 사료된다.

키워드 : 다중 프로세서, 코달 링, CC-NUMA, 캐쉬 교체 정책

Abstract The chordal ring based CC-NUMA system contains many links to transmit transactions between a local node and a remote node because of its structural characteristics. However, the inclination that the transactions concentrate on the ring link increases both the traffic of the ring link and the response time, which degrades the overall performance of the chordal ring based CC-NUMA system. In this paper we suggest a new remote cache replacement policy that considers both the number of total links and the number of ring links to traverse for the transactions. Our proposed replacement policy can balance data between the ring link and the chordal link properly because it reflects the characteristics of chordal ring based CC-NUMA system well.

Key words : multiprocessor, chordal ring, CC-NUMA, cache replacement policy

1. 서 론

최근 대용량 네트워크 전송 기술의 발달은 링을 이용하여 데이타를 전송, 처리하는 다중 프로세서 시스템의 처리속도를 향상시켰다. 이러한 대용량 다중 프로세서 시스템은 프로세서가 공유 메모리를 접근하는 방법에 따라 크게 UMA(Uniform Memory Access)와 NUMA (Non-Uniform Memory Access)로 나눌 수 있다[1,2]. UMA 방식의 시스템은 각각의 다른 프로세서가 공유메모리를 접근하는 경로가 모두 동일한 경우를 말하는데, 이러한 방식을 갖는 다중 프로세서 시스템은 확장성이 좋지 않기 때문에 비교적 작은 규모의 다중 프로세서 시스템에서 많이 사용된다. 반면에 NUMA 방식의 다중 프로세서 시스템은 각각의 다른 프로세가 공유메모리를 접근하는 경로가 일정하지 않는 경우를 말한다.

서울대학교에서 개발된 PANDA 시스템은 하나의 트랜잭션을 여러 개의 패킷으로 나누어 전송하는 레지스터 삽입방식을 사용하는 CC-NUMA(Cache Coherent NUMA) 시스템이다[3]. 이중 링 구조의 PANDA II[4]는 500MByte/sec. 대역폭의 링크를 이중으로 구성하되 서로 회전 방향을 다르게 하여 대역폭 증가와 지연 시간 감소를 가지도록 한 구조이다. 이러한 이중 링 구조의 PANDA II 시스템은 트랜잭션을 회전 방향이 반대

· 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2002-041-D00443)
† 비 회 원 : 수원대학교 컴퓨터학과
gruf97@hanmail.net
saintboy@empal.com
dark_engel@hanmail.net
†† 종신회원 : 수원대학교 컴퓨터학과 교수
stjhang@suwon.ac.kr
논문접수 : 2003년 12월 26일
심사완료 : 2004년 7월 29일

인 2개의 링크를 이용하여 분산시킬 수 있는 장점이 있지만 원격 노드 개수가 증가하면 단일 전송 패킷의 응답 지연 시간이 증가된다는 문제가 있다[10].

이러한 문제를 해결하기 위해서 지점 간 링크를 이용하는 이중 링 구조의 CC-NUMA 시스템에 코달 링(Chordal Ring) 네트워크 토폴로지를 적용한 코달 링 구조의 CC-NUMA 시스템이 제안되었다[5,6]. 지점 간 링크를 이용하는 코달 링 구조의 CC-NUMA 시스템은 원격 노드를 모두 연결하고 있는 링 링크와 Skip Distance에 따라 건너뛰면서 원격 노드를 연결하고 있는 코달 링크가 존재한다. 이러한 코달 링 구조의 CC-NUMA 시스템은 Skip Distance에 따라 건너뛰면서 원격 노드를 연결하고 있는 코달 링크를 이용함으로써 단일 전송 패킷의 응답 지연 시간을 감소시키고자 제안되었다.

하지만, 제시된 코달 링 구조를 갖는 CC-NUMA 시스템[5,6]은 이중 링과 달리 한 개의 링 링크만이 모든 노드를 연결하기 때문에 링크 이용률 측면에서 효율적이지 못하다. 즉, 코달 링 구조의 CC-NUMA 시스템의 모든 원격 노드를 연결하고 있는 링 링크의 트래픽이 상대적으로 코달 링크의 트래픽보다 많이 발생하는 경향을 가진 구조이다. 이러한 링 링크로 트래픽이 몰리는 경향은 2장에서 서술한 바와 같이 코달 링 구조의 CC-NUMA 시스템에 있어서 중요한 성능하락 요인이 될 수 있다. NUMA 방식의 경우에는 또한 메모리에 접근하는 경로가 일정하지 않기 때문에 메모리 접근 시간 또한 일정하지 않으며, 노드 사이의 물리적 거리에 비례해서 메모리 접근 시간이 다르게 된다. 따라서 CC-NUMA 시스템에서는 원격 노드에서 현재 노드로 가져오는데 시간이 더 많이 걸리는 정보를 그렇지 못한 정보보다 더 오랫동안 원격 캐쉬에 유지하여 사용하는 것이 전체적인 성능 향상의 요건이 될 수 있다. 즉, 원격 캐쉬 교체시 접근 시간이 더 오래 걸리는 노드의 정보를 교체시키는 대신 접근 시간이 더 적게 걸리는 노드의 정보를 교체시켜서 차후 그 원격 노드에서 현재 노드로 가져올 때 걸리는 시간을 줄이는 것이 CC-NUMA 시스템의 전체적인 성능 향상에 기여할 수 있다. 본 논문에서는 이러한 구조적인 특징들을 고려하여 CC-NUMA 시스템, 특히 코달 링 구조의 CC-NUMA 시스템을 위한 새로운 원격 캐쉬 교체 정책을 제시하고자 한다.

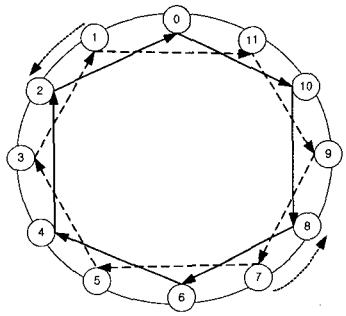
본 논문의 구성은 다음과 같다. 2장에서 기존의 코달 링 구조의 CC-NUMA 시스템의 구조와 문제점을 분석을 할 예정이고, 3장에서는 본 논문에서 제안하는 원격 캐쉬 교체 정책을 서술한다. 4장에서는 Augmint[7,8]를 이용한 SPLASH II[9] 벤치마크 프로그램의 시뮬레이션을 통해 기존의 코달 링 구조의 CC-NUMA 시스템을 기준으로 본 논문에서 새롭게 제안한 원격 캐쉬 교체 정책을 적용한 코달 링 구조의 CC-NUMA 시스템의 성능을 비교할 예정이며, 5장에서는 이러한 성능평가에 근거하여 결론을 맺을 예정이다.

2. 기존 시스템의 분석

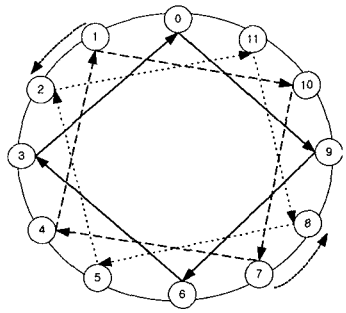
2.1 코달 링 구조의 CC-NUMA 시스템

PANDA와 PANDA II 시스템은 스누핑 방식으로 캐쉬 일관성을 유지[3,4]하고, 방송 트랜잭션을 지원하므로 각 노드를 연결하는 지점간 링크로 구성된 링은 논리적으로 버스와 동일한 동작을 수행하게 된다. 코달 링 구조의 CC-NUMA 시스템[5,6]도 동일하게 스누핑 방식으로 캐쉬 일관성을 유지하며, 지점간 링크로 구성된 방향 분리 코달 링의 상호 연결망을 가지는 시스템이다.

코달 링 구조의 CC-NUMA 시스템에서 각 노드에는 최대 4개의 프로세서 모듈, 지역 메모리, I/O 제어기와 노드 제어기가 지역 버스에 연결되어 있다. 노드의 지역 버스는 스누핑 캐쉬 일관성 프로토콜[11]에 의해서 동작하며, 요청 트랜잭션과 응답 트랜잭션을 분리하는 split 버스로 구성되어 있다. 12개의 노드로 구성된 코달 링의 Skip Distance에 따른 구조의 예는 그림 1에 도시되어 있다.



(a) Skip Distance가 2인 Chordal Ring (Chordal Ring 2)



(b) Skip Distance가 3인 Chordal Ring (Chordal Ring 3)

그림 1 노드 수가 12인 시스템 구성도

그림에서 원으로 표시한 부분은 최대 4개의 프로세서 모듈과 지역 메모리, I/O 제어기와 노드 제어기를 가지는 하나의 노드를 의미하며 바깥쪽의 링크가 코달 링 구조 CC-NUMA 시스템의 모든 원격 노드를 연결하고 있는 링 링크, 안쪽의 링크가 원격 노드를 Skip Distance에 따라 건너뛰면서 연결하고 있는 코달 링크이다.

그림 2는 한 노드 구조를 나타내고 있다. 각 노드에는 최대 4개까지의 프로세서 모듈과 지역 메모리, I/O 제어기(MIOC), 노드 제어기(node controller)가 지역 버스에 연결되고 있다. 지역 메모리는 분산형 공유 메모리로서 전체 시스템 메모리 주소 영역의 일부를 구성하고 있다. I/O 제어기는 프로세서에서 요구한 I/O 장치에 대한 읽기 및 쓰기를 수행한다. 모든 I/O 관련 트랜잭션은 지역 버스에 연결된 PCI 버스를 통해 처리하게 된다.

