이중 링 CC-NUMA 시스템에서 링 구조 변화에 따른 시스템 성능 분석

(Analysis of System Performance by changing the Ring Architecture on Dual Ring CC-NUMA System)

윤 주 범 † 장 성 태 †† 전 주 식 †††
(Joo Beom Yun) (Seong Tae Jhang) (Chu Shik Jhon)

요 약 NUMA 구조는 원격 메모리에 대한 접근이 불가피한 구조적 특성 때문에 상호 연결망이 시스템 성능을 좌우하는 큰 변수가 된다. 기존에 대중적으로 사용되던 버스는 물리적 확장성 및 대역폭에서 대규모 시스템을 구성하는 데 한계를 보인다. 이를 대체하는 고속의 지점간 링크를 사용한 이중 링 구조는 버스가 가지는 확장성 및 대역폭의 한계라는 단점을 개선하였으나, 많은 노드가 연결되는 경우에는 응답 지연시간이 증가하는 문제점을 가지고 있다. 본 논문에서는 스누핑 프로토콜이 적용된 이중 링 구조에서 노드 개수 증가에 따른 응답 지연시간 증가의 문제점을 보완하기 위해 코달 링 구조로의 변화를 제안하고, 이 구조에 효과적인 링크 제어기를 설계한다. 또한 확률 구동 시뮬레이터를 통해 본 논문을 통해 제시한 코달 링 구조가 시스템의 성능 및 응답 지연시간에 미치는 영향을 알아본다.

키워드 : 다중프로세서 시스템, 지점간 링크, SCI, 이중 링, 코달 링, 대역폭, 이용률, 응답지연시간, 건너뜀 거리

Abstract Since NUMA architecture has to access remote memory, an interconnection network determines the performance of CC-NUMA system. Bus, which has been used as a popular interconnection network, has many limits to build a large-scale system because of the limited physical scalability and bandwidth. Dual ring interconnection network, composed of high-speed point-to-point links, is made up for resolving the defects of the bus for large-scale system. But, it also has a problem that the response latency is rapidly increased when many nodes are attached to snooping based CC-NUMA system with dual ring. In this paper, we propose a chordal ring architecture in order to overcome the problem of the dual ring on snooping based CC-NUMA system, and design an efficient link controller adopted to this architecture. We also analyze the effects of chordal ring architecture on the system performance and the response latency by using probability-driven simulator.

Key words : Multiprocessor system, point-to-point links, SCI, dual ring, chordal ring, bandwidth, utilization, response latency, skip distance

1. 서 론

최근의 공유 메모리 다중 프로세서 시스템에서, NUMA 구조는 확장성 및 프로그래밍 용이성의 측면에서 장점을 가진다. 하지만, 시스템의 크기가 증가함에 따라 원격 메모리에 접근하는 빈도수가 증가하므로 원격 메모리의 접근을 줄이거나 접근 속도를 높이는 것이 NUMA 구조의 성능 향상을 위해 필수적이다. NUMA 구조의 성능 향상을 위한 노력 중에서, 원격 캐쉬를 두면서 그 크기를 증가시키는 것은 원격 메모리의 접근을 줄이는 방법[1]이며, 상호 연결망의 속도를 증가시키는 것은 원격 메모리에 접근하는 속도를 높이는 방법이다. 이에 대한 대안으로 COMA 시스템이 제안되었으나, 캐쉬 일관성을 유지하는데 드는 비용이 크고, 구현하기에 복잡하다는 단점으로 인해 캐쉬 크기가 증가하고 있는 NUMA 시스템에 대해 이점이 줄어들고 있다[1].

공유 메모리 다중 프로세서 시스템에서 가장 흔히 사

† 비 회 원 : 국가보안기술연구소 연구원
netair@etri.re.kr
†† 종신회원 : 수원대학교 전자계산학과 교수
stjhang@mail.suwon.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학과 교수
csjhon@riact.snu ac.kr
논문접수 : 2001년 5월 28일
심사완료 : 2001년 11월 21일

용되는 상호 연결망은 공유 버스이다. 공유 버스는 구현상의 복잡도가 낮고 비용이 적게 드는 장점을 가지지만, 확장성 및 대역폭의 한계를 이유로 주로 소규모의 시스템에서 사용된다. 이러한 공유 버스의 단점을 보완하기 위해 최근에 널리 사용되는 상호 연결망이 이중 링(dual ring)이다. 이중 링은 지점간 링크를 사용하여 노드들을 양방향으로 연결함으로써 공유 버스에 비해 속도의 제약이 적고, 확장성과 대역폭이 높다는 장점을 가진다. 지점간 링크라는 특성 때문에 공유 버스와는 달리 방송(broadcast)이 어렵다는 특징을 가지므로 이중 링을 사용하는 기존의 많은 시스템들은 디렉토리를 사용한 캐쉬 일관성 유지 방법을 사용하였으나, 스누핑 캐쉬 일관성 유지 방법을 사용한 이중 링 구조도 제안된 바 있으며[2], 이의 성능이 디렉토리 캐쉬 일관성 유지 방법을 사용한 구조에 비해 우수함이 발표되었다[3].

상호 연결망으로서 버스의 한계를 극복하고자 링을 사용하려고 하는 노력은 꾸준히 계속되어 왔다. Sequent에서는 4개의 펜티엄 프로 프로세서가 P6버스에 의해 UMA구조로 묶인 노드들을 SCI 링크[4]를 사용해 연결한 STiNG[5]을 발표했다. STiNG은 디렉토리를 사용한 캐쉬 일관성 유지방법을 사용하며, 데이타를 공유하고 있는 노드들을 연결리스트(linked-list)로 연결하여 공유 데이타에 대한 읽기 및 쓰기 관리를 수행한다. Toronto 대학에서 만든 NUMAchine은 디렉토리를 사용한 캐쉬 일관성 유지방법을 사용하는 링 구조의 네트워크를 계층적 구조로 확장한 것으로, 64개의 프로세서를 연결한 프로토타입이 제작되었다[6]. Michel Dubois가 제안한 Express Ring[7]은 스누핑 방식의 캐쉬 일관성 유지방법을 사용한 시스템이다. Express Ring은 슬롯 링(Slotted Ring) 방식으로 구성되어, 링크의 대역폭을 충분히 사용하기 어렵다는 단점을 가진다. 서울대학교의 PANDA 연구실에서 제안한 PANDA II (Progressive Approach of NUMA model based on Distributed shared memory Architecture II) 시스템[2]은 스누핑 방식의 캐쉬 일관성 유지방법을 사용하며, 4개의 펜티엄 프로 프로세서가 묶인 하나의 노드를 양방향 지점간 링크로 연결하였다. 이 시스템의 특징은 링을 통한 방송 트랜잭션을 지원하여, 링을 가상적인 버스로 구성하였다는 것이다.

