SCI 기반 PC 클러스터링을 위한 CC-NUMA 프로토타입 카드의 설계와 성능

(Design and Performance of a CC-NUMA Prototype Card for SCI-Based PC Clustering)

오 수 철 † 정 상 화 ††
(Soo-Cheol Oh) (Sang-Hwa Chung)

요 약 고성능 PC 클러스터 시스템을 구축하기 위해서는 네트워크 접근 시간을 최소화하는 것이 중요하다. SCI 기반 PC 클러스터 시스템에서는 각 노드에 네트워크 캐쉬를 유지함으로써 네트워크 접근 시간을 줄이는 것이 가능하다. 본 논문에서는 SCI 기반 PC 클러스터 시스템을 위해서 네트워크 캐쉬를 활용하는 CC-NUMA 카드를 제안한다. CC-NUMA 카드는 각 노드의 PCI 슬롯(slot)에 plug-in되는 형태이며, 공유메모리, 네트워크 캐쉬, 네트워크 제어 모듈을 포함한다. 네트워크 캐쉬는 클러스터 노드의 PCI 버스상에 존재하는 공유메모리를 캐쉬하며 공유메모리와 네트워크 캐쉬 사이의 일관성은 IEEE SCI 표준에 의해 유지된다. 본 연구에서는 SCI 기반 PC 클러스터 시스템의 성능을 측정하기 위하여 CC-NUMA 프로토타입 카드를 개발하였으며, 이를 기반으로 하여 클러스터 시스템을 구축하였다. 실험결과, CC-NUMA 카드를 장착한 클러스터 시스템이 네트워크 캐쉬를 활용하지 않는 NUMA 기반 클러스터 시스템에 비해서 우수한 성능을 보임을 알 수 있었다.

키워드 : 네트워크 캐쉬, CC-NUMA, SCI, PC 클러스터, PCI

Abstract It is extremely important to minimize network access time in constructing a high-performance PC cluster system. For an SCI-based PC cluster, it is possible to reduce the network access time by maintaining network cache in each cluster node. This paper presents a CC-NUMA card that utilizes network cache for SCI-based PC clustering. The CC-NUMA card is directly plugged into the PCI slot of each node, and contains shared memory, network cache, and interconnection modules. The network cache is maintained for the shared memory on the PCI bus of cluster nodes. The coherency mechanism between the network cache and the shared memory is based on the IEEE SCI standard. A CC-NUMA prototype card is developed to evaluate the performance of the system. According to the experiments, the cluster system with the CC-NUMA card showed considerable improvements compared with an SCI-based cluster without network cache.

Key words : Network cache, CC-NUMA, SCI, PC cluster, PCI

1. 서 론

최근에 사회의 모든 분야에서 이루어지는 정보화와 인터넷의 급속한 보급으로 웹서버, 전자상거래서버, VOD서버 등을 포함한 여러 분야에서 고성능 서버에 대한 수요가 증가하고 있다. 그러나, 고성능 서버는 고가이기 때문에 구입 및 활용에 상당한 어려움을 가지고 있다. 이러한 문제를 해결하기 위한 방안으로서, 고속 마이크로프로세서를 장착한 저가의 PC를 고속 네트워크로 연결하여 하나의 컴퓨팅 시스템으로 사용하려는 클러스터 시스템이 등장하였다. 클러스터 시스템은 캐비넷 기반의 중대형 서버에 비해 확장성이 우수하고, 가격 대 성능비가 높다는 장점이 있다.

PC 기반 클러스터가 중대형 컴퓨터에 필적하는 단일 컴퓨팅 시스템으로서의 성공 여부는 PC를 연결하는 네트워크 성능과 지원소프트웨어에 달려 있다. 클러스터

· 본 연구는 한국과학재단 목적기초연구(2002-2-30300-002-3)지원으로 수행되었음.
† 비 회 원 : 부산대학교 컴퓨터공학과
osc@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
shchung@pusan.ac.kr
논문접수 : 2001년 2월 5일
심사완료 : 2001년 9월 12일