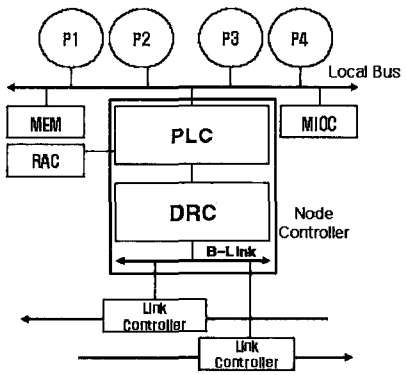


그림 2 노드 구조

노드 제어기는 원격 노드로의 접근 지연 시간을 단축하기 위한 원격 캐쉬(RAC : Remote Access Cache)를 유지하며 지역 버스에서 발생한 요청에 대해 RAC로부터 데이타 제공, 전역 링크를 통한 원격 노드로부터의 데이타 제공, 전역 링크에서 발생한 요청에 대해 응답 책임을 진다. 이와 동시에 노드 제어기는 지역 버스 및 전역 링크에서 발생하는 모든 트랜잭션을 스누핑하여 메모리-캐쉬 일관성 유지를 위해 필요한 제어를 수행한다. 코달 링 구조의 CC-NUMA에서 사용하고 있는 트랜잭션과 캐쉬 일관성 유지 프로토콜은 기존의 PANDA II 시스템에서 사용된 트랜잭션과 캐쉬 일관성 프로토콜[4]을 그대로 적용할 수 있다.

2.2 기존 코달 링 구조 CC-NUMA 시스템의 문제점

그림 3과 같은 이중 링 구조의 PANDA II 시스템과 달리 그림 1과 같은 코달 링 구조의 CC-NUMA 시스템[5,6]은 원격 노드 모두를 연결하고 있는 링 링크가 단방향인 1개 밖에 존재하지 않기 때문에 트랜잭션이 링 링크로 몰리는 경향이 있다. 이러한 경향은 링 링크의 이용률 증가로 병목현상(bottleneck)을 일으키게 되어 코달 링 구조 CC-NUMA 시스템의 전체 성능을 하락시킬 수 있다. 그러므로 이러한 링 링크의 트랜잭션의 일부를 코달 링크로 적절히 분산시킨다면 코달 링 구조 CC-NUMA 시스템 전체의 성능향상을 가져올 수 있을 것이다.

하지만, 기존의 코달 링 구조 CC-NUMA 시스템[5,6]의 원격 캐쉬 교체 정책(replacement policy)은 NUMA 시스템 구조의 특성과 코달 링 구조의 특성을 반영하고

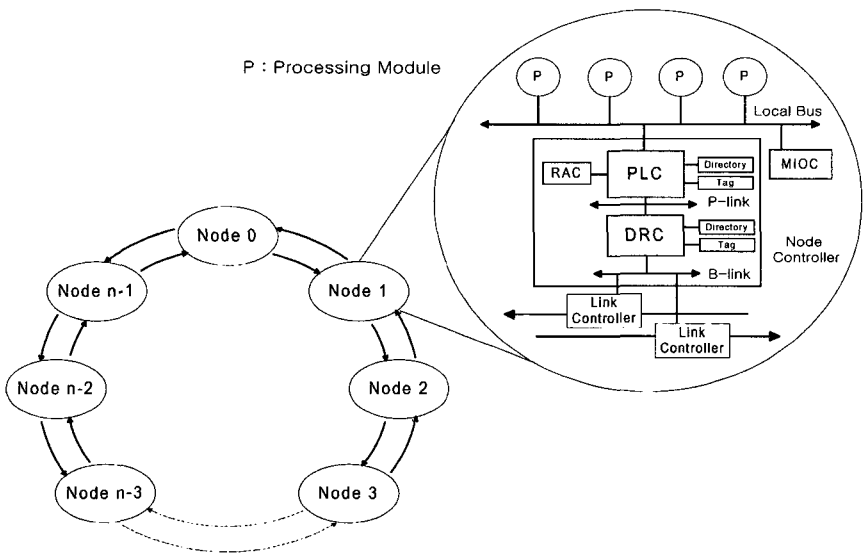


그림 3 이중 링 구조 PANDA II 시스템

있지 않다. 기존에 널리 사용되고 있는 원격 캐쉬 교체 정책인 LRU와 LFU, 임의 교체 정책들은[1,2,11] 지역 프로세서가 원격 캐쉬의 캐쉬 라인을 접근하는 패턴만을 사용하여 교체할 캐쉬 라인을 결정하는 방법이다. 따라서 원격 캐쉬의 캐쉬 라인을 교체할 때마다 캐쉬 라인 교체 비용이 달라지며, 이로 인해 교체된 캐쉬 라인에 다시 접근할 때 발생하는 접근 비용이 달라지는 NUMA 시스템의 구조적 특성을 고려하지 않는 방법이라고 할 수 있다. 이러한 캐쉬 교체 정책들은 NUMA 방식의 대규모 다중 프로세서 구조보다는 UMA 방식의 소규모 다중 프로세서 구조에 적합한 원격 캐쉬 교체 정책이고 UMA 구조에서 개발된 정책들이다. 상술한 바와 같이 CC-NUMA 시스템은 노드들 간의 거리에 따라 메모리를 접근하는 비용이 다르기 때문에 단순히 프로세서가 원격 캐쉬의 캐쉬 라인을 접근하는 패턴만을 가지고 다음의 교체할 캐쉬 라인을 결정하는 방법은 적절치 않을 수 있다. 따라서 링 링크의 트랜잭션을 코달 링크로 적절히 분산시켜서 링 링크와 코달 링크 사이의 부하 균형을 이루는 동시에 CC-NUMA 시스템의 구조적 특징인 노드들 간의 거리에 따른 메모리 접근하는 비용을 같이 줄일 수 있는 원격 캐쉬 교체 정책이 필요할 것이다. 이를 위해 코달 링 구조의 CC-NUMA 시스템에서 코달 링크 이용률에 비해 링 링크 이용률이 더 큰 이유를 예를 들어 분석하면 다음과 같다.

(1) 방송용 데이타 요청 패킷의 경우

코달 링 구조의 CC-NUMA 시스템에서 방송용 데이타 요청 패킷은 스누핑을 위해 코달 링에 있는 모든 노드로 보내는 패킷이기 때문에 링 링크를 이용하여 전송하며 이 요청에 대해 각 노드는 스누핑을 수행한다. 그림 3과 같은 이중 링 구조의 PANDA II 시스템과 달리 코달 링 구조의 CC-NUMA 시스템에서는 그림 1과 같이 노드를 모두 연결하고 있는 링 링크가 단방향이기 때문에 이 링 링크를 이용하여 방송용 데이타 요청 패킷을 전송해야 하며, 이로 인해 링 링크의 이용률이 코달 링에 비해 상대적으로 증가되는 경향이 있다.

(2) 단일 전송 패킷 전송 시, 링 링크를 이용하는 것이 최소의 링크를 형성하는 경우

코달 링 구조의 CC-NUMA 시스템에서 현재 노드에서 목적지 원격 노드로 메모리 블록을 포함하는 단일 전송 패킷(예를 들어 write-back 트랜잭션을 위한 패킷)을 전송할 경우 가장 최소의 링크를 지나가는 길을 이용하여 전송한다. 따라서 현재 노드에서 목적지 원격 노드로 단일 전송 패킷을 전송할 때, 링 링크로 지나가는 쪽이 가장 최소의 링크를 형성한다면 단일 전송 패킷은 링 링크를 이용해서 전송을 하게 된다. 그림 4의 예에서 큰 원에 있는 노드들은 노드 0번에서 시작되는 단일 전송 패킷을 받을 때, 링 링크만을 이용해서 단일 전송 패킷을 받는 목적지 원격 노드들을 나타낸다. 그림 4에서 노드 0번에서 노드 2번으로 단일 전송 패킷을 주기 위해서는 코달 링크를 이용하여 지나가는 것 보다 링 링크를 사용하는 것이 최소의 링크를 지나가게 되므로 이런 경우에는 코달 링크를 이용하지 않고 링 링크만을 이용해서 단일 전송 패킷을 주게 된다. 그림 4에서 작은 원은 노드 0번에서 시작되는 단일 전송 패킷을 받을 때, 지나가는 링 링크의 수와 코달 링크의 수가 동일한 경우를 나타낸다.

(3) 단일 전송 패킷을 전송할 때, 링 링크와 코달 링크를 동시에 사용하는 경우

현재 노드에서 목적지 원격 노드로 메모리 블록을 포함하는 단일 전송 패킷을 전송할 때, 링 링크와 코달 링크 둘 다 이용해야 전송이 가능한 목적지 원격 노드들이 존재한다. 이때, 링 링크와 코달 링크를 동시에 사용하는 경우는 전체적인 링크 이용률을 증가시키고 부가적으로 링 링크 이용률 또한 같이 증가하게 된다.