이중 링 구조가 버스에 비해 속도 및 물리적 확장성에서 장점을 가지지만, 지점간 링크로 구성되는 특징 때문에 링에 연결되는 노드들의 수가 증가하면 할수록 원격 노드로의 요청 및 응답에 걸리는 시간은 점점 증가하게 된다. 이중 링 구조가 가지는 확장성에서의 장점을 살리기 위해서는 노드 수의 증가에 따른 지연시간의 증가를 줄이며, 지속적인 성능 향상을 꾀할 수 있는 방안을 강구해야 한다.

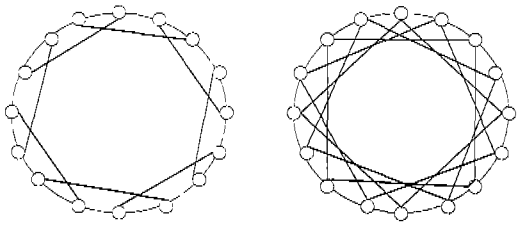
본 논문에서는 이중 링 구조에서 스누핑 프로토콜을 사용하는 PANDA II 시스템을 코달 링 구조(Chordal Ring Architecture)로 변환하는 방안을 제시하고자 한다. 기존 PANDA II 시스템에서의 한 노드에는 입력과 출력이 2개씩, 전체 4개의 포트가 존재한다. 따라서 본 논문에서는 기존의 노드 구조를 그대로 유지하기 위하여 노드 차수(node degree)가 4인 코달 링 구조만 고려하기로 한다. 이중 링과, 건너뛰 거리(skip distance)를 변화시킨 코달 링을 비교하여 장단점을 분석해 보고, 코달 링을 적용하기 위한 새로운 링크 제어기(link controller)의 구조에 대해 고려해 보기로 한다.

2장에서는 코달 링을 적용한 PANDA 시스템의 전체 구조와 동작, 링크 제어기의 구조와 동작을 소개한다. 또한 캐쉬 일관성 프로토콜도 살펴본다. 3장에서는 이중 링과 여러 건너뛰 거리를 갖는 코달 링 구조를 설명하고 정성적 분석을 하기로 한다. 4장에서는 3장에서 소개한 구조들의 특성 및 성능을 모의 실험 결과를 통해 분석해 본다. 5장에서는 4장의 내용을 근거로 결론을 맺도록 한다.

2. 시스템의 구조와 동작

2.1 코달 링의 구조

본 논문에서 채택하고자 하는 상호 연결 망의 기반이 되는 코달 링에 대해서 살펴보자. 노드 차수가 2인 단일 링에서 노드 차수를 3이나 4로 증가시키면 그림 1의 (a), (b)와 같은 두 개의 코달 링을 만들 수 있다. 이 코달 링들을 만들기 위해서는 여러 개의 링크를 추가하면 되는데, 일반적으로 더 많은 링크가 추가될수록 노드 차수가 높아지고 네트워크의 지름은 짧아진다. 더 높은 노드 차수를 갖는 코달 링을 구현하기 위해서는 더 많은 링크를 필요로 한다[8]. 노드 차수와 네트워크의 지름 사이의 트레이드오프(tradeoff)는 가격 대 성능뿐만 아니라, 주어진 상호 연결 망의 확장성(scalability)과 결함 허용성까지 영향을 주게 된다[8].



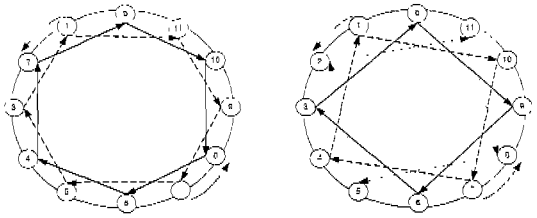
(a) 노드 차수 3인 코달 링 (b) 노드 차수 4인 코달 링

그림 1 두 개의 코달 링

위와 같이 여러 노드 차수의 코달 링 중에서 새로 제안하는 코달 링은 노드 차수가 4인 것이다. 이것은 기존 PANDA II 시스템에서 이중 링 구조를 채택하였기 때문인데 PANDA II 구조에서는 한 노드에 각각 입력과 출력이 2개씩이므로 전체 4개의 포트가 존재한다. 따라서 본 논문에서는 기존 노드의 구조를 그대로 유지하기 위해 노드 차수가 4인 코달 링만을 고려하기로 한다. (또한 이후 본 논문에서 코달 링이라고 하는 것은 노드 차수가 4인 코달 링을 의미하는 것임.)

2.2 시스템의 전체 구조

본 논문에서 제시하는 시스템은 CC-NUMA 구조의 스누핑 캐쉬 일관성 프로토콜을 사용하며, 지점간 링크로 구성된 방향 분리 코달 링의 상호 연결 망을 가지는 PANDA 시스템이다. PANDA 시스템에서의 링은 방송 트랜잭션(broadcast)을 지원하므로, 논리적으로 버스와 동일한 동작을 수행한다. 이를 사용하여 스누핑(snooping) 방식으로 캐쉬 일관성을 유지하게 된다. 링 구조에서 스누핑 방식의 캐쉬 프로토콜을 사용하는 또 다른 시스템인 Express Ring에서는 슬롯 링(Slotted Ring)방식으로 구성되어 있는 반면에, PANDA 시스템에서는 레지스터 삽입 방식을 사용한다. 슬롯 링은 데이타를 전송하고자 하는 클러스터가 링을 돌고 있는 크기가 각기 다른 슬롯들 중 전송할 데이타와 크기가 맞는 빈 슬롯이 지나갈 때야 비로소 전송을 수행할 수 있으므로, 링의 이용률이 낮은 단점이 있는 반면에[7], 레지스터 삽입 방식은 각 링크 사이에 존재하는 FIFO 큐를 사용해 전송하는 방식으로 슬롯 링에 비해 링의 이용률을 높일 수 있는 장점을 가진다[3]. 새로 제안하는 시스템은 그림 2의 (a), (b)와 같은 구조를 가진다.



(a) 길이가 2인 코달 링 (코달 링 2)　　(b) 길이가 3인 코달 링 (코달 링 3)

그림 2 노드 수가 12인 시스템 구성도

위 구조에서 링크를 연결하는 방법에는 건너뛰 거리가 2가 되는 방법(길이 2)과 건너뛰 거리가 3이 되는 방법(길이 3)이 있다. 그러나 건너뛰 거리가 2 이상이 되게 연결하게 되면 시계 방향으로 이동하다가 목적 노드가 가까워 옴에 따라서 다시 바깥쪽 링을 따라 패킷을 싣게 된다. 이것은 바깥 링크의 이용률(utilization)을 높이는 결과를 초래하므로 바람직한 결과가 나올 수 없게 된다. 따라서 노드를 적당히 건너 뛰면서 링크를 연결하는 방법이 바람직하며, 이것은 전체 시스템의 노드 수에 따라 달라질 것으로 생각할 수 있다. 어떤 노드의 개수에서 건너뛰 거리를 얼마로 정할지는 시뮬레이션을 통하여 구해 볼 것이다.