시스템을 위한 네트워크로는 메시지 패싱에 기반한 Fast Ethernet, Myrinet 등과 분산공유메모리에 기반한 SCI를 들 수 있다. LAN용으로 개발된 Fast Ethernet은 100 Mbps의 낮은 대역폭과 TCP/IP와 같은 복잡한 프로토콜의 사용으로 시스템에서 필요로 하는 충분한 대역폭을 제공하지 못하고 있다. Myrinet[1]은 160MByte/sec의 최대 대역폭을 가지는 네트워크로, 네트워크 연결시 크로스바 스위치를 사용함으로 네트워크 구성 비용이 높다. 또한, Fast Ethernet과 Myrinet이 사용하는 메시지 패싱 방식은 빈번한 데이타 복사와 OS 시스템 콜로 인한 소프트웨어 오버헤드가 크기 때문에 네트워크의 물리적 성능을 저하시키고 있다. 이러한 소프트웨어 오버헤드를 최소화하기 위해서 메시지 패싱 과정에서 OS의 개입을 배제한 사용자수준 인터페이스가 개발되었다. 대표적인 시스템으로 AM[2], FM[3], U-Net[4]이 있으며, 최근에는 Myrinet을 위한 사용자수준 인터페이스로 GM[5]이 개발되었다.

SCI(Scalable Coherent Interface : ANSI/IEEE standard 1596-1992)는 1μs이하의 낮은 지연시간과 1GByte/sec의 대역폭을 가지는 네트워크이다. SCI는 point-to-point connection에 기반하여 ring 및 switch topology를 지원하며, 최대 64K개의 노드 연결이 가능하다. 또한, DSM 및 캐쉬 일관성 유지 프로토콜을 지원함으로 다양한 형태의 NUMA 및 CC-NUMA 시스템의 제작이 가능하며, 상용 CC-NUMA 서버인 IBM의 NUMA-Q[6], Data General의 Aviion[7]에 의해서 그 성능이 입증되었다.

본 연구에서는 클러스터 시스템의 네트워크로 SCI를 사용하였다. SCI 기반 클러스터 시스템의 경우, 각 노드에 네트워크 캐쉬를 유지함으로서 네트워크 접근 시간을 최소화시킬 수 있다. 그러나, Dolphin[8], LRR-TUM[9], TUC[10]에서 개발한 SCI 네트워크 카드는 CC-NUMA 기능을 지원하지 않는다. 본 논문은 SCI 기반 PC 클러스터 시스템을 위해서 네트워크 캐쉬를 활용하는 CC-NUMA 카드를 제안한다. CC-NUMA 카드는 각 노드의 PCI 슬롯에 plug-in되는 형태이며 공유메모리, 네트워크 캐쉬, 네트워크 제어 모듈로 구성된다. 공유메모리와 네트워크 캐쉬사이의 일관성은 IEEE SCI 표준에 의해서 유지된다. 본 논문에서는 최종 CC-NUMA 카드 구현에 앞서 CC-NUMA 프로토타입 카드를 개발하고 그 성능을 측정하였다.

2. CC-NUMA 기반 PC 클러스터 시스템

본 논문에서 제안하는 CC-NUMA 카드에 기반한 PC 클러스터 시스템의 구조는 <그림 1>과 같다. CC-NUMA 카드는 PC의 PCI 슬롯에 plug-in되는 형태이며 공유메모리 제어 모듈, 공유메모리, 네트워크 캐쉬, 네트워크 제어 모듈 및 SCI 디렉토리로 구성된다. PC는 메인보드상에서 시스템 버스 인터페이스를 제공하지 않음으로 CC-NUMA 카드는 PCI 버스상에 위치한다. 본 시스템에 존재하는 메모리는 지역메모리와 공유메모리로 구성된다. 지역메모리는 각 노드의 시스템 버스에 장착되어 있는 메모리로 지역 CPU의 L1, L2 캐쉬에 의해서 캐쉬된다. 또한 PCI 버스상에 위치한 CC-NUMA 카드는 시스템 버스를 snoop할 수 없으므로 지역 메모리를 다른 노드와 공유할 수 없다. 이를 해결하기 위해서 본 시스템은 PCI 버스에 장착되는 CC-NUMA 카드에 공유 메모리를 위치시킨다. 공유메모리는 원격 노드의 네트워크 캐쉬에 의해서 캐쉬된다. 지역 CPU가 PCI 버스상에 위치한 공유메모리와 네트워크 캐쉬를 접근할 때, L1과 L2 캐쉬는 사용되지 않는다. 이것은 PCI specification 2.1에 정의된 캐쉬 지원 기능이 대부분의 PC chip set에서 지원되지 않기 때문이다.