그림 5에서 0번 노드에서 11번 목적지 원격 노드로 메모리 블록을 포함하는 단일 전송 패킷을 전송하는 예

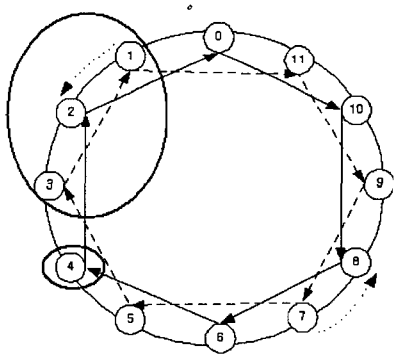


그림 4 링 링크를 이용하는 데이타 전송의 예

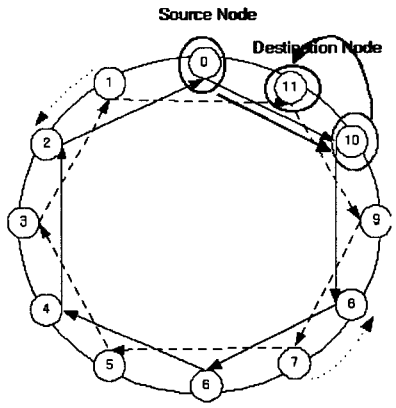


그림 5 2개의 링크를 지나가는 데이타 전송의 예

를 들어보자. 노드 0번에서 노드 11번으로의 단일 전송 패킷 전송을 하기 위해서는 노드 0번에서 노드 10번을 거친 후 노드 11번으로 전송해야 최소 링크를 이용하게 된다. 이 때, 노드 0번에서 나온 단일 전송 패킷을 노드 10번에서 받았다면 노드 10번에서는 그 패킷을 노드 11번으로 전송하기 위하여 코달 링크에서 링 링크로 단일 전송 패킷을 옮겨야 한다. 이러한 작업은 그림 2의 노드 구조에서 2개의 링크 컨트롤러와 B-Link를 이용해서 이루어질 수 있으며 이 경우 두개의 링크의 이용률 증가와 함께 링크를 옮기기 위한 전송 지연이 발생하게 된다.

3. 코달 링 구조 CC-NUMA 시스템을 위한 새로운 원격 캐쉬 교체 정책

3.1 새로운 원격 캐쉬 교체 정책

CC-NUMA 시스템의 경우 접근 시간이 적게 걸리는 원격 노드의 정보를 교체시키고 접근 시간이 더 오래 걸리는 원격 노드의 정보를 캐쉬 내에 더 많이 유지하여 차후 원격 노드에서 가져올 때 걸리는 시간을 줄이는 원격 캐쉬 교체 정책이 전체적인 성능 향상에 기여할 수 있다. 또한 본 논문에서 목표로 하는 코달 링 구조의 CC-NUMA 시스템은 원격 노드 모두를 연결하고 있는 단 방향 링 링크가 1개 밖에 존재하지 않는다. 이러한 코달 링 구조의 CC-NUMA 시스템의 구조적인 특성은 상술한 바와 같이 모든 원격 노드를 연결하고 있는 단방향인 링 링크의 이용률을 보다 많이 증가시킴으로서 시스템에 전체적인 성능을 하락시킬 가능성이 있다.

하지만 기존의 CC-NUMA 시스템들은 이러한 CC-NUMA의 구조적인 특성을 반영하지 않고 원격 캐쉬 교체 정책으로 LRU와 LFU, 임의 교체등을 사용한다. LRU와 LFU, 임의 교체 정책은 프로세서가 원격 캐쉬의 캐쉬 라인을 접근하는 패턴을 사용하여 교체할 캐쉬 라인을 결정하는 방법이다. 따라서 노드들 간의 거리에 따라 메모리를 접근하는 비용이 다른 CC-NUMA 시스템에서는 프로세서가 원격 캐쉬의 캐쉬 라인을 접근하는 패턴만을 가지고 다음의 교체할 캐쉬 라인을 결정하는 방법은 적절치 않을 수 있다. 위에서 언급한 바와 같이 링 링크의 트랜잭션을 코달 링크로 적절히 분산시키면서 CC-NUMA 시스템의 구조적 특징인 노드들 간의 거리에 따른 메모리 접근하는 비용을 같이 줄일 수 있는 원격 캐쉬 교체 정책이 필요하며, 본 논문에서 이러한 정책을 제시하고자 한다. 본 논문에서 제시하고 있는 원격 캐쉬 교체 정책의 알고리즘을 요약하면 다음과 같다.

(1) 원격 캐쉬에 존재하는 특정 Set의 캐쉬 라인의 상태가 전부 Modified일 경우

(a) 원격 캐쉬 교체 정책이 실행되는 지역 노드(이하 현재 노드로 칭함)에서 목적지 원격 노드로 축출되는 메모리 블록을 포함하는 단일 전송 패킷을 전송하고 차후 현재 노드의 원격 캐쉬 미스로 인해 그 원격 노드로부터 축출된 메모리 블록을 다시 전송받기 위해 지나가야 하는 총 링크의 수가 최대인 원격 노드의 메모리 블록을 가급적 교체하지 않고 오랫동안 원격 캐쉬에 남겨둔다. 즉, 그림 5의 노드 0번의 원격 캐쉬의 동일 set에 노드 2번의 메모리 블록과 노드 9번의 메모리 블록이 존재할 경우에 대한 예가 그림 6에 나타나 있다. 노드 0번의 원격 캐쉬에 노드 10에 저장되어 있는 새로운 메모리 블록을 넣기 위해서 캐쉬 라인 교체를 실행한다면 그림 6과 같이 총 링크의 수가 6인 노드 9번의 데이타보다는 총 링크의 수가 3으로 비교적 짧은 노드 2번의 데이타를 우선적으로 교체하는 것이 링크 트래픽과 차후 원격 캐쉬 미스 시 메모리 접근 시간을 줄이는데 기여할 것이다.

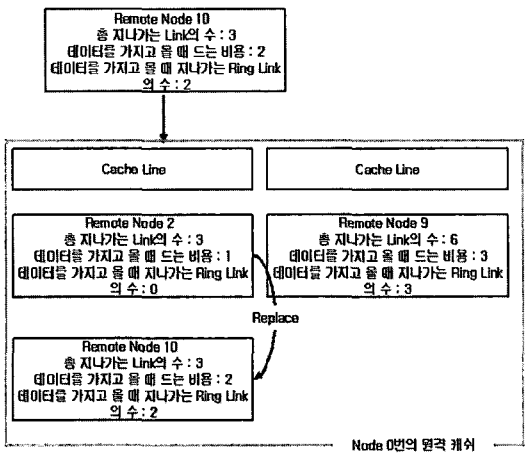


그림 6 새로운 원격 캐쉬 교체 정책의 예

(b) 코달 링 구조의 CC-NUMA 시스템은 그 구조적 특징 때문에 현재 노드와 원격 노드들 사이의 단일 전송 패킷을 주고받을 때 지나가는 총 링크의 수가 동일한 경우가 많다. 이러한 경우에는 메모리 접근 지연 시간을 줄이는데 초점을 맞춰 차후 축출된 메모리 블록에 다시 접근하기 위해 지나가는 링크의 수가 최대인 원격 노드의 메모리 블록을 더 오랫동안 원격 캐쉬에 남겨둔다.

그림 7은 노드 0번을 기준으로 할 때, 축출된 메모리 블록을 포함하는 단일 전송 패킷을 전송하고 차후 그

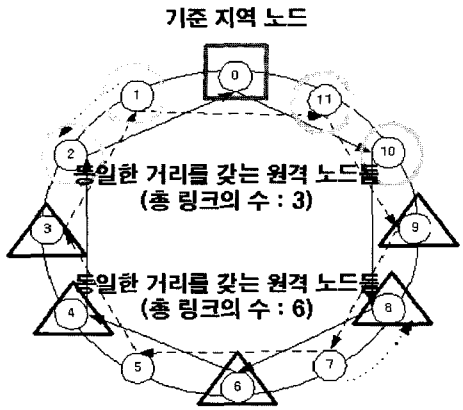


그림 7 총 링크수가 동일한 경우의 예

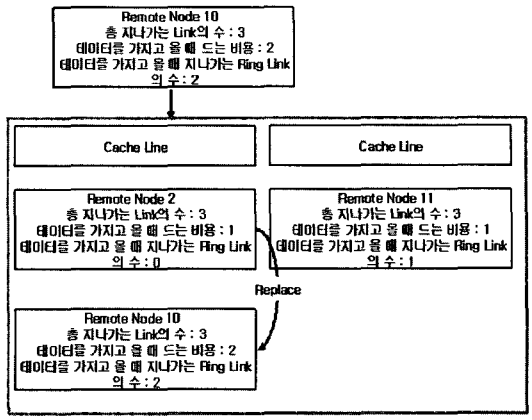


그림 8 새로운 원격 캐쉬 교체 정책의 예
(총 링크의 수가 같을 때)

메모리 블록을 포함하는 단일 전송 패킷을 다시 받기 위해서 거쳐야하는 링크의 수가 동일한 경우를 나타낸 것이다. 현재 노드인 노드 0번의 원격 캐쉬 입장에서 본다면 원으로 표시되어 있는 노드들은 요청한 메모리 블록을 포함하고 있는 단일 전송 패킷을 주고받기 위해서 지나가는 링크의 수가 3으로 동일하다. 노드 0번의 원격 캐쉬 입장에서 본다면 세모로 표시되어 있는 노드들은 단일 전송 패킷을 주고받기 위해서 지나가는 총 링크의 수가 6으로 동일한 노드들을 표시한 것이다. 이러한 노드들의 데이타가 현재 노드인 노드 0번의 원격 캐쉬의 동일 set, 다른 캐쉬 라인에 존재한다면 데이타를 가지고 오기 위해서 지나가는 링크의 수가 적은 것을 우선적으로 교체하는 것이 메모리 접근 지연 시간을 줄이는데 기여할 수 있을 것이다.

(c) 만약 (b)의 경우도 동일하다면 링 링크 트래픽을 최소화하는데 초점을 맞춰 축출된 메모리 블록을 포함하는 단일 전송 패킷을 전송하고 차후 그 메모리 블록을 포함하는 단일 전송 패킷을 다시 받기 위해 지나가는 링 링크의 수가 최대인 원격 노드의 메모리 블록을 더 오랫동안 원격 캐쉬에 남겨둔다. 그림 8에서는 현재 노드인 노드 0번의 원격 캐쉬에 총 링크의 수가 3으로 동일하며, 축출된 메모리 블록 전송하고 차후 그 메모리 블록에 접근하기 위해서 지나가는 링크의 수가 동일한 노드 2번의 메모리 블록과 노드 11번의 메모리 블록이 동일 set, 다른 캐쉬 라인에 존재할 경우의 예를 보여주고 있다.

