이중 연결 구조 CC-NUMA 시스템의 효율적인 상호 연결망 구성 기법

서 효 중†

요 약

반도체 미세 공정의 개발과 더불어, 높아진 집적도 및 동작 클럭의 고속화로 단일 프로세서 시스템 성능은 지속적으로 개선되고 있다. 이 결과 기가헬즈 이상의 클럭 속도를 가지는 개인용 컴퓨터가 보편적인 데스크 탑 시스템으로 자리잡게 되었으며, 불과 수년 전의 고가 대형 시스템은 점차 이러한 작은 시스템들을 상호연결망으로 연결한 형태로 급속히 대체되어가고 있다. 이러한 구조의 클러스터 컴퓨터는 높은 확장성과 고성능을 얻을 수 있으므로, 점차 그 영역을 확대해나가고 있으나, 상호연결망의 대역폭 및 지연에 따라 성능 제한 요소는 여전히 존재하고 있으며, 이러한 이유로 SCI, Myrinet, Gigabit Ethernet 등 고속의 상호연결망이 클러스터 시스템의 연결 구조로 사용되고 있다. 프로세서 속도의 개발과 더불어 상호연결망의 속도 또한 개선되어 왔는데, 상호연결망은 그 대역폭을 늘리는 것과, 상호연결망을 이용한 경우의 통신 시간 지연의 축소로 볼 수 있다. 대역폭의 확장 및 지연시간의 단축은 상호연결망의 고속화를 통하여 이루어질 수 있으나, 작은 면적에 집적되어 있는 프로세서와는 달리, 보다 넓은 면적에 펼쳐져 있는 상호연결망의 동작 속도는, 물리적 거리에 의한 지연으로 인하여 개선의 난이도가 높으며, 따라서 클러스터 시스템의 확장 규모는 상호연결망의 병목 현상에 의하여 제한된다고 할 수 있다. 이러한 이유로 보다 높은 대역폭의 상호연결망을 구현하려는 노력은 복수개의 연결 구조를 이용한 형태로 개선되어 왔으며, 고속으로 동작하는 SCI 점 대 점 연결구조를 이용한 다중 연결 형태의 시스템이 활발히 연구되어 왔다. 본 논문은 이러한 이중 점 대 점 연결 구조 시스템의 성능 제한 요소인 접근 시간 및 효율을 개선하기 위하여, 두개 중 하나의 점 대 점 연결을 링 형태로, 나머지 하나는 링을 몇 개의 노드의 묶음으로 분할하여 연결하는 구성을 제시하였으며, 방송 및 일 대 일 전송에 적합한, 간단하고 효율적인 경로 설정 방법과 적절한 묶음의 수를 제시하였다. 본 논문에 제시한 구조의 시스템의 성능 측정의 비교 대상으로, 최신 시스템에 채용되어 있는 반대방향 이중 링 구조를 비교 대상으로 하였으며, 반대방향 이중 연결 구조에 비하여 본 논문에 제시한 상호연결망 구성 및 트랜잭션 경로 설정 방법이 상대적으로 우수함을 시뮬레이션을 통하여 검증하였다. 실험 결과, 본 논문에서 제안한 상호연결망 구조 및 트랜잭션 경로 설정 방법을 이용한 경우, 반대방향 이중 링 구조의 시스템 구조에 비하여 단위 트랜잭션의 처리 시간이 1.05~1.11배 향상되었으며, 시스템의 성능은 1.42~2.1배 향상되었다.

An efficient interconnection network topology in dual-link CC-NUMA systems

Hyo-Joong Suh†

ABSTRACT

The performance of the multiprocessor systems is limited by the several factors. The system performance is affected by the processor speed, memory delay, and interconnection network bandwidth/latency. By the evolution of semiconductor technology, off the shelf microprocessor speed breaks beyond GHz, and the processors can be scalable up to multiprocessor system by connecting through the interconnection networks. In this situation, the system performances are bound by the latencies and the bandwidth of the interconnection networks. SCI, Myrinet, and Gigabit Ethernet are widely adopted as a high-speed interconnection network links for the high performance cluster systems. Performance improvement of the interconnection network can be achieved by the bandwidth extension and the latency minimization. Speed up of the operation clock speed is a simple way to accomplish the bandwidth and latency betterment, while its physical distance makes the difficulties to attain the high frequency clock. Hence the system performance and scalability suffered from the interconnection network limitation. Duplicating the link of the interconnection network is one of the solutions to resolve the bottleneck of the scalable systems. Dual-ring SCI link structure is an example of the interconnection network improvement. In this paper, I propose a network topology and a transaction path algorism, which optimize the latency and the efficiency under the duplicated links. By the simulation results, the proposed structure shows 1.05 to 1.11 times better latency, and exhibits 1.42 to 2.1 times faster execution compared to the dual ring systems.

키워드 : 다중프로세서 시스템(Multiprocessor System), 이중 연결 구조(Dual Link Interconnection), 트랜잭션 경로설정(Transaction Path), CC-NUMA

※ 본 연구는 2003년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음.
† 정 회 원 : 가톨릭대학교 컴퓨터정보공학부 교수
논문접수 : 2003년 10월 9일, 심사완료 : 2004년 1월 16일

1. 서 론

점차 고속화, 고성능화 되어가는 마이크로프로세서의 발달에 따라, 개인용 컴퓨터 또는 워크스테이션에 사용되는 프로세서를 이용하여 보다 대형의 시스템을 구성하려는 연구는 활발이 이루어져 왔다. 이러한 움직임은 학계 뿐만 아니라 중대형 시스템을 개발하는 산업계에까지 확대되어 고성능의 클러스터 구조 시스템은 중대형 시스템으로서 급속히 확대되고 있다. 클러스터 시스템의 단위 노드로서 사용되는 시스템은 일반적인 데스크 탑 컴퓨터로부터 워크스테이션, 서버 등 다양하게 사용되고 있다.