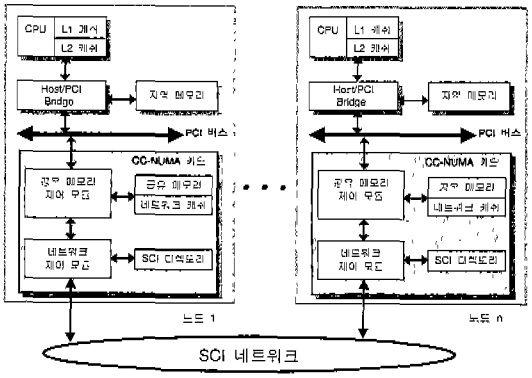


그림 1 CC-NUMA 기반 PC 클러스터 시스템

각 노드에 존재하는 공유메모리들은 하나의 SCI 전역 주소 공간으로 매핑되며 각 노드는 동일한 SCI 주소를 가지고 공유메모리에 접근한다. SCI 디렉토리는 SCI에 기반한 공유리스트 정보를 저장한다. 공유메모리 제어 모듈은 지역 및 원격 CPU에서 전달된 공유메모리에 대한 read/write 트랜잭션을 처리하며, SCI의 캐쉬 일관성 유지 알고리즘을 사용하여 공유메모리와 네트워크 캐쉬사이의 캐쉬 일관성을 유지시키는 역할을 한다. 또한, 캐쉬 일관성 유지작업시 필요한 원격 노드와의 트랜잭션 교환은 네트워크 제어 모듈을 통하여 이루어진다. 네트워크 제어 모듈은 SCI 디렉토리를 관리하며, 공유메모리 제어 모듈 및 원격 노드에서 전송된 read/write/

invalidation 트랜잭션을 처리한다.

기존의 CC-NUMA 시스템인 NUMA-Q, Aviion은 시스템 버스에 장착된 메인 메모리를 공유대상으로 하며, 공유 메모리 제어 모듈이 시스템 버스에 장착되어 있다. 이러한 시스템은 100MHz이상의 고속으로 동작하는 시스템 버스를 기반으로 함으로 전체 시스템이 고성능을 가지는 장점이 있으나, 공유 메모리 제어 모듈의 개발 비용이 높다는 단점이 있고, PC 기반 시스템에서는 구현이 불가능하다. PC 기반 시스템은 공유 메모리 제어 모듈이 PCI 버스상에 위치하여 공유 메모리에 대한 접근 시간이 증가하는 단점은 있으나, 저가의 비용으로 PC 클러스터상에 CC-NUMA 시스템 구현이 가능하여 가격 대 성능비가 기존의 CC-NUMA 시스템에 비하여 높다.

3. CC-NUMA 프로토타입 카드

본 논문에서는 2장에서 제시한 CC-NUMA 카드를 개발하기에 앞서, <그림 2>와 같은 구조의 프로토타입 카드를 개발하였다. 프로토타입 카드는 공유메모리 제어 모듈의 구현에 주력하고자 개발되었으며, 네트워크 제어 모듈은 Dolphin사의 PCI-SCI 카드 D310을 활용하였다. 따라서, 프로토타입 시스템의 개발은 SCI 네트워크의 세부 동작에 신경 쓰지 않고, CC-NUMA 제어 메카니즘 구현에 주력하였다. 공유메모리 제어 모듈은 1개의 FPGA (Xilinx Virtex XCV400-4-HQ240, System Gates : 468K)로 개발하였으며, PCI 인터페이스와 캐쉬 컨트롤러를 포함하고 33MHz로 동작한다. PCI 인터페이스는 PCI specificatioin 2.1 기준에 의해 개발하였으며 지역 CPU 및 PCI-SCI 카드와의 트랜잭션 교환을 담당한다. 캐쉬 컨트롤러는 SCI의 typical set을 구현하였으며, 캐쉬 일관성 유지 작업을 수행한다. 공유메모리와 네트워크 캐쉬는 DRAM에 저장하며, 태그정보는 속도를 위해서 SRAM에 저장된다. 태그정보는 SCI 공유 상태 및 분산 디렉토리 정보를 저장한다. 현재 구현된 시스템은 4Mbyte의 공유메모리와 1Mbyte의 네트워크 캐쉬를 포함한다. 각 노드에 위치하는 공유메모리의 크기는 4GB 주소공간안에서 클러스터 시스템을 구성하는 노드수에 따라 변경 가능함으로, 각 노드의 공유메모리 크기는 수백Mbyte까지 확장 가능하다.

