리피터 노드를 이용한 Scalable CC-NUMA 시스템

(Scalable CC-NUMA System using Repeater Node)

경 진 미 † 장 성 태 ††

(Jin Mi Kyoung) (Seong Tae Jhang)

요 약 CC-NUMA구조에서는 원격 메모리에 대한 접근이 불가피한 구조적인 특성 때문에 상호 연결망이 성능을 좌우하는 큰 변수로 작용한다. 기존에 사용되는 버스는 대역폭의 한계와 물리적 확장성 때문에 대규모의 시스템에는 적합하지 않다. 이를 대체하는 고속의 지점간 링크를 도입한 이중 링 구조는 이러한 버스의 한계를 극복하고는 있지만 많은 노드를 거쳐야 하는 문제로 인해 응답 지연 시간이 증가하는 단점을 안고 있다. 본 논문에서는 요청과 응답 패킷의 지연 시간을 줄이는 방안으로 리피터 노드를 이용한 다중 링을 제안한다. 제안된 시스템은 링과 링 사이의 구조가 대칭형을 이루고 있어 요청을 내보내는 링을 제외한 다른 링의 hop수는 똑같은 수치를 갖고 있으며, 이중 링에 비해 최대의 hop수와 최소의 hop수의 차가 적고 평균 hop수 또한 적어 좋은 성능을 보인다. 본 논문에서는 또한 이러한 구조를 유지하기 위한 리피터 노드의 구조를 제안하며 리피터 노드의 구조와 노드의 확장에 따른 다양한 성능을 확률 구동 시뮬레이터를 사용하여 평가를 수행한다.

키워드 : 다중 프로세서, 이중 링, CC-NUMA

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

Key words : multiprocessor, dual ring, CC-NUMA

1. 서 론

최근의 공유 메모리 다중 프로세서 시스템에서, CC-NUMA 시스템이 널리 사용되고 있는 것은 확장성과 프로그래밍이 용이하다는데 기인한다[1,2]. 하지만, 시스템의 크기가 증가함에 따라 원격 메모리로의 빈번한 접근은 메모리 접근 지연 시간(memory access latency)을 증가시킨다. 원격 메모리로의 접근 횟수를 줄이거나 접근 속도를 높이는 것이 CC-NUMA 구조의 성능 향상을 위해 필수적이다[3,4]. 현재 공유 메모리 다중 프로세서 시스템에 가장 널리 사용되는 상호 연결망은 공유 버스이다. 공유 버스는 구현상의 복잡도가 비교적 낮고 비용이 적게 드는 장점을 가지지만, 확장성 및 대역폭의 한계를 이유로 주로 소규모의 시스템에서 사용된다.

공유 버스의 단점을 보완하기 위해 최근에 널리 사용되는 상호 연결망이 이중 링(dual ring)이다. 이중 링은 지점간 링크를 사용하여 노드들을 양방향으로 연결함으

· 본 연구는 한국과학재단 2001년 목적기초연구(지역대학 우수과학자 지원 연구. 과제번호 : R05-2001-000-00983-0) 지원으로 수행되었음.

† 정 회 원 : (주)써머스테크놀로지 솔루션 다비전
jinmi35@hananet.net
†† 종신회원 : 수원대학교 컴퓨터학과 교수
stjhang@mail.suwon.ac.kr

논문접수 : 2002년 1월 23일
심사완료 : 2002년 7월 19일

로써 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 방법[12, 13]을 사용하였으나, 스누핑 캐쉬 일관성 유지 방법을 사용한 이중 링 구조도 제안된 바 있으며[6,7], 이것의 성능이 디렉토리 캐쉬 일관성 유지 방법을 사용한 구조에 비해 우수함이 발표되었다[6,7].

서울대학교의 PANDA 연구실에서 제안한 PANDA II 시스템[9]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. 이러한 이중 링 구조의 CC-NUMA 시스템은 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 시간이 증가하게 된다. 이중 링 구조가 가지는 확장성에서의 장점을 살리기 위해서는 노드 수의 증가에 따른 접근 지연 시간의 증가를 줄이며, 지속적인 성능 향상을 꾀할 수 있는 방안을 강구해야 한다.

본 논문에서는 이러한 문제를 해결하기 위해 스누핑 기반 이중 링 CC-NUMA 시스템인 PANDA II의 구조를 개선하여 방송 패킷을 동시에 중첩해서 여러 노드로 전송함으로써 요청의 전송 시간을 줄이고, 빠른 응답 시간을 제공할 수 있는 새로운 CC-NUMA 시스템을 제시하고자 한다. 본 논문에서 제시하는 시스템의 가장 큰 장점은 더 많은 노드를 장착할 수 있는 확장 가능한 구조라는 것이다. 2장에서는 기존의 PANDA II 구조에 대해 간단한 소개와 문제점을 기술하고 본 논문에서 제안한 시스템의 개념과 동작에 대해 살펴본다. 3장에서는 본 논문에서 제시한 리피터(Repeater)노드의 구조와 리피터 노드를 사용한 Scalable CC-NUMA 시스템의 동작에 대해 살펴보고 이를 정량적으로 분석하며, 4장에서는 PANDA II 시스템과의 성능 비교를 수행한다. 5장에서는 제안한 시스템의 성능 평가를 수행하여 기존의 구조에서 개선된 점을 유도하고, 6장에서는 이를 바탕으로 결론을 맺는다.

2. PANDA II 시스템의 분석

스누핑 캐쉬 일관성 프로토콜을 사용하는 CC-NUMA 구조의 PANDA II 시스템의 전체 구조는 그림 1과 같다. 각 노드는 단방향 지점간 링크 두 개를 이용한 방향 분리 이중 링으로 연결된다. 요청을 전송하는 방송 패킷의 경우, 목적지 메모리 블록의 주소가 홀수이면 시계 방향의 링을 통해, 짝수이면 반시계 방향의 링을 통해 전송된다. 요청에 대한 응답 패킷의 경우에는, 목적지 노드가 결정되어 있으므로 목적지 노드의 ID를 이용하여 가까운 방향의 링을 통해 전송된다.

