Torus Ring : 계층 링 구조의 변형을 통한 상호 연결망의 성능 개선

(Torus Ring : Improving Performance of Interconnection Networks by Modifying Hierarchical Ring)

곽 종 욱 † 반 형 진 †† 전 주 식 †††
(Jong Wook Kwak) (Hyong Jin Ban) (Chu Shik Jhon)

요 약 다중 프로세서 시스템에서 노드 간의 연결을 제공하는 상호 연결망이 전체 시스템의 성능에서 차지하는 비중은 매우 크다. 상호 연결망의 형태는 여러 종류가 있을 수 있으나 Mesh, 링, 계층 링 등의 형태가 많이 사용된다.

이 논문에서는 기존의 계층 링을 수정한 Torus Ring을 제안한다. Torus Ring은 계층 링과 완전히 동일한 복잡도를 가지면서도 지역 링 간의 연결 방법만을 변경한 형태의 상호 연결망이다. 이 연결망은 역방향 인접 링에 대한 요청에서 홉 수의 이득을 봄으로써 평균 홉 수를 감소시킨다. 또한 접근의 지역성을 고려하지 않은 균등분포의 가정 하에서도 평균 홉수의 기대값에서 계층링과 동일한 값을 가지며, 실제 병렬 프로그램이 수행되는 환경에서는 인접링에 대한 통신 비율이 증가할 가능성이 크기 때문에 더 큰 홉 수의 이익을 기대할 수 있다. 이에 따라 상호 연결망의 요청과 응답의 지연 시간이 최대 19%까지 감소하였으며, 이러한 응답 지연 시간의 단축이 수행 시간을 최대 10% 정도까지 감소시키는 결과를 가져왔다.

키워드 : 다중 프로세서 시스템, 상호 연결망, 링 연결망, 계층 링 연결망, 망 토폴로지

Abstract In multiprocessor systems, interconnection network design is critical for overall system performance. Popular interconnection networks, which are generally considered, are meshes, rings, and hierarchical rings.

In this paper, we propose "Torus Ring", which is a modified version of hierarchical ring. Torus Ring has the same complexity as the hierarchical rings, but the only difference is the way it connects the local rings. It has an advantage over the hierarchical rings when the destination of a packet is the neighbor local ring in the reverse direction. Though the average number of hops in Torus Ring is equal to that of the hierarchical rings when assuming the uniform distribution of each transaction, the benefits of the number of hops are expected to be larger because of the spatial locality in the real environment of parallel programming. In the simulation results, latencies in the interconnection network are reduced by up to 19%, and the execution times are reduced by up to 10%.

Key words : Multiprocessor System, Interconnection Network, Ring Network, Hierarchical Ring Network, Network Topology

1. 서 론

고성능 컴퓨터 시스템의 급격한 성능 향상에 따라 사용자들의 컴퓨터 시스템의 성능 향상에 대한 기대치는 점차 증가하게 되었으며 이에 부응하기 위한 노력의 일환으로 다중 프로세서 시스템이 사용되고 있다. 다중 프로세서 시스템에서 사용되는 메모리 모델은 공유 메모리 모델과 메시지 전달 메모리 모델로 나누어 볼 수 있다[1].

또한 다중 프로세서 시스템은 메모리 접근 시간에 따라 NUMA 시스템과 UMA 시스템으로 분류할 수 있는데, 특히 분산된 공유 메모리를 사용하는 대규모의 시스템에서는 시스템 내의 여러 곳에 메모리가 분산 배치되어 각 메모리에 접근하는 시간이 상이한 NUMA 시스

† 학생회원 : 서울대학교 전기컴퓨터공학부
leoniss@panda.snu.ac.kr
†† 비 회 원 : 삼성전자
jin284@panda.snu.ac.kr
††† 종신회원 : 서울대학교 전기컴퓨터공학부 교수
csjhon@riact.snu.ac.kr
논문접수 : 2004년 9월 1일
심사완료 : 2005년 1월 21일

템이 주로 사용된다. 이러한 NUMA 시스템에서는 각 노드를 연결하는 상호 연결망이 시스템의 성능을 크게 좌우하게 된다. 그림 1은 4장의 시스템 성능 분석에서 논의하게 될 모의 실험 환경 하에서 수행된 병렬 프로그램들의 수행 시간 구성 요소를 그래프로 나타낸 것이다. 그림 1에서 보여지듯이, 원격지 노드 메모리에 대한 접근인 "remote"에 할애되는 부분이 전체 프로그램 수행 시간에서 평균 34%, 최대 77%까지 차지한다.

이러한 상호 연결망의 성능 향상을 위한 노력으로는 상호 연결망의 성능 자체를 개선하는 방법과 상호 연결망의 이용을 최적화하는 방법이 있다. 전자로는 연결망의 형태(Topology)를 개선하거나, 라우팅이나 흐름 제어(Flow Control) 개선 등의 방법이 있을 수 있겠고, 후자로는 원격 접근 캐시(Remote Access Cache)의 성능을 개선하는 등 상호 연결망의 이용을 최대한 줄여 빠른 메모리 접근을 가능하게 하려는 방법 등이 있다.

본 논문에서는 전자와 같은 노력의 연장선상에서, 새로운 다중 프로세서 상호 연결망의 형태를 제안하고 그의 성능을 기존의 상호 연결망과 비교 평가해 본다.

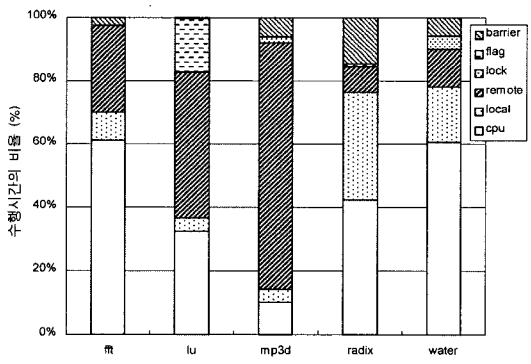


그림 1 병렬 프로그램의 수행 시간 (Mesh, 16노드)

본 논문에서는 기존의 다양한 형태의 상호 연결망들의 성능을 비교해보고, 계층 링의 경우 어떻게 구성하는 것이 가장 성능 향상에 유리한지를 알아본다. 또한 계층 링을 변형한 형태인 Torus Ring을 제안하고 이 연결망이 기존의 계층 링 연결망에 비해 어느 정도의 성능 향상을 가져오는지, 또 어떻게 구성하는 것이 가장 효율적인지 실험을 통해서 알아본다.

본 논문의 구성은 다음과 같다. 2장에서는 Mesh, 단일 링, 계층 링 등의 기존 상호 연결망에 대해서 알아보고 이들의 성능을 평가하는 기준에 대해서 설명한다. 3장에서는 본 논문에서 제안하는 Torus Ring의 형태에 대해서 설명하고 그 특징과 구현에 대해서 논의한다. 4장에서는 시뮬레이션을 통해 위에서 설명된 상호 연결망들의 성능을 비교, 평가한다. 끝으로 5장에서는 결론 및 향후 과제를 설명한다.

2. 관련 연구

다중 프로세서 시스템에서 각 프로세서 상호간의 통신을 지원하기 위한 네트워크 연결망을 다중 프로세서 노드 간 상호 연결망이라고 한다. 상호 연결망은 그 형태나 성능 등에 따라서 여러 방법으로 분류될 수 있는데, 이 절에서는 본 논문에서 논의하고자 하는 네트워크 형태(Topology)에 따른 분류 중, 비교의 대상이 되는 연결망의 형태들을 논의한다.

2.1 Mesh 연결망

Mesh 연결망은 그림 2와 같은 형태의 상호 연결망으로 가장 널리 사용되는 형태의 상호 연결망이라고 할 수 있다. 그림 2의 사각형은 프로세서 노드를 포함하고 있는 스위치를 의미하며, 화살표로 표시된 선들은 패킷이 이동할 수 있는 링크를 나타낸다.

Mesh 연결망의 특징은 각 스위치에서 X와 Y축 어느 쪽으로든 패킷을 보낼 수 있어 평균 홉 수가 적으며, 연결망을 둘로 나누는 단면의 대역폭인 단면 대역폭(Bisection Bandwidth)이 높아 동시에 전송할 수 있는 대역폭이 크다는 것이다. $k \times k = N$ 으로 구성된 Mesh일 때 단면 너비(Bisection Width)는 k가 되고, 연결망 내에서 가능한 가장 긴 홉 수인 지름(Diameter)은 $2(k-1)$로 짧은 편이다. 다만 주변부를 제외한 모든 스위치가 양방향의 Degree-4를 가짐으로 복잡도가 상당히 큰 편이라고 할 수 있다.