위의 그림 8은 노드 0번의 원격 캐쉬에 노드 2번의 메모리 블록과 노드 11번의 메모리 블록을 저장하고 있을 경우, 새롭게 들어오는 노드 10번의 메모리 블록을 위해서 어떤 라인의 데이타를 우선적으로 교체할지에 대한 것을 나타내고 있다. 노드 2번의 메모리 블록과 노드 11번의 메모리 블록은 노드 0번을 기준으로 하였을 때, 총 링크의 수와 메모리 블록을 가지고 올 때 지나가는 링크의 수가 동일하기 때문에 메모리 블록을 현재 노드인 노드 0번에 가지고 오기 위해서 지나가는 링 링크의 수로 결정해야 한다. 따라서 먼저 노드 2번의 메모리 블록이 우선적으로 교체되어야 한다.

(2) Shared인 캐쉬 라인이 하나 존재하고 나머지 캐쉬 라인의 상태가 Modified일 경우

이 경우에는 본 논문에서 제시한 원격 캐쉬 교체 정책을 사용하지 않고, Shared인 캐쉬 라인을 교체하게 된다. Shared인 캐쉬 라인은 다른 원격 노드에 유효한 데이타가 존재하기 때문에 그 블록을 원격 노드로 축출할 필요가 없기 때문이다[12].

(3) Shared인 캐쉬 라인이 2개 이상이 존재할 경우

(a) 이러한 경우에는 우선 메모리 접근 지연 시간을 줄이는데 초점을 맞춰 차후 그 메모리 블록을 원격 노드로부터 현재 노드로 가져올 때 지나가는 링크의 수가 최대인 원격 노드의 메모리 블록을 더 오랫동안 원격 캐쉬에 남겨둔다.

(b) 만약 (a)의 경우가 동일하다면 링 링크 트래픽을 최소화하는데 초점을 맞춰 차후 그 메모리 블록을 원격 노드로부터 현재 노드로 전송하기 위해 지나가는 링 링크의 수가 최대인 원격 노드의 메모리 블록을 더 오랫동안 원격 캐쉬에 남겨둔다.

상술한 바와 같이 본 논문에서 새롭게 제안하는 원격 캐쉬 교체 정책의 핵심은 CC-NUMA 시스템의 특성을 원격 캐쉬 교체 정책에 반영하여 현재 노드와 원격 노드 간의 데이타를 주고받을 때 지나가는 총 링크의 수를 우선적으로 고려를 한다. 또한 코달 링 구조의 CC-NUMA 시스템은 현재 노드와 원격 노드 간의 데이타를 주고받기 위해서 지나가는 총 링크의 수가 동일한 원격 노드가 많다는 특징이 있는데, 이러한 구조적

특징을 원격 캐쉬 교체 정책에 반영하기 위해서는 또 다른 기준이 마련되어야 한다. 즉 현재 노드와 원격 노드 간의 데이타를 주고받기 위해서 지나가는 총 링크의 수가 동일할 경우에는 현재 노드에서 요청한 데이타를 원격 노드에서 현재 노드로 전송할 때 지나가는 총 링크의 수가 최소인 것을 먼저 교체한다. 지역 노드에서 요청한 데이타를 원격 노드에서 현재 노드로 전송할 때 지나가는 총 링크의 수를 고려하는 이유는 현재 노드의 프로세서에서 요청한 데이타를 받기 전까지 해당 프로세서는 다음 작업을 할 수 없기 때문이다. 따라서 현재 노드에서 요청한 데이타를 원격 노드에서 현재 노드로 전송할 때는 데이타 전송이 신속하게 이루어져야 하며, 이것이 원격 캐쉬 교체 정책의 교체할 캐쉬 라인을 결정할 때 중요한 요소가 될 수 있다.

3.2 제시한 원격 캐쉬 교체 정책의 구현

본 논문에서 제안하는 새로운 원격 캐쉬 교체 정책을 지원하기 위해서는 그림 2의 노드 제어기에 원격 노드에 따라 데이타를 주고받기 위해서 지나가는 링크의 개수에 대한 정보를 제공하는 링크 수 결정자를 더 유지해야 한다. 즉 본 논문에서 제안하고 있는 원격 캐쉬 교체 정책으로 교체할 캐쉬 라인을 결정하기 위해서는 현재 노드와 목적지 원격 노드 간의 데이타를 주고받기 위해서 지나가는 링크 개수를 알아야 하는데 링크 수 결정자를 사용하지 않는 경우에는 부가적으로 링크의 수를 카운트할 수 있는 하드웨어를 장착해야한다. 링크 수 결정자는 캐쉬 라인의 태그로부터 필요한 정보를 받아서 해당 캐쉬 라인에 저장된 메모리 블록을 저장하고 있는 원격 노드의 정보를 파악하며, 교체할 캐쉬 라인의 결정에 필요한 현재 노드와 해당 목적 노드사이에 단일 전송 패킷을 주고받기 위해서 지나가는 링크의 개수에 대한 정보를 제공한다.

본 논문에서 제시하고 있는 원격 캐쉬 교체 정책은 현재 노드와 목적지 원격 노드들 간에 단일 전송 패킷을 주고받기 위해서 지나가는 총 링크의 수와 원격 노드에서 현재 노드로 단일 전송 패킷을 주기 위해서 지나가는 링크의 수 등을 고려해서 원격 캐쉬에 있는 캐쉬 라인을 교체하는 정책이다. 이러한 정책은 현재 노드와 원격 노드들 사이에 단일 전송 패킷을 주고받기 위해서 지나가는 링크의 트래픽을 효율적으로 분배할 수 있지만, 현재 노드와 원격 노드들 사이에 단일 전송 패킷을 주고받기 위해서 지나가는 총 링크의 수가 최대인 노드의 데이타는 계속해서 캐쉬 라인을 할당받게 되는 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해서 그림 9와 같이 각 캐쉬 라인마다 에이징 비트를 둔다. 에이징 비트는 지역 노드와 원격 노드 사이에 단일 전송 패킷을 주고받기 위해서 지나가는 총 링크의 수가 최대인 노드의 데이타가 원격 캐쉬에 계속해서 할당되는 것을 방지해 주는 방법이다. 에이징 비트는 일종의 포화 카운터 비트로 동일 set의 다른 캐쉬 라인이 교체될 때마다 하나씩 증가하게 된다. 만약 에이징 비트에 있는 값이 최대가 된다면 그 캐쉬 라인의 데이타는 오랫동안 교체되지 않고 캐쉬에 머물러 있었다는 의미이기 때문에 지나가는 총 링크의 수와 관계없이 교체하게 된다.

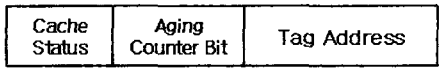


그림 9 원격 태그 캐쉬 상태 정보

만약 현재 노드내의 지역 프로세서가 원격 캐쉬에 존재하는 캐쉬 라인에 접근하게 되면 해당 데이타의 캐쉬 라인의 에이징 비트의 값을 클리어시켜 최근에 노드내의 프로세서에서 접근한 캐쉬 라인을 보다 더 오랫동안 원격 캐쉬에 존재할 수 있게 한다. 이러한 점은 LRU 캐쉬 교체 정책의 기능도 동시에 수행하여 원격 캐쉬를 보다 효율적으로 사용할 수 있도록 해 준다. 에이징 카운터 비트는 캐쉬 라인의 상태가 Invalid 이외의 상태라면 해당 캐쉬 라인의 상태와 관계없이 계속해서 갱신된다. 이는 지역 프로세서에서 접근하지 않고 오랫동안 원격 캐쉬에 저장되어 있는 캐쉬 라인을 효율적으로 축출하기 위해서이다.

동일 set의 하나의 캐쉬 라인의 상태가 Shared이고 나머지 캐쉬 라인의 상태는 Modified인 경우 원격 캐쉬의 상태가 Shared이고 가장 최근에 접근을 하였더라도 앞서 서술한 방법에 따르면 캐쉬 라인의 상태가 Shared인 캐쉬 라인을 계속해서 축출하게 된다. 그 이유는 캐쉬 라인의 상태가 Shared인 메모리 블록은 다른 노드에 유효한 데이타가 존재하기 때문에 축출이 될지라도 메모리 블록의 전송을 동반하지 않기 때문이다[12]. 이러한 가장 최근에 지역 프로세서에 접근한 메모리 블록을 가지는 캐쉬 라인을 축출하는 것은 캐쉬 지역성(Cache Locality)측면에서 볼 때 좋지 않다. 이러한 문제를 해결하기 위해서는 에이징 카운터 비트가 0인 캐쉬 라인(지역 프로세서에서 가장 최근에 접근한 메모리 블록)의 상태가 Shared인 경우 해당 캐쉬 라인을 축출하지 않고 다른 캐쉬 라인에서 축출될 캐쉬 라인을 선택하게 된다. 위와 같이 수정된 원격 캐쉬 교체 정책을 적용한다면, 노드안의 지역 프로세서에서 가장 최근에 접근한 캐쉬 라인을 효율적으로 유지할 수 있기 때문에 가장 최근에 접근한 캐쉬 라인의 축출로 인한 캐쉬 미스를 좀 더 줄일 수 있을 것이다. 또한 원격 캐쉬의 동일 set에 2개 이상의 캐쉬 라인이 Shared일 경우, 본 논문에서 제시

한 원격 캐쉬 교체 정책과 에이징 카운터 비트의 값을 이용해서 다음에 축출될 캐쉬 라인을 결정하게 된다. 만약 Shared인 캐쉬 라인 중 에이징 카운터 비트의 값이 최대가 되는 캐쉬 라인이 존재한다면 해당 캐쉬 라인을 제거하게 된다. 바로 위에서 언급을 하였듯이 캐쉬 라인의 상태가 Shared라면 다른 노드에 유효한 데이타가 존재할 가능성이 높기 때문에 축출 시 메모리 블록의 전송을 동반하지 않기 때문에 전체적인 시스템 성능에 도움이 될 수 있을 것이다[12].