그림 2의 (a)를 보면 3개의 링이 존재한다. 가장 바깥쪽의 큰 링과 안쪽의 길이가 2인 2개의 코드(chord) 링이 존재하는데, 가장 바깥쪽의 링과 안쪽의 두 개의 코드 링의 방향은 서로 반대방향으로 되어 있다. 가장 바깥쪽의 링은 주로 스누핑을 위한 방송 패킷들에 사용되고, 안쪽 코드 링에서 돌던 단일 전송 패킷(unicast)이 방향을 바꿀 때 바깥쪽의 링으로 옮겨 타게 된다. 또한 단일 전송 패킷이 소스 노드에서 목적 노드까지 거리가 바깥쪽의 링을 타고 도는 것이 안쪽코드 링을 타는 것보다 홉(hop)이 적을 경우 바깥쪽의 링을 타고 돌 수도 있다. 그러나 단일 전송 패킷이 바깥쪽의 링의 타고 도는 경우보다 안쪽의 코드 링을 타는 경우가 홉 측면에서 유리하다면 안쪽 코드 링을 타고 돌게 된다. 이때 안쪽 코드 링은 몇 개의 노드를 건너 뛰어서 연결되기 때문에 안쪽 코드 링을 타고 돌다가 목적 노드를 지나게 된다면 다시 바깥 링을 타고 오던 방향으로 가게 된다. 물론 이것은 건너뛰 거리(또는 길이)보다 적게 되기 때문에 많은 시간을 바깥쪽 링에서 소비하지는 않을 것이라 생각된다. 이때 링크의 방향 전환은 노드 안쪽의 B-링크를 통해 다른 링크 제어기(link controller)로 전송됨으로써 이루어진다. 그리고 바깥쪽의 큰 원의 링크를 링 링크, 안쪽의 링크를 코드 링크라 하겠다. 그림 2에서 원 안의 숫자는 노드 번호를 의미하며 2.3절에서 노드의 구조를 살펴 볼 것이다.

2.3 노드의 구조

그림 3은 PANDA 시스템의 한 노드 구조를 나타내고 있다. 각 노드에는 네 개의 프로세서 모듈과 지역 메모리(MEM), 입출력 제어기(IOC) 및 노드 제어기(node controller)가 지역 버스에 연결되어 있다. 노드의 지역 버스는 요청 트랜잭션과 응답 트랜잭션을 분리하는 분리(split) 버스 프로토콜을 지원하며 스누핑 방식에 의한 캐쉬 일관성 유지 프로토콜을 사용한다. 각 프로세서 모듈은 2차 캐쉬(L2 cache)까지 내장하고 있으며 지역 메모리는 분산형 공유 메모리로서 전체 시스템 메모리 물리 주소 영역의 일부를 구성하고 있다. 입출력 제어기는 프로세서에서 요구한 입출력 디바이스에 대한 읽기 및 쓰기를 수행해 준다. 모든 입출력 관련 트랜잭션은 지역

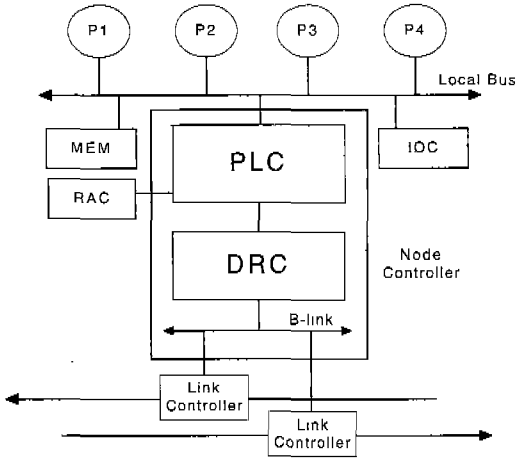


그림 3 노드 구조도

버스에 연결된 PCI 버스를 통해 처리하게 된다.

노드 제어기는 원격 노드로의 접근 지연 시간을 단축하기 위한 원격 캐쉬(RAC : Remote Access Cache)를 유지하며 지역 버스에서 발생한 요청에 대해 원격 캐쉬로부터 데이타 제공 또는 전역 링크를 통한 원격 노드로부터의 데이타 제공을 담당하고 마찬가지로 전역 링크에서 발생한 요청에 대해 응답할 책임을 진다. 이와 동시에 노드 제어기는 지역 버스 및 전역 링크에서 발생하는 모든 트랜잭션을 스누핑하여 메모리-캐쉬 일관성 유지를 위해 필요한 제어를 수행한다. 이때 지역 버스에서 발생하는 트랜잭션을 처리하는 부분이 지역버스 제어기(PLC: Panda Local bus Controller)이고, 전역 링크에서 발생하는 트랜잭션을 처리하는 부분이 이중 링 제어기(DRC: Dual Ring Controller)이다.

2.3.1 노드 제어기

노드 제어기는 지역 버스 제어기, 이중 링 제어기, 원격 캐쉬(RAC) 및 태그(tag), 지역 메모리 디렉토리, B-링크, 그리고 링크 제어기로 이루어져 있다. 원격 캐쉬는 원격 메모리에 대한 응답 지연 시간을 줄이고, 원격 메모리 접근 요구들로 인한 전역 링크의 트래픽(traffic)을 감소시키기 위한 것으로 지역 메모리에서 서비스 할 수 없는 트랜잭션에 대한 처리를 가능하게 한다.

지역 메모리 디렉토리와 원격 캐쉬 태그는 지역 메모리의 상태 및 원격 캐쉬의 상태를 캐쉬 라인별로 유지하는 부분으로, 상태 검색 및 갱신 요구의 반복으로 인해 전체 시스템의 병목점(bottleneck)이 되는 것을 방지하기 위해 지역 버스 제어기와 이중 링 제어기에서 각각 독립적으로 접근할 수 있도록 이중화되어 있다.

이중 링 제어기는 시스템 버스를 통해 요구 패킷이 들어오면 원격 캐쉬 태그나 지역 메모리 디렉토리를 참조하여 응답 책임이 있는 경우 지역 버스 제어기로 전달하고, 책임이 없는 경우에는 다음 노드에 전달한다. 지역 버스 제어기는 지역 P6 버스 상의 트랜잭션을 관찰하여 원격 캐쉬 태그와 지역 메모리 디렉토리에 대한 캐쉬 일관성 유지의 책임을 갖는다. 원격 메모리 영역에 대한 요구 중 원격 캐쉬에 유효한 값이 있는 경우 이를 서비스하고, 지역 노드에서 처리할 수 없는 경우에는 트랜잭션을 이중 링 제어기로 전달한다. 또한, 이중 링 제어기가 보내온 요청 및 응답 트랜잭션의 처리도 담당한다. 링크 제어기는 SCI 링크를 통해 전달받은 패킷을 지역 노드로 올리거나 바이패스(bypass) 시키고, 지역 노드에서 나온 노드를 원하는 SCI 링크를 통해 보내는 역할을 한다.