이러한 클러스터 시스템의 구현은 메시지 패싱 방식을 사용하는가, 또는 공유메모리 방식을 사용하는가에 따라 그 운영의 난이도 및 프로그램의 구현 형태에서 차이를 나타내게 되는데, 일반적으로 공유메모리 방식의 경우가 메시지 패싱 방식의 운영 형태에 비해 보다 용이한 프로그래밍 및 기존 프로그램의 이식이 가능한 것으로 알려져 있다[1].

공유메모리 방식으로 클러스터 시스템이 구현될 경우, 시스템은 CC-NUMA(Cache Coherent Non-Uniform Memory Access) 구조로 구현된다. CC-NUMA 시스템의 경우 각 프로세서에서 클러스터 시스템 내의 모든 메모리에 직접 접근이 가능하며, 메모리 일관성 유지 프로토콜에 따라서 메모리의 일관성 유지를 위한 트랜잭션이 노드 내 및 노드 간 상호연결망을 통하여 전송된다. 이러한 트랜잭션의 전달에 따라서 각 프로세서의 프로그램 수행이 이루어지며, 트랜잭션의 전달 경로가 되는 상호연결망은 시스템 성능을 제한하는 중요한 요소로 작용하게 된다[2].

클러스터 시스템에 있어서 상호연결망의 중요성 및 구조는 이러한 이유로 강조되어 왔으며, 현재 각 노드를 연결하는 상호연결망으로 Fast Ethernet, Gigabit Ethernet, Myrinet, SCI(Scalable Coherent Interface)등이 채용되고 있다. 이들 중 특히 SCI의 경우 1GB/s의 대역폭을 지원하고 점대 점 연결 및 스위치 구조를 사용하고 있으므로, IBM의 NUMA-Q등 산업계와 학계의 고성능 시스템에서 사용되고 있다[3]. 한편 이러한 상호연결망을 통하여 전송할 트랜잭션의 경로설정 또한 고속으로 처리되어야 하며, 경로설정의 속도 또한 시스템 성능 제한 요소로 작용한다[4].

따라서 시스템 성능의 제한 요소로 작용하는 상호연결망의 고성능화를 위한 연구는 필수적이며, 고성능화를 위한 한 형태로써 복수 경로를 이용하는 경우는 이중 버스, 이중 링 등으로 제시된 바 있다. 그러나 복수 경로의 설정은 트랜잭션의 전송 경로를 설정하는 알고리즘과 연결 구성을 필요로 하게 되며, 경로 선택 알고리즘 및 연결 구성이 효율적으로 설계되지 못할 경우 높아진 대역폭을 제대로 활용할 수 없게 된다. 또한 경로의 설정 알고리즘의 복잡도가 높을 경우, 알고리즘으로 인한 경로 처리 설정 효율로 인하여 상호연결망으로 전송될 트랜잭션의 대역폭을 제한할 수 있으므로, 경로 선택 알고리즘은 단순하면서도 효율적으로 설계되어야 한다.

CC-NUMA 시스템과 같은 공유메모리에 기반한 클러스터 시스템이 구현될 경우, 각 노드간에 일관성 유지를 위한 트래픽과 실제 데이터를 전송하는 트래픽이 존재한다. 이 두 트래픽은 차지하는 대역폭의 크기면에 있어서 차이를 나타내며, 일관성 유지를 위하여 참여할 노드의 수와 데이터 전송을 위하여 참여하여야 할 노드의 수는 큰 차이를 나타내게 된다[5].

본 연구는 복수개의 상호연결망을 사용하는 클러스터 시스템의 트랜잭션의 성질에 따라서 참여하여야 할 노드의 수가 다름에 주목하여, 이러한 유형에 따른 지연 시간을 단축할 수 있는 연결 구조를 제시하고, 각 트랜잭션에 따라서 보다 효율적인 상호연결망 전송 경로를 제공하는 알고리즘을 고안하며, 이러한 경로선택 알고리즘이 고 대역폭의 상호연결망 성능을 제한하지 않도록 충분히 단순한 방법을 이용하도록 하여, 상호연결망으로 인한 병목 현상을 완화시켜 상대적으로 시스템의 성능 향상을 꾀하는 것이다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구에 대하여 살펴보고, 3장에서 본 논문에서 제안하는 연결 구조와 트랜잭션의 경로 설정 방법을 설명하며, 4장에서 시뮬레이션 도구와 환경 및 결과를 제시하고, 5장에서 결론을 맺는다.

2. 관련 연구

작은 규모의 데스크 탑 시스템이나 워크스테이션을 서로 연결하여 보다 고성능의 시스템으로 확장하기 위한 접근은 최근 수년 동안 다중 프로세서 시스템의 주요 연구 분야로 대두되어 있다. 이는 기존의 MPP(Massively Parallel Processing) 방식 등을 이용한 대형 시스템에 비하여 상대적으로 낮은 가격으로 높은 성능을 얻을 수 있음이 보고되었기 때문이며, 이러한 클러스터 시스템은 중대형 시스템을 급속도로 대체해 나가고 있는 실정이다[6].

전통적으로 다중 프로세서 시스템의 구현 형태는 메시지 전송 방식과 공유메모리 방식으로 나누어 볼 수 있는데, 프로세서 간의 명시적 데이터 전송을 필요로 하지 않는 공유메모리 전송 방식은 각 프로세서에게 단일 시스템 이미지를 제공할 수 있는 구조로써, 병렬 프로그램을 위한 별도의 프로세서간의 통신을 명시적으로 프로그램에 구현하지 않아도 되는 장점으로 기존의 프로그램 이식 및 프로그램의 구현에 상당한 이점을 가지고 있다. 반면 단일한 메모리 이미지를 모든 프로세서에 제공하게 되므로, 각 프로세서에서 접근하는 메모리 사이의 일관성 유지를 위한 부하가 발생하게 되며, 일반적으로 이러한 메모리 일관성 유지 작업은 프로그래머나 소프트웨어에서 고려하지 않아도 문제가 발생하지 않도록 하드웨어를 이용하여 구현한다[7].