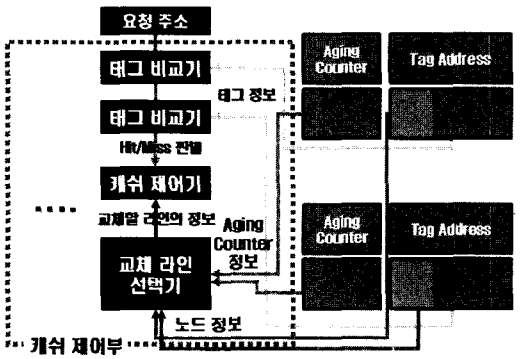


그림 10 본 논문에서 제시하고 있는 원격 캐쉬 교체 정책을 지원하기 위한 캐쉬 제어부의 실시 예

그림 10은 2-way Set Associative 캐쉬 구조에 대한 본 논문에서 제시하고 있는 원격 캐쉬 교체 정책을 구현하기 위해 필요한 구체적인 하드웨어 구조의 실시 예를 나타내고 있다. LRU 원격 캐쉬 교체 정책과 다른 부분은 태그 비교기 이외에 각각의 원격 노드와 관계되어 있는 링크의 정보를 저장하고 있는 링크 수 결정자가 존재하는 것과 함께 원격 캐쉬의 각 캐쉬 라인에 하나의 노드 데이타가 계속해서 할당되는 것을 막기 위한 그림 9에서와 같은 에이징 카운터 비트가 존재하는 것이다. 그림 10에서 교체 라인 선택기는 선택된 set의 캐쉬 라인으로부터의 노드 정보를 이용하여 링크 수에 대한 정보를 제공하고 상술한 링크 수 결정자를 내포하고 있으며 링크 수 결정자로부터의 정보와 에이징 카운터 정보를 이용하여 교체할 캐쉬 라인에 대한 정보를 제공한다.

4. 성능 분석

4.1 모의 실험 환경 및 실험 인자

본 논문에서 제안한 새로운 원격 캐쉬 교체 정책을 평가하기 위해서 모의실험을 실행하였다. 모의실험에는 Execution-Driven 방식의 시뮬레이터인 Augmint[7,8]을 사용하였다. 모의실험 환경에서 각 노드는 2GHz 클록의 CPU를 가정했으며, 이 CPU가 100MHz의 지역 버스에 연결되어서 동작을 하는 것을 가정했다. 또한 각 노드를 연결시켜주는 단방향 지점간 링크는 500MHz로 동작하고, 16bit 전송이 가능하므로 1GByte/Sec의 대역폭을 가정했다. 모의실험에 사용된 여러 시스템 인자 값에 대한 요약은 다음과 같다.

표 1 모의 실험 인자

인자	값
프로세서 수	48
한 노드내의 프로세서 수	2
노드 수	24
프로세서 캐시의 크기	8KB
원격 캐시의 크기	32KB
프로세서 클럭 속도	2GHz
노드간 링 연결 클럭 속도	500MHz
프로세서 캐시 접근 시간	2 CPU 클럭
버스 요청 명령 전송 시간	9 버스 클럭
RAC Access Time	2 버스 클럭
메모리 Access Time	20 버스 클럭
버스의 단위 블록 데이타 전송 시간	50 버스 클럭
링 구조에서 인접 노드간 요청 명령 전송 시간	120 버스 클럭
링 구조에서 인접 노드간 블록 데이타 전송 시간	240 버스 클럭

4.2 벤치마크 프로그램

본 논문의 모의실험 입력 프로그램으로 사용된 것은 SPLASH-II에서 제시된 병렬 프로그램이다[9]. SPLASH-II 벤치마크 프로그램은 과학, 공학, 그래픽 등의 분야에서 사용되는 계산유형을 포함하는 여러 개의 응용 프로그램들을 제공한다. 본 연구에 사용된 각각의 SPLASH-II 벤치마크 프로그램은 다음과 같다.

표 2 벤치마크 프로그램 종류와 관련 인자

응용 프로그램	인자
FFT	64K Complex Doubles
LU	512×512 Matrix (16×16 Blocks)
BARNES	8K Particles(Bodies)
RADIX	1M Key Integers, Radix 1024

FFT는 Fast Fourier Transformation을 수행하는 프로그램으로, 전치 행렬을 구하는 과정에서 모든 프로세서들 간에 상호적인 통신을 발생시킨다. LU는 N×N 조밀 행렬 A를 하삼각 행렬 L과 상삼각 행렬 U의 곱 L×U로 나타내기 위해서 분할하는 프로그램이다. BARNES는 Hierarchical N-body Problem 방식을 사용하여 각 입자들 사이에 상호 작용을 구하는 프로그램이다. RADIX는 Radix Sort를 수행하는 프로그램으로, 정렬 과정에서 발생하는 치환이 모든 프로세서 사이에 상호

적인 통신을 발생시킨다.

4.3 실험 결과

다음 그림 11과 그림 12는 SPLASH II 벤치마크 프로그램[15]중 RADIX, FFT, LU, BARNES를 MINT 시뮬레이터[7]를 이용하여 얻은 확률 결과치의 평균 값들을 확률 구동(probability-driven) 시뮬레이터인 SES/Workbench[14]를 이용하여 기존의 코달 링 구조의 CC-NUMA 시스템에서 Skip Distance에 따라 달라지는 링 링크 이용률과 코달 링크 이용률을 구한 것이다[5,6]. 이중 링 구조의 PANDA II 시스템은 링 링크 이용률과 코달 링크 이용률의 차이가 거의 없는 반면에, 코달 링 구조의 CC-NUMA 시스템에서는 코달 링크의 Skip Distance가 2인 경우에 링 링크 이용률과 코달 링크 이용률의 차이가 가장 많이 난다는 것을 알 수 있다. 또한 코달 링크의 Skip Distance가 3과 4인 코달 링 CC-NUMA 시스템에서는 상대적으로 링 링크 이용률과 코달 링크 이용률의 차이가 적은 것을 볼 수 있다.