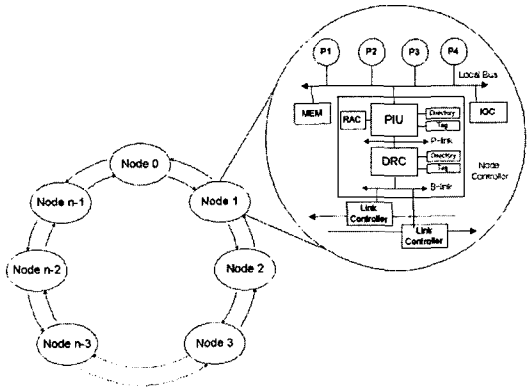

그림 1 PANDA II 시스템의 전체 구조도

PANDA II 시스템에서의 링은 스누핑 방식으로 캐쉬 일관성을 유지하고, 방송 트랜잭션을 지원하므로 논리적으로 버스와 동일한 동작을 수행한다. 각 노드에는 4개의 프로세서 모듈, 지역 메모리, I/O 제어기와 노드 제어기가 지역 버스에 연결되어 있다. 노드의 지역 버스는 스누핑 캐쉬 일관성 프로토콜에 의해서 동작하며, 요청 트랜잭션과 응답 트랜잭션을 분리하는 split 버스이다. PANDA II 시스템에서 사용하는 링크 제어기는 IEEE SCI 표준[8] 규약을 따르며 지정한 500MHz를 지원하는 Dolphin 社의 LC3 칩[15,16]을 사용한다. LC3 칩은 링크 쪽의 대역폭은 800MByte/s를 양방향으로 지원하며 링에서 노드로 들어오는 패킷을 16개까지 담을 수 있는 노드방향 큐, 노드에서 링으로 내보내는 패킷 역시 16개까지 담을 수 있는 링 방향 큐로 구성되어 있다. 이중 링 제어기와는 B-link를 통해서 패킷을 전달하는데 B-link는 64 bit, 155MHz의 동작까지 지원하며 1240MByte/s의 대역폭을 가진다

PANDA II 시스템에서의 각 노드 제어기는 이중 링 제어기와 링크 제어기(LC3)가 서로 패킷을 주고받을 수 있는 버스 형태의 상호 연결망인 B-link와 이중 링 제어기(Dual Ring Controller : DRC), 지역 버스 제어기(Processor Interface Unit : PIU), 원격 캐쉬(Remote

Access Cache : RAC), 메모리의 상태를 관리하는 메모리 디렉토리와 원격 캐쉬의 상태를 관리하는 원격 캐쉬 태그가 각각 PIU와 DRC에 달려있는 형태로 구성되어 있다.

표 1과 표 2는 PANDA II 시스템에 적용된 캐시 일관성 프로토콜에 사용된 지역 버스 트랜잭션과 링 트랜잭션의 종류를 기술한 것이며 캐쉬 일관성 프로토콜의 자세한 사항은 참고 문헌 [7,9]에 기술되어 있다.

표 3 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

표 4 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

상술한 PANDA II 시스템의 가장 큰 문제점은 노드의 수가 많아지면 노드 간 전송에 따른 응답 지연 시간이 길어져 고속 지점간 링크의 장점을 살릴 수가 없게 된다. 예를 들어 노드 개수가 24개일 때의 최대 응답 지연 시간은 N/2(N: 노드 수) 즉 12개의 hop을 거쳐야 하며, 평균 응답 시간은 6 hop을 차지하게 되며, 노드의 개수가 증가할수록 응답할 때까지 거쳐야하는 hop의 수가 많아지게 된다. 이러한 문제를 해결하는 방안으로 본 논문에서 제시한 시스템은 표 1과 표 2에서와 같은 메모리 블록에 대한 요청을 포함하는 다양한 방송 패킷이 리피터 노드를 지날 때 다른 링의 노드로 중첩해서 전송하여 응답 시간을 보다 적은 hop 수로 줄일 수 있으며 보다 나은 확장성을 제공할 수 있다.

3. 개선된 스누핑 기반 이중 링 구조의 CC-NUMA 시스템

3.1 리피터 노드를 이용한 구성

본 논문에서 제시하는 Scalable CC-NUMA 시스템의 기본 개념은 PANDA II에서 사용된 스누핑 캐쉬 일관성 프로토콜을 그대로 사용하는 동시에, 링을 기존의 한 개의 이중 링에서 여러 개의 이중 링으로 나누어 그룹화하여 링 내의 노드들 간의 hop수를 줄이고자 한다. 링으로 보내는 방송 패킷이 링과 링을 연결하는 노드를 지날 때 이 노드가 이 방송 패킷을 두 개의 링으로 동시에 전송하여 전송 시간의 중첩을 유도하며, 이러한 특징이 기존의 PANDA II 보다 좋은 성능을 나타내게 된다. 이때 링과 링을 연결하는 노드가 본 논문을 통해 제시하는 리피터 노드이다.

그림 2는 노드의 개수가 12개인 PANDA II 구조를 예시한다. PANDA II는 참조하려는 메모리 블록을 요청하는 방송 패킷의 경우 0번 노드에서 11번 노드까지 어느 노드에서 시작을 해도 전체 노드를 한번씩은 방문하며, 이 예의 경우 평균 12번의 hop을 꼭 지나게 된다. 그 요청에 대한 응답 패킷의 경우에는 해당 요청을 보낸 노

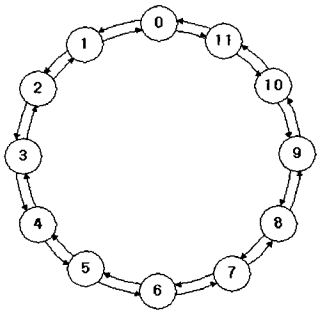

그림 2 PANDA II의 링 구조(노드수:12)

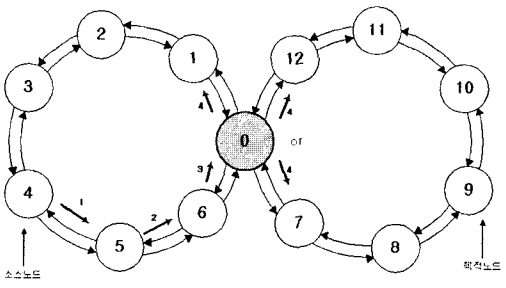

그림 3 리피터 노드(0번 노드)를 이용한 Ring2(2는 링의 개수)

드까지의 최단 경로를 찾아가도록 구성되어 있어 최대 N/2(N:노드의 수) hop을 거치게 되며, 그림 3의 예에서는 최대 hop 수가 6, 평균 hop 수가 3을 나타낸다.