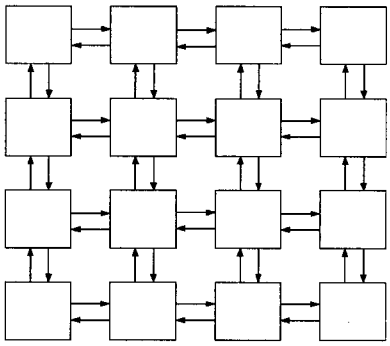


그림 2 Mesh 연결망

2.2 링 연결망

링 연결망은 그림 3과 같이 프로세서 노드를 링 형태로 연결한 비교적 간단한 형태의 연결망이다. 링 연결망은 그 간단한 구성 방법으로 인해 노드의 추가 및 제거가 쉽고, 낮은 복잡도로 인해 빠른 속도의 운용이 가능

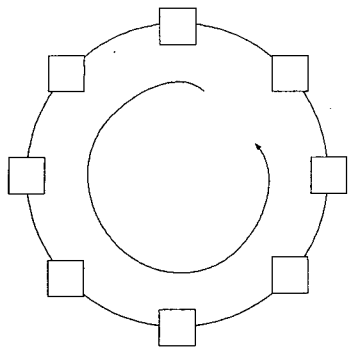


그림 3 링 연결망

하다. 또한 링은 캐시 일관성(Cache Coherence)과 메모리 일관성(Memory Consistency)을 효과적으로 구현할 수 있는 특징을 가지는데, 요청/응답의 배치와 방송(Broadcast)의 용이함이 그것이다. 예컨대 하나의 무효화(Invalidation) 요청이 링 연결망을 순회하면서 여러 캐시 라인을 무효화 할 수 있도록 구현될 수 있다. 이 같은 이익의 극대화를 위해 스누핑(Snooping) 프로토콜을 링 연결망에서 사용하여 성능을 향상시키는 연구도 진행된 바 있다[2].

링 연결망 중에서도 패킷의 전달 방향을 양방향으로 하거나, 단방향의 링을 이중으로 사용하는 등 여러 구성 방법이 있다. 단방향의 링을 이중으로 사용하는 경우에는 트랜잭션의 방향에 따라 링을 구별해 사용하는 방법이 성능면에서 우수함이 밝혀져 있다[3].

링 연결망의 단면 너비는 2이며 지름은 전체 노드 수가 N일 때 N-1이다. 각 스위치는 Out Degree-1의 복잡도를 가진다. Mesh 연결망 등에 비해서 스위치 복잡도가 낮아서 보다 높은 속도의 패킷 전송이 가능한 장점이 있으나 지름이나 단면 대역폭 등에서 노드 수가 많아질수록 한계를 보인다.

2.3 계층 링 연결망

그림 4와 같이 단일 링 형태의 지역 링을 상위 계층의 전역 링이 연결하고 있는 형태가 계층 링 연결망이다. 계층 링 연결망은 단일 링 연결망의 장점들을 거의 그대로 취하면서도 단일 링 연결망보다 노드 수의 증가에 따른 부담이 적다는 장점이 있다. 데이타의 지역성이 어느 정도 존재한다면, 128 프로세서 노드 정도까지는 계층 링 연결망이 동일한 비용의 Mesh 연결망에 비해서 성능이 나쁘지 않다는 연구 결과도 있다[4,5]. Hector[6]나 NUMAchine[7]과 같은 시스템은 노드 내부에서는 버스를 이용하여 프로세서들을 연결하고 각각의 노드를 연결하는 상호 연결망은 2단계 계층 링 연결망을 이용한다. 스누핑 프로토콜을 사용한 링 연결망에서도 프로토콜의 확장을 통해 계층 링 연결망으로 확장

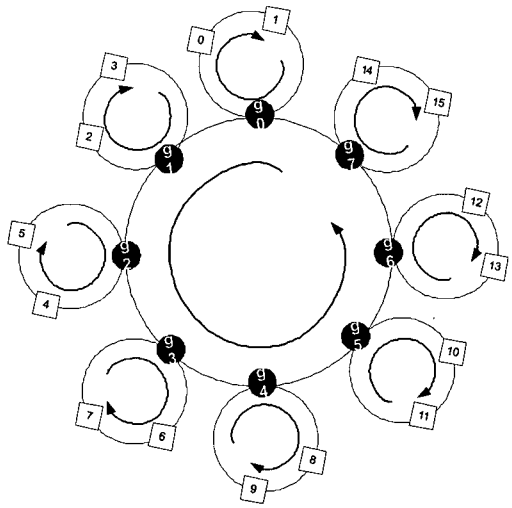


그림 4 2단계 계층 링 연결망

하면 전체 시스템 성능이 크게 개선된다.

여러 개의 단일 링을 링이 아닌 다른 형태의 여러 단계의 연결망으로 연결하는 연구들도 있었는데, Multi-stage Ring Network(MRN)[8]이나 Shuffle-Ring[9], Recursive Cube of Rings(RCR)[10] 등을 들 수 있다. 이들 연구의 공통된 특징은 고정된 낮은 크기의 스위치 Degree를 이용하면서 단순한 계층 링보다 단면 대역폭의 향상이나 지름의 감소를 추구했다는 것이다. 하지만 이러한 연결망의 경우 단일 링에서의 장점이었던 패킷 방송의 용이함과 노드의 추가 및 제거의 편리성은 취할 수 없게 된다. 한편, 스누핑 캐시 일관성 프로토콜을 사용하는 링 구조 NUMA 시스템에서 연결망의 구조를 계층링으로 확장하며 이때 추가적으로 필요한 효율적인 스누핑 캐시 일관성 프로토콜을 제안한 연구[11]도 있다. 또한, NUMA 시스템의 연결망으로서 링 구조가 가지는 단점을 극복하고자 다양한 형태의 링 구조 변형을 제안하고 이를 지원하기 위한 리피터 구조를 제시한 연구[12]도 있다.

2단계 계층 링은 지역 링과 전역 링으로 구성될 수 있는데, 이는 $m \times n$과 같이 나타낼 수 있고, m은 지역 링의 개수, n은 지역 링 내의 노드의 개수를 의미한다. 따라서 그림 4는 8×2의 계층 링 형태라고 할 수 있다. 그림에서 흰색의 정사각형은 프로세서 유닛이 달려 있는 노드를 의미하고, 검은 색의 원은 프로세서 유닛이 없는 전역 링 스위치를 의미한다. 정사각형 내의 번호는 해당 프로세서 노드의 번호를 의미하고, 전역 스위치 안의 번호는 전역 스위치의 번호를 의미하는데, 프로세서 노드와 구분하기 위해 앞에 g를 붙여 지칭한다. 또한 전역 스위치 gn에 연결된 지역 링을 지칭할 때 간단히

n번 지역 링이라고 일컫도록 하겠다.

3. Torus Ring의 구성과 특징

3.1 Torus Ring의 구성

본 논문에서 제안하는 Torus Ring은 그림 5와 같은 형태의 링 연결 구조이다.

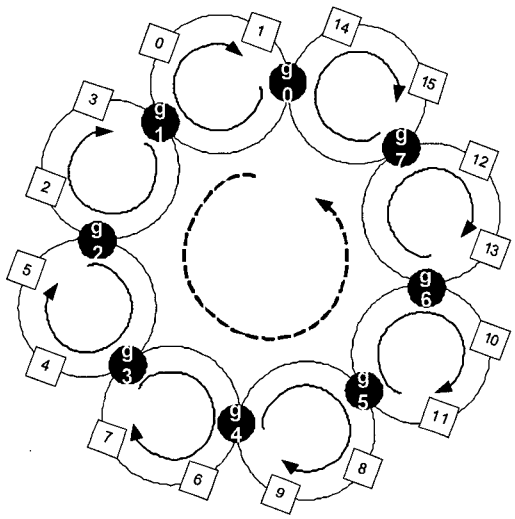


그림 5 Torus Ring 연결망

사각형 내의 숫자는 이전의 연결 구조 그림들에서와 마찬가지로 프로세서 유닛이 달려 있는 노드 번호를 의미하며, 검은색의 원은 프로세서 유닛이 없이 스위치 역할만 하는 "전역 링" 스위치를 의미한다. Torus Ring에서는 계층 링에서와 같이 명시적인 전역 링이 존재하는 것은 아니지만, 검은색의 스위치들이 인접한 지역 링을 연결하고 있으므로 이들을 편의상 전역 링 스위치라고 부르고, 이 스위치들을 통한 지역 링 간의 통신도 전역 링을 통한 통신이라고 지칭하도록 하겠다. Torus Ring도 계층 링과 마찬가지로 동일한 수의 노드를 가진 시스템이라도 여러 가지 방법으로 구성할 수 있는데, 이 구성을 역시 $m \times n$(지역 링의 개수 m, 지역 링 내의 노드 수 n)으로 지칭한다. 즉 그림 5의 경우 8×2 Torus Ring이라고 할 수 있다.