이와 같은 노드 구조에서 상호 연결 망을 설계할 때 고려해야 할 부분으로는 B-링크와 링크 제어기가 있다. B-링크(Backside Link)는 PANDA II 시스템에서 채택한 링크 제어기인 LC3 칩이 노드 방향으로 통신을 하기 위한 인터페이스로 역시 Dolphin 사가 제안한 버스 형태의 상호 연결 망이다. B-링크는 64 bit 데이타 링크로 155MHz 속도까지 동작한다. 따라서 대역폭은 1240MByte/Sec. 이다. B 링크에서 패킷 전송은 링크 중재 기간으로 나뉘어 전송하게 되는데, 이중 링 제어기와 두 개의 LC3는 이 링크 중재 기간에 요구를 보내면 B-링크의 중재를 거쳐 패킷 전송이 가능할 때 B-링크로 패킷을 전송한다[9].

2.3.2 링크 제어기

PANDA II 시스템에서는 이중 링이기 때문에 링크 제어기를 2개 사용한다. 각각의 링크 제어기는 Dolphin 사에서 제공하는 SCI 링크 제어 칩(Link Controller Chip : LC3)이다. LC3는 링크 방향으로의 인터페이스는 ANSI/IEEE SCI 프로토콜을 만족시키면서 패킷 전송을 처리하고, 노드 방향 인터페이스는 자사에서 개발한 B-링크 프로토콜을 따른다. LC3 칩의 대역폭은 1GByte/sec으로 링에서 노드로 들어오는 패킷을 16개까지 담을 수 있는 노드 방향 큐, 노드에서 링으로 내보내는 패킷 역시 16개까지 담을 수 있는 링 방향 큐로 구성되어 있다[10].

LC3의 동작은 다음과 같다. SCI 패킷은 헤더를 가지는데 목적 노드의 주소를 포함하고 있다. 이것을 보고 스트리퍼 블록(striper block : Strip)이 그 패킷이 자신의 노드에 해당하는 것인지의 여부를 판별해서 자신의 노드에 해당하는 것이면 노드 쪽으로 전송하기 위한 큐

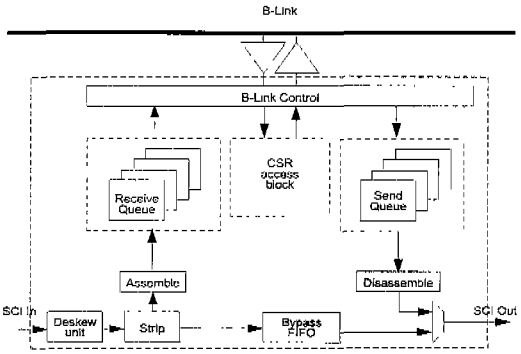


그림 4 LC3 내부 구조도

로 삽입하게 되고 아니면 바이패스 FIFO로 넘겨주게 된다. 만약 큐의 공간에 여유가 없다면 받은 패킷을 없애고 그 패킷을 보낸 노드에게 재시도 하라는 메시지를 보낸다. 바이패스 FIFO는 지역 노드에서 링으로 전송하는 패킷에 대해 우선 순위를 가지므로 바이패스 FIFO가 비었을 경우에만 노드로부터 나오는 패킷을 전송할 수 있다[10]. 그림 4는 LC3의 내부 블록도이다.

이와 같은 LC3의 기능 중에 새로운 구조를 채택하기 위해서는 더 필요한 기능이 있다. 코달 링의 바깥쪽 링에 해당하는 LC3는 위에서 기술한 대로 작동하면 되지만, 코달 링의 안쪽 링에 해당하는 LC3는 위의 기능과 단일 전송 패킷의 방향을 바꾸어 주는 기능이 필요하게 된다. 이 기능은 스트리퍼 블록이 목적 노드 주소를 보고 자신의 노드이거나 자신보다 시계 방향으로 건너뛸 거리 미만의 노드 주소를 가질 때(코드 링크가 반 시계 방향으로 구성되었을 경우), 수신 큐(receive queue)에 패킷을 저장한다. 예를 들어, 건너뛸 거리가 2인 코달 링2 의 경우, (현재 노드번호 + 1) 이하의 주소일 때 수신 큐에 패킷을 저장한다. 그리고 B-링크 제어기는 패킷의 목적 노드 주소를 보아서, 자기 노드이면 이중 링 제어기로 보내고 자기 노드가 아니면 B-링크에 연결된 다른 LC3로 보내주면 된다. 이것이 패킷의 이동 방향을 바꾸는 방법이 된다. 따라서 패킷의 이동 방향을 바꾸어 주는 부분을 LC3에서 지원해 주는 것 이외에는 또 다른 부가적인 하드웨어 지원이 필요 없게 된다. 또한, LC3에서 위와 같은 기능을 제공해 주기 위해서는 LC3의 스트립 모듈에서 비교 연산을 수행하는 부분만 바꾸어 주면 된다.

2.4 캐쉬 일관성 프로토콜

이 장에서는 새로 제안된 시스템에 적용될 캐쉬 일관성 프로토콜에 대한 설명으로 구성된다. 새로 제안된 시스템에서 사용할 캐쉬 일관성 프로토콜은 PANDA II 시스템에서 사용된 캐쉬 일관성 프로토콜[11]을 그대로 적용할 수 있다. 또한 이 캐쉬 일관성 프로토콜은 "지점간 링크를 이용한 이중 링 스누핑 버스 다중 프로세서 시스템의 설계와 검증"[11]에서 증명이 된 것이다.

2.4.1 트랜잭션

2.4.1.1 지역 버스 트랜잭션

지역 버스 트랜잭션은 노드 내의 지역 버스에서 정의되는 트랜잭션으로 그 종류는 표 1과 같다.

표 1 지역 버스 트랜잭션 종류

지역 버스 트랜잭션	트랜잭션 설명
BRL	Bus Read Line
BRP	Bus Read Partial
BRIL	Bus Read Invalidate Line
BIL	Bus Invalidate Line
BWL	Bus Write Line
BWP'	Bus Write Partial
BLR	Bus Locked Read
BLW	Bus Locked Write

2.4.1.2 링 트랜잭션

링 트랜잭션은 지역 버스 제어기와 이중 링 제어기가 주고받는 트랜잭션으로 링을 통해 다른 노드로 전해진다. 또한 링으로부터 이 트랜잭션들을 받게 되면 지역 버스 제어기로 올려보내서 지역 버스 제어기가 처리를 하거나 링 버스 트랜잭션에 해당되는 지역 버스 트랜잭션을 지역 버스로 발생시킨다. 링 트랜잭션의 종류는 표 2와 같다. 지역 버스 트랜잭션이나 링 트랜잭션에 대해 자세한 사항은 "지점간 링크를 이용한 이중 링 스누핑

표 2 링 트랜잭션의 종류

링 트랜잭션	관련 지역 버스 트랜잭션
Read Shared	BRL, BRP
Read Exclusive	BRIL, BIL, BWP(writeback 영역)
Invalidate	BIL
Write Partial	BWP(write through 영역)
Writeback Shared	BWL, BLR
Writeback Exclusive	BWL, BLR
Locked Read	BLR
Locked Write	BLW
Purge	BRIL, BIL
Flush	BRL

버스 다중프로세서 시스템의 설계와 검증"[11]을 참조하면 된다.