최근, 고속의 마이크로프로세서에 적용된 기술은 버스를 이용한 다중 프로세서 시스템을 보다 쉽게 구현할 수 있게 하였으며, 인텔 등 프로세서를 설계하는 회사들은 두 개 내지 네 개의 프로세서를 버스를 통하여 연결할 수 있도록 제공하고 있다. 버스를 통하여 수 개의 프로세서가 연결된 SMP(Symmetric Multiprocessor) 시스템은 고성능 워크스테이션 및 서버로 자리잡고 있다. SMP 다중 프로세서 시스템이 용이해짐과 반대로, 보다 많은 개수의 프로세서를 연결하여 고성능의 MPP 시스템을 구현하는 것은 상대적으로 높은 구현 가격과 난이도, 복잡한 프로그램 구현 등의 단점을 가지게 되었으며, 높은 가격 대 성능비를 나타내는 소규모의 SMP 시스템을 상호연결망을 이용하여 구성하려는 접근은 자연스럽게 클러스터 시스템의 구현으로 발전하게 되었다. 이러한 클러스터 시스템은 미국 Berkeley 소재의 캘리포니아 주립대학의 NOW 프로젝트는 클러스터 시스템의 대표적인 연구 사례로서 SUN의 워크스테이션과 인텔 펜티엄 PC들을 다수 연결한 형태이다[8].

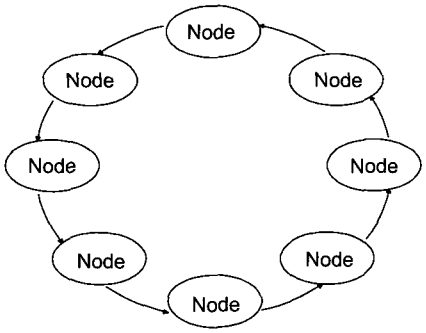
한편 리눅스를 이용한 시스템 또한 다수 연구되어 왔는데, NASA와 Maryland 주립대학에서 연구된 BEOWOLF 프로젝트는 그 대표적인 경우로, 프로세서들 간에 작업을 공유함으로써 높은 성능을 나타내었다[9]. 리눅스를 이용한 시스템은 IBM Cluster Server 등 산업계에서도 널리 채용되고 있다.

클러스터 시스템은 메시지 패싱 방식의 보편화된 라이브러리인 MPI(Message Passing Interface)를 이용한 방식이 다수 연구되고 있으며, 국내의 다수 대학에서도 관련 연구 및 시스템의 구현 연구가 진행되고 있다[10]. 또한 클러스터 시스템 간의 통신 모델로 Compaq, Intel, Microsoft 사는 Cornell 대학의 U-Net을 기반으로 한 VIA(Virtual Interface Architecture)를 발표하였으며, VIA는 다양한 플랫폼에 적용될 수 있는 표준화된 API를 제공함으로써 다양한 상호연결망에 적용할 수 있게 되었다[11].

상호연결망의 연결 구조 및 형태에 대한 연구 또한 활발히 이루어졌으며, IBM의 NUMA-Q 시스템의 원형인 Sequent의 StiNG[12], Debois의 Express Ring[13], Data General의 AViiON 시스템[14] 서울대학교 컴퓨터공학과의 PANDA 시스템[15] 등은 점 대 점 연결을 이용한 링 구조로 이루어진 시스템이다. 링 구조는 차후 보다 높은 성능과 확장성을 위해 Data General의 AViiON 시스템과 서울대학교 컴퓨터공학과의 PANDA-II 시스템 등에서 이중 링 구조로 발전하였으며, 이러한 이중 링 구조는 크로스바 스위치 등에 비해 높은 성능과 확장성을 나타냄이 보고되어 있다[16].

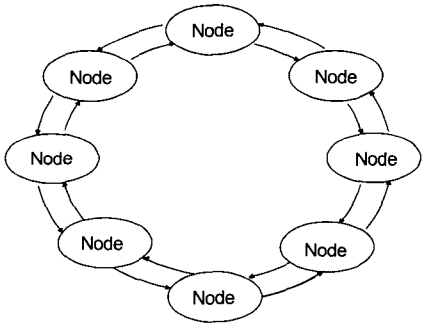
3. 연결 구조 및 트랜잭션 전송 경로

(그림 1)은 점 대 점 연결을 이용하여 링 구조로 구성된 시스템이다. 각 노드는 수 개의 프로세서로 구성된 SMP 시스템이며, 각 노드는 서로 점 대 점 연결을 통하여 링 형태로 연결되어 있다. 이러한 형태는 IBM의 NUMA-Q, Data General의 AViiON시스템 등 산업계와 Express Ring, PANDA 시스템 등 학계에서 채용되었다[15].



(그림 1) 점 대 점 연결을 이용한 링 구조의 시스템

(그림 2)는 위와 같은 점 대 점 연결을 이용한 링 구조의 시스템이 두 개의 점 대 점 연결을 통하여 확장된 형태이다. 이러한 구조는 AViiON이나 PANDA-II 시스템에서 사용되었으며, 두 개의 연결 링크와 링크 제어기를 가지고 있다. 각 노드의 두 개의 점 대 점 연결에서 트랜잭션은 서로 반대 방향으로 전송되도록 구성되어 있다.