Torus Ring은 기본적으로는 전체 노드를 몇 개의 지역 링으로 나누어 이 지역 링을 연결하는 링을 구성한다는 점에서 계층 링과 비슷하다고 할 수 있다. 하지만 여러 지역 링을 연결하는 방법에 있어서, 계층 링이 명시적으로 전역 링을 만들어 연결한다면 Torus Ring은 지역 링의 링크를 전역 링이 공유한다는 것으로 볼 수 있다. 즉 계층 링에서는 패킷의 출발지 노드와 목적지 노드가 서로 다른 링에 존재할 때에 지역 링의 링크와 따로 존재하는 전역 링의 링크를 거쳐 지나가야 하지만, Torus Ring에서는 전역 링의 링크가 따로 구성되지 않고 지역 링의 링크를 이용하도록, 인접한 두 개의 지역 링들끼리 전역 링 스위치를 이용해 직접 붙어 있는 구조를 취하고 있다. 이에 따라 하나의 지역 링에 전역 링 스위치가 2개씩 붙어 있는 형태가 된다.

Torus Ring에서의 라우팅은 계층 링에서와 마찬가지로 간단하다. 지역 링 내에서 서비스 되어야 하는 패킷의 경우에는 전역 링 스위치의 링 내 패킷 전달로 전송이 이루어지고, 다른 지역 링으로 전송되어야 하는 패킷의 경우에는 전역 링 스위치 간의 연결 링크들(그림 5의 g0→g1, g1→g2 등과 같은 링크)을 이용해 다른 지역 링으로 패킷을 전송한다. 교착 상태를 회피하기 위한 라우팅의 제한 방법은 뒤쪽에서 설명하도록 하겠다.

3.2 Torus Ring의 특징

Torus Ring의 특징을 알아보기 위해 Torus Ring이 일반적인 계층 링보다 성능이 좋을 수 있는 경우와 그렇지 않은 경우를 나누어 생각해 보도록 하겠다. 혼란을 막기 위해 아래의 설명은 노드의 배치 방법이 각각 계층 링의 경우 그림 4, Torus Ring의 경우 그림 5와 같다고 가정하고 있다. 그리고 Torus Ring이나 계층 링에서 각각의 지역 링은 2개씩의 인접한 지역 링을 가지는데, 전역 링의 패킷 전송 방향에 따라 이들을 순방향 인접 링, 역방향 인접 링으로 구분하여 부르기로 한다.

먼저 Torus Ring이 계층 링보다 성능이 좋을 수 있는 경우는 인접한 지역 링간에서 통신이 이루어지는 경우이다. 인접한 지역링간의 통신은 요구에 대한 요청과 그에 대한 응답으로 구분해 보았을때 "순방향 인접링에 대한 요청/역방향 인접링에 대한 응답"과 "역방향 인접링에 대한 요청/순방향 인접링에 대한 응답"으로 구분지어 볼 수 있다. 이 경우 Torus Ring은 최종적으로 $m-2$ 홉의 이득을 볼 수 있으며, 특히 역방향 인접 링간의 통신에 있어서 Torus Ring은 목적지 지역 링과 출발지 지역 링을 직접 연결하고 있는 전역 링 스위치를 사용할 수 있으므로 전역 링에서의 추가적인 비용 없이 0 홉으로 지역 링을 바꿔 탈 수 있다. 계층 링의 해당 경우에서는 $m-1$ 홉의 링크를 건너가야 목적지 링으로 갈 수 있다.

Torus Ring에서 지역 링을 건너가는 데에 0 홉이 걸리는 경우는 역방향 인접 링이 목적지 지역 링인 경우이고, 1 홉이 걸리는 경우는 순방향 인접 링이 목적지 지역 링인 경우이다. 계층 링에서 1 홉이 걸리는 경우는 순방향 인접 링이 목적지 지역 링인 경우이고, $m-1$ 홉이 걸리는 경우는 역방향 인접 링이 목적지 지역 링인 경우이다. 순방향의 경우 링을 건너기 위한 홉 수는 계

층 링과 Torus Ring의 경우에 있어서 1홉으로 같지만, 사실상 목적지 노드를 기준으로 보았을 때 Torus Ring이 계층 링에서보다 홉 수가 1 홉 더 많다. 이것은 두 연결 구조 모두 전역 링을 사용해 지역 링을 바꾸는 데는 동일하게 1 홉이 걸리지만 Torus Ring은 목적지 지역 링에 도착하고 나서 또 하나의 전역 링 스위치를 지나야 하기 때문이다. 예컨대 노드 0에서 노드 2로 전송되는 패킷의 경우(그림 5 참조) Torus Ring에서는 0→1→g0→g1→g2→2의 5 홉, 계층 링에서는(그림 4 참조) 0→1→g0→g1→2의 4 홉이 걸리게 된다.

하지만 목적지 지역 링이 역방향 인접 링인 경우에는 Torus Ring이 계층 링과 비교했을 때 m-1 홉의 이득을 그대로 볼 수 있다. 그림 5와 같은 Torus Ring의 구성상 역방향 인접 링에서는 목적지 지역 링에 도착한 후에는 다시 전역 링 스위치를 지나지 않기 때문이다. 노드 0에서 노드 14로 전송하는 패킷의 경우가 이에 해당한다고 할 수 있다. Torus Ring에서는 0→1→g0→14의 3 홉, 계층 링에서는 0→1→g0→g1→·····→g7→14의 10 홉이 걸리게 됨을 알 수 있다.

Torus Ring이 계층 링보다 성능이 나쁜 경우는 지역 링 내에서 통신이 이루어 지는 경우나 인접하지 않은 목적지 링으로 패킷을 보내야 하는 경우이다. 지역 링 내에서의 통신의 경우에는 계층 링에서는 1개의 전역 링 스위치만 통과하면 되지만 Torus Ring에서는 2개의 전역 링 스위치를 통과해야 하기 때문에 적어도 1 홉이 반드시 추가될 수 밖에 없다. 또한 목적지 지역 링이 출발지 지역 링과 인접하지 않을 때, 목적지 지역 링 도착 이후 또 하나의 전역 링 스위치를 거쳐야 하기 때문에 Torus Ring이 계층 링에 비해서 1 홉씩 손해를 보게 된다.

이렇듯 Torus Ring은 계층 링에 비해 장점을 가질 수 있는 경우도 있으나 성능상 손해를 가져오는 경우도 있는데, 연결망에서의 통신의 특성을 고려해 보면 Torus Ring이 계층 링에 비해서 사실상 가져오는 이득은 더욱 커진다고 할 수 있다. 다중 프로세서를 연결하는 내부 연결망의 프로토콜들의 특성상 대부분의 경우 요청이 보내지면 그에 따른 응답이 최초의 요청 노드로 보내지는데, 이 경우 인접한 지역 링간의 통신에서 순방향 목적지 지역 링은 역방향 목적지 지역 링으로, 역방향 목적지 지역 링은 순방향 목적지 지역 링으로 바뀌게 되므로 이 요청/응답을 한 묶음으로 생각해서 계산한다면 평균적으로 $(m-1)+(-1)=m-2$ 홉의 이득을 보게 된다고 할 수 있다. 또한 지역 링 내에서의 통신이나 인접하지 않은 링 간의 통신에서 발생하는 계층 링에 비한 손해는 1 홉으로 고정적이다. 따라서 이는 지역 링의 크기가 커질수록 상대적으로 영향력이 작아지게 되는데, 이것은 후에 실험을 통해서 결과를 알아보도록 하겠다. 한편 패킷 홉 수의 균등분포(Uniform Distribution)를 가정했을 때에 Torus Ring의 계층 링에 대한 평균적인 홉 수의 이익 및 손해의 기대값은 다음과 같다.

$$(m-1)\times\frac{1}{m}-1\times(1-\frac{1}{m})=0$$

즉 패킷의 균등분포를 가정한 경우에도 Torus Ring은 홉 수에서 손해를 보는 것이 아니며, 실제 병렬 프로그램이 수행될 때에는 가까운 인접 링에 대한 통신의 비율이 공간 지역성에 의해 $\frac{1}{m}$ 보다는 클 가능성이 높기 때문에 홉 수 상의 더욱 큰 이익을 기대해 볼 수 있다.

3.3 Torus Ring의 라우팅 방법

Torus Ring에서의 라우팅은 교착 상태를 회피하기 위해서 높음 채널과 낮음 채널 두 개의 가상 채널을 사용한다[13]. 계층 링에서 서로 다른 지역 링 간의 통신에서는 전역 링에서 가상 채널을 변경함으로써 채널 종속성을 끊어주지만[14], Torus Ring에서는 각각의 지역 링의 가상 채널을 사용해 종속성을 제거해 준다. 구체적인 라우팅 방법은 아래와 같다.