2.4.2 L1, L2 캐쉬의 상태 전이

프로세서 모듈에 포함된 L1 및 L2 캐쉬는 Write-Back/Write-Invalidate 스누핑 프로토콜을 사용한다. 이 캐쉬들은 Goodman의 Write-once 프로토콜[12]을 축약한 MSI 프로토콜을 사용하는데, 이는 계층적 캐쉬 구조를 가지는 시스템에서 읽기 전용 라인에 대한 쓰기 요구가 언제나 상위 캐쉬 및 지역 버스로 보여야 하기 때문이다. 또한 MSI 프로토콜은 Illinois 프로토콜[1]을 사용하는 캐쉬 제어기에 대해서도 읽기 요구 시 언제나 공유 상태로 데이타를 전송해서 쉽게 구현할 수 있다.

2.4.3 원격 캐쉬의 상태 전이

원격 캐쉬의 상태 전이는 I, S, MS, M의 네 가지 상태를 둔 Berkeley 프로토콜[1]을 사용한다. Berkeley 프로토콜은 소유권(Ownership)의 개념을 둠으로써 수정 상태에서의 읽기 요청에 대해 캐쉬간 전송을 통해서 데이타를 전송해준다. 메모리에 대한 갱신은 수정된 캐쉬 라인 교체 시에만 발생한다. 원격 캐쉬에서 Berkeley 프로토콜을 사용하는 이유는 링으로의 트랜잭션 수를 조금이라도 줄이기 위함이다. 원격 캐쉬의 상세한 상태 설명과 상태 전이도는 "지점간 링크를 이용한 이중 링 스누핑 버스 다중 프로세서 시스템의 설계와 검증"[11]을 참조하면 된다.

2.4.4 메모리 디렉토리의 상태 전이

메모리 디렉토리는 C(clean), G(gone), S(shared), L(locked)의 상태를 가지고 전이한다. 이 상태를 보고 현재 응답의 책임(소유권)이 홈노드에 있는지 아니면 다른 노드에 있는 지를 결정한다.메모리 디렉토리의 상세한 상태 설명과 상태 전이도는 "지점간 링크를 이용한 이중 링 스누핑 버스 다중 프로세서 시스템의 설계와 검증"[11]을 참조하면 된다.

3. 코달 링의 구성과 특징

이 장에서는 코달 링의 여러 구조들을 소개하고 이들의 정성적인 분석을 통해 이중 링과의 성능 차이 등을 예측해 본다.

3.1 이중 링 (코달 링1)

이중 링은 그림 5처럼 건너뛰 거리가 1인 코드 링크를 갖는 구조이다. 이 구조가 현재 PANDA II 시스템의 구조이며, 이 구조에서 단일 전송 패킷은 현재 자기 노드와 목적 노드와의 거리를 시계 방향(②식)과 반 시계 방향(①식)으로 계산하여 가까운 쪽으로 패킷을 실어

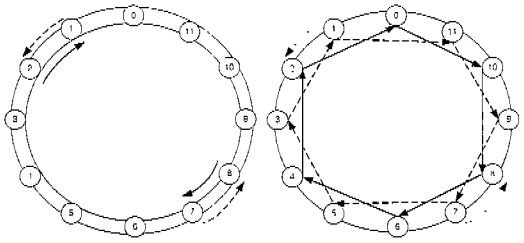


그림 5 이중 링(코달 링1) 그림 6 노드 수 12인 코달 링2

보내게 된다. 이 구조의 장점은 라우팅이 간단하고 확장성이 있다는 것이다. 그러나 이 구조에서 단일 전송 패킷의 링 지연 시간은 최대 n/2 번의 홉이 된다(n은 노드의 개수). 또한 이와 같은 긴 지연 시간 때문에 링의 이용률이 높을 것으로 생각된다. 이 구조에서 자기 노드에서 목적 노드까지의 거리(distance)는 다음 식에 의해 구할 수 있다. (Ntotal는 전체 노드 수, Nsrc는 소스 노드 번호, Ndest는 목적 노드 번호)

① 반시계 방향 :

```
distance=(Ntotal+(Ndest-Nsrc)) % Ntotal;
```

② 시계 방향 :

```
distance=(Ntotal-(Ndest-Nsrc)) % Ntotal;
```

3.2 코달 링 2

코달 링2는 그림 6과 같이 바깥쪽에서 모든 노드를 연결한 링 링크와 안쪽의 건너뛰 거리가 2인 코드 링크로 이루어져 있다. 단일 전송 패킷을 보낼 때는 현재 노드에서 목적 노드와의 거리를 시계 방향(③식)과 반 시계 방향(①식)으로 전송할 때의 홉 수를 계산하여 홉 수가 적은 방향 링크로 패킷을 보낸다. 이때 안쪽 링크를 따라 가다가 목적 노드를 건너뛰게 되면, 현재 도착 노드에서 노드 안의 B-링크를 타고 다른 링크 제어기로 이동하여 바깥쪽 링크에 패킷을 실어 시계 방향으로 이동하게 된다. 이때 바깥쪽 링크의 시계 방향으로 가는 홉은 최대 1이 된다.

③ 코달링 2, 시계 방향 :

```
distance=((Ntotal+(Nsrc-Ndest)) % Ntotal) / 2;
if ((Nsrc-Ndest) % 2 != 0)
        distance=distance+2;
```

이 구조에서 바깥쪽의 링 링크는 방송 패킷을 모두 전송해야 하고, 때때로 단일 전송 패킷을 전송해야 하기 때문에, 부하가 과다하게 몰릴 염려가 있으나, 통상 방송 패킷은 데이타가 없어서 링 지연 시간이 짧기 때문에 이중 링에 비해서 높은 링 이용률을 가질 것 같지는 않다. 코달 링2는 이중 링의 장점인 확장성을 가지면서

링 지연 시간을 줄일 수 있기 때문에, 노드 수가 20 정도 이하인 구조에서 가장 좋을 것으로 생각된다.

3.3 코달 링 3과 코달 링 4

그림 7은 코드 링크의 건너뜀 거리를 3으로 연결한 구조이다. 이 구조는 노드 수가 많아짐에 따라, 이중 링과 코달 2의 링 지연 시간보다 더 적은 링 지연 시간을 가질 것으로 생각된다. 그러나 안쪽 코드 링크를 돌다가 목적 노드를 건너뛰게 되면 최대 2 홉의 바깥 링 링크를 타야 하기 때문에 바깥 링 링크의 이용률이 증가할 수도 있을 것 같다.