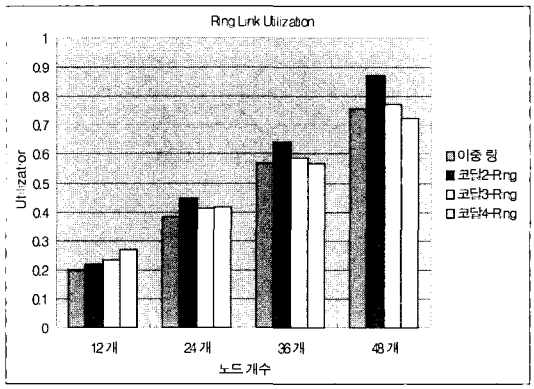


그림 11 기존 코달 링 구조의 CC-NUMA 시스템의 링 링크 이용률

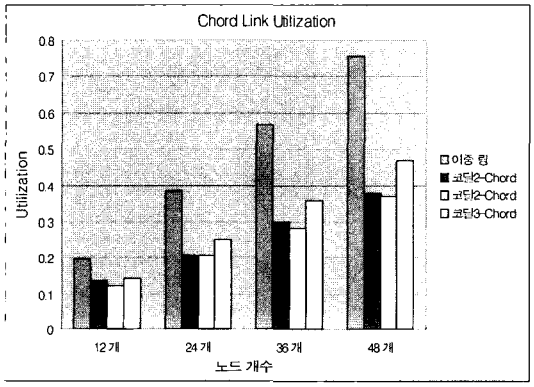


그림 12 기존 코달 링 구조의 CC-NUMA 시스템의 코달 링크 이용률

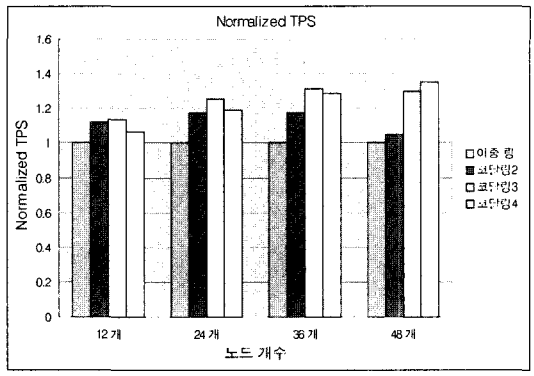


그림 13 기존의 코달 링 구조의 CC-NUMA 시스템의 성능 비교

그림 13은 SPLASH II 벤치마크 프로그램[15]중 RADIX, FFT, LU, BARNES를 MINT 시뮬레이터[7]를 이용하여 얻은 확률 결과치의 평균 값들을 확률 구동(probability-driven) 시뮬레이터인 SES/Workbench [14]를 이용하여 기존의 코달 링 구조 CC-NUMA 시스템의 성능[5,6]을 비교한 것이다. 그림 13에서 TPS (Transaction Per Second)의 비교에 의해 이중 링에 비해 코달 링의 성능이 좋아지는 것을 볼 수 있으며, 코달 링크의 Skip Distance가 3인 코달 링 시스템의 경우가 전체적으로 좋은 성능이 나오는 것을 볼 수 있다. 이것은 그림 11과 그림 12에서 보듯이 Skip Distance가 3인 코달 링 시스템의 링 링크와 코달 링크의 이용률 차이가 비교적 적은 것이 시스템 성능에 큰 영향을 준 것으로 생각할 수 있다. 그림 13에서 노드의 개수가 48개일 경우는 코달 링크의 Skip Distance가 4인 코달 링 시스템의 경우가 가장 성능이 좋은 것을 보여주고 있다. 노드의 개수가 48개일 경우, 코달 링크의 Skip Distance가 3인 코달 링 시스템의 링 링크의 이용률과 코달 링크 이용률의 차이가 급격하게 증가됨에 비해서 코달 링크의 Skip Distance가 4인 코달 링 시스템인 경우 링 링크의 이용률과 코달 링크의 이용률 차이가 감소하게 되는 것을 위의 그림 11과 그림 12를 통해서 알 수 있으며 이것이 두 시스템의 성능 차이에 결정적인 이유로 작용한 것으로 판단된다. 전체적으로 링 링크의 이용률과 코달 링크의 이용률의 차이에 따라서 링 링크로 트래픽이 몰리는 경향이 성능의 차이를 발생시키고, 이러한 트래픽의 부하 균형을 효율적으로 맞추는 것이 코달 링 구조 CC-NUMA 시스템의 성능을 보다 더 높일 수 있는 방법이 될 수 있을 것으로 사료된다.

표 3과 표 4, 그림 14, 그림 15, 그림 16은 12개의 노드를 가지고 원격 캐쉬가 2-Way Set Associative Cache인 코달 링 구조의 CC-NUMA시스템에서 임의

표 3 임의 교체를 사용하였을 때, 축출된 메모리 블록을 전송하기 위해서 지나가는 총 링크의 수

	노드 0번으로 메모리 블록 이동 시			노드 0번에서 메모리 블록 축출 시			총 링크의 수		
	코달 링크의 수	링 링크의 수	두 개 링크의 차	코달 링크의 수	링 링크의 수	두 개 링크의 차	코달 링크의 수	링 링크의 수	두 개 링크의 차
코달 링-2	1.854	0.890	0.964	1.781	0.909	0.872	3.636	1.818	1.818
코달 링-3	1.636	1.090	0.546	1.363	1.363	0	3	2.454	0.546
코달 링-4	1.090	1.636	0.546	1.090	1.636	0.546	2.181	3.281	1.1

표 4 본 논문에서 제시한 원격 캐쉬 교체 정책을 적용하였을 때, 축출된 메모리 블록을 전송하기 위해서 지나가는 총 링크의 수

	노드 0번으로 메모리 블록 이동 시			노드 0번에서 메모리 블록 축출 시			총 링크의 수		
	코달 링크의 수	링 링크의 수	두 개 링크의 차	코달 링크의 수	링 링크의 수	두 개 링크의 차	코달 링크의 수	링 링크의 수	두 개 링크의 차
코달 링-2	1.309	0.709	0.6	1.29	1	0.29	2.6	1.709	0.89
코달 링-3	1.2	0.781	0.42	0.945	1.509	0.56	2.145	2.29	0.15
코달 링-4	0.854	1.145	0.29	0.963	1.545	0.58	1.818	2.690	0.87

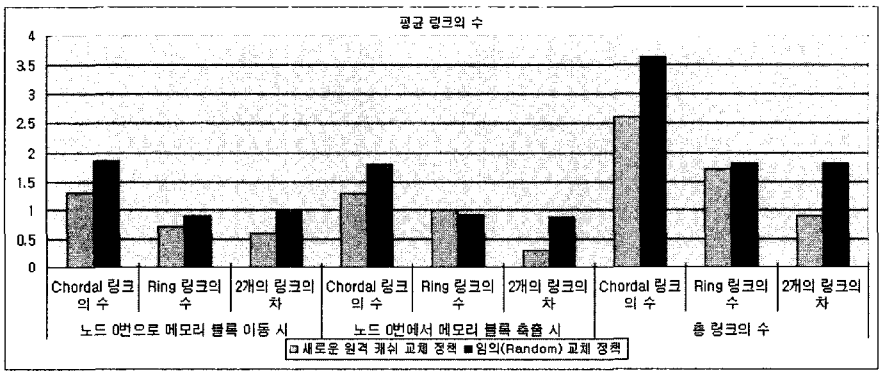


그림 14 코달 링-2에서의 평균 링크의 수

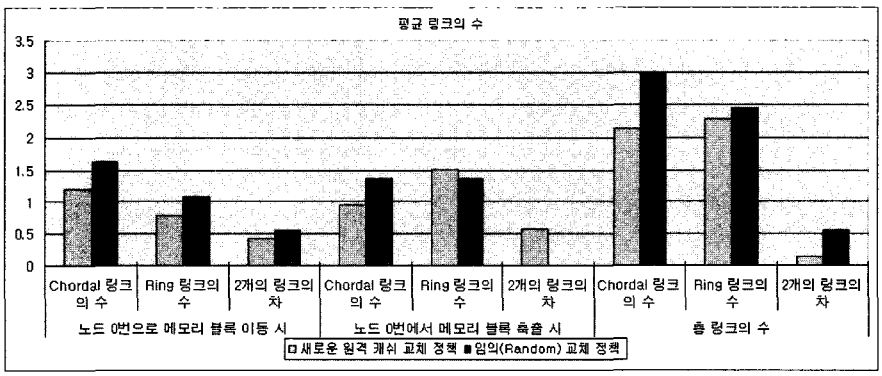


그림 15 코달 링-3에서의 평균 링크의 수

(Random)교체 정책과 본 논문에서 제안하고 있는 새로운 원격 캐쉬 교체 정책을 원격 캐쉬 교체 정책으로 각각 적용하였을 경우, 평균 링크의 수를 정량적으로 계산한 것이다. 노드 0번에 존재하는 원격 캐쉬에서 특정 캐쉬 블록을 제거하기 위해서 지나가는 링크의 수(노드 0번에서 메모리 블록 축출 시)와 축출된 캐쉬 블록을 다시 가지고 오기 위해서 지나가는 링크의 수(노드 0번으로 메모리 블록 이동 시)를 계산할 때, 지나가는 링크의 트래픽은 없다고 가정했다. 전체적으로 임의(Random) 교체 정책보다는 본 논문에서 제안하고 있는 새로운 원

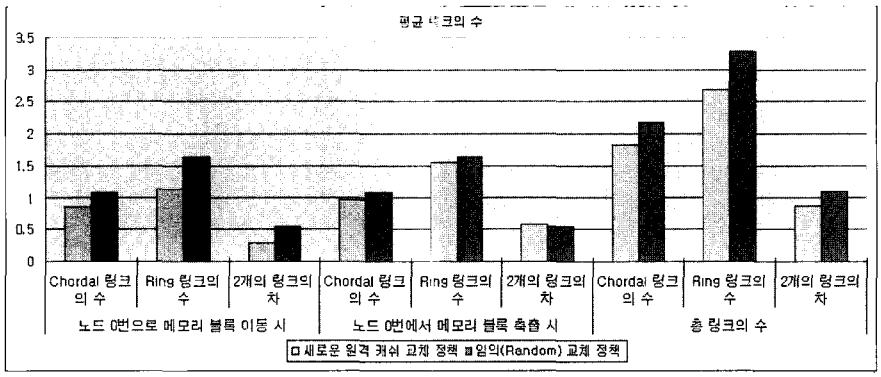


그림 16 코달 링-4에서의 평균 링크의 수

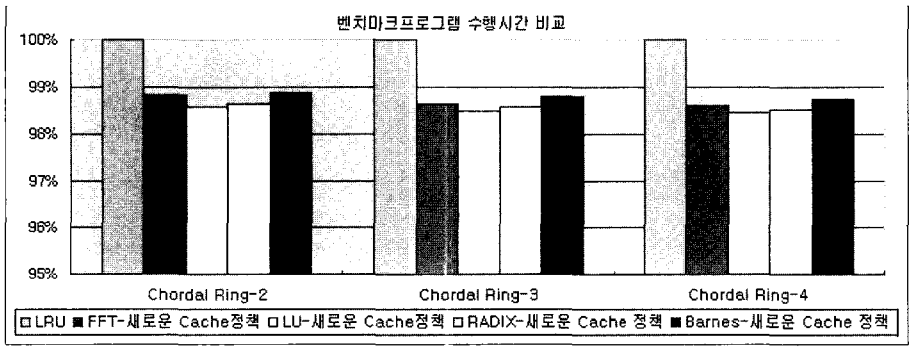


그림 17 2 Way Set Associative Cache일 경우의 벤치마크 프로그램 수행시간

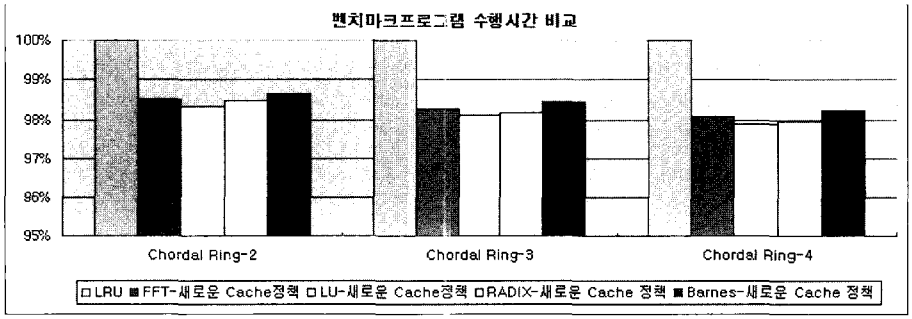


그림 18 4 Way Set Associative Cache일 경우의 벤치마크 프로그램 수행시간

격 캐쉬 교체 정책을 적용할 경우 데이타를 주고받기 위해서 지나가는 평균 링크의 수가 전체적으로 감소한 것을 볼 수 있다. 또한 데이타를 주고받기 위해서 지나가는 코달 링크와 링 링크의 수에 차이가 또한 감소한 것을 볼 수 있다. 즉 임의 교체 정책보다는 본 논문에서 제안하고 있는 새로운 원격 캐쉬 교체 정책을 코달 링 구조의 CC-NUMA 시스템의 원격 캐쉬 교체 정책으로 사용하였을 경우, 전체적인 트래픽 감소와 함께 효율적으로 트래픽을 2개의 링크로 분산시킬 수 있다는 것을 의미한다.