패킷이 최초 출발지 지역 링 내의 지역 링 스위치에 도착했을 때, 그 스위치는 패킷의 목적지 노드가 이 지역 링 내에 있는 것인지를 보고 그렇다면 단일 링에서와 마찬가지의 방법으로 라우팅한다. 곧 지역 링 내에서의 목적지 노드 번호가 현재 패킷이 위치한 노드 번호보다 크면 높음 채널, 작으면 낮음 채널로 라우팅한다. 그렇지 않고 패킷이 전역 링 스위치를 통해 다른 지역 링으로 나가야 하는 경우라면, 해당 전역 링 스위치를 지역 링 스위치들 중 가장 작은 노드 번호를 가지는 것으로 간주하여 항상 낮음 채널로 라우팅한다. 따라서 다른 지역 링으로 나가야 하는 패킷의 경우 출발지 지역 링에서는 항상 낮음 채널을 사용하게 된다.

전역 링 스위치에서는 도착한 패킷의 목적지 지역 링이 자신이 속한 지역 링이라면 항상 높음 채널을 사용해 그 지역 링을 돌도록 해 준다. 그렇지 않다면 전역 링을 회전하면서 목적지 지역 링을 찾아야 하는데, 이때는 전역 링 내에서의 자신의 번호(그림 5의 검은 원 안의 숫자, 몇 번째 지역 링과 연결된 전역 링 스위치인지를 나타낸다)보다 목적지 전역 링의 번호가 크면 높음 채널을, 목적지 전역 링의 번호가 작으면 낮음 채널을 사용한다. 그리고 마지막으로 목적지 지역 링으로 연결해 주는 전역 링 스위치에 도착하면 전술한 바와 같이 높음 채널을 사용해 목적지 노드까지 찾아가면 된다.

이와 같은 방법으로 라우팅을 하면 목적지 전역 링의 번호가 출발지 전역 링의 번호보다 크기 때문이든 아니면 목적지 전역 링의 번호가 작더라도 목적지 지역 링

에 도착했기 때문이든 항상 낮음 채널→높음 채널의 방법으로 채널을 변경해 나간다. 따라서 역시 낮음 채널과 높음 채널 각각의 채널 종속성 그래프가 사이클이 없음을 보이면 이 라우팅 방법은 교착 상태가 없음을 보일 수 있게 된다.

그림 6은 채널 종속성 그래프를 보여주기 위한 Torus Ring 구조의 예시이다. 이해를 돕기 위해 각 링크 옆에 채널 번호를 표시하였다. 그림 6에서는 각 링크에 하나의 채널 번호만을 표시하였지만, 각 링크마다 낮음과 높음의 두 개의 채널이 존재한다.

그림 7은 그림 6의 채널 종속성 그래프[14]이다. 채널 종속성 그래프의 정점은 상호 연결망에서의 채널을 의미하고, 그래프의 정점들을 연결하는 화살표는 화살표의 시작 정점(채널)에서 화살표의 끝 정점(채널)으로 라우팅이 가능함을 나타낸다. 왼쪽의 (a)는 낮음 채널을 나타낸 것이고 오른쪽의 (b)는 높음 채널을 나타낸 것이다. 편의상 (a)의 채널 번호와 (b)의 채널 번호를 동일하게 표시하였지만 이들은 실제로는 한 링크에 존재하는 서로 다른 채널이다. 그림 7에서 보는 바와 같이, 1) 지역 링 내부, 2)인접 지역 링 사이, 3)인접하지 않는 지역 링 사이의 낮음 채널과 높음 채널의 그래프에는 모두 사이클이 없고 따라서 이 라우팅 방법은 교착 상태가 없다.

그림 7의 채널 종속성 그래프에 대해서 설명을 하기 위해 지역 링 내의 프로세서 노드에서 나오는 채널을 나타낸 정점은 지역 정점, 전역 링 스위치에서 나오는 채널을 나타낸 정점은 전역 정점이라고 지칭하기로 한다. 하나의 전역 정점은 높음 그래프와 낮음 그래프 각각에 2개씩 존재하는데, 이는 전역 스위치가 본래 두 개

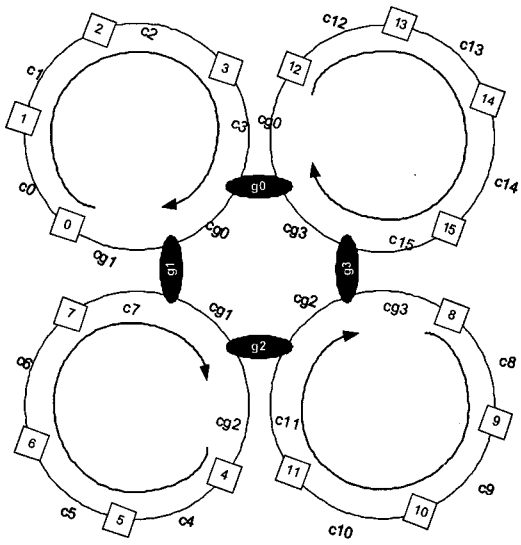
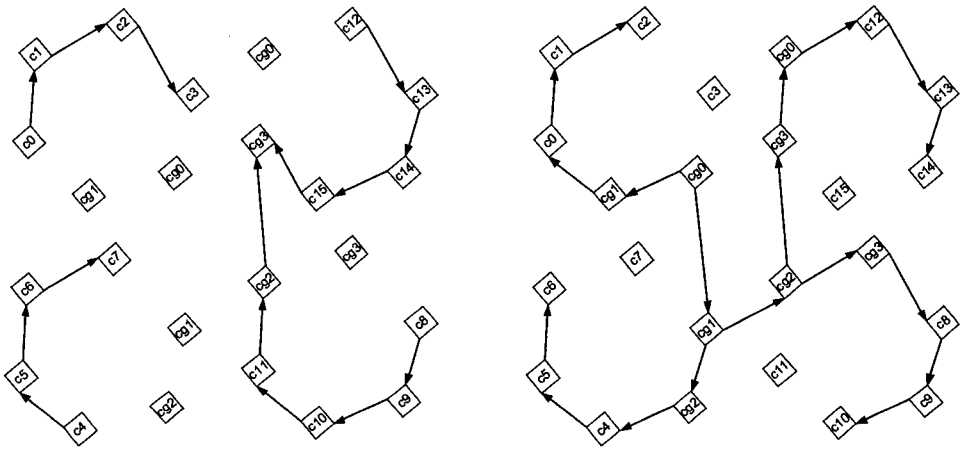


그림 6 채널을 표시한 Torus Ring 구조

의 지역 링을 연결하기 때문이다. 이들을 구분하기 위해 지역 링 번호를 콜론(:) 앞에 나타내어 표시하기로 한다. 예를 들어 높음 채널의 3번 지역 링의 0번 전역 정점은 (b) 3:cg0와 같이 표기한다.

먼저 패킷의 목적지 지역 링이 출발지 지역 링과 동일한 경우를 살펴본다. 이 경우 패킷은 목적지 노드의 번호에 따라 낮음 채널을 택하거나 높음 채널을 택하는데, 목적지 노드 번호가 작아서 낮음 채널을 택한 경우에는 지역 링 내에서 가장 큰 번호의 지역 정점까지 낮음 채널로 라우팅 되고, 그 이후는 전역 링 스위치이다. 따라서 높음 채널로 변경되어 라우팅 된다. 결국, 낮음



(a) 낮음 채널 (b) 높음 채널

그림 7 채널 종속성 그래프

채널의 그래프에서는 지역 링 내에서 두 번째로 작은 번호의 지역 정점에서부터, 가장 큰 번호의 지역 정점까지 화살표를 표시하게 된다(예: (a) c1→c2→c3). 그 후에는 높음 채널로 변경되므로, 마지막 지역 정점에서 해당 링의 높음 채널의 전역 정점 중 그 번호가 작은 정점으로 화살표를 표시하게 된다. 예컨대 (a) c3→(b) 0:cg0의 경우이다. 이후 큰 번호의 전역 정점, 첫 번째 번호의 정점까지 화살표가 연결되게 된다((b) 0:cg0→ 0:cg1→c0).

목적지 노드 번호가 커서 처음부터 높음 채널을 택한 경우에는 도착할 때까지 높음 채널을 계속 사용하게 된다. 높음 채널의 그래프에서는 지역 링 내에서 가장 작은 번호의 지역 정점에서부터 두 번째로 큰 번호의 지역 정점에서까지 화살표를 표시한다((b) c0→c1→c2).

한편 패킷의 목적지 지역 링이 출발지 지역 링과 다른 경우를 살펴보면, 먼저 출발지 노드에서는 전술한 바와 같이 전역 링 스위치를 가장 작은 노드 번호로 간주하여 항상 낮음 채널을 사용한다. 0번 지역 링의 c0에서 나가는(outgoing) 화살표가 있는 것도 이것 때문이다. 낮음 채널의 다른 지역 링들도 가장 작은 번호의 지역 정점에서 다음 번호의 지역 정점으로 나가는 화살표가 존재한다. 전역 스위치에 도달하면, 목적지 전역 링의 번호와 현재의 전역 링의 번호에 따라 높음 채널과 낮음 채널을 선택하여 라우팅 한다. 이에 따라 전역 링 정점에 화살표가 그림 7과 같은 방법으로 표시된다. 인접한 역방향으로의 통신은 해당 스위치를 통해 곧바로 이동하기 때문에 전역 링을 이동할 필요가 없다. 이상을 바탕으로 제안된 Torus Ring에서의 라우팅 방식에 대한 예시를 들면 다음과 같다.