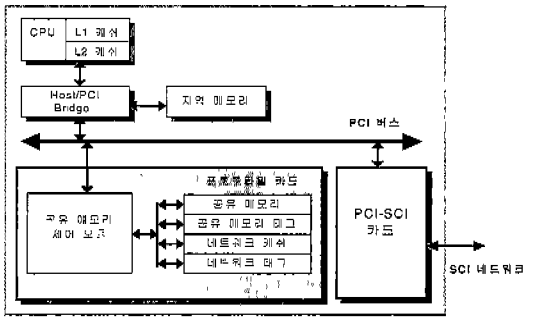
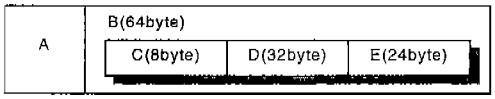


그림 2 CC-NUMA 프로토타입 카드 구조

프로토타입 시스템에서는 PCI delayed 트랜잭션을 사용하여 지역 CPU의 공유메모리 read/write 트랜잭션을 처리한다. PCI read/write 트랜잭션은 PCI specification 2.1에 정의된 시간(16 PCI clocks)안에 정상 종료할 수 없는 경우에 PCI delayed 트랜잭션을 사용하여 트랜잭션을 처리하며, 처리 순서는 다음과 같다. 지역 CPU에서 공유메모리 read/write 트랜잭션이 발생하면 Host/PCI bridge와 PCI 버스를 통하여 프로토타입 카드에 전달된다. 프로토타입 카드는 PCI 버스에서 트랜잭션에 관한 정보를 읽은 후, 요구된 트랜잭션이 완료될 때까지 PCI target ready 신호를 assert하지 않아서 트랜잭션을 retry시킨다. Host/PCI bridge는 프로토타입 카드에서 응답을 받을 때까지 트랜잭션을 계속 retry한다. 트랜잭션이 프로토타입 카드에서 수행 완료되면, 프로토타입 카드는 retry된 트랜잭션에 응답을 하여 트랜잭션을 정상 종료시킨다. 본 연구는 공유메모리 제어 모듈의 동작속도 향상 및 최적화를 수행하고 SRAM 및 DRAM에 대한 접근 속도를 향상시킴으로써, 공유메모리 및 네트워크 캐쉬에 대한 트랜잭션을 PCI delayed 트랜잭션을 사용하지 않고 처리할 수 있도록 시스템 성능을 향상시킬 것이다.

지역 CPU에서 원격 메모리 read/write 트랜잭션이 발생하면 CC-NUMA 카드는 PCI 버스를 통하여 PCI-SCI 카드에 트랜잭션의 전송을 요구한다. 그리고 원격 노드의 PCI-SCI 카드에 도착한 트랜잭션은 PCI 버스를 통하여 CC-NUMA 카드에 전달된다. 이러한 메커니즘을 지원하기 위해서는 PCI-SCI 카드가 지역 메모리가 아닌 CC-NUMA 카드를 대상으로 PCI 트랜잭션을 발생시켜야 하며, 이를 위해서 Dolphin사의 PCI-SCI 카드 드라이버를 수정하였다.



A : PCI-SCI 카드의 SCI 패킷 헤더
C : 프로토타입 카드의 원격 트랜잭션 헤더
E : 더미 데이타
B : PCI-SCI 카드의 SCI 패킷 데이타
D : 프로토타입 카드의 원격 트랜잭션 데이타