그림 17, 그림 18은 각각의 노드 안에 존재하는 4개의 CPU의 속도가 2GHz이고, 총 노드의 개수가 24개인 코달 링 구조의 CC-NUMA 시스템에 대해 성능 평가를 한 것이다. 그림 17은 2-Way Set Associative Cache에서 원격 캐쉬 교체 정책 중 LRU의 수행시간을 100%로 보았을 때, 본 논문에서 새롭게 제안하고 있는 원격 캐쉬 교체 정책을 적용한 후, 각각의 벤치마크 프로그램의 수행시간을 비교한 비교표이다. 전체적으로 수

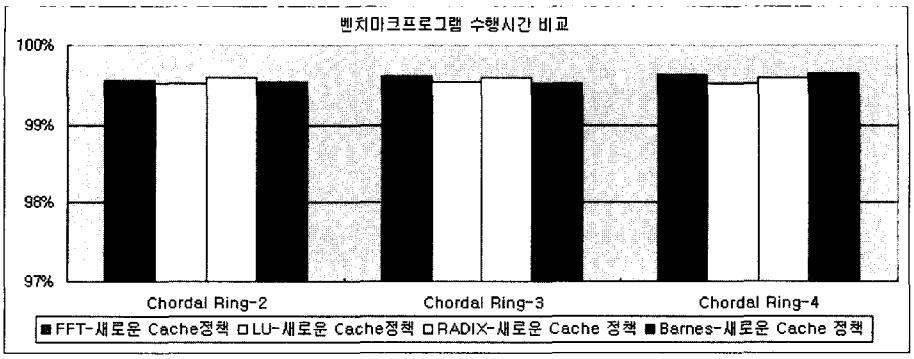


그림 19 4-way Set Associative일 경우, Shared 상태의 에이징 카운터 비트 적용 여부에 따른 성능 비교표

행시간이 LRU보다 빨라지는 것을 볼 수 있으며, 각각의 벤치마크 프로그램 중 LU가 전체적으로 성능이 좋게 나오는 것을 볼 수 있다. 4-Way Set Associative Cache에서 적용한 그림 18의 결과에서도 기존의 원격 캐쉬 교체 정책인 LRU 보다는 본 논문에서 제시한 새로운 원격 캐쉬 교체 정책에서 각각의 벤치마크 프로그램 수행시간이 향상되는 것을 볼 수 있다.

위에 보는 그림과 같이 벤치마크 프로그램의 수행시간을 볼 때, 전체적으로 본 논문에서 제안하고 있는 원격 캐쉬 교체 정책이 성능이 좋게 나오고 있다. 그 이유는 앞에서 서술한 바와 같이 본 논문에서 제시하는 원격 캐쉬 교체 정책에서 메모리 블록을 전송하기 위해 지나가는 링 링크와 코달 링크의 수가 전체적으로 감소하였고, 링 링크와 코달 링크의 부하균형을 효율적으로 이룬 것에 기인한다고 사료된다. 또한 하나 이상의 원격 캐쉬에 있는 캐쉬 라인의 상태가 Shared인 경우에도 해당 캐쉬 라인 가장 최근에 접근한 메모리 블록이라면 축출하지 않는 점 또한 성능 향상의 한 요인으로 작용한 것으로 사료된다. 2-Way Set Associative Cache에 적용한 경우보다 4-Way Set Associative Cache에 적용한 경우가 더 향상된 성능을 보이는데, 그 이유는 2-Way Set Associative Cache에서 보다 4-Way Set Associative Cache에서 에이징 카운터 비트의 효과에 의해서 원거리 노드의 메모리 블록이 축출된 확률이 적어지기 때문이다.

그림 19는 24개의 노드를 가지고 원격 캐쉬가 2-Way Set Associative Cache인 코달 링 구조의 CC-NUMA시스템에서 본 논문에서 제시하고 있는 원격 캐쉬 교체 정책에서 캐쉬 라인의 상태가 Shared일 경우에 에이징 카운터 비트를 적용하지 않은 각각의 수행시간을 100%로 했을 때, 캐쉬 라인의 상태가 Shared일 경우에도 에이징 카운터 비트를 적용한 경우의 수행시간을 비교한 것이다. 결과에서 보듯이 캐쉬 라인의 상태가 Shared일 경우에도 에이징 카운터 비트의 값을 적용하였을 때의 수행시간이 그렇지 않은 것 보다 더 향상되는 것을 볼 수 있다. 이러한 점은 Shared 상태인 캐쉬 라인이 두 개 이상 존재할 경우, 교체 할 때 전송해야 하는 거리와 에이징 카운터로 다음 교체할 라인을 결정하는 것이 성능면에서 좋다는 것을 나타낸다.

그림 20은 24개의 노드를 가지고 원격 캐쉬가 2-Way Set Associative Cache인 코달 링 구조의 CC-NUMA 시스템에서 Invalid 상태를 제외한 원격 캐쉬에 존재할 수 있는 모든 캐쉬 상태에 대해서 얼마나 많은 확률로 교체가 되는지를 나타내는 표이다. LRU의 교체 비율표를 본다면 Modified 상태, Modified-Shared 상태, Shared 상태에 대해서 균등한 비율로 교체되는 것을 볼 수 있다. 본 논문에서 제안한 원격 캐쉬 교체 정책은 상대적으로 Shared 상태가 자주 교체되는 것을 볼 수 있다. 이것은 본 논문에서 제안한 원격 캐쉬 교체 정책을 적용하게 되면 캐쉬 라인의 상태가 Shared 상태로 존재하는 라인인 경우 Modified 상태 보다 먼저 교체를 하게 되기 때문이다. 따라서 Shared 상태로 존재하게 되는 캐쉬 라인의 교체 비율이 다른 상태에 비해

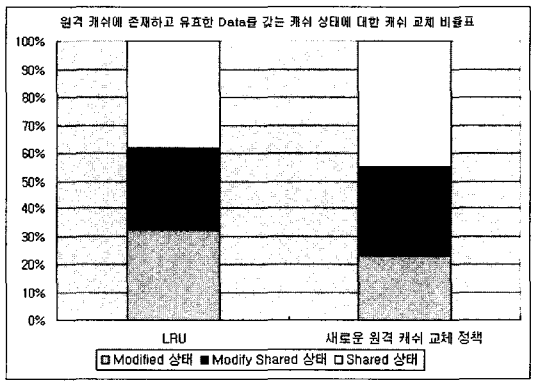


그림 20 원격 캐쉬의 캐쉬 상태에 따른 교체 비율

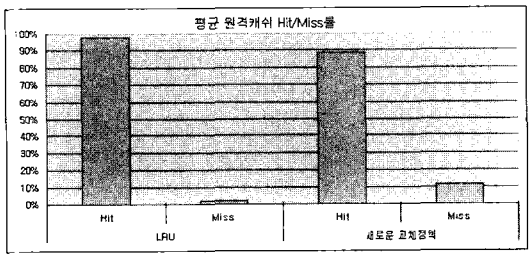


그림 21 원격 캐쉬 평균 Hit/Miss율

서 많은 것을 볼 수 있다.

그림 21은 24개의 노드를 가지고 원격 캐쉬가 2-Way Set Associative Cache인 코달 링 구조의 CC-NUMA시스템에서 모든 벤치마크 프로그램에 대한 원격 캐쉬의 평균 Hit/Miss율을 나타내고 있다. 본 논문에서 제시하고 있는 원격 캐쉬 교체 정책은 LRU 정책을 사용한 것 보다는 낮은 Hit율을 보이고 있는데, 이는 프로세서가 공유 메모리를 접근하는 패턴을 이용해서 축출된 캐쉬 라인을 교체하는 것이 아니라, 코달 링의 구조적 특징을 사용하기 때문에 Hit율이 감소하는 경향을 보인다. 또한 본 논문에서 제안한 원격 캐쉬 교체 정책에서는 가장 최근에 접근한 캐쉬 라인의 상태가 Shared 상태일 경우, 다른 모든 캐쉬 라인의 상태가 Modified 상태라면 가장 최근에 접근을 하였던 Shared 상태의 캐쉬 라인이 교체되기 때문에 기존의 원격 캐쉬 교체 정책인 LRU보다는 Hit율이 감소되는 원인으로 사료된다.