1. 지역링 내부

1) 출발지 0 → 도착지 3

(b)c0 → (b)c1 → (b)c2

2) 출발지 1 → 도착지 0

(a)c1 → (a)c2 → (a)c3 → (b)0:cg0 → (b)0:cg1 → (b)c0

2. 인접 지역링

1) 출발지 5 → 도착지 10

(a)c5 → (a)c6 → (a)c7 → (b)1:cg1 → (b)2:cg2 → (b)2:cg3 → (b)c8 → (b)c9

2) 출발지 5 → 도착지 0

(a)c5 → (a)c6 → (a)c7 → (b)0:cg1

3. 비인접 지역링

1) 출발지 0 → 도착지 10

(a)c0 → (a)c1 → (a)c2 → (a)c3 → (b)0:cg0 → (b)1:cg1 → (b)2:cg2 → (b)2:cg3 → (b)c8 → (b)c9

2) 출발지 10 → 도착지 0

(a)c10 → (a)c11 → (a)2:cg2 → (a)3:cg3 → (b)0:cg0 → (b)0:cg1

표 1에 그래프에 표시되지 않은 경로와 몇몇 혼동이 쉬운 전역 정점의 라우팅 경로를 정리하였다.

표 1 채널 종속성 그래프의 경로

출발지 지역 링과 도착지 지역 링이 동일한 경우		
지역 링 0, 1, 2, 3		(a) c3→ (b) 0:cg0, (a) c7→(b)1:cg1 (a) c11→(b)2:cg2, (a)c15→(b)3:cg3
출발지 지역 링과 도착지 지역 링이 다른 경우		
지역 링 0	역	(a) c3→(b) 3:cg0
	순	(a) c3→ (b) 0:cg0, (b) 0:cg0→(b) 1:cg1
지역 링 1	역	(a) c7→(b) 0:cg1
	순	(a) c7→(b) 1:cg1, (b) 1:cg1→(b) 2:cg2
지역 링 2	역	(a) c11→(b) 1:cg2
	순	(a) c11→(b) 2:cg2, (b) 2:cg2→(b) 3:cg3
지역 링 3	역	(a) c15→(b) 2:cg3
	순	(a) c15→(b) 3:cg3, (b) 3:cg3→(b) 0:cg0

3.4 Torus Ring과 타 연결망과의 차이점

Torus Ring의 스위치 구현은 계층 링의 구성 요소와 동일한 구성 요소를 사용하여 구현되었다. 다만 각각의 구성 요소를 연결하는 방법이 계층 링과 다를 뿐이다. 표 2에서 볼 수 있듯 Torus Ring은 계층 링과 그 복잡도가 동일하다(지역 링 스위치는 입력 1과 출력 1, 전역 링 스위치는 입력 2와 출력 2).

표 2는 각 연결망의 특성을 비교해 나타낸 도표이다. Mesh를 제외한 연결망은 모두 단방향 연결망으로서 이 경우 Degree는 Out-Degree를 나타내었다.

4. 시스템 성능 분석

본 논문에서의 모의 실험은 Execution-driven 시뮬레이터인 RSIM[15]으로 진행되었다. RSIM은 명령어 수준의 병렬성(Instruction-Level Parallelism)을 적극적으

표 2 연결망의 특성 비교

	Mesh ($k\times k$)	링	계층 링(m+n)	Torus Ring
Degree	4	1	1,2	1,2
지름 (Diameter)	$2(k-1)$	$N-1$	$2n+m-1$	$2n+2$
링크 수	$2N-2k$	N	$m(n+1)$	$m(n+2)$
단면 너비 (Bisection Width)	k	2	2	2

로 이용할 수 있는 공유 메모리 다중 프로세서 시스템을 시뮬레이션 한다. 또한 효율적인 메모리 시스템, 다중 프로세서 일관성 프로토콜, 다중 프로세서 연결망 등을 모든 자원에서의 병목 현상들까지 포함해 모델링하고 있다.

4.1 모의 실험 환경

모의 실험에 사용된 인자는 표 3과 같다. RSIM은 파이프라인 되는 웜홀(Wormhole) 라우팅 네트워크를 모델링 한다. 웜홀 라우팅 네트워크에서의 최소 전송 단위인 flit의 크기는 8 바이트로 설정하였고 이는 또한 상호 연결망의 연결 링크 폭도 의미하게 된다.

연결망에 존재하는 스위치들의 버퍼 크기는 64개의 flit이 들어갈 수 있는 공간으로 설정하였다. 이 인자는 스위치 내의 각각의 방향에 해당하는 모든 버퍼들에 적용된다.

표 3 모의 실험 인자

인자	값
노드 수	16~128
프로세서 속도	1 GHz
사이클 당 명령어 반입	4
Active list 크기	64
사이클 당 명령어 완료	4
연산 유닛	2 ALU, 2 FPU
연산 지연 시간	Int 1, Float 3
캐시 라인 크기	64 bytes
L1 캐시 크기	16 KB, direct-mapped
L1 접근 지연 시간	1 cycle
L2 캐시 크기	64KB, 4-way
L2 접근 지연 시간 (태그, 데이타)	3, 5 cycles
메모리 접근 지연 시간	18 cycles
연결망 속도	150 MHz
Flit 크기 (= 링크 폭)	8 bytes
Flit 지연 시간	4 cycles
연결망 스위치 버퍼 크기	64 flits

4.2 벤치마크 프로그램

모의 실험에 사용된 벤치마크 프로그램은 RSIM에서 수행할 수 있도록 컴파일 된 SPLASH[16]와 SPLASH-2 [17]에서 추출한 5개의 프로그램으로, 각각의 특징은 아래와 같다.

FFT는 여러 프로세서들 간의 통신을 최소화하도록 최적화된 Fast Fourier Transform 연산을 수행하는 프로그램이다. 노드 간의 통신은 세 번의 행렬 전치 과정에서 발생하는데, 모든 프로세서가 상호 통신하는 All-to-All 형식의 통신이 발생된다. LU는 조밀 행렬을 하위와 상위의 삼각 행렬의 곱으로 분해하는 프로그램이다. 노드 간의 통신을 줄이기 위해 2차원 산포 분해(Scatter Decomposition)를 사용해 블록의 소유권이 할당되어 각 블록은 그 블록을 소유하고 있는 프로세서가 업데이트하게 한다. 블록의 크기는 캐시 적중 실패율을 낮추도록 적당히 커야 하며, 또 노드간 부하의 분배를 위해서는 적절히 작아야 한다. 이 논문에서는 8의 블록 사이즈를 사용했다. MP3D는 분자의 궤도를 시뮬레이션 하는 프로그램으로 분자들이 안정화 상태에 도달한 이후의 통계 분석을 통해 흐름의 영역을 예측해 낸다. 수행 시 몇 단계(time-step)를 수행할 것인지 인자로 줄 수 있는데, 각 단계마다 initialize, move, add, reservoir-move, reservoir-collide의 과정을 거친다. 작업의 분배는 분자의 단위로 이루어져서 특정 분자는 항상 한 프로세서에 의해서 이동된다. 하지만 이 분자의 위치 배정은 실제 공간에서의 위치와는 관련이 없이 이루어져 프로세서 지역성이 상당히 떨어진다. 이 논문에서는 LOCKING 옵션을 사용해 컴파일 된 버전을 사용하고, 2 단계를 수행한다. Radix는 병렬 기수 정렬을 수행하는 프로그램으로 각각의 프로세서가 전역 히스토그램을 이용해 새로운 배열 키를 배치하는 과정에서 모든 프로세서의 상호 통신(All-to-All)을 발생시킨다. 이 배치 과정은 본래 쓰기 연산이므로, 키는 주로 쓰기에 의해서 교환된다. Water는 액체 상태의 물 분자 간의 힘과 포텐셜을 계산하는 프로그램이다. 사용자가 지정한 만큼의 단계를 수행하는데, 각 단계는 육면체 내의 물 분자의 움직임을 계산하기 위한 뉴튼 방정식을 설정하고 풀이하는 과정을 포함한다. 각 프로세서는 할당 받은 분자들과, 데이타 배열 내에서 그 분자들과 인접한 다음 $n/2$ 분자들의 상호 작용을 계산한다. 따라서 각 프로세서는 전체 프로세서의 절반과 통신을 하게 된다.

각 벤치마크 프로그램의 문제 크기(Problem Size)는 표 4와 같다.