(그림 2) 반대방향 이중 링 구조의 시스템

3.1 트랜잭션의 종류

스누핑에 기반한 일관성 유지 기법을 사용하는 Express Ring이나 PANDA 시스템의 경우, 링 상에 발생하는 트랜잭션은 요청과 응답으로 분리되며, 요청의 경우 트랜잭션을 모든 노드가 참여하여 관여한다. 즉, 메모리에 대한 접근이 한 노드에서 발생할 때, 노드 내에서 메모리 라인이 제공되지 못하여 원격 노드로 요청이 되어야 할 경우, 그 메모리 접근은 모든 노드가 참여하여 메모리 라인을 전송할 책임이 있는 노드를 판단하여야 하며, 요구된 메모리 라인에 대한 소유 권한을 가지고 있는 노드는 요구한 노드에게 해당되는 메모리 라인에 대한 전송을 일 대 일 전송의 형태로 응답한다. 만일 요청 트랜잭션이 쓰기 접근과 같이 모든 노

드의 캐시 상태를 변화시키는 경우 모든 노드는 이에 해당되는 작업을 각 노드 내의 여러 캐시에 반영하여야 한다. 이러한 동작은 요청 트랜잭션을 관찰함으로써 이루어지며, 각 노드 제어기는 자신의 캐시 및 메모리 상태에 따라서 노드 내의 지역 버스에 해당되는 트랜잭션을 발생 시킬지의 유무를 결정해야 한다. 다음 <표 1>은 발생되는 트랜잭션과 유형을 나타낸 것이다.

〈표 1〉 트랜잭션의 종류와 유형

트랜잭션 이름	트랜잭션 종류	데이터 유무
읽기 요청	방송 트랜잭션	무
쓰기 요청		
무 효 화		
읽기 응답	일 대 일 트랜잭션	유
쓰기 응답		
되 쓰 기		

3.2 반대방향 이중 링 구조의 트랜잭션 전송 경로 설정 방법

위의 (그림 2)와 같이 반대방향의 이중 링 구조의 경우, 경로에 따른 부하의 분배 및 트랜잭션의 지연 감소를 위하여 트랜잭션의 종류 및 접근 주소에 따른 경로를 설정한다. 경로 설정 방법으로 접근하는 주소의 최하위 비트에 따라 연결 경로를 선택하는 방법, 요청 및 응답에 따라 전송 경로를 달리하는 방법, 해당되는 주소의 메모리를 가지는 노드와의 경로길이에 따라 설정하는 방법, 데이터 전송을 일으키는 노드간의 물리적 거리에 따라 전송 경로를 설정하는 방법 등이 있다.

요청 트랜잭션의 경우, 모든 노드에 전송이 전달되고 참여하여야 하므로, 링 상으로 전송한 요청이 처음 요청을 전송한 노드로부터 동일 방향의 링크를 통하여 전달되고, 중간 경로에 있는 각 노드는 전달하는 트랜잭션의 내용을 확인하고 일관성 유지 작업을 수행하게 된다. 따라서 요청 트랜잭션의 종료는 특정 노드에서 발생한 요청이 링 상에 전달되어 다시 자신에게 돌아올 때 요청의 전달이 완료된다.

요청은 모든 노드에 전달되어야 하므로, 양 방향의 링에 대하여 적절한 부하로 분배되어야 하며, 본 논문에서는 해당되는 요청의 물리적 메모리가 존재하는 홈 노드 쪽으로 요청이 전달되는 것을 가정하였다[16].

요청에 대한 응답은 요청한 노드와 응답을 전송하는 노드 간의 일 대 일 전송이므로, 양 방향의 링 중 보다 짧은 경로를 통하여 전달한다.

3.3 이중 링크를 이용한 시스템 구성

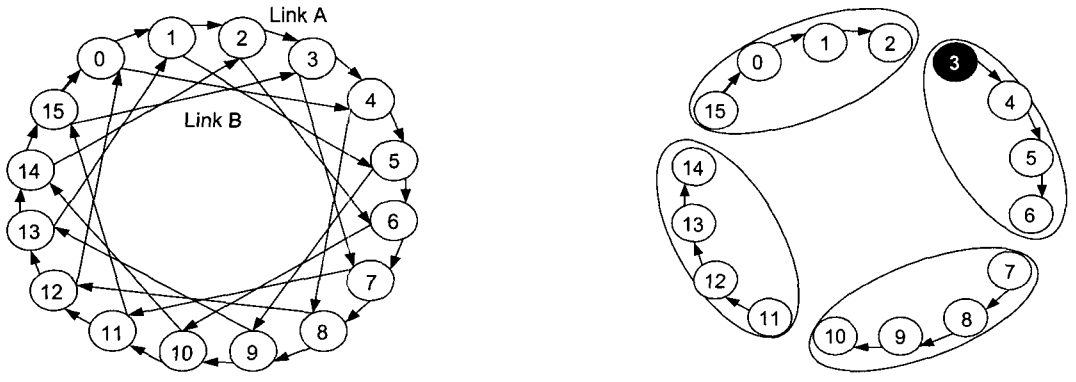
본 논문에서 제안하는 구조는 각 트랜잭션에 대하여, 요청이 발생한 노드 위치를 기준으로 하여 몇 개의 부분으로 링이 분할되어 처리할 수 있는 형태로 변형시킨 것이다. 안쪽의 링크는 각 묶음에서 같은 순서에 해당하는 노드끼리 연결시킴으로서 묶음 간의 연결을 설정한다. 링의 중심을 기준으로 바깥 쪽 연결을 '링크 A'로, 안쪽 연결은 '링크 B'로 지칭한다.

(그림 3)(a)는 16개의 노드를 갖는 시스템을 네 개의 부분묶음으로 분할한 시스템의 링크 형태이다. 이러한 분할은 각 트랜잭션을 발생시킨 노드를 기준으로 구성된다. (그림 3)(b)는 노드 3이 트랜잭션을 발생시키는 경우이며, 이 때 개념적으로 구성되는 트랜잭션 묶음을 표시한 것이다.