그림 8은 코드 링크의 건너뜀 거리를 4로 연결한 구조이다. 이 구조는 처음의 노드 수가 작을 때는 안 좋은 성능을 보이다가 노드 수가 아주 많아지면 코달 링 3보다 링 지연 시간이 감소할 것으로 생각된다. 그러나, 코달 링 4의 경우 목적 노드를 건너뛸 경우 최대 3 홉의 바깥 링 링크를 타야 하기 때문에 바깥 링 링크의 이용률이 더욱 증가하여 링 이용률 측면에서 좋은 구조가 될 것 같지는 않다. 이 구조들의 반 시계 방향의 홉 수 계산은 이중 링과 동일하고, 시계 방향으로 코드 링크를 이용할 경우의 홉 수 계산은 다음과 같다.

④ 코달링 3, 시계 방향 :

```
distance=((Ntotal+(Nsrc-Ndest)) % Ntotal) / 3;
if (((Nsrc - Ndest) % 3) != 0)
distance=distance+4-((((Nsrc-Ndest)+Ntotal) %
        Ntotal) % 3);
```

⑤ 코달링 4, 시계 방향 :

```
distance=((Ntotal+(Nsrc-Ndest)) % Ntotal) / 4;
if (((Nsrc - Ndest) % 4) != 0)
distance=distance+5-((((Nsrc-Ndest)+Ntotal) %
        Ntotal) % 4);
```

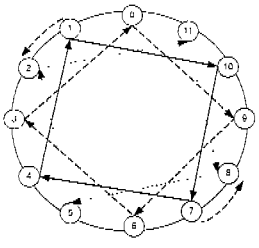


그림 7 코달 링3 (n=12)

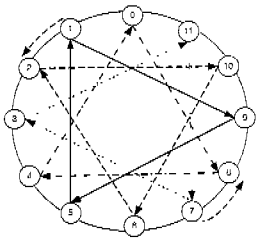


그림 8 코달 링4 (n=12)

4. 모의실험 및 성능 분석

이 장에서는 모의실험 환경 및 인자에 대해 설명하고 모의실험 결과를 분석하도록 한다.

4.1 모의실험 환경 및 인자

본 논문에서는 앞에서 제안한 코달 링 시스템과 이중 링 시스템을 모의실험하기 위한 도구로 확률 구동(probability-driven) 시뮬레이터인 SES/Workbench를 사용한다. SES/Workbench는 큐잉 모델에 기반 하여 모델링 되어 있는 시스템의 성능을 분석하기 위해 과학이나 공학 분야에서 쓰이고 있는 모의실험 도구이다[13]. SES/Workbench의 입력 값은 다음과 같이 구하여 모의실험 하였다.

모의실험에서 SES/Workbench의 입력으로는 SPLASH-2에서 제공하는 프로그램 중 FFT, Radix, LU, Barnes를 프로그램 구동(program-driven) 방식의 시뮬레이터인 MINT[14]에서 수행하여 나온 결과치를 사용하였다. FFT는 FFT(Fast Fourier Transform)를 수행하는 프로그램이고, LU는 행렬의 LU 변환을 수행하는 프로그램이고, Radix는 기수 정렬을 수행하는 프로그램이고, Barnes는 3차원 우주에서 중력에 의한 여러 입자의 행동을 시뮬레이션 하는 프로그램이다[15]. 프로그램의 인자들은 표 3과 같다.

표 3 MINT에서 사용된 응용 프로그램의 인자

응용 프로그램	인자
FFT	262144 Complex Doubles
Radix	2M integers, radix 1024
LU	512*512 matrix, 16*16 blocks
Barnes	8K particles

표 4 MINT 모의 실험의 시스템 관련 인자

인 자	값
프로세서 수	32
한 노드내의 프로세서 수	2
노드 수	16
프로세서 캐시의 크기	16KB
원격 캐시의 크기	512KB
프로세서 클럭 속도	500MHz
노드간 링 연결 클럭 속도	500MHz
프로세서 캐시 접근 시간	2 프로세서 클럭
버스 요청 명령 전송 시간	9 버스 클럭
버스의 단위 블록 데이타 전송 시간	18 버스 클럭
링 구조에서 인접 노드간 요청 명령 전송 시간	100링 클럭
링 구조에서 인접 노드간 블록 데이타 전송 시간	200링 클럭

또한, MINT의 입력으로 사용한 시스템 관련 인자는 표 4와 같다. 표 3과 표 4의 인자들을 사용하여 MINT를 수행하고, 여기서 나온 결과치를 이용하여 이중 링 제어기와 그 아래 부분인 B-링크, 링크 제어기, 전역 링크 등

을 SES/Workbench를 이용하여 모델링 하였다. MINT를 수행하고 나온 결과치는 트랜잭션의 종류별 발생 빈도수였고, 이 결과를 이용하여 트랜잭션 종류별로 발생 확률 값을 구했으며 그 값은 표 5와 같다. 그리고 SES/Workbench에서는 이렇게 구한 확률 값만큼 트랜잭션을 생성시켰고, 트랜잭션의 작동을 실제 PANDA 시스템에서의 작동과 똑같이 하였다. 또한, SES/Workbench의 모의실험에서 사용된 인자는 표 6과 같다.

표 5 트랜잭션 종류별 발생 확률 값

트랜잭션 종류	확 률
Read request	0.66353
Flush request	0.03872
Read request (for write)	0.13681
Purge request	0.00305
WriteBack request	0.10729
Invalidation request	0.05060
Request 소계	1
Read response	0.74766
Flush response	0.06674
Write response	0.18080
Purge response	0.0048
Response 소계	1

표 6 SES/Workbench에서 사용된 인자

인 자	값
B-링크 클럭	155MHz
전역 링크 클럭	500MHz
노드 개수	12, 24, 36, 48
SCI 패킷 헤더	32 Byte
캐쉬 라인 크기	64 Byte
코달 링의 코드 링크의 건너뜀 거리	2, 3, 4

모의실험한 대상은 이중 링(코달링 1으로 볼 수도 있음), 코달링2 (코드 링크의 건너뜀 거리가 2), 코달링3 (코드 링크의 건너뜀 거리가 3), 코달링4 (코드 링크의 건너뜀 거리가 4)와 같이 모두 4가지의 구조이다.

4.2 모의실험 결과 및 분석

모의 실험 결과는 4가지 구조에 대해 요청 트랜잭션의 응답 지연 시간, 응답 트랜잭션의 응답 지연 시간을 제시하고, 링크 이용률, TPS(Transaction Per Second)을 제시하도록 한다. 또한 제시된 모의 실험 결과를 분석하여 각 모델의 특성을 생각해 보기로 한다.

4.2.1 시스템별 응답 지연 시간의 변화

그림 9는 노드 개수 변화에 따른 요청 트랜잭션과 응답 트랜잭션의 평균 응답 지연 시간의 변화를 나타낸 것이다. 응답 지연 시간은 요청 트랜잭션과 응답 트랜잭션으로 나누어 측정하였다. 그림 9에서 노드의 개수가 증가하게 되면 전반적인 시스템들의 응답 지연 시간이 늘어나게 된다. 특히, 이중 링의 경우 노드 수와 링 이용률이 거의 선형적인 관계를 보여준다. 이는 노드 개수가 증가함에 따라 트랜잭션이 더 많은 링크를 순회하기 때문에 링에서 보내는 시간이 많아지기 때문이다. 코달링 2, 코달링 3, 코달링 4의 경우도 노드 수가 증가함에 따라 응답 지연 시간이 증가하나, 이중 링에 비하여 완만한 기울기를 보여주고 있다.