표 4 벤치마크 프로그램의 문제 크기

벤치마크	문제 크기
FFT	216 complex doubles
LU	256x256 matrix
MP3	50000 molecules
Radix	512K integers
Water	512 molecules

4.3 모의 실험 결과 및 분석

본 실험에서 사용된 성능 평가의 기준은 각 벤치마크 프로그램의 응답 지연 시간과 프로그램 수행시간이다. 또한 Torus Ring과 계층 링은 같은 수의 노드를 가진 시스템 일지라도 구성하는 방법이 다양한데 이에 따른

성능 변화도 살펴본다. 그리고 이를 통해 찾은 가장 우수한 성능의 Torus Ring 구성을 다른 연결망들의 성능과 상호 비교해 본다.

4.3.1 상호 연결망 구성별 응답 지연 시간

다음 그림 8에서 그림 11은 계층 링(hring)과 Torus Ring(tring)의 구성 별 응답 지연 시간(Latency)을 보여주고 있다. 응답 지연 시간은 L2 캐시(Level 2 Cache)에서 적중 실패가 발생한 시점부터 요청된 블록이 L2 캐시에 도달할 때까지의 걸린 시간을 프로세서 클럭 수로 나타낸 것을 의미하는 것으로, 그림에서는 각 요청에 대한 응답 지연 시간의 전체적인 평균을 보여주고 있다. 이 응답 지연 시간은 상호 연결망에서의 지연 시간뿐만 아니라 노드 내에서 프로세서와 연결되어 있는 버스, 디렉터리, 메모리 등에서의 지연 시간까지도 함께 포함하게 된다.

각각의 그림에서 볼 수 있듯이 대부분의 구성에서 Torus Ring이 계층 링보다 응답 지연 시간이 짧음을 알 수 있다. 앞서 분석하였듯이 Torus Ring의 형태는 각 벤치마크 프로그램들의 연결망 내에서의 홉 수를 감소시켰고, 이에 따라 상호 연결망의 요청과 응답의 지연 시간이 최대 19%까지 감소하였다.

한편 MP3D는 64노드 이상에서의 응답 지연 시간이 다른 프로그램에 비해서 현저히 큰데, 이것은 MP3D에서 사용되는 락(Lock) 때문이다(그림자 표시). 그림 12는 8x8 형태로 구성된 64노드의 계층 링에서 각 벤치마크 프로그램의 수행 시간을 여러 구성 요소로 분리하여 보여주는 그래프인데, MP3D의 수행 시간 중에서 락에 할애된 시간이 매우 큰 비중을 차지하고 있는 것을 볼 수 있다. RSIM에서 제공되는 락 알고리즘은 Test-and-Test-and-Set 방법으로, 여러 프로세서가 동시에 하나의 락을 얻기 위해 Test-and-Set과 Test를 반복적으로 수행하게 되면 UPGRADE 혹은 READ_OWN과 READ_SH가 하나의 캐시 라인에 여럿 요청되게 되어 응답이 되돌아 오는데 까지 많은 시간이 걸리게 되는 것이다.

4.3.2 상호 연결망 구성 별 수행 시간

이 절에서는 계층 링과 Torus Ring의 구성별 수행 시간에 대해 설명한다. 또한, 계층 링과 Torus Ring을 구성하는 방법에 따른 성능의 변화를 비교해 살펴보도록 하겠다. 계층을 구성하는 방법이 중요한 이유는 벤치마크 프로그램의 지역성(Locality)에 따라 동일한 지역 링 내에서 해결될 수 있는 트랜잭션의 수가 달라질 수 있어서 구성 방법에 따른 성능의 편차가 생길 수 있기

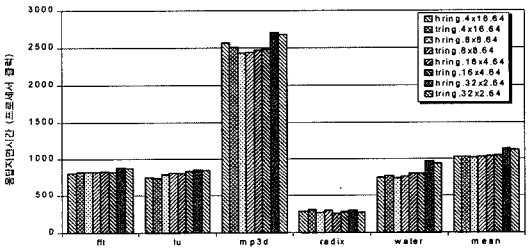


그림 10 계층 링과 Torus Ring의 응답 지연 시간(64노드)

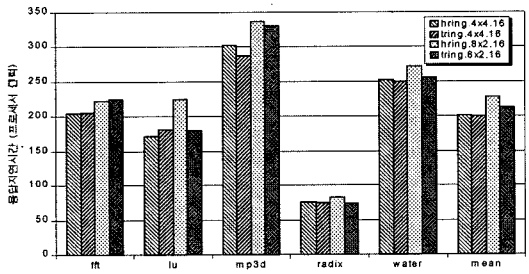


그림 8 계층 링과 Torus Ring의 응답 지연 시간(16노드)

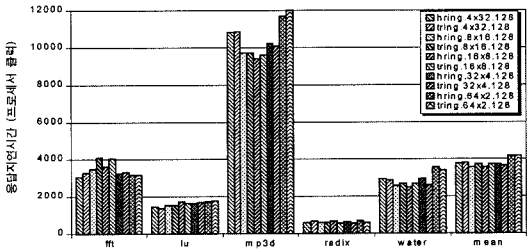


그림 11 계층 링과 Torus Ring의 응답 지연 시간(128노드)

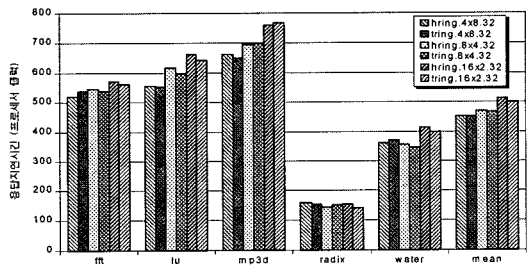


그림 9 계층 링과 Torus Ring의 응답 지연 시간(32노드)

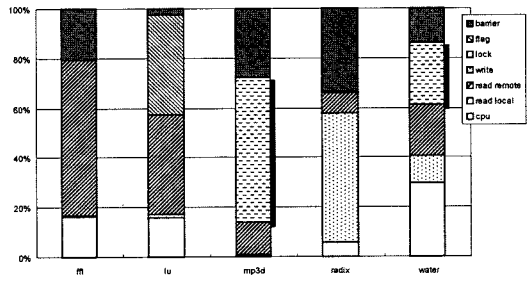


그림 12 계층 링(8x8)의 실행 시간 구성

때문이다.

물론 계층 링이나 Torus Ring에서, 지역 링 내에서 해결될 수 있는 트랜잭션의 비율(링 지역성)이 크다고 해서 무조건 성능이 좋아진다고는 할 수 없다. 이것은 본래 단일 링의 한계가 확장성의 부족에서 기인하는 것과 마찬가지인 것으로, 지역 링의 크기가 지나치게 커지면 링 지역성은 많이 증가하게 되지만 반대로 지역 링 내에서의 평균 홉 수 또한 증가하게 되기 때문이다.

그림 13에서 그림 16은 노드 수 별로 지역 링 개수에 따른 계층 링과 Torus Ring의 수행 시간을 보여주고 있다. 여기에서 수행 시간은 각각의 벤치마크 프로그램을 수행시켜서 프로그램이 시작될 때부터 끝날 때까지 걸린 프로세서 클럭 수를, 단일 링에서 해당 프로그램을 수행했을 때의 프로세서 클럭 수에 정규화시켜서 나타낸 것을 의미한다. 그래프를 보면 동일한 구성 내에서 계층 링보다 Torus Ring이 대부분의 경우에 더 좋은 성능을 보이고 있다. 성능이 향상된 경우 동일 구성의 계층 링 대비 최대 10% 정도의 이익을 볼 수 있음을 알 수 있다.

성능의 향상은 대체적으로 노드 수에 상관없이 고르지만 적은 수의 노드에서 성능 향상이 조금 더 큰 편이다. 또한 특정 노드 수의 모든 구성에서 Torus Ring이 계층 링에 비해 성능이 좋은 것은 아니다. 일부 Torus Ring은 동일한 구성의 계층 링에서보다 성능이 나쁜 경우가 있다. 하지만 이 경우에도 해당 노드 수에서 가장 성능이 좋은 경우의 구성은 Torus Ring인데, 32노드의 MP3D(tring.4x8.32가 32노드 내에서 가장 성능이 좋음), 64노드의 LU(tring.4x16.64)와 Water(tring.8x8.64) 등이 그 경우이다. 하지만 Torus Ring이 모든 구성에서 성능이 나쁜 경우도 있어서, 64노드의 Radix, 128의 MP3D와 Radix 등이 그러하다.

이러한 성능의 유, 불리는 앞에서 Torus Ring의 특징에서 언급하였듯이 Torus Ring이 계층 링보다 유리한 경우인, 역방향의 인접한 링에 대한 접근이 충분히 많은지에 따라 결정된다. 즉 각 벤치마크의 통신 패턴에 따른다고 볼 수 있는데, 역방향의 인접한 링에 대한 접근은 직관적으로 생각해 보았을 때 $\frac{1}{m}$의 비율이다. 실제로 프로그램을 수행시켰을 때도 이 정도의 비율에 근접한 결과를 볼 수 있다. 표 5는 64노드를 기준으로 보았을 때 벤치마크 프로그램의 수행 중에 상호 연결망에 나타난 모든 요청과 응답에 대해서 패킷의 목적지 노드가 역방향 인접 링에 있을 경우의 비율을 나타난다.