3.4 방송 트랜잭션의 전달

방송 트랜잭션은 모든 노드에 전달되어야 한다. 방송 트랜잭션의 처리 경로는 방송의 요청을 각 분할된 묶음에 전달하는 경로와 각 묶음 내에서 자신의 묶음 내에 있는 노드로 트랜잭션을 전달하는 경로, 그리고 전달된 트랜잭션에 대한 확인 트랜잭션을 발생시키는 세 가지 경로로 볼 수 있다.

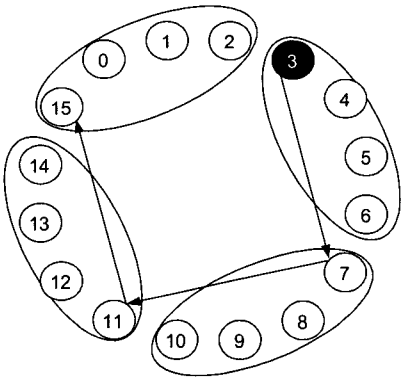
방송의 요청을 각 분할된 묶음에 전달하는 경로는 링크 B를 이용하여 전달되며, 링크 B 경로를 통하여 전달 받은



(a) 링크 연결 구성 (b) 트랜잭션 발생시의 묶음 구성

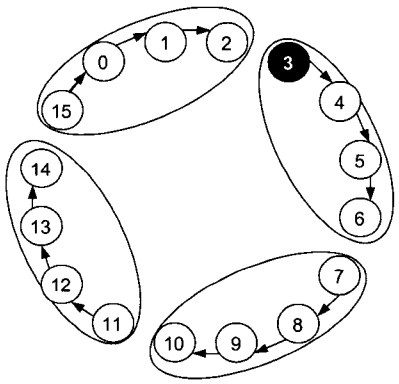
(그림 3) 제안하는 링크의 연결 구성 및 트랜잭션에 따른 묶음 구성

노드는 다시 다음 묶음의 노드에게 링크 B를 이용하여 전달된다. 이 경로를 표시하면 (그림 4)와 같다.



(그림 4) 링크 B를 통한 방송 트랜잭션 전달

방송 요청을 발생시켰거나 전달 받은 노드는, 자신의 묶음 내의 노드에게 링크 A를 통하여 요청을 전달하며, 각 묶음의 마지막 노드는 더 이상 요청을 전달하지 않는다. (그림 5)는 묶음 내에서 요청된 방송 트랜잭션의 전달을 나타낸 것이다.

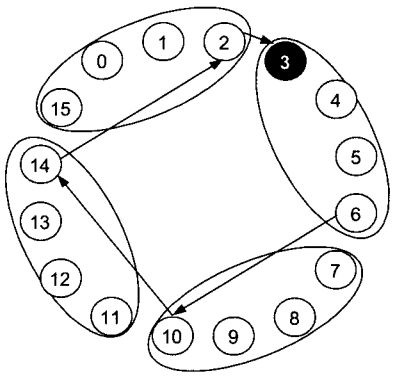


(그림 5) 링크 A를 통한 묶음 내 방송 트랜잭션 전달

여러 연결구조를 갖는 시스템에서 각 노드는 자신의 요청 트랜잭션에 대한 응답을 기다리고 있는 상태일 수 있으며, 이 경우 동일한 메모리 라인에 대한 요청이 다른 노드로부터 발생하였을 경우, 이러한 요청을 우선순위에 의해 받아들이거나 거부할 수 있다. 이 처리를 위해 요청 트랜잭션은 확인 트랜잭션을 반드시 필요로 하며, 이러한 확인 트랜잭션이 요청을 발생시킨 노드에 전달되어야 요청 트랜잭션이 모든 노드에 대하여 받아들여진 것이다.

확인 트랜잭션은 각 묶음 내에서 요청 트랜잭션을 링크 A를 이용하여 더 이상 전달할 곳이 없는, 즉 묶음 내의 마지막 노드가 발생시키며, 이 마지막 노드는 링크 A를 통하여 확인 트랜잭션을 발생시킨다.

요청 트랜잭션을 발생시킨 노드의 경우 묶음의 마지막 노드가 요청 트랜잭션을 받게 되면 링크 B를 통하여 확인 트랜잭션을 발생시키게 되며, 요청 트랜잭션을 링크 B를 통하여 받아들인 묶음의 경우, 묶음 내의 마지막 노드가 요청 트랜잭션을 받아들이고, 이전 묶음의 마지막 노드로부터 확인 트랜잭션이 링크 B를 통하여 전송되었을 경우 다시 링크 B를 통하여 확인 트랜잭션을 전달하게 된다. 마지막 묶음 내의 마지막 노드는 링크 A를 통하여 요청 트랜잭션이 자신에게 전달되고 링크 B를 통하여 확인 트랜잭션을 받을 경우 링크 A를 통하여 마지막 확인 트랜잭션을 전달하게 되며, 이를 받게 되는 노드는 애초 요청 트랜잭션을 발생시킨 노드가 된다. (그림 6)은 이러한 확인 트랜잭션이 전달되는 경로이다.



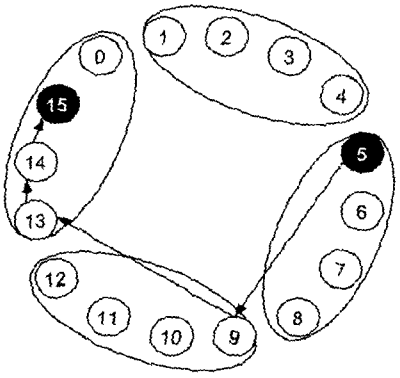
(그림 6) 각 묶음내 마지막 노드의 확인 트랜잭션 전달

이러한 트랜잭션 경로는 요청 트랜잭션의 발생에 위치에 따라 정적 경로가 설정되므로, 트랜잭션이 전달되는 경로에서 트랜잭션의 발생 노드 번호에 의해 각 노드는 전달 경로를 정적으로 설정한다. 트랜잭션의 전달에 있어 각 노드의 전달 경로 설정은 트랜잭션을 발생시킨 노드 번호에 해당되는 근원지 주소(Source ID) 값과 목적지에 해당되는 목적지 주소(Target ID) 값에 의하여 판단하며, 방송 트랜잭션의 경우 미리 고정한 값을 목적지 주소에 사용한다.