그림 3은 PANDA II 시스템의 요청과 응답에 걸리는 지연 시간을 줄이기 위해 본 논문에서 제시하는 리피터 노드를 이용하여 다시 구성한 이중 링 구조의 CC-NUMA 시스템을 예시한다. 리피터 노드는 2개의 링을 연결하는 노드로 하나의 노드에서 시작한 방송 패킷을 자신의 링 뿐만 아니라 다른 링으로 전송하는 역할과 다른 링을 한 바퀴 회전한 방송 패킷을 그 링에서 제거하는 역할을 수행한다. 리피터 노드를 거치면서 방송 패킷은 두 개의 링으로 동시에 전송되므로 빠르게 방송 패킷을 전송할 수 있으며, 이에 따라 빠른 응답을 기대할 수 있다.

그림 3은 소스 노드(요청을 보낸 노드)가 4번이며, 요청을 담고 있는 방송 패킷의 전송 방향은 시계 반대 방향인 경우를 예시한다. 4번 노드에서 출발하여 5번, 6번 노드를 거쳐 0번 리피터 노드를 거치면서 방송 패킷은 1번과 7번 혹은 1번과 4번 노드로 중첩된 시간에 전송된다. 소스 노드가 포함되지 않은 링도 이 방송 패킷을 반시계 방향으로 전송한다고 가정할 때 1번 노드와 7번 노드, 2번 노드와 8번 노드, 3번 노드와 9번 노드는 거의 중첩된 시간에 이 방송 패킷을 전송하며, 이는 요청 지연 시간을 줄이고 빠른 응답을 얻는 효과를 얻을 수 있다. 물론 링을 돌고 있는 패킷이 없을 경우 같은 시간에 전송하는 것이 가능하고 이미 전송중인 패킷이 있다면 그 패킷이 지난 후 큐에 놓인 순서대로 링을 돌게 된다. 그러나 이런 제약사항은 기존의 PANDA II에서도 동일하며, 제시한 구조가 PANDA II 보다 빨리 요청을 방송할 수 있게 된다. 응답 패킷도 기존의 PANDA II처럼 가장 짧은 경로를 통해 오게 되는데 그림 3의 예에서는 목적 노드인 9번에서 시작해 8번, 7번, 0번, 6번, 5번, 그리고 소스 노드인 4번으로 응답한다. 이 구조에서 리피터를 하나의 노드처럼 생각해 hop 수를 계산할 때 포함시켰지만 리피터 노드는 기존의 노드와는 많은 구조상의 차이를 보이며, 상술한 바와 같은 단순한 기능을 수행한다. 그림 3은 리피터 노드가 hop 수에 포함되므로 하나의 노드처럼 고려해 번호를 부여했다.

3.2 리피터 노드를 이용한 다른 구성 예

리피터 노드와 링의 수에 변화를 주면서 더 좋은 성능을 보이는 구조를 찾는 방안으로서 그림 4, 그림 5, 그리고 그림 6이 제시될 수 있다.

그림 4는 리피터 노드를 두 개 사용한 구조이다. 예를 들면 4번의 소스 노드가 전송한 방송 패킷을 5번 리피터 노드가 자신의 링에 포함되어 있는 1번 노드에 그 패

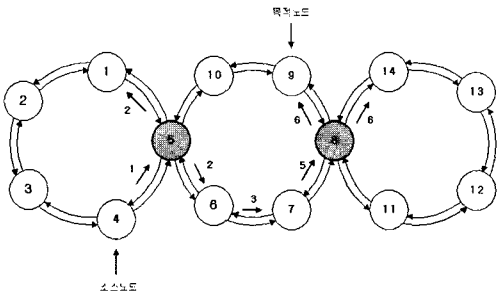

그림 4 Ring3(리피터 노드 2개 사용)

킷을 전송함과 동시에 6번 노드에도 전송한다. 결국 1번과 6번, 2번과 7번은 거의 같은 시간에 그 패킷을 전송할 수 있으며, 리피터 노드가 5번과 8번 두 개이므로 8번에 도착한 패킷은 9번과 14번에 동시 전송되어 결국 13번 노드를 돌 때까지는 중첩된 시간을 유도하여 12번과 11번 노드만이 다른 시간동안 그 패킷을 전송하게 된다. 응답은 역시 최단 경로를 찾아 10번, 5번, 4번으로 전송된다.

그림 5는 각 링에 있는 노드의 수를 줄이기 위해 리피터를 3개 사용한 구조이다. 각 링에는 3개의 노드가 있고, 링의 수는 총 4개가 된다. 3번 노드를 소스 노드라 할 때 전송된 방송 패킷은 4번 리피터 노드를 지나 같은 시간에 1번 노드와 5번 노드로 전송되며, 6번 노드를 지나 같은 시간에 7번 노드와 10번 노드에 전송되며 8번 노드를 지나 같은 시간에 9번 노드와 15번 노드로 전송된다. 15번 노드를 지나 14번 13번 그리고 8번 노드를 지나면 모든 방송 패킷의 전송은 끝나게 된다.