그림 3 프로토타입 카드의 SCI 패킷 구조

프로토타입 시스템은 원격 노드와의 트랜잭션 교환시 PCI-SCI 카드에서 지원하는 SCI write 트랜잭션을 사용하며, SCI write 트랜잭션의 64 byte 패킷 데이타에 프로타입 카드가 전송하는 원격 트랜잭션에 관한 헤더 및 데이타를 <그림 3>처럼 포함하여 전송한다. 프로토타입 카드가 PCI-SCI 카드에 SCI 패킷 데이타의 전송을 요구하면, PCI-SCI 카드는 SCI 패킷 헤더를 추가하여 원격노드로 전송한다. SCI 패킷이 원격 노드의 PCI-SCI 카드에 도착하면, PCI-SCI 카드는 SCI 패킷 헤더를 제거하고 PCI 버스를 통하여 SCI 패킷 데이타를 프로토타입 카드로 전송한다. 따라서, 프로토타입 카드가 원격 노드에서 전송된 트랜잭션에 관한 정보를 가지기 위해서는 SCI 패킷 데이타 영역에 프로토타입 카드에서 사용할 수 있는 헤더정보를 포함해야 한다. 또한, 프로토타입 카드의 원격 트랜잭션 헤더 및 데이타의 크기는 모두 40byte이므로 SCI write 트랜잭션의 64byte 패킷 데이타를 채우기 위해서 나머지 24byte는 더미 데이타로 채운다. SCI는 64byte의 캐쉬 라인 크기를 지원하지만, 프로토타입 시스템에서는 원격 트랜잭션 헤더 및 데이타를 64byte SCI 패킷 데이타에 포함하기 위해서 캐쉬 라인 크기를 32byte로 하였다. 본 연구의 최종 목표인 공유메모리 제어 모듈과 네트워크 제어 모듈을 한 장의 보드에 통합한 구현에서는 캐쉬 라인 크기를 64byte로 확대할 예정이다.

4. 실험 및 분석

본 논문에서는 실험을 통하여 CC-NUMA 프로토타입 카드에 기반한 CC-NUMA 방식과 Dolphin사의 PCI-SCI 카드에 기반한 NUMA 방식의 성능을 측정하였다. NUMA 방식은 지역메모리를 공유메모리로 활용하며, 네트워크 캐쉬를 지원하지 않는다. 각 노드는 단일 Intel Pentium II 350MHz를 사용한 PC이며 운영체제로 리눅스를 사용하였다.

4.1 지역 메모리, 공유 메모리, 원격 메모리 접근 시간

2 노드 환경에서 시스템 버스에 존재하는 지역 메모리, PCI 버스에 존재하는 공유메모리, 원격노드의 PCI 버스에 존재하는 원격 메모리에 대한 메모리 접근 시간은 <표 1>과 같다. <표 1>에서 CC-NUMA 시스템의 원격 메모리 read와 write 시간이 같은 이유는 2 노드 환경에서의 원격 메모리 read/write transaction이 1회의 SCI transaction만 발생시킴으로 consistency 유지에 필요한 시간이 동일하기 때문이다. 3 노드 이상의 환경일 때, write transaction에서 sharing list 삭제등을 포함한 consistency문제가 발생하며, 원격 메모리 접근 시간은 read 보다 증가한다.

CC-NUMA 시스템의 원격 메모리 접근 시간은 12.7 μs로 NUMA 시스템의 3배 정도임을 알 수 있다. 앞에서 설명한 바와 같이, CC-NUMA 시스템의 원격 메모리 접근 시간은 PCI-SCI 카드 사용으로 인한 오버헤드를 포함한다. 오버헤드는 PCI 버스를 통하여 데이타를 전송하는 시간과 <그림 3>에서와 같이 더미 데이타를 처리하는 시간으로 구성된다. 프로토타입 카드에서 원격 메모리 read를 수행할 경우, 데이타를 포함하지 않는 request 패킷은 56byte의 더미 데이타를 포함하며, 데이타를 포함하는 response 패킷은 24byte의 더미 데이타를 포함한다. 이러한 더미 데이타 처리와 PCI 버스 사용으로 인한 오버헤드는 실험을 통한 시스템 분석결과 약 4.32μs이다.

본 논문은 더미 데이타 처리로 인한 오버헤드를 없애기 위해서 더미 데이타를 전송하지 않는 실험을 수행하였다. 이 경우 원격 메모리 접근 시간은 16μs로 더미 데이타를 포함하는 경우보다 성능이 좋지 않다. 이것은 PCI-SCI 카드에 존재하는 write merge 시간 때문이다 [11]. PCI-SCI 카드는 PCI 버스를 통하여 들어오는 데이타를 64byte 패킷 데이타를 사용하는 SCI write 트랜잭션을 사용하여 원격노드로 전송한다. 이때, 전송하고자 하는 데이타의 크기가 64byte 보다 작을 경우, 64byte를 채우기 위한 write merge 시간이 존재한다. 이 시간은 프로토타입 시스템에서 3.8μs로 설정되어 있다. 원격 메모리 접근은 request와 response를 포함하여 PCI-SCI 카드를 2번 사용하게 됨으로 총 오버헤드는 7.6μs가 된다. 따라서, 프로토타입 시스템에서는 더미 데이타를 포함시켜 64byte 패킷을 완성한 뒤 전송하는 것이 더 좋은 성능을 가진다. 본 연구는 네트워크 제어 모듈과 공유메모리 제어 모듈을 한 장의 보드로 통합하는 것을 최종목표로 하며, 이 경우, 7.6μs의 오버헤드가 없어져서 원격 메모리 접근 시간은 8.4μs가 될 것으로 예상한다.