3.5 일 대 일 트랜잭션의 전송

일 대 일 트랜잭션의 전송은 응답과 되쓰기 트랜잭션에 해당되는데, 일 대 일 전송으로 이루어지므로 트랜잭션을 발생시킨 노드를 기준으로 하여 형성되는 묶음에서, 보내는 노드와 받을 노드가 동일한 묶음에 속하지 않을 경우 링크 B를 통하여 전송하며, 전송 경로상 노드가 전송의 목적지 노드와 같은 묶음에 해당될 때, 링크 A를 통하여 전송한다. (그림 7)은 5번 노드로부터 15번 노드로 일 대 일 트랜잭션이 전송되는 경우의 설정 경로이다.

방송 트랜잭션과 마찬가지로, 일 대 일 트랜잭션의 경우도 트랜잭션을 발생시킨 노드와 목적지 노드에 따라 정적으로 설정되므로 전달 경로의 각 노드는 트랜잭션 내의 노드 번호에 따라 경로를 설정한다.



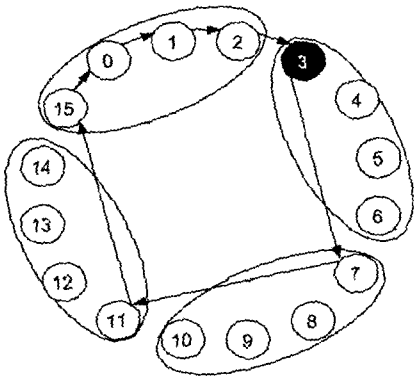
(그림 7) 일 대 일 트랜잭션 전송

3.6 트랜잭션의 경로 길이

다음은 계산식에 사용된 기호의 의미이다.

- N : 총 노드의 개수
- i : 한 묶음 내 노드의 개수
- k : 총 묶음의 개수 $N = ki$가 성립된다. $N > k$이고, (k, i는 각각 1보다 큰 정수)

반대방향 이중 링 구조에서, 방송으로 발생하는 요청 트랜잭션을 처리하기 위한 경로 길이 d는 총 노드의 수와 같으므로 $d = N$이다. 반면에 논문에서 제안한 링의 방송형태 요청 트랜잭션을 처리하기 위한 최장 경로 길이 d'은 트랜잭션을 발생한 노드부터 링의 전달방향으로 가장 멀리 있는 묶음의 첫 노드까지 전송되는 링크 B를 통한 $k-1$ 단계의 전달과, 마지막 묶음 내부에서 링크 A를 통한 $i-1$단계의 전달, 그리고 확인 트랜잭션이 마지막 묶음의 마지막 노드에서 요청을 발생시킨 노드로 전달하는 1단계의 링크 A를 통한 전달 경로가 된다. (그림 8)은 방송 형태인 요청 트랜잭션이 발생 전달되는 최장 경로를 표시한 것이다.



(그림 8) 요청 트랜잭션의 최장 경로

결국, 최장 경로는 $d' = k-1+i-1+1 = k+i-1$이 된다.

이 경로길이와 반대방향 이중 링의 경로의 차이 $d-d'$은 다음과 같다.

$dd' = Nk - Ii + 1 = ki - (k+i) + 1$이고, 모든 k와 i에 대하여 $ki - (k+i) + 1 > 1$이므로, 본 논문에서 제안한 방법은 방송 트랜잭션에 대해서 최소한 한 단계 이상의 경로를 단축하며, k, i가 커짐에 따라서 경로의 차이는 점차 증가하게 된다. 또한 $ki = N$이므로, 방송 트랜잭션은 총 노드의 개수가 많아질수록 개선이 나타난다.

일 대 일 트랜잭션의 경우 모든 노드 조합에 대해 동일한 비율로 트랜잭션이 발생한다고 할 경우 다음과 같은 평균 전송 경로를 가진다.

- 반대방향 이중 링의 평균 일 대 일 전송 경로 : $D = N(N+2)/8$
- 제안한 구조의 평균 일 대 일 전송 경로 : $D' = N(k-1)/2$
- 전송 경로의 차이 : $D - D' = N(N4k+6)/8$

따라서 $N4k+6 > 0$이면 항상 제안된 구조가 보다 짧은 전송 경로를 가지게 되므로, $N > 4k-6$을 유지하는 k 만큼의 묶음으로 나누어 링크 B를 구성하게 되면, 일 대 일 전송에서도 보다 짧은 전송 경로를 구성하게 된다.

4. 시뮬레이션 방법 및 결과

여러 개의 프로세서를 사용하는 다중 프로세서 시스템의 시뮬레이션 도구로 본 논문에서는 Augmint를 사용하였다[17]. Augmint는 Mint를 기반으로 x86 기계의 프로그램을 사용할 수 있도록 변경한 다중 프로세서 시뮬레이터로써 시험 대상의 실행 파일에 프로그램을 삽입하여 메모리 접근을 추적할 수 있다. 시뮬레이터 프로그램은 전단부와 후단부로 나뉘는데, 전단부는 프로세서로부터 발생하는 메모리 접근을 생성하고, 후단부는 프로세서 이외의 모든 메모리 접근 경로 및 네트워크를 필요에 따라서 구현하도록 되어 있으며, 본 연구에서는 반대방향 이중 링 및 제안한 구조의 시스템을 구현하였다.

성능 평가에 사용한 프로그램은 SPLASH-2 벤치마크[18] 프로그램의 일부를 사용하였으며, 다음 <표 2>와 같은 부하를 사용하였다.