노드 수가 12개인 경우 코달링 2의 지연 시간이 가장 짧은 것으로 나타나 20개 이하의 작은 노드 개수에서는 코달링 2의 응답 지연 시간이 가장 좋은 것으로 생각된다. 그러나, 노드 수가 48개로 늘어남에 따라 코달링 3에서 코달링 4로 응답 지연시간이 가장 적음이 관찰된다. 노드 수가 48개인 경우에는 코달링 4의 응답 지연 시간이 가장 적음이 관찰됨으로써, 노드 수가 많아짐에 따라 건너뜀 거리가 높은 것이 좋은 것으로 생각된다. 한편, 아주 작은 노드 개수(8개 이하)일 경우는 이중 링의 성능이 좋을 것으로 생각할 수도 있다.

그림 9를 보면 응답 트랜잭션의 경우 코달링이 이중 링에 비해 거의 대부분 응답 지연 시간 측면에서 좋은 성능을 나타냈다. 응답 트랜잭션의 응답 지연 시간 감소는 코달링 2의 경우 25% 정도이며, 코달링 3의 경우 30%, 코달링 4의 경우 노드 48개에서 35% 정도이다. 이 결과는 코달링이 단일 트랜잭션의 링 지연 시간을 감소시키려는

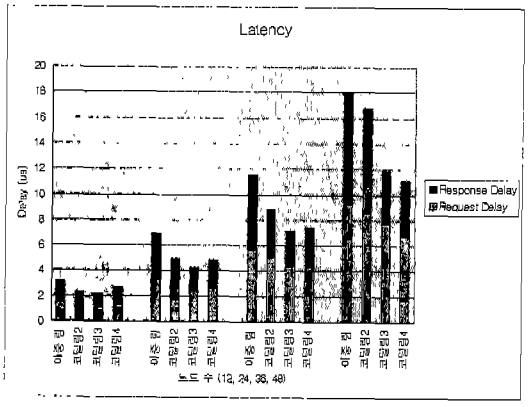


그림 9 응답 지연시간

의도에 잘 일치하는 것으로 예측이 맞았음을 나타내지만 예상보다 높은 수치이다. 그리고, 요청 트랜잭션의 경우는 코달링에서 바깥쪽 링 링크에 트랜잭션이 몰려 나쁜 성능을 나타내리라는 예상과는 달리 노드 수가 증가함에 따라 요청 트랜잭션의 응답 지연 시간도 감소하게 된다. 이는 노드 수가 증가함에 따라 코달링의 바깥쪽 링의 이용률이 낮아지는 것과 요청 트랜잭션에 포함된 단일 전송 패킷(Writeback, Flush request 등)의 응답 지연 시간이 작아짐에 따른 것으로 분석된다.

4.2.2 시스템별 링 이용률 변화

그림 10과 그림 11은 노드 개수 변화에 따른 링크 이용률의 변화를 나타낸 것이다. 그림 10은 링 링크의 이용률을 나타낸 것이고, 그림 11은 코드 링크의 이용률을 나타낸 것이다. 그림 10과 그림 11을 보면 노드 수가 늘어남에 따라 링의 이용률이 전반적으로 증가한다. 이는 전반적인 시스템의 병목 지점이 전역 링크임을 의미한다. 또한, 코달링의 경우 바깥쪽 링크의 링 이용률이 안쪽 코드 링크의 링 이용률보다 높음을 나타내고 있다.

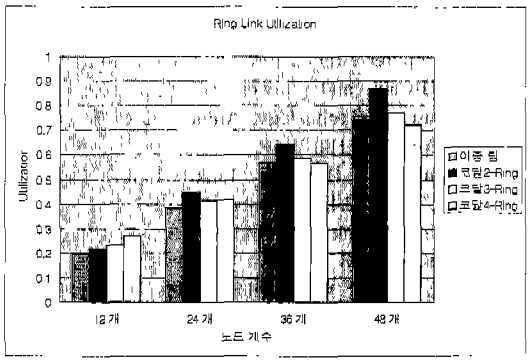


그림 10 링 이용률 (링 링크)

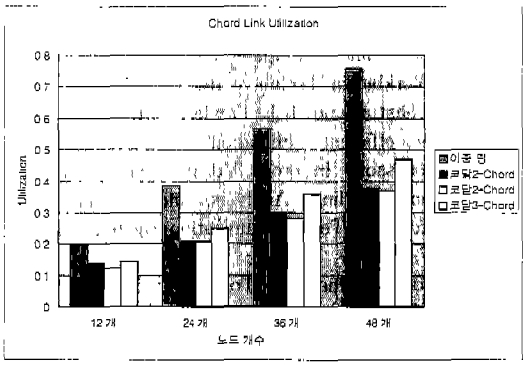


그림 11 링 이용률 (코드 링크)

하지만, 이 차이는 노드 수가 12개인 경우 코달링 2가, 노드 수 24개인 경우 코달링 3이, 노드 수 36, 48 개인 경우 코달링 4의 안과 밖의 링크 이용률의 차이가 작아져서 노드 수에 맞는 최적의 코달링 구조를 선택할 수 있다. 또한, 노드 수 36개부터는 코달링의 링 이용률이 이중 링보다 작아지게 되어 더 좋은 링 이용률을 나타낸다. 이는 코달링 3과 코달링 4의 링 지연시간이 감소하여 트랜잭션이 링에 머무르는 시간이 작아지기 때문에 전체 링크 이용률이 작아지는 것이라고 생각된다.

노드 수가 증가함에 따라, 코달링의 건너뜀 거리가 증가함에 따라 더 많은 단일 전송 패킷들이 바깥쪽의 링크보다는 안쪽의 링크로 이동하는 것이 유리하게 되기 때문에 안쪽 링크의 이용률이 높아지는 것 같다. 따라서 건너뜀 거리가 커질수록 바깥 링크에 패킷이 몰릴 것이라는 예상은 빗나가게 되었다.