표 1 메모리 접근 시간

	CC NUMA 시스템		NUMA 시스템	
	Read	Write	Read	Write
지역 메모리	17ns	23ns	17ns	23ns
공유 메모리	1.2μs	1.2μs	17ns	23ns
원격 메모리	12.7μs	12.7μs	4.02μs	2.59μs

CC-NUMA 카드의 공유메모리 접근 시간은 PCI 버스상에 위치하기 때문에 상당히 크다. 따라서 본 시스템에서는 네트워크 캐쉬 사용으로 발생한 시간 이득이 공

표 2 2 노드 클러스터에서 메모리 read 시간

(단위 : ms)

	Read 회수	CC-NUMA 시스템							NUMA 시스템
		네트워크 캐쉬 hit ratio							
		94%	90%	86%	82%	78%	74%	70%	
지역 메모리	100,000	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
공유 메모리	100,000	120.6	120.7	120.6	121.0	120.7	120.7	120.5	2.1
원격 메모리	100,000	206.4	252.1	297.2	342.6	387.9	433.2	476.8	519.7
합 계	300,000	328.8	374.6	419.6	465.4	510.4	555.7	599.1	523.6

표 3 3 노드 클러스터에서 메모리 read 시간

(단위 : ms)

	Read 회수	CC-NUMA 시스템							NUMA 시스템
		네트워크 캐쉬 hit ratio							
		94%	90%	86%	82%	78%	74%	70%	
지역 메모리	100,000	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
공유 메모리	100,000	124.9	127.2	130.1	132.7	135.1	138.1	140.0	2.1
원격 메모리	200,000	427.3	528.4	628.8	733.7	835.6	929.7	1045.6	1019.3
합 계	400,000	554.0	657.4	760.7	868.2	972.5	1069.6	1187.4	1023.2

표 4 4 노드 클러스터에서 메모리 read 시간

(단위 : ms)

	Read 회수	CC-NUMA 시스템							NUMA 시스템
		네트워크 캐쉬 hit ratio							
		94%	90%	86%	82%	78%	74%	70%	
지역 메모리	100,000	1.8	1.8	1.8	1.8	1.8	1.8	1.8	1.8
공유 메모리	100,000	125.1	127.2	125.0	130.6	132.3	133.4	135.1	2.1
원격 메모리	300,000	643.4	789.1	936.7	1086.2	1242.0	1387.8	1537.2	1510.5
합 계	500,000	770.3	918.1	1063.5	1218.6	1376.1	1523.0	1674.1	1514.4

유메모리 접근으로 발생하는 오버헤드를 보상할 수 있어야 한다. 이것을 측정하기 위해서 아래의 실험을 수행하였다.

4.2 Hit ratio에 변환에 따른 시스템 성능

2, 3, 4 노드 클러스터상에서 네트워크 캐쉬 hit ratio를 변화시키면서 CC-NUMA 시스템과 NUMA 시스템의 성능을 측정한 결과는 <표 2, 3, 4>와 같다. 본 실험에서는 지역, 공유, 원격 메모리에 대한 데이타 분할 비율을 결정하기 위해서 matrix multiplication 알고리즘의 경우를 기준으로 하였다. Matrix A와 B의 multiplication에서 A는 row major, B는 column major로 저장되며, matrix A와 B는 <그림 4>처럼 각 노드에 균0등하게 할당한다. 또한 A는 공유되지 않으므로 각 노드의 지역메모리에 할당하고, B는 모든 노드에서 공유하게 됨으로 각 노드의 공유메모리에 할당한다.