시뮬레이션을 위하여 사용한 노드의 구조는 32노드의 시스템을 기준으로 하였으며, 각 노드 내에는 한 개씩의 프로세서를 가지고 있는 형태를 가정하였고, 프로세서 속도 및 링크의 속도 등은 다음 <표 3>과 같이 최신 프로세서의 속도 및 링크의 속도를 기준으로 하였다.

<표 2> 시뮬레이션에 이용된 프로그램 및 부하

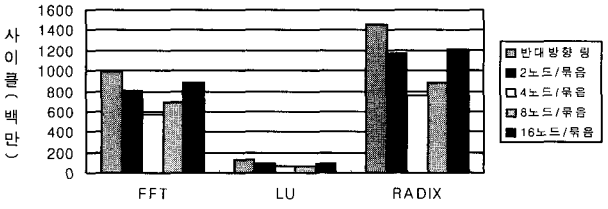
프로그램	부 하
FFT	-m16 -p32 n2048 -l5
LU	-n128 -p32 -b16
RADIX	-p32 -n131072 -r1024 -m2097152

〈표 3〉 실험 대상 시스템 환경

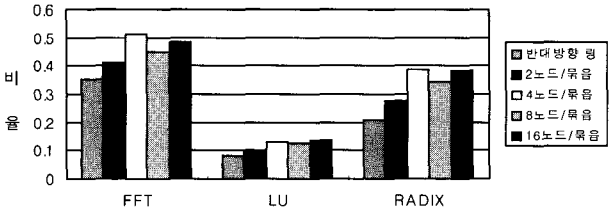
항 목	값
프로세서 클럭 속도	1GHz
시스템의 프로세서 수	32개
프로세서 당 캐시 크기	64Kbyte
프로세서 캐시 연관	4way
노드 개수	32개
노드 내 버스 속도	266MHz
노드 당 원격 캐시 크기	256Kbyte
원격 캐시 연관	8way
링크 전송 대역폭	1Gbyte/s
캐시 교체 정책	최근 최소 사용(LRU)

4.1 트랜잭션 처리 시간 지연 및 링크의 사용률

다음 (그림 9)는 각 프로세서에서 발생한 메모리 접근이 노드 내에서 처리되지 못하고 트랜잭션을 발생시킬 경우, 프로세서로부터의 접근 요청으로부터 트랜잭션을 발생시키고 처리 완료될 때까지 걸린 시간 지연이며, (그림 10)은 프로그램의 실행 시간 동안의 링크의 점유율을 나타낸다.



(그림 9) 트랜잭션이 발생한 경우의 메모리 접근 시간 지연



(그림 10) 링크 점유율

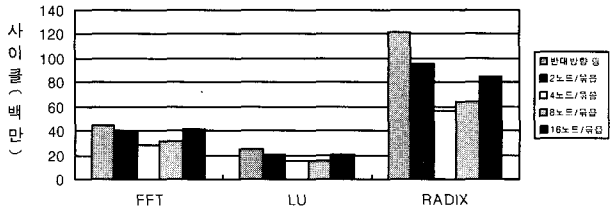
프로세서의 접근 요청으로부터 처리가 완료될 때까지의 시간은 모든 프로그램에서 8개의 묶음으로 분할하고, 각 묶음에 4개의 노드를 가지도록 한 경우가 가장 빠른 처리가 이루어졌음을 나타내며, FFT와 RADIX에 비하여 LU가 빠른 처리를 보인다. 이러한 이유는 (그림 10)에서 보여지는 것과 같이 LU에서 링크 점유율이 낮기 때문에 나타나며, 링크의 경쟁이 낮기 때문에 처리 지연 시간이 짧게 나타난 것이다.

또한 묶음을 몇 개로 구성하는가에 따라, 서로 다른 지연을 나타내었는데, 2개의 묶음으로 나누거나, 16개의 묶음으로 나누었을 경우, 상대적으로 긴 처리 지연 시간을 나타내었으며, 이러한 처리 지연동안 사용한 링크의 대역 폭은 16개의 묶음으로 나누었을 경우가 상대적으로 적게 나타났다.

링크 점유율은 프로그램의 수행시간동안 어느 정도의 링크 점유가 일어났는가를 나타내는가를 의미하므로, 본 논문에서 제안한 구조가 보다 빠른 시간에 프로그램 수행을 완료할 경우 높은 링크 점유율을 보이게 되므로, 프로그램의 실행 시간에 대비하여 비교해야 한다.

4.2 실행 시간

다음 (그림 11)은 각 프로그램을 수행하는 데 걸린 시간이다.



(그림 11) 프로그램 실행 시간

모든 경우에서 프로그램의 실행 시간은 반대방향 이중 링 구조에 비하여 빠른 처리를 나타내었으며, 묶음 당 4개의 노드를 할당하고, 총 8개의 묶음으로 나누었을 경우가 가장 높은 성능을 나타내었다. 그 다음으로 빠른 성능을 보인 경우는 묶음 당 8개의 노드를 할당하고 총 4개의 묶음으로 나눈 경우이다.

앞서 보았던 링크 점유율과 실행 시간을 대비하여 볼 때, 묶음 당 2개의 노드를 할당한 경우와 16개의 노드를 할당한 경우는 유사한 정도의 실행시간을 나타내었다. 반면 앞서 링크 점유율은 묶음 당 두 개의 노드를 할당한 경우가 상대적으로 낮은 점유율을 보였으므로 이 두 경우에서는 묶음 당 적은 노드를 할당하는 경우가 보다 많은 노드를 할당하는 경우에 비해 높은 확장성을 가질 수 있음을 알 수 있다.