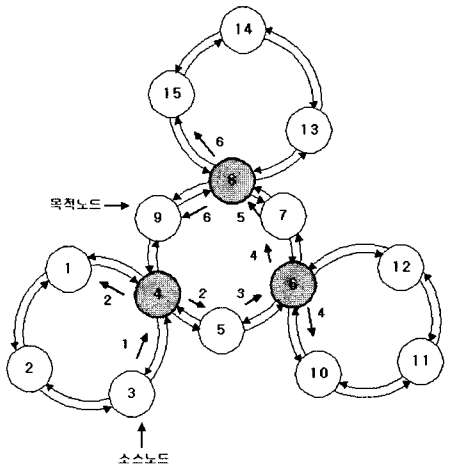

그림 5 Ring4(A) (리피터 노드 3개 사용)

그림 6은 안쪽 링을 모두 리피터 노드만으로 구성한 것이다. 각각의 리피터 노드가 마치 자신이 포함된 노드가 있는 링을 책임지는 형태의 구성을 한 것으로 세 개의 리피터의 내용은 일관성을 유지해야하는 문제를 내포하고 있어서 이 구조를 하나의 리피터 노드만으로 구성하는 구조로의 가능성을 띠게 된다. 이 구조에 위의 예제를 적용하면 4번 노드를 시작해 13번 리피터 노드를 통과하면서 1번 노드와 14번 리피터 노드는 같은 시간에 방송 패킷을 전송하며, 14번 노드를 지나면서 15번 리피터 노드와 5번 노드도 동시에 방송 패킷을 전송한다. 15번 리피터 노드를 지나면서는 안쪽의 리피터 노드만으로 연결된 링은 방송 패킷을 전송을 종료하며, 12번 11번 10번 9번 노드를 지나면서 모든 방송 패킷의 전송을 종료하게 된다.

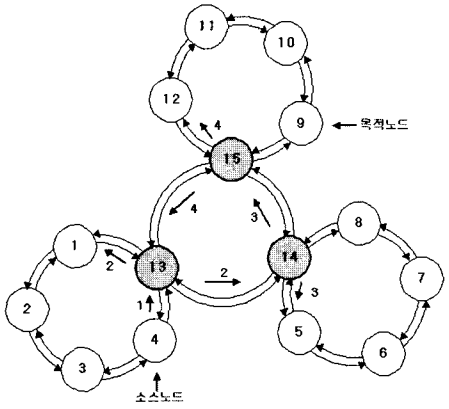

그림 6 Ring4(B) (리피터 노드 3개 사용)

3.3 확장된 리피터 노드를 이용한 링 구조

기본 노드 구조에서는 in 포트와 out 포트의 수가 2개씩 연결되어 총 4개의 포트가 연결되며, 앞서 제시한 리피터 노드는 기본 노드의 구조를 응용한 것으로서 2개까지의 링을 연결할 수 있는 구조를 가정했다. 만일 리피터 노드가 최대 4개까지의 링을 연결할 수 있도록 확장된 구조를 갖는다면 이 확장된 리피터 노드는 16개까지의 포트를 필요로 하게 된다. 즉, 링이 한 개씩 늘어날 때마다 이중링 구조를 유지하기 위해 in 포트와 out 포트가 2개씩 총 4개의 포트가 필요하며, 4개 이상의 링을 연결할 경우에 리피터 노드의 입출력 포트의 증가로 인한 리피터 노드 제어기의 I/O pin 수의 증가, 리피터 노드의 복잡도 증가, 버퍼 크기의 증가 등이 발생해서 구현 비용과 복잡도가 크게 증가한다. 이런 이유로 본 논문에서는 최대 4개까지의 링을 유지할 수 있는 확장된 리피터 노드로 한정하기로 한다.

그림 7은 4개까지의 링을 연결할 수 있는 확장된 리피터 노드 하나만 두고 네 방향으로 노드를 분산시켜 각 링과 링 사이의 요청 지연 시간과 응답 지연 시간을 줄이도록 구성된 구조를 예시하고 있다. 하나의 링에서 나온 트랜잭션은 다른 링들로 동시에 전송될 수 있으며 이러한 중첩된 시간을 이용해 요청 지연 시간과 이로 인한 응답 지연 시간도 줄일 수 있다. 앞의 예제처럼 4번 소스 노드를 시작으로 시계 반대 방향으로 방송 패킷이 전송된다고 할 때 0번 리피터 노드를 지나면서 다른 세계의 링으로 그 방송 패킷을 동시에 전송하게 되어 더 많은 중첩된 시간을 유도하며 빠른 요청 시간과 응답 지연 시간을 기대할 수 있다.

같은 수의 노드를 연결함에 있어서 앞에서 서술한 바와 같이 하나의 링보다는 두 개의 링으로 나누고, 두 개의 링보다는 세 개, 네 개의 링으로 나누어 노드를 분산시키는 것이 응답 지연 시간을 줄이는데 효과를 나타냈으며, 이와 같은 구조는 확장 가능성 면에 있어서 훨씬 효과적이어서 PANDA II의 확장성 문제를 해결하는 하나의 방안으로서 제시될 수 있다. 그림 7은 4개까지의 링을 연결할 수 있도록 확장된 리피터 노드를 이용하여 12개의 노드에서 더 좋은 성능을 보여주는 구조로 노드를 24개까지 확장했을 때도 다른 구조보다 더 좋은 성능을 보이게 된다.

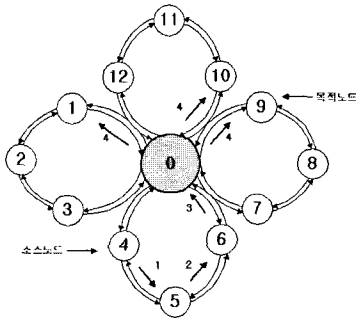

그림 7 Ring4(C) (Repeater 노드 1개 사용)

리피터 노드를 이용한 3.2절과 3.3절에 제시된 링 구조는 기존의 PANDA II 구조에서 단일 링으로 연결된 구조에서의 응답 지연 시간 등의 성능을 개선하기 위한 구조로서, 본 논문을 통해 제시한 리피터 노드를 이용해 링으로 노드의 수를 분산시켜 줌으로써 부가적인 하드웨어의 필요 없이도 보다 나은 효과를 기대해 볼 수 있으며 노드의 수를 24개 또는 48개를 확장하게 되어도

기존의 구조보다 훨씬 나은 결과를 기대해 볼 수 있다. 또한 노드의 수가 24개만 되더라도 프로세서의 수는 24*4=96개로 대규모의 시스템이라고 볼 수 있으며, 효과적인 병렬처리를 얻어 낼 수 있다.

3.4 리피터 노드를 이용한 구조의 정량적 분석

표 3은 응답 패킷이 전송되어올 때 평균 걸리는 hop의 수로 노드의 수가 12개와 24개인 경우를 고려했으며 이를 그래프로 나타내면 그림 8과 같다. 각각의 링의 구조는 리피터 노드를 사이에 두고 노드간의 거리가 대칭형으로 볼 수 있다. 따라서 최대와 최소 hop수의 차이가 PANDA II 시스템보다도 적게 되고 이에 따른 평균 hop 수도 적게 나타난다. PANDA II의 최대 응답 지연 hop 수는 N/2(N:노드 수)로 리피터 노드를 둔 구조의 N/r(r:ring 수)보다도 더 크다.