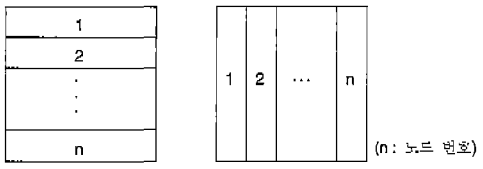


그림 4 Matrix multiplication을 위한 메모리 할당

각 메모리에 대한 read 비율은 분할된 데이타 크기에 비례한다. 4노드 실험의 경우, 각 노드는 지역메모리, 공유메모리에 대해서 10만번, 원격메모리에 대해서 30만번의 read를 수행한다. 본 실험에서는 네트워크 캐쉬 hit ratio를 94%에서 70%까지 변화시키면서 지역, 공유, 원격 메모리 read 시간을 측정하였다. 네트워크 캐쉬 hit ratio는 같은 캐쉬 라인에 대한 read 회수를 변화시킴으

로써 조정하였으며, 이를 통하여 CC-NUMA 시스템이 NUMA 시스템에 비해서 좋은 성능을 가지는 네트워크 캐쉬 hit ratio의 값의 범위를 알 수 있었다.

<표 2, 3 ,4>에 나타난 값은 각 메모리에 대해서 표에서 지정한 횟수만큼 read한 시간의 총합이다. 원격 메모리 read 시간의 경우, CC-NUMA 시스템은 네트워크 캐쉬 사용으로 NUMA 시스템보다 상당히 빠르다. 그러나, CC-NUMA 시스템의 공유메모리 read 시간은 PCI 상에 위치함으로 NUMA 시스템 보다 느리다. 전체 메모리 read 시간은 2, 3, 4 노드의 경우, 네트워크 캐쉬 hit ratio가 약 77%, 76%, 75% 일 때 CC-NUMA 시스템과 NUMA 시스템의 성능이 교차됨을 알 수 있다. CC-NUMA 시스템의 성능이 노드수가 증가함에 따라 좋아지는 것은 노드수 증가에 비례해서 원격 메모리 접근이 증가하기 때문이다.

CC-NUMA 시스템에서 네트워크 캐쉬 hit ratio에 따른 write 실험을 수행하지 않았으며, CC-NUMA 시스템과 NUMA 시스템의 성능이 교차하는 네트워크 캐쉬 hit ratio를 다음과 같이 예상하였다. 원격 메모리 hit시 CC-NUMA 시스템은 NUMA 시스템에 비해서 1회당 1.39μs 빠르며, hit되지 않을 경우는 10.11μs가 느리다. 지역 메모리 접근 시간 및 공유 메모리 접근 시간을 포함한 다른 factor는 원격 메모리 접근 시간에 비해 작기 때문에 무시한다. 원격 메모리 hit로 인한 성능 향상과 hit 되지 않을 경우의 성능 감소의 합이 0이 되는 지점이 두 시스템의 성능이 교차하는 네트워크 캐쉬 hit ratio의 값이다. 2 노드의 경우, hit ratio는 약 88%(=10.11/(1.39+10.11))로 예측된다. 3 노드 CC-NUMA 시스템의 경우, sharing-list 삭제를 위한 1회의 SCI transaction이 추가 수행되어 원격 메모리 write시간이 증가한다. 이 경우, hit ratio는 약 93%(=(10.11*2)/(1.39+10.11*2))가 된다. 4 노드의 경우, 3 노드와 유사하게 2회의 SCI transaction이 추가 수행되어 hit ratio는 약 96%(=(10.11*3)/(1.39+10.11*3))가 된다. SPLASH-2의 메모리 read와 write의 평균비율이 2:1인 것을 고려하면[12], 두 시스템의 성능이 교차하는 hit ratio는 약 82%가 될 것으로 예상한다.

SPLASH-2[13]의 hit ratio가 평균 90%이상인 것을 고려한다면, 실험결과는 매우 우수하다고 판단된다. 공유메모리 제어 모듈과 네트워크 제어 모듈이 한 장의 보드에 통합 구현되는 최종 CC-NUMA 시스템에서는 앞서 언급한 바와 같이 PCI-SCI 카드 사용으로 인한 오버헤드가 없어짐에 따라 성능은 더욱 향상될 것으로 예측된다.