5. 결 론

최근 마이크로프로세서의 동작 클럭은 기가헬즈 이상으로 발전하였고, 발전한 반도체 미세공정 기술의 덕분으로 점차 고속화되어 가고 있다. 이러한 프로세서들을 연결하여 보다 높은 성능의 시스템을 구성하는 기술은 지속적으로 연구되어 왔으며, 높은 병렬성으로 얻어진 처리 능력의 확장은 복잡한 계산 및 처리를 필요로 하는 응용에 적절히 사용되고 있다.

그러나 고속으로 급속히 발전해나가는 프로세서의 성능에 비하여 상대적으로 저속인 상호연결망의 처리 능력은 시스템의 성능 및 확장성의 제한 요소로 작용하였다.

따라서 상호연결망의 연결 구성 및 방법에 관한 연구가 지속적으로 이루어졌으며, 최근 수 년간 SCI와 같은 점 대

점 연결 구조를 이용한 링 구조의 시스템이 활발히 연구, 개발된 바 있다.

이러한 점 대 점 연결 구조의 시스템 또한 상호연결망의 처리 능력 확장과 지연 시간의 개선을 위하여 복수개의 점 대 점 연결을 갖는 구조로 발전해왔으며, 복수개의 연결을 구성하는 방법과 연결 경로를 설정하는 방법에 따라서 트랜잭션의 지연 시간 및 시스템 전체의 성능에 영향을 주게 된다.

본 논문은 두 개의 점 대 점 연결 중 한 연결을 링 형태로 구성하고 연속된 몇 개의 노드를 묶음으로 분할하여, 나머지 한 연결을 다른 묶음 내의 일정한 노드와 연결되도록 배치하고, 이러한 이중 연결 구조에서 방송 트랜잭션에 적합한 전송 방법과 일 대 일 트랜잭션에 적합한 전송 방법을 고안하고 제시하였다.

본 논문에서 제시한 전송 방법은 트랜잭션의 종류 및 전송 노드의 고유 주소에 따라서 정적인 경로 설정을 수행하므로, 경로 설정을 위한 부가적인 처리시간 지연을 발생시키지 않으며, 시뮬레이션을 통하여 이러한 시스템이 반대방향 이중 링 구조의 시스템에 비하여 1.42~2.1배 빠른 수행 시간을 보임을 입증하였다.

이러한 결과는 노드 당 2개의 링크를 갖는 CC-NUMA 시스템 구성에서 방송 트랜잭션과 일 대 일 트랜잭션에 대하여 적합한 시스템 구조와 적절한 트랜잭션 전송 경로 설정방법을 제시한 것이며, 차후 과제로서 2개의 링크를 갖는 시스템에서 일 대 일 전송만으로 트랜잭션이 구성될 경우 적절한 시스템 구조와 전송 경로 설정 방법의 연구가 진행 중이다.

참 고 문 헌

[1] D. S. Henty, "Performance of Hybrid Message-Passing and Shared-Memory Parallelism for Discrete Element Modeling," Proc. of the 2000 ACM/IEEE conf. on Supercomputing, article No.10, 2000.

[2] G. A. Abandah and E. S. Davidson, "Characterizing Distributed Shared Memory Performance : A Case Study of the Convex SPP1000," IEEE Trans. on Parallel and Distributed Systems, Vol.9, No.2, pp.206-216, Feb., 1998.

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

[4] Jonathan Kay and Joseph Pasquale, "Profiling and Reducing Processing Overheads in TCP/IP," IEEE Trans. on Networking, Vol.4n, No.6, pp.817-828, 1996

[5] H. Oi and N. Ranganathan, "Performance Analysis of the Bidirectional Ring-Based Multiprocessor," Proc. of ISCA 10th International Conf. on Parallel and Distributed Computing Systems, pp.397-400, October, 1997.

[6] http://www.ibm.com/servers/eserver/clusters.

[7] S. J. Eggers and R. H. Katz, "Evaluating the performance of four snooping cache coherency protocols," Proc. of the 16th International Symposium on Computer Architecture, New York, June, 1989.

[8] T. Anderson and D. Culler and D. Patterson, "A Case for NOW (Networks of Workstations)," IEEE Micro, Vol.15, No.1, pp.54-64, Feb., 1995.

[9] http://www.beowulf.org.

[10] Message Passing Interface Forum, MPI : A Message-Passing Interface Standard, UT-CS-94-230, 1994.

[11] Intel, Compaq and Microsoft Corporations, Virtual Interface Architecture specification Version 1.0, December, 1997, http://www.viarch.org/.

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

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

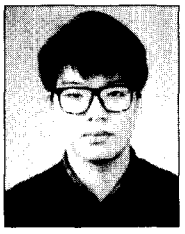
[14] http://www.dg.com/.

[15] http://panda.snu.ac.kr/nrl/.

[16] Hitoshi Oi and N. Ranganathan, "A Comparative Study of Bidirectional Ring and Crossbar Interconnection Networks," Proc. of the 1998 International Conf. on Parallel and Distributed Processing Techniques and Applications, pp. 883-890, July, 1998.

[17] A-T. Nguyen, M. Michael, A. Sharma and J. Torrellaz, "The Augmint multiprocessor simulation toolkit for Intel x86 architecture," Proc. of the IEEE International Conference on Computer Design, Oct., 1996.

[18] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta. "Methodological considerations and characterization of the SPLASH-2 parallel application suite," Proc. of the 22th Annual International Symposium on Computer Architecture, pp.24-36, 1995.



서 효 중

e-mail : hjsuh@catholic.ac.kr

1991년 서울대학교 이학사

1994년 서울대학교 공학석사(컴퓨터공학)

2000년 서울대학교 공학박사(컴퓨터공학)

2002년 지씨티 리서치 선임연구원

2003년~현재 서울대학교 컴퓨터연구소 객원연구원

2003년~현재 가톨릭대학교 컴퓨터정보공학부 전임강사

관심분야 : 컴퓨터 구조, 병렬처리 시스템, 내장형시스템, 클러스터 시스템