표 3 응답 패킷의 평균 hop 수

	Ring1 PANDA II	Ring 2	Ring3	Ring4 (A)	Ring4 (B)	Ring4 (C)
Node 12	3	2.41	2.27	2.0	1.95	1.89
Node 24	6	4.70	4.45	3.71	3.24	3.15

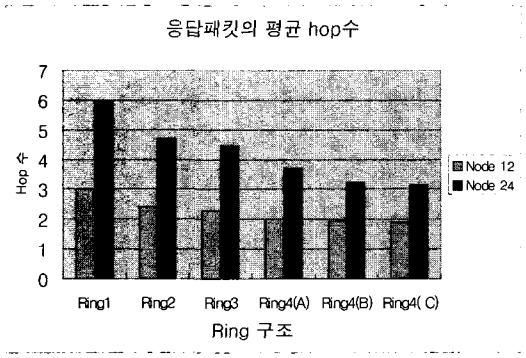

그림 8 응답 패킷의 평균 hop수에 대한 그래프

표 4는 노드 수가 12개, 24개인 경우에 대한 방송 패킷이 거치는 평균 hop 수를 나타내며 이를 그래프로 나타내면 그림 9와 같다. 이 그림에서 알 수 있듯이 노드의 개수가 많아질수록 본 논문에서 제시한 구조가 PANDA II에 비해 hop 수의 감소 비율이 더 크다. 노드의 개수가 2배가 되도 평균 걸리는 hop 수는 기존의 PANDA II보다 그의 두 배에 가까운 이익을 볼 수 있다. 이러한 특징은 리피터 노드를 이용한 구조가 더 많은 노드를 장착할 수 있는 확장 가능한 구조임을 입증하는 근거라 할 수 있다.

표 4 요청 패킷의 평균 hop 수

	Ring1 PANDA II	Ring 2	Ring3	Ring4 (A)	Ring4 (B)	Ring4 (C)
Node 12	12	9.692	8.714	7.8	7.6	5.54
Node 24	24	17.28	16.46	16	11.57	10.08

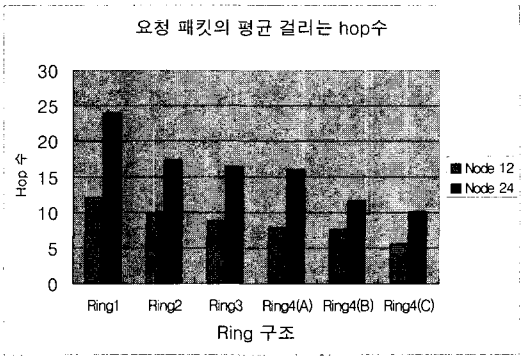

그림 9 요청 패킷의 평균 걸리는 hop 수에 대한 그래프

4. 리피터 노드의 구조

제안된 리피터 노드의 기본 동작은 노드에 포함된 B-link 제어기를 링의 개수만큼 유지해 이를 단순히 패킷 제너레이터(Packet Generator)를 통해 다른 링으로 포워드(Forward) 시키는 것이다. 그림 10의 PBTL(Pending Broadcast Transaction List)은 포워드한 방송 패킷에 대한 정보를 담아두고 다른 노드에서 보낸 방송 패킷이 현재 Pending 중인 트랜잭션과 동일한 블록에 대한 요청일 경우 그 요청을 재전송할 것을 요구하는 기능을 갖게 된다. 가장 단순하게는 그림 10의 PBTL 없이 단순한 포워드만을 위한 기능을 고려해 볼 수 있으나 한번 전송한 방송 패킷에 대한 응답 패킷이 돌아오지 않은 상태에서 같은 주소 영역에 대한 요청을 또다시 내보낸다는 것은 불필요하고 비용의 낭비도 가져오므로 PBTL을 유지하는 것이 보다 나은 성능 향상을 얻을 수 있다.

리피터 노드는 소스 노드가 포함된 링이 아닌 다른 링으로 전송한 방송 패킷이 링을 한바퀴 돌고 올 때까지 이 PBTL에 있는 정보를 유지한다. 이렇게 하는 이유는 만약 소스 노드와 목적 노드가 같은 링에 있게 될 때 최단 경로를 찾아 응답을 하게 되는데 이 최단 경로에 리피터 노드가 포함되지 않는다면 이 리피터 노드를 지나지 않고 응답을 마치게 된다. 이렇게 되면 리피터 노드에는 계속해서 그 트랜잭션이 Pending중이게 되며, 이미 종료한 트랜잭션에 대한 요청이 계속 재시도되는 문제가 발생하므로 이를 고려해 PBTL을 유지해서 방

송 패킷이 전체 노드를 한바퀴 돌 때까지만 Pending 정보를 유지하게 된다. 본 논문의 시뮬레이션 부분에는 PBTL의 설계를 제외한 단순한 기능을 고려했으며 이와 같은 기능에 포함된 여러 성능 향상 요인들은 차후에 논의할 예정이다.