5. 결론 및 향후 연구 과제

본 논문에서는 SCI 기반 PC 클러스터 시스템을 위한 CC-NUMA 카드를 제안하였다. CC-NUMA 카드는 각 노드의 PCI 슬롯에 plug-in되는 형태이며, 공유메모리, 네트워크 캐쉬, 네트워크 제어 모듈을 포함한다. 네트워크 캐쉬는 PCI 버스에 존재하는 공유메모리를 캐쉬하며, IEEE SCI 표준에 기반한 캐쉬 일관성 유지 알고리즘을 사용한다. 최종 CC-NUMA 카드 구현에 앞서 프로토타입 카드를 구현하였으며, 2, 3, 4 노드 클러스터 실험을 통하여, 네트워크 캐쉬 hit ratio가 약 75%이상일 때 CC-NUMA 시스템의 성능이 네트워크 캐쉬를 활용하지 않는 NUMA 시스템에 비해서 우수함을 알 수 있었다. 또한, 공유메모리가 PCI 버스상에 위치함으로서 발생하는 오버헤드를 네트워크 캐쉬를 통하여 극복할 수 있다는 것을 확인하였다. 본 연구진은 현재 공유메모리 제어 모듈과 네트워크 제어 모듈이 한 장의 보드에 통합되는 최종 CC-NUMA카드를 구현 중에 있다.

참 고 문 헌

[1] http://www.myri.com

[2] A. Mainwaring and D. Culler, "Active Message Applications Programming Interface and Communication Subsystem Organization", *Technical Document*, 1995.

[3] S. Pakin, V. Karamcheti and A. A. Chien. Fast Messages (FM): Efficient, Portable Communication for Workstation Clusters and Massively-Parallel Processors. *IEEE Concurrency*, Vol. 5, Issue. 2, pp. 60-72, 1997.

[4] A. Basu, V. Buch, W. Vogels and T. von Eicken. U-Net: A User-Level Network Interface for Parallel and Distributed Computing. *Proceedings of the 15th ACM Symposium on Operating Systems Princi-ples*, pp. 40-53. Copper Mountain, Colorado, December 3-6 1995.

[5] Myricom, Inc. The GM API. White Paper. Myricom, Inc., 1997.

[6] http://www.sequent.com/whitepapers/numa_arch.html

[7] R. Clark. SCI Interconnect Chipset and Adapter: Building Large Scale Enterprise Servers with Pent-ium Pro SHV Nodes. White Paper. Data General Corporation, 1999.

[8] http://www.dolphinics.com

[9] Wolfgang Karl, Markus Lebercecht, Martin Schulz, Supporting Shared Memory and Message Passing on Cluster of PCs with a SMiLE,*CANPC* 99,

Orlando, USA (together with HPCA-5), January, 1999.
[10] Mario Trams, Wolfgang Rehm, Daniel Balkanski, Stanislav Simeonov, Memory Management in a combined VIA/SCI Hardware, *IPDPS 2000 Workshops*, pp. 4-15 Cancun, Mexico, May 2000.
[11] Dolphin Interconnect Solutions, PCI-SCI card IRM driver version 1.5.0, Dolphin Interconnect Solutions, 1998.
[12] Woo S. C., Ohara M., Torrie E., Pal Singh J., Gupta A., "The SPLASH-2 Programs: Characterization and Methodological Considcrations", *Proceedings of the 22nd ISCA*, pp. 24-36, June 1995.
[13] David A. Patterson, Joh L. Hennessy, Computer Architecture A Quantitative Approach, *Morgan Kaufmann Publishers*, pp. 687-689, 1996.

오 수 철

1995년 부산대학교 컴퓨터공학과 학사. 1997년 부산대학교 컴퓨터공학과 석사. 1997년 ~ 1998년 LG전자 멀티미디어 연구소 연구원. 1998년 ~ 현재 부산대학교 컴퓨터공학과 박사과정. 관심분야는 클러스터 시스템, 병렬처리, SCI, CC-NUMA, VOD

정 상 화

1985년 서울대학교 전기공학과 학사. 1988년 Iowa State University 전기 및 컴퓨터공학과 석사. 1993년 University of Southern California 전기 및 컴퓨터공학과 박사. 1993년 ~ 1994년 University of Central Florida 전기 및 컴퓨터공학과 조교수. 1994년 ~ 현재 부산대학교 컴퓨터공학과 부교수. 관심분야는 클러스터 시스템, 병렬처리, 정보검색, VOD, Infiniband