이와 같은 인접한 링에 대한 통신 비율은 제안된 Torus Ring의 성능 향상과 밀접한 관련을 가지며, 각각의 벤치마크 프로그램들이 많은 지역성을 내포하고 있을 경우 인접한 지역링에 대한 통신의 비율이 커지게 된다. 표 5에서도 나타나듯이, 4x16의 Radix, 8x8의 LU와 Radix, 16x4의 LU 등이 상대적으로 나머지 벤치마크 프로그램들에 비해서 비율이 낮음을 볼 수 있다. 그리고 이들은 계층 링과의 수행 시간 비교에서 조금씩 그 성능이 떨어짐을 알 수 있다. 이런 현상은 다른 노드 구성에서도 마찬가지로 발견할 수 있다.

그림 17은 32 노드의 경우 나타난 계층 링과 Torus

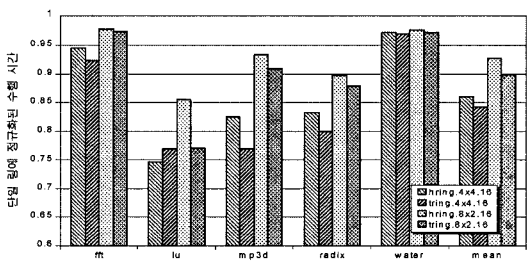


그림 13 계층 링과 Torus Ring의 수행 시간(16노드)

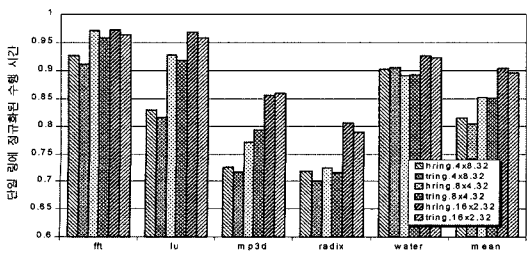


그림 14 계층 링과 Torus Ring의 수행 시간(32노드)

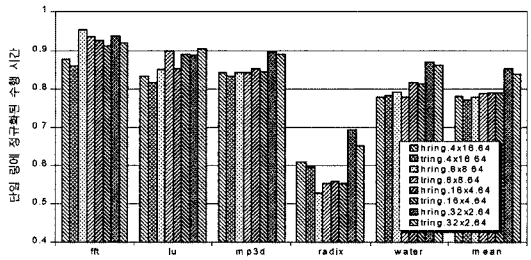


그림 15 계층 링과 Torus Ring의 수행 시간(64노드)

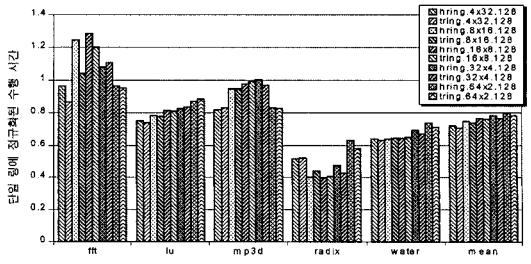


그림 16 계층 링과 Torus Ring의 수행 시간(128노드)

표 5 Torus Ring이 계층 링보다 유리한 경우의 비율, 64노드 (단위: %)

Tring 4x16 64	fft	24.83	Tring 8x8 64	fft	12.38	Tring 16x4 64	fft	6.2
	lu	25.15		lu	9.12		lu	4.12
	mp3d	24.34		mp3d	12.63		mp3d	6.3
	radix	21.7		radix	11.5		radix	7.7
	water	24.67		water	14.49		water	7.29

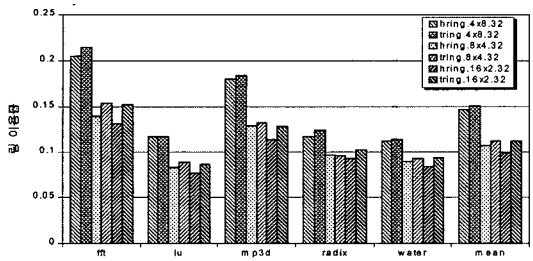


그림 17 계층 링과 Torus Ring의 이용률(32노드)

Ring의 링 이용률을 보여준다. 링 이용률(Utilization)은 상호 연결망에 존재하는 모든 스위치들에 대해서, 내부의 버퍼들이 사용된 시간을 전체 수행 시간으로 나눈 값의 평균을 나타낸다. 이와 같은 링 이용률은 상호 연결망 전체적으로 스위치 내부의 버퍼가 얼마나 오랜 시간 동안 이용되고 있는지, 또한 상호 연결망 내의 패킷들이 얼마나 자주 정체되는지를 나타내는 척도라고 할 수 있다. 앞선 모의 실험에서, 수행 시간에서 우수한 결과를 보이는 벤치마크 프로그램 중에서도 간혹 응답 지연 시간의 평균에서 더 많은 프로세서 클럭을 요구하는 경우를 발견할 수 있는데 이에 대한 원인은 다음과 같다. 그림 17을 보면 32 노드를 기준으로 링 이용률이 전반적으로 Torus Ring이 높다는 것을 알 수 있다. 이는 링의 부하가 심하게 걸리지 않은 상태에서 링의 이용률을 높여 동시에 더 많은 요청을 처리 했기 때문이다. 이 경우 각각의 요청에 대한 응답 지연 시간은 더 오래 걸릴 수 있지만, 이들의 모든 요구가 상호 중첩된(overlap) 형태의 결과로 나타나는 실행 시간 측면에서는 유리한 결과를 보이는 원인으로 해석 할 수 있다.

그림 18과 그림 19는 Torus Ring이 계층 링과 비교했을 때 성능의 우위가 어디에서 무엇으로부터 기인하는지를 시각적으로 보여주기 위해 FFT의 홉 수 분포를 나타냈다. 그림에서 X축은 홉 수를 나타내고 Y축은 해당 홉 수가 연결망에 나타난 빈도를 의미한다. 전체적인 그래프의 형태는 계층 링과 Torus Ring이 비슷함을 볼 수 있다. 하지만 계층 링의 경우 지역 링 내의 노드 수인 n의 값에 해당하는 홉 수의 빈도가 높고, Torus Ring의 경우 n+1값에 해당하는 홉 수의 빈도가 특히 높다. 이것은 벤치마크의 지역성에 기인하는 것으로, 상호 연결망에 나타난 요청이나 응답의 목적지 노드가 역

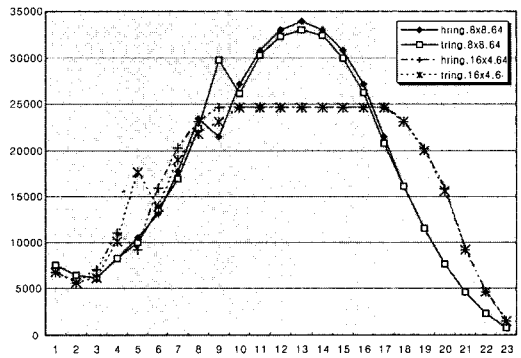


그림 18 FFT의 홉 분포, 64노드

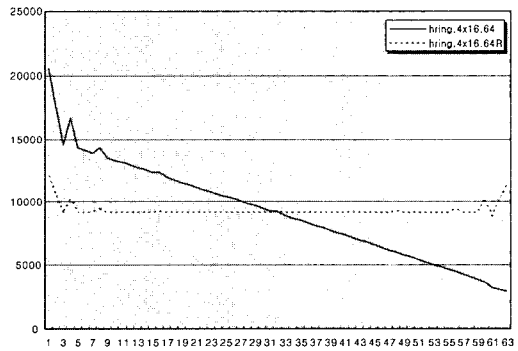


그림 19 FFT의 지역성

방향의 인접한 노드인 경우 지역 링을 한 바퀴 돌아서 도달하므로, 이는 계층 링과 Torus Ring에서 각각 n홉과 n+1 홉이 걸리기 때문이다. Torus Ring에서의 n+1 홉의 빈도는 계층 링의 n 홉에서의 빈도보다 상대적으로 많은데, 이것은 n+1 홉 보다 큰 수의 홉에서 빈도가 줄어들어 보다 n+1 홉의 빈도가 늘어났기 때문이다. 그림 18의 경우 8×8 구성에서는 10홉에서 18홉의 빈도가 줄고 9홉의 빈도가 증가했고, 16×4 구성에서는 6홉에서 10홉의 빈도가 감소하고 5홉의 빈도가 증가했다. 이러한 경향은 Torus Ring의 전체 평균 홉 수가 감소하는 결과를 가져온다.