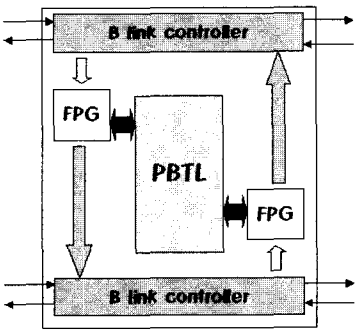

그림 10 리피터 노드의 구조

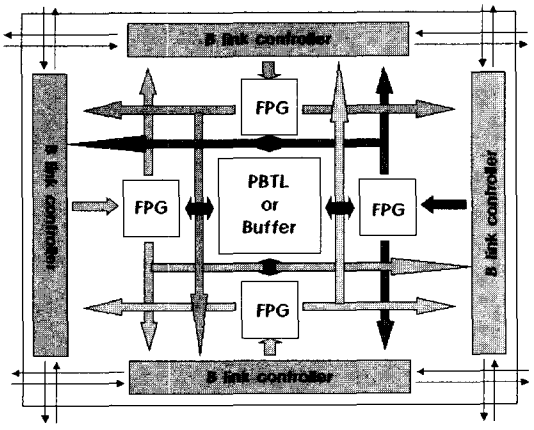

그림 11 4개까지의 링을 연결하도록 확장된 리피터 노드의 구조

그림 10의 리피터 노드 구조는 링을 2개 연결할 경우에 해당하며, 그림 11의 구조는 그림 7에서 설명한 바와 같이 4개까지의 링을 연결할 수 있도록 확장된 리피터 노드의 구조를 예시하고 있으며, 하나의 방송 패킷을 다른 3개의 링에 전송하기 위해 방송 패킷을 단순하게 이웃 노드로 전송하는 FPG를 통해 다른 세 방향으로 그 요청을 방송할 수 있고 그 이외의 것은 그림 10과 동일하다. 한가지 더 고려해 볼 수 있는 것은 Pending중인 트랜잭션에 대해 같은 영역에 대한 요청을 재전송할 것을 요구하지 않고 버퍼를 두어 이에 대한 정보를 갖고 있다가 응답 요청이 오면 요청을 보낸 모든 노드에 응답하는 것이다. 리피터 노드에 성능 개선을 위한 이러한 다양한 기능을 추가하기 위해서는 리피터 노드에 대한 보다 세심한 연구와 리피터 노드의 복잡도 증가에 따른 시스템 성능에 미치는 영향 등을 종합적으로 평가해야 하기 때문에 버퍼를 두는 등의 리피터 노드 자체의 성능 개선에 대한 다양한 방법은 차후에 고려해 볼 예정이다. 그림 11은 그림 10의 확장된 구조로 B-link가 있는 4 방향으로 링을 연결해 노드를 분산해 줄 수 있다. PBTL을 유지하는 것은 상술한 바와 같이 차후에 고려할 예정이며 본 논문의 시뮬레이션에서는 PBTL과 버퍼는 고려하지 않았다.

본 논문에서 제안한 리피터 구조에서는 각각의 입력 링으로부터 전달되어오는 패킷들을 해당 출력 링에 연관된 큐에 잠시 담아두었다가 각 출력 링의 링크 이용이 가능해질 때 해당 패킷을 전송하는 FIFO 방식으로 구성된다. 리피터 노드에 연관된 링의 개수가 늘어날 경우에는 그에 비례해서 출력 링에 연관된 큐의 크기를 적정하게 늘려주어야 한다. 앞에서도 설명한 것처럼 리피터 노드는 해당 패킷을 단순히 바이패스시키는 간단한 역할을 수행하며, 리피터 노드의 입출력 핀의 개수와 복잡도를 고려하여 최대 4개의 링이 연결된다는 것을 가정하면 출력 링의 큐가 overflow되는 문제는 많이 발생하지 않을 것으로 사료된다. 향후 큐의 overflow를 줄이고, 트래픽을 줄이는 다양한 방안에 대한 연구와 링의 수가 증가하는 경우에 링을 분할하는 적절한 방법, 서로 다른 링에서 전달된 패킷을 처리하는 우선 순위 방식에 대한 연구가 필요하다.

5. 성능 분석

5.1 모의실험 환경 및 인자

본 논문에서는 앞에서 제안한 리피터 노드를 이용한 구조와 PANDA II를 모의실험하기 위한 도구로 확률 구동(probability-driven) 시뮬레이터인 SES/Workbench를 사용한다. SES/Workbench는 큐잉 모델에 기반하여 실제 상용화된 다중 프로세서 시스템의 성능 평가에 쓰이고 있는 모의실험 도구이다[10]. SES/Workbench의 입력 값은 다음과 같이 구하여 모의실험 하였다. 모의실험에서 SES/Workbench의 입력 인자의 확률 값들은 SPLASH-2[14]에서 제공하는 프로그램 중 FFT, Radix, LU, Barnes를 프로그램 구동(program-driven) 방식의 시뮬레이터인 MINT[11]에서 수행하여 나온 결과치를 사용하였다.

표 5와 표 6의 인자들을 사용하여 MINT 시뮬레이션

을 수행하고, 여기서 나온 결과치를 이용하여 이중 링 제어기와 그 아래 부분인 B-링크, 링크 제어기, 전역 링크 등을 SES/Workbench를 이용하여 모델링하였다. MINT 시뮬레이터를 통해 트랜잭션의 종류별 발생 빈도 수를 구하고, 이 결과를 이용하여 트랜잭션 종류별로 발생 확률 값을 구했으며 그 값은 표 7과 같다. SES/Workbench에서는 이렇게 구한 확률 값만큼 트랜잭션을 생성하였으며 SES/Workbench의 모의실험에서 사용된 인자는 표 8과 같다.

표 5 MINT에 사용된 응용 프로그램 인자

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

표 6 MINT 모의 실험의 시스템 관련 인자

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

표 7 트랜잭션 종류별 발생 확률 값

트랜잭션 종류	확률
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 8 SES/Workbench에서 사용된 인자

인자	값
B-Link 클럭	155MHz
전역 링크 클럭	500MHz
노드 개수	12, 24
SCI 패킷 헤더	32 Byte
캐쉬 라인 크기	64 Byte

모의실험 대상은 PANDA II와 리피터 노드를 사용한 구조에서 링을 2개 둔 구조와, 링을 4개 사용한 구조이다.

5.2 모의실험 결과 및 분석

모의 실험 결과는 4가지 구조에 대해 요청 트랜잭션의 지연 시간, 응답 트랜잭션의 응답 지연 시간을 제시한다. 또한 제시된 모의 실험 결과를 분석하여 각 모델의 특성을 생각해 보기로 한다.