그림 22는 24개의 노드를 가지고 원격 캐쉬가 2-Way Set Associative Cache인 코달 링 구조의 CC-NUMA시스템의 각각의 노드들에 존재하는 2개의 링크의 이용률의 변화를 표시한 것이다. 본 논문에서 제안하고 있는 원격 캐쉬 교체 정책을 원격 캐쉬에 적용했을 경우에는 기존의 원격 캐쉬 교체 정책인 LRU와 비교해서 링 링크의 이용률은 감소하였고 코달 링크의 이용률은 증가되었다. 기존의 원격 캐쉬 교체 정책에서 효율적으로 사용하지 못했던 코달 링크의 이용률을 증가시키고, 비교적 이용이 많았던 링 링크의 이용률을 감소시켜 링크간의 트래픽 불균형을 해소함으로써, [그림 21]의 성능 하락 요인에도 불구하고 전체적으로 코달 링 구조의 CC-NUMA 시스템의 성능을 증가시킨 것으로 사료된다.

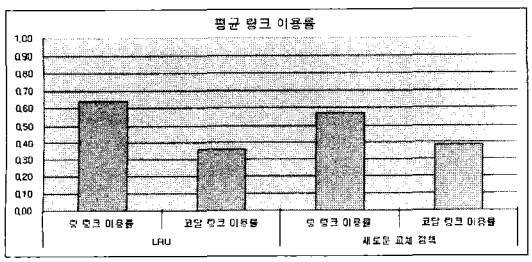


그림 22 평균 링크 이용률

본 논문에서 제안하고 있는 원격 캐쉬 교체 정책은 원격 캐쉬 교체 정책에 코달 링 구조의 CC-NUMA 시스템의 하드웨어적 특성을 감안했다는 점이 기존의 원격 캐쉬 교체 정책과는 다른 차이점이라 할 수 있다. 본 논문에서 제안하고 있는 원격 캐쉬 교체 정책의 주된 개념은 노드에 존재하는 링크의 링크 이용 효율성을 높여서 코달 링 구조의 CC-NUMA 시스템의 성능을 향상시키는데 있다고 할 수 있다. 따라서 지역 프로세서의 원격 캐쉬 블록을 접근하는 패턴을 고려하지 않기 때문에 원격 캐쉬의 Hit율 감소라는 문제점이 생긴다는 단점이 있다. 그러나 위의 결과를 종합적으로 살펴본다면, 코달 링 구조의 CC-NUMA 시스템에서는 원격 캐쉬의 Hit율 감소에 따른 성능하락 요소 보다는 링크 이용 효율성에 따른 성능향상 요소가 전체적인 성능에 영향을 많이 준다는 것을 알 수 있다.

5. 결 론

코달 링 구조의 CC-NUMA 시스템은 모든 노드를 연결하고 있는 링 링크와 Skip Distance에 따라 원격 노드를 건너뛰면서 연결하는 코달 링크로 구성이 된다. 이 시스템에 있어서 성능 하락 요인 중 하나인 링 링크로 트랜잭션이 몰리는 것을 방지하기 위해서는 최소의 링크를 지나가는 링 링크를 이용하지 않고 지나가는 링크의 수가 증가되더라도 코달 링크만을 이용해서 데이타를 주고받는 방법이 있다. 그러나 이 방법은 링 링크 이용률이 감소한 만큼 코달 링크의 이용률이 증가된다는 점과 지나가는 링크의 증가에 따른 데이타 접근 시간이 더 걸리는 점 때문에 좋은 방법이라고 할 수 없다. 따라서 본 논문에서는 CC-NUMA 시스템의 구조적인 특징에 따른 데이타 전송 경향의 고려와 링 링크를 보다 더 많이 지나가는 원격 노드의 데이타를 현재 노드의 원격 캐쉬에서 가급적 교체하지 않는 방법으로 코달 링크와 링 링크간의 부하 균형을 유지하는 코달 링 구조의 CC-NUMA 시스템에 알맞은 원격 캐쉬 교체 정책을 제안하였다.

기존의 원격 캐쉬 교체 정책은 모든 캐쉬 라인의 교체 결정을 지역 프로세서에 접근하는 패턴을 이용하여 결정한다. 이것은 프로세서가 있는 위치에 따라 공유 메모리를 접근하는 비용이 달라지는 NUMA 시스템에서는 멀리위치하고 있는 메모리 블록의 접근과 교체에 의한 비용을 전혀 고려하지 않는 원격 교체 방법이다. 그러나 본 논문에서 제시하고 있는 원격 캐쉬 교체 정책

은 CC-NUMA 시스템의 구조적 특징과 함께, 코달 링 구조의 CC-NUMA 시스템의 구조적 특징을 반영함으로써 효율적인 트래픽 분산을 통한 성능을 향상시킬 수 있다는 데에 큰 의미가 있을 것이다. 이를 위해 본 논문에서 제시한 원격 캐쉬 교체 정책은 데이타를 주고받기 위해서 지나가는 총 링크의 수가 적을 것을 우선적으로 교체함으로써 CC-NUMA 시스템의 구조적인 특징을 원격 캐쉬 교체 정책에 적용하였다. 또한 코달 링 구조의 CC-NUMA 시스템의 구조적인 특징인 링 링크로 트래픽이 몰리는 현상을 해결하기 위해서 원격 캐쉬에 있는 데이타 중 데이타를 주고받기 위해서 지나가는 링 링크의 수가 적은 것을 우선적으로 교체하는 방법을 사용하였다. 이렇게 본 논문에서 제시하고 있는 원격 캐쉬 교체 정책은 CC-NUMA의 구조적인 문제점을 적극적으로 해결하기 위한 방안이라고 생각이 되어진다.

향후 연구로는 보다 더 적극적으로 코달 링크와 링 링크를 동시에 사용하는 데이타 전송의 수를 감소시킬 수 있는 방안에 대한 연구를 계속할 예정이며 Eager Write-Back[13] 정책을 본 논문에서 제시한 원격 캐쉬 교체 정책에 접목하는 연구도 수행할 예정이다. 현재, 제시한 캐쉬 교체 정책을 구동하는 캐쉬 제어기를 FPGA 수준에서 구현하는 연구도 진행하고 있다.

참 고 문 헌

[1] J. L. Hennessy and D.A Patterson, "Computer Architecture : A Quantitative Approach," Second Edition, Morgan Kaufmann Publishers, Inc, 1996.

[2] Kai Hwang and Zhiwei Xu, "Scalable parallel Computing : Technology, Architecture, Programming," McGraw-Hill, 1998.

[3] Sung Woo Chung, Seong Tae Jhang, and C. S. Jhon, "PANDA : Ring Based Multiprocessor System using New Snooping Protocol," ICPADS'98, pp.10-17, December 1998.

[4] 정성진, "지점간 링크를 이용한 이중 링 스누핑 버스 다중 프로세서 시스템의 설계와 검증", 서울대학교 석사학위 논문, 2000.

[5] 윤주범, 장성태, 전주식, "이중 링 CC-NUMA 시스템에서 링 구조 변화에 따른 시스템 성능 분석", 한국정보과학회 논문지 : 시스템 및 이론, 제29권 2호, pp.105-115, 2002.

[6] Joo Beom Yun, Cheol Won Lee, Seong Tae Jhang, and Chu Shik Jhon, "Analysis of System Performance by Changing the Ring Architecture on Dual Ring CC-NUMA System," ICPADS'2002, Dec. 2002.

[7] JACK E. Veenstra, Robert J. Fowler, "MINT : A front end for efficient simulation of shared-memory multiprocessors", In Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems(MASCOTS), pp. 201-207, 1994.

[8] A.-T. Nguyen, M. Michael, A. Sharma, J. Torrellas, "The Augmint multiprocessor simulation toolkit for Intel x86 architectures," 1996 International Conference on Computer Design (ICCD '96).

[9] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and Anoop Gupta, "Methodological Considerations and Characterization of the Splash-II Parallel Application Suite," In Proceedings of the 22th International Symposium on Computer Architecture, pp. 24-36, May 1995.

[10] 경진미, 장성태, "이중 링 CC-NUMA 시스템에서 리피터 노드를 이용한 링구조 변화에 따른 성능 분석", 한국정보과학회 논문지 : 시스템 및 이론, 제29권 9호, pp.503-513, 2002.

[11] D.E. Culler and J.P. Singh, "Parallel Computer Architecture: A Hardware /Software Approach," Morgan Kaufmann Publishers, 1999.

[12] Kevin MlLepak and Mikko H. Lipsasti, "Silenet Stores and Store Value Locality," IEEE Transactions on Computers, Vol.50, No.11, Nov. 2001.

[13] Hsien-Hsin S. Lee, Gary S. Tyson, Matthew K. Farrens, "Eager writeback - a technique for improving bandwidth utilization," Proceedings of the 33rd annual ACM/IEEE international symposium on Microarchitecture, December 2000.

[14] Scientific and Engineering Software inc., "SES/Workbench Technical Reference," 1995.

[15] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. "Methodological considerations and characterization of the SPLASH-2 parallel application suite," In Proc. 22th Annual International Symposium on Computer Architecture, 1995.

김 수 한

2001년 2월 수원대학교 컴퓨터학과 학사
2003년 2월 수원대학교 컴퓨터학과 석사
2003년 3월~현재 수원대학교 컴퓨터학과 박사과정 재학 중. 관심분야는 컴퓨터 구조, 다중 프로세서 시스템, 임베디드 시스템 설계

김 인 석

1997년 2월 수원대학교 전자계산학과 학사. 1999년 2월 수원대학교 컴퓨터학과 석사. 2002년 2월 수원대학교 컴퓨터학과 박사 수료. 관심분야는 컴퓨터 구조, 다중 프로세서 시스템, 임베디드 시스템 설계

김 봉 준
1997년 2월 수원대학교 전자계산학과 학사. 1999년 2월 수원대학교 컴퓨터학과 석사. 2003년 2월 수원대학교 컴퓨터학과 박사 수료. 관심분야는 컴퓨터 구조, 다중 프로세서 시스템

장 성 태
정보과학회논문지 : 시스템 및 이론
제 31권 제 8 호 참조