4.2.3 시스템별 성능 변화

그림 12는 시스템별로 구한 TPS(Transaction Per Second)를 이중 링의 TPS에 정규화시켜 나타낸 그림이다. 그림 12에서 보면 노드 수가 늘어남에 따라 코달링의 성능이 월등히 좋아진다. 특히 노드 수가 많아짐에 따라, 코달링의 건너뜀 거리가 클수록 성능 향상이 높은데, 이는 응답 지연 시간이 전체 시스템의 병목 현상이었음을 말해 주는 것이라 생각된다. 노드가 12개인 경우는 코달링 4의 성능이 오히려 나쁜데, 이것은 바깥쪽의 링 이용률이 높아 대기하는 시간이 많기 때문이라고 생각된다. 따라서 노드 수가 12개 이하인 경우는 코달링 2의 성능이 가장 최적이라고 생각된다. 또한 노드 수가 36개인 경우는 코달링 3의 성능이 가장 우수하고 성능 향상은 이중 링에 비해 30% 정도 된다고 생각한다. 노드 수 48개에서는 코달링 4의 성능이 가장 우수한 것으로 분석된다. 특히, 노드 수 48개의 코달링 4의 경우는

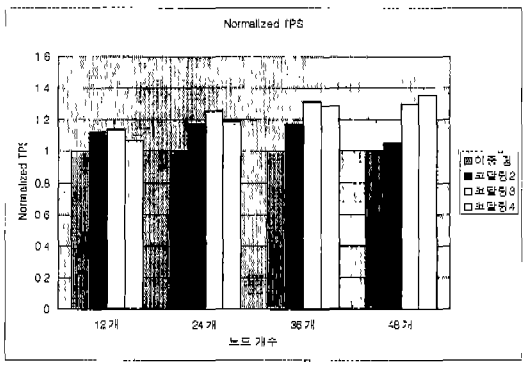


그림 12 시스템별 성능 변화

이중 링에 비해 38% 정도의 성능 향상이 관측된다.

5. 결론 및 향후 과제

일반적으로 NUMA 구조의 상호 연결 망으로 사용되어 오던 버스 구조가 물리적 확장성 및 대역폭에서 그 한계를 나타내었기 때문에 버스 구조의 단점을 보완할 수 있는 상호 연결망인 고속의 지점간 링크를 이용한 시스템들이 개발되고 있다. 그러나 지점간 링크를 사용한다는 기본적 특성 때문에 노드 개수가 증가하면 많은 링크를 거쳐야 하며, 이에 따라 응답 지연 시간이 증가하는 단점을 안고 있다.

본 논문에서는 PANDA 시스템에서 사용된 이중 링 구조가 노드 수의 증가에 따른 응답 지연 시간 증가의 문제점을 해결하기 위해 코달 링 구조의 도입을 제안하였고, 이 구조에 필요한 링크 제어기 구조를 제시하였다. 코달 링 구조의 시스템은 기존의 노드 구조나 캐쉬 일관성 프로토콜의 수정 없이 링크의 연결만 바꾸어주면 되는 구조이기 때문에 구현이 복잡하지 않다. 따라서 링의 장점인 확장성과 단순성(simplity)을 그대로 유지하게 된다.

이중 링과 여러 구조의 코달링을 모의실험한 결과, 노드 수가 작은 경우(노드 수 12)에는 코달링2 가 좋은 성능을 나타내었다. 그러나 노드 수 12개와 24 개에서 링 이용률은 이중 링이 가장 좋게 나타났으나, 코달 링과의 차이가 작았다. 노드 수 24, 36 개인 경우는 코달링3 이 가장 좋은 성능을 나타내었다. 이때, 이중 링과의 링 이용률 차이도 거의 비슷하게 나타났다. 노드 수 48개 이상인 경우는 코달링4가 가장 좋은 성능을 나타냈으며, 링 이용률 측면에서도 가장 좋은 성능을 나타내었다. 따라서, 노드 수가 증가함에 따라 코달링의 코드 링크의 건너뛰 거리가 증가해 가는 구조가 유리해진다.

본 논문에서는 기존 링크 제어기에서 패킷 전송의 방향을 바꾸게 해 주는 기능이 필요하다고 가정하였다. 그러나, 이 기능을 링크 제어기가 아닌 이중 링 제어기에서 지원해 주는 방법에 대한 연구도 진행되어야 할 것으로 생각된다. 이 과제만 해결된다면 기존 PANDA 시스템에서 새로운 비용이 들지 않고도 응답 지연 시간이 월등히 작은 시스템을 이용할 수 있을 것으로 생각되기 때문이다.

참 고 문 헌

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

[2] 장병순, "PANDA 시스템에서 링 대역폭 확장을 위한 효율적인 방안", 서울대학교 석사학위 논문, 1999.

[3] Sung Woo Chung, Seong Tae Jhang and Chu Shik Jhon, "PANDA : Ring-Based Multiprocessor System using New Snooping Protocol," In The Proceeding of ICPADS'98, pp 10-17, Dec. 1998.

[4] IEEE Computer Society, "IEEE Standard for Scalable Coherent Interface(SCI)," Institute of Electrical and Electronics Engineers, August 1993.

[5] Tom Lovett and Russell Clapp, "STiNG : A CC-NUMA Computer System for the Commercial Marketplace," In Proceedings of the 23th International Symposium on Computer Architecture, pp. 308-317, May 1996.

[6] Z. Vranesic, S. Brown, M. Stumm, S. Caranci, A. Grbie, R. Grindley, M. Gusta, O. Krieger, G. Lemieux, K. Loveless, N. Manjikian, Z. Zilic, T. Abdelrahman, B. Gamsa, P. Pereira, K. Sevcik, A. Elkateeb, S. Srbljic, "The NUMAchine Multiprocessor," Department of Computer Science, Toronto Univ., 1995

[7] L. Barroso and M. Dubois, "The Performance of Cache-Coherent Ring-based Multiprocessors," In Proceedings of the 20th International Symposium on Computer Architecture, pp.268-277, May 1993.

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

[9] Dolphin, "A Backside Link(B-Link) for SCI nodes," Draft 2.4, September 21, 1995.

[10] Dolphin, "Link Controller 3 Specification," November, 1998.

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

[12] J. R. Goodman, "Using Cache Memory to Reduce Processor-Memory Traffic," In Proceedings of the 10th International Symposium on Computer Architecture, pp. 124-131, June 1983.

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

[14] J.E. Veenstra and R.J. Fowler. "MINT: a front end for efficient simulation of shared-memory multiprocessor". In proc. 2nd International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pages 201-207, 1994.

[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.

윤 주 범

1999년 2월 고려대학교 이학사. 2001년 2월 서울대학교 대학원 컴퓨터공학과 석사. 2001년 3월 ~ 현재 한국전자통신연구원 부설 국가보안기술연구소 연구원. 관심분야는 컴퓨터 구조, 정보보안.

장 성 태

1986년 2월 서울대학교 전자계산기공학과 공학사. 1988년 2월 서울대학교 대학원 컴퓨터공학과 석사. 1994년 2월 서울대학교 대학원 컴퓨터공학과 박사. 1994년 3월 ~ 현재 수원대학교 전자계산학과 교수. 관심분야는 다중 프로세서컴퓨터구조, 병렬처리, 캐쉬 일관성 유지 프로토콜

전 주 식

1975년 2월 서울대학교 응용수학과 학사. 1977년 2월 한국과학기술원 전산학과 석사. 1983년 2월 미국 Univ. of Utah 박사. 1983년 ~ 1985년 Univ. of Iowa 조교수. 1985년 ~ 현재 서울대학교 컴퓨터공학과 교수. 1994년 ~ 현재 컴퓨터신기술공동연구소 소장. 관심분야는 컴퓨터 구조, 병렬처리, VLSI/CAD