5.2.1 시스템별 요청 지연 시간의 변화

그림 12는 트랜잭션의 발생 빈도별 트랜잭션의 요청 지연 시간을 보여주는 것이다. Ring2는 리피터 노드 한 개로 링을 2개 만든 구조로써 하나의 링에 노드수가 6개인 구조를 말하는 것이다. Ring4는 리피터 노드에 링을 4개 연결하고 한 링에 노드를 3개씩 연결한 구조이다. 이 그림에서 PANDA II의 구조와 비교해 Ring2나 Ring4는 하나의 트랜잭션의 중복 요청을 통한 요청 지연 시간에 많은 이득을 볼 수 있으며, 이는 Ring을 4개까지 연결한 구조에서 보다 효과적이라 할 수 있다. 100, 200, 300의 값은 트랜잭션의 발생 시간 간격으로 시간 간격이 클수록 발생 빈도가 낮아지며, 발생 빈도가 낮아질수록 보다 좋은 결과를 얻을 수 있다. 리피터 노드를 이용한 구조에서는 요청시간의 이득이 가장 큰 장점이며, 트랜잭션의 빈도와 적절한 큐의 사이즈를 통해 최상의 결과를 얻어낼 수 있다.

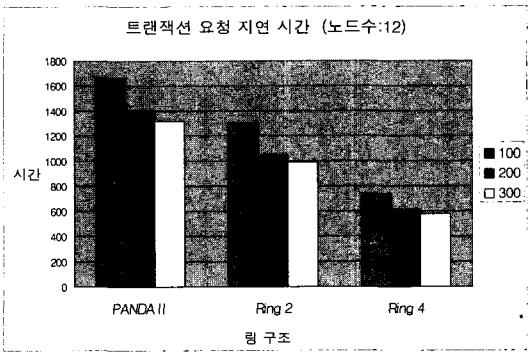

그림 12 트랜잭션 발생 빈도별 요청지연시간(노드수:12)

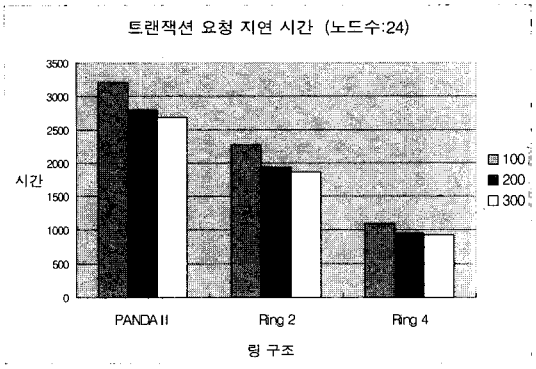

그림 13 트랜잭션 발생 빈도별 요청지연시간(노드수:24)

그림 13은 노드수가 24인 경우의 모의실험 결과이다. 노드수가 24인 Ring4의 성능이 노드수가 12인 PANDA II보다 요청 시간 지연이 더 적다. 이는 리피터 노드를 이용한 구조에서는 노드수가 늘더라도 요청 시간의 부담을 줄일 수 있음을 보이며 이는 빠른 응답시간을 기대할 수 있어 전체적인 성능을 높이는 이유라고 볼 수 있다.

5.2.2 시스템별 응답 지연 시간의 변화

그림 14와 그림 15는 트랜잭션의 발생 빈도에 따른 응답 지연 시간을 보여준다. 그림 14는 노드 수가 12, 그림 15는 노드 수 24이며, 요청 지연 시간과 마찬가지로 응답 지연 시간 또한 리피터 노드를 이용한 구조에서 기존의 구조보다 좋은 성능을 보인다. 특히 노드를 링에 분산해 주는 Ring4 구조는 요청과 응답 지연 시간 모두에 있어서 많은 효과를 얻어 낼 수 있으며 이는 더 많은 노드를 장착할 수 있는 확장 가능한 구조임을 보여준다.

그림 16과 그림 17은 요청 트랜잭션과 응답 트랜잭션의 합을 각 구조별로 보여주는 그림으로 한번의 트랜잭션이 요청되면서 응답될 때까지의 평균 지연 시간을 나타내 주는 것이다. 그림에서와 같이 노드 수가 12인 구조에서의 링 구조 변화에 따른 지연 시간의 감소 비율보다 노드 수 24인 구조에서의 지연시간 감소 비율이 더 큼을 볼 수 있으며, 이는 확장성에 있어서 리피터 노드를 이용한 구조의 장점을 보여주는 동시에 기존의 PANDA II에서 보여주는 확장성 문제를 해결할 수 있음을 의미한다.

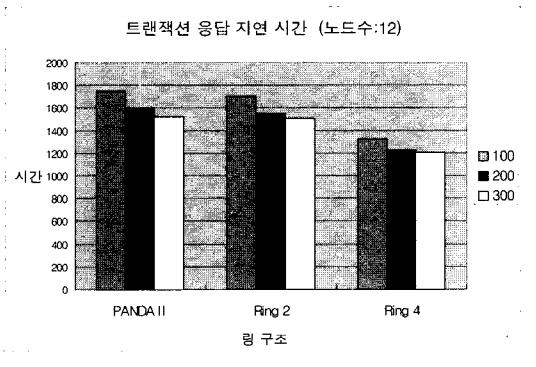

그림 14 트랜잭션 발생 빈도별 응답 지연시간(노드수:12)

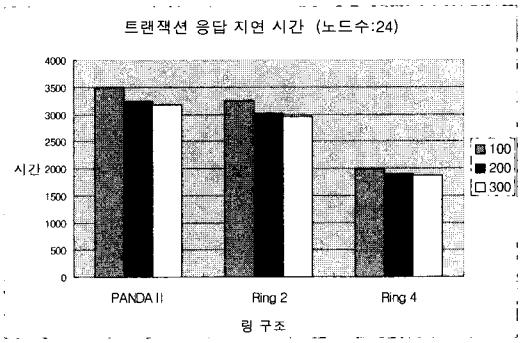

그림 15 트랜잭션 발생 빈도별 응답 지연 시간(노드수:24)

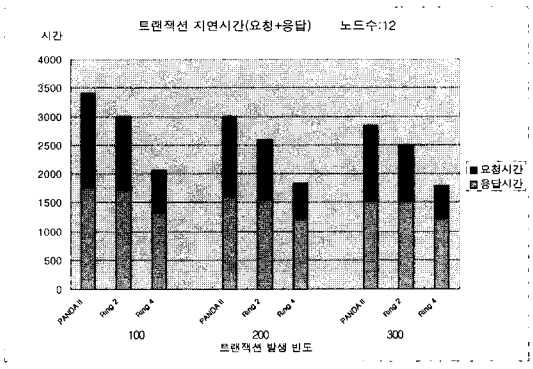

그림 16 트랜잭션의 지연시간 (노드수:12)