그림 19는 4×16으로 구성된 64노드의 계층 링에서 추출한 상호 연결망의 요청과 응답을 이용하여 그려진 그래프이다. 실선은 특정 토폴로지와는 무관하게 단순히 출발지 노드 번호와 목적지 노드 번호의 차이를 나타낸

것이고, 점선은 각 요청이나 응답이 단일 링에서 나타났을 때를 가정한 경우의 홉 수를 나타낸 것이다. 따라서 실선은 연결망의 형태와 무관한 FFT의 지역성을 보여준다고 할 수 있다. 곧 그림 18의 계층 링의 n홉과 Torus Ring의 n+1홉의 돌출은 그림 18의 점선이 양측에서 모두 높은 빈도를 보인다는 것에서 기인한다.

한편 정해진 노드 수 내에서 어떤 계층 구성 방법이 효율적인가에 대해서는, Radix를 제외하고는 대체적으로 지역 링의 개수가 적은 쪽의 구성에서 치우쳐서 성능이 좋고, 지역 링의 수가 늘어날수록 성능이 떨어지는 경향을 보여주고 있다. 극단적으로 FFT의 경우에는 지역 링의 개수를 최대한 줄이고 각 지역 링에 많은 노드를 배치한 경우(링 지역성을 가장 높인 경우)가 성능이 가장 좋은 것으로 나타난다. 그 외의 벤치마크 프로그램들은 대체적으로 지역 링의 개수가 4개 정도일 때 가장 성능이 좋은 것으로 나타났다. Torus Ring은 이러한 구성일 때 대개 계층 링보다 수행 시간이 적음으로, 동일 구성 내에서 계층 링보다 성능이 좋지 않은 경우라 할지라도, 동일 노드 수 내에서 가장 성능이 좋은 구성은 Torus Ring인 경우가 많은 것이다.

Radix는 지름이 가장 짧은 경우의 계층 구성(지역 링의 개수가 전체 노드 수의 제곱근에 가까운 경우)에서 가장 성능이 좋은 경향을 보인다. Radix는 연결망의 홉 수의 분포와 가장 일치하는 수행 시간을 보여주는 프로그램이라고 할 수 있다.

5. 결 론

본 논문에서는 계층 링과 형태는 비슷하지만 지역 링을 연결하는 방법에 있어서 차이를 둔 Torus Ring이라는 상호 연결망 형태(Topology)를 제시하며, 이 형태에서의 라우팅 방법이 교착 상태가 존재하지 않음을 보였다. 또한 Torus Ring에서의 응답 지연 시간이 줄어듦으로 해서 프로세서의 전체 수행 시간을 감소시킬 수 있음을 실험을 통해서 확인하였다.

Torus Ring은 지역 링의 연결에 따로 전역 링을 두어 링크를 구성하는 계층 링과는 달리, 지역 링의 링크를 곧바로 인접한 두 지역 링과의 연결에 활용하는 상호 연결망의 형태이다. 이 연결망에서는 역방향의 인접한 지역 링에 접근하는 경우의 홉 수가 계층 링에 비해서 m+1 만큼 감소한다. Torus Ring이 지역 링보다 홉 수가 적을 수 있는 경우의 확률은 $\frac{1}{m}$ 정도이고 실제로 벤치마크 프로그램을 수행하였을 때 이에 따른 성능 향상을 관찰할 수 있었다. 다만 지역 링과 전역 링의 링크가 공유되는 측면 때문에 전체적인 연결망의 이용률이 증가할 수 있지만, 실험 결과에서 보여주듯이 전역 링의 링크가 병목을 초래할 정도의 급격한 이용률의 증가는 없는 것으로 판단된다.

향후 성능 향상을 보이지 않는 일부 벤치마크 프로그램의 지역성을 잘 활용할 수 있도록 Torus Ring 연결망 내에 노드를 효율적으로 배치하는 방법이나, 전역 링 스위치의 이용률을 낮출 수 있는 방안에 대한 연구가 필요할 것으로 보인다.

참 고 문 헌

[1] David E. Culler and Jaswinder Pal Singh with Anoop Gupta, "Parallel Computer Architecture : A Hardware/Software Approach," Morgan Kaufmann Publishers, Inc, 1998.

[2] Sung Woo Chung, Seong Tae Jhang and Chu Shik Jhon, "PANDA: ring-based multiprocessor system using new snooping protocol," Proceedings of International Conference on Parallel and Distributed Systems, pp. 10-17, Dec 1998.

[3] Byoung Soon Jang, Sung Woo Chung, Seong Tae Jhang and Chu Shik Jhon, "Efficient Schemes to Scale the Interconnection Network Bandwidth in a Ring-based Multiprocessor System," SAC-2001 (16th ACM Symposium on Applied Computing), Las Vegas, United States, pp.510-516, March 2001.

[4] G. Ravindran and M. Stumm, "A performance comparison of hierarchical ring- and mesh-connected multiprocessor networks," Third International Symposium on High-Performance Computer Architecture, pp. 58-69, Feb 1997.

[5] G. Ravindran and M. Stumm, "On topology and bisection bandwidth of hierarchical-ring networks for shared-memory multiprocessors," 5th International Conference On High Performance Computing, pp. 262-269, Dec 1998.

[6] Z.G. Vranesic, M. Stumm, D.M. Lewis and R. White, "Hector: a hierarchically structured shared-memory multiprocessor," Computer, Vol.24, Iss.1, pp. 72-79, Jan. 1991.

[7] R. Grindley, T. Abdelrahman, S. Brown, S. Caranci, D. DeVries, B. Gamsa, A. Grbic, M. Gusat, R. Ho, O. Krieger, G. Lemieux, K. Loveless, N. Manjikian, P. McHardy, S. Srbljic, M. Stumm, Z. Vranesic and Z. Zilic, "The NUMAchine multiprocessor," Proceedings of International Conference on Parallel Processing, pp. 487-496, 2000.

[8] Dongho Yoo, Inbum Jung, Seung Ryoul Maeng and Hyunglae Roh, "Multistage ring network: a new multiple ring network for large scale multiprocessors," Proceedings of International Workshops on Parallel Processing, pp. 290-294, 1999.

[9] Guihai Chen and F.C.M. Lau, "Shuffle-Ring: overcoming the increasing degree of hypercube,"

Proceedings of Second International Symposium on High-Performance Computer Architecture, pp. 130-138, Feb 1996.

[10] "Recursive cube of rings: a new topology for interconnection networks," Y. Sun, P.Y.S. Cheung, X. Lin, IEEE Transactions on Parallel and Distributed Systems, Vol.11, Iss.3, pp. 275-286, Mar 2000.

[11] 성현중, 김형호, 장성태, 전주식, "스누핑 프로토콜을 사용하는 NUMA 시스템의 계층적 링 구조로의 확장", 정보과학회 논문지(A), pp. 1305~1317, Vol. 26, No. 11, Nov. 1999.

[12] 경진미, 김인석, 김봉준, 장성태 "리피터 노드를 장착한 이중 링 CC-NUMA 시스템", 한국정보과학회:학술대회지, 2002, 10 v.2002, n.한국정보과학회 02 가을 학술발표논문집(1), pp.697-699

[13] W.J. Dally, "Virtual-Channel Flow Control," IEEE Trans. Parallel and Dist. Sys., vol. 3, no. 2, pp. 194 - 205, Mar. 1992.

[14] G. Ravindran, "Performance Issues in the Design of Hierarchical-ring and Direct Networks for Sharedmemory Multiprocessors," Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Toronto, Jan 1998.

[15] Vijai S. Pai, Parthasarathy Ranganathan and Sarita V. Adve, "RSIM Reference Manual," Dept. of Electrical and Computer Engineering, Rice University, Technical Report 9705, 1997.

[16] J. P. Singh, W. D. Weber, and A. Gupta, "Splash: Stanford parallel applications for shared memory," Tech. Rep. CSL-TR-91-469, Stanford University, 1991.

[17] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," Proceedings of the 22nd International Symposium on Computer Architecture, 1995.

곽 종 욱

1998년 2월 경북대학교 컴퓨터공학과 공학사. 2001년 8월 서울대학교 대학원 컴퓨터공학과 석사. 2002년 3월~현재 서울대학교 대학원 전기컴퓨터공학부 박사과정. 관심분야는 컴퓨터 구조, 고성능 병렬처리시스템. 저전력 내장형 시스템

반 형 진

2002년 8월 고려대학교 경제학과 학사 2004년 8월 서울대학교 전기컴퓨터공학과 석사. 2004년 9월~현재 삼성전자. 관심분야는 고성능 병렬 프로세서 구조, 프로세서 연결망 구조

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 전기컴퓨터공학부 교수. 1994년~1999년 서울대학교 컴퓨터신기술공동연구소 소장. 관심분야는 컴퓨터 구조, 병렬처리, VLSI/CAD