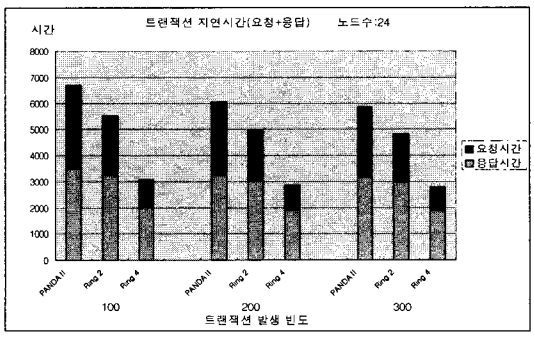

그림 17 트랜잭션의 지연시간 (노드수:24)

6. 결 론

CC-NUMA 구조에서 상호 연결망으로 쓰이는 버스의 확장성 및 대역폭의 한계에 대한 해결책으로 고속의 지점간 상호 연결망에 대한 많은 연구가 이루어지고 있으나 이 구조는 노드의 개수가 증가함에 따라 거쳐야하는 링크가 많아져서 응답 지연 시간이 증가되는 문제가 있다.

본 논문에서는 PANDA II의 링 구조를 본 논문에서 제시한 리피터 노드를 이용해 여러 링으로 분리해 줌으로써 각 링에 노드를 분산시키고 노드간 패킷 전송 시간을 중첩시켜서 전체 노드에 걸리는 시간을 줄이는 결과를 얻어 낼 수 있었으며 이는 요청 패킷뿐만 아니라 응답 패킷에도 좋은 결과를 가져올 수 있다. 각 링이 리피터 노드를 사이에 두고 대칭 구조를 보여줌에 따라 각 노드간의 hop의 최대치와 최소치간의 간격을 줄일 수 있으며, 각 링에 연결된 노드의 수가 기존의 것보다 적기 때문에 각 노드간의 지연 시간을 줄일 수 있다. PANDA II 구조보다 노드간의 hop의 수가 적어지므로 요청 시간 및 응답 시간 지연을 줄일 수 있는 장점 외에 더 많은 노드를 장착할 수 있는 확장 가능한 구조인 것이 기존의 구조에서 보이는 단점을 해결한 것이다. 또한 리피터 노드에 PBTL이나 버퍼를 이용해 보다 효과적으로 트랜잭션을 처리해 줄 수 있을 것으로 사료되며, 기존의 PANDA II보다 적은 hop으로 응답을 할 수 있다는 것은 이 구조가 갖는 장점이라고 볼 수 있다.

본 논문에서 제시한 리피터 노드는 그림 3, 그림 4, 그림 5, 그림 6에서 설명한 것처럼 이웃 노드에서 전달되어온 방송 패킷을 단순히 여러 곳으로 동시에 전송하는 역할을 수행하며, 전송된 각각의 패킷은 리피터 노드를 제외한 서로 다른 노드에 중복해서 전송되지 않기 때문에 리피터 노드로 인한 새로운 트래픽은 거의 발생하지 않는다. 본 논문에서 제시한 모의 실험에서는 리피터 노드에서 유발될 수 있는 지연 시간을 충분히 고려하였다. 모의 실험 결과 리피터 노드로 인한 트래픽 유발 효과는 미미한 정도였으며, 리피터 노드내의 queue depth를 통해 그 영향을 조사해 보았을 때 리피터 노드를 통과하면서 나타나는 부하는 전체 성능에 거의 영향을 미치지 않는 수준으로 판단된다. 이는 리피터 노드를 단순하게 바이패스시키는 수준의 간단한 역할로만 사용함으로 얻어지는 것으로서 보다 많은 기능을 할 수 있도록 리피터 노드를 구조화하게 되면 이에 대한 부하 및 트래픽에 대한 고려가 필요하며, 리피터 노드의 복잡도 증가에 따른 시스템의 성능에 미치는 영향에 대한 종합적인 평가가 필요할 것으로 사료된다.

향후 계획으로는 앞에서도 언급했듯이 리피터 노드내에 PBTL을 유지해 Pending 중인 트랜잭션과 같은 메모리 블록에 대한 요청이 리피터 노드를 지날 때 재시도 요청을 내는 방법에 관한 연구와 버퍼를 두어 메모리 블록을 유지하도록 하여 이 리피터 노드를 지나는 응답에 대한 책임을 리피터 노드의 버퍼가 담당하도록 하는 연구를 계속할 예정이다. 또한 머지 버퍼를 두어 같은 메모리 블록에 대한 트랜잭션을 요청한 노드들로 동시에 응답 트랜잭션을 전송하는 방안에 대한 연구뿐만 아니라, 링의 수가 증가하는 경우에 링을 분할하는 적절한 방법, 서로 다른 링에서 전달된 패킷을 처리하는 우선 순위 방식에 대한 연구도 계속할 예정이다.

참 고 문 헌

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

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

[3] Zhang, Z. and J. Torrellas. "Reducing Remote Conflict Misses : NUMA with Remote Cache versus COMA," In Proc. of the 3rd IEEE Symp. on High Performance Computer Architecture (HPCA-3), pp. 272-281, Feb. 1997.

[4] 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.

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

[7] 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.

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

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

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

[11] 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.
[12] 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.
[13] Daniel Lenoski, Anoop Gupta et al. "The Stanford Dash Multiprocessor," IEEE Computer, March 1992.
[14] 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.
[15] Dolphin, "A Backside Link(B-Link) for SCI nodes," Draft 2.4, September 21, 1995.
[16] Dolphin, "Link Controller 3 Specification," November, 1998.

경 진 미
2000년 2월 수원대학교 전자계산학과 학사. 2002년 2월 수원대학교 컴퓨터학과 석사. 2002년 2월 ~ 현재 (주)써머스테크놀로지 솔루션 디비젼 소속 . 관심분야는 컴퓨터 구조, 다중 프로세서 시스템, 네트워크 및 시스템 프로그래밍

장 성 태
정보과학회논문지 : 시스템 및 이론
제 29 권 제 2 호 참조