고성능 PC 클러스터링을 위한 SCI 기반 Network Cache Coherent NUMA 시스템의 설계 및 구현

(Design and Implementation of an SCI-Based Network Cache Coherent NUMA System for High-Performance 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 공유 메모리를 PCI 버스상에 위치시킴으로써 네트워크 캐쉬 지원을 가능하게 하였으며, 이에 기반한 Network Cache Coherenet NUMA(NCC-NUMA) 시스템을 제안하고, 핵심 모듈인 NCC-NUMA 카드를 개발하였다. NCC-NUMA 카드는 각 노드의 PCI 슬롯(slot)에 plug-in되는 형태이며, 공유메모리, 네트워크 캐쉬, 공유메모리 제어 모듈 및 네트워크 제어 모듈을 포함한다. 공유메모리와 네트워크 캐쉬 사이의 일관성은 IEEE SCI 표준에 의해 유지된다. NCC-NUMA 시스템의 성능 측정을 위해 SPLASH-2 벤치마크를 수행하였으며, NCC-NUMA 시스템이 네트워크 캐쉬를 활용하지 않는 NUMA 기반 클러스터 시스템에 비해서 최대 56%의 성능향상을 보임을 알 수 있었다.

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

Abstract It is extremely important to minimize network access time in constructing a high-performance PC cluster system. For PC cluster systems, it is possible to reduce network access time by maintaining network cache in each cluster node. This paper presents a Network Cache Coherent NUMA (NCC-NUMA) system to utilize network cache by locating shared memory on the PCI bus, and the NCC-NUMA card which is core module of the NCC-NUMA system is developed. The NCC-NUMA card is directly plugged into the PCI slot of each node, and contains shared memory, network cache, shared memory control module and network control module. 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. According to the SPLASH-2 benchmark experiments, the NCC-NUMA system showed improvements of 56% compared with an SCI-based cluster without network cache.

Key words : Network Cache, CC-NUMA, Shared Memory, SCI, PC Cluster, PCI

1. 서 론

최근에 사회의 모든 분야에서 고성능 서버에 대한 수요가 증가하고 있으며, 이러한 요구를 수용하기 위해서 PC 클러스터 시스템이 등장하였다. PC 클러스터 시스템은 고속 CPU를 장착한 저가의 PC를 고속 네트워크로 연결하여 하나의 컴퓨팅 시스템으로 사용하는 것이다. 이러한 PC 클러스터 시스템에서 CPU는 굉장히 빠른 속도로 발전하지만, 네트워크의 발전은 이를 따르지 못하고 있으며, 결국 전체 PC 클러스터 시스템의 성능 향상을 지연시키고 있다. 따라서, PC 클러스터 시스템이 중대형 컴퓨터에 필적하는 컴퓨터 시스템으로 성공하기 위해서는 PC를 연결하는 네트워크의 성능이 뒷받침되어야 한다. PC 클러스터 시스템을 위한 네트워크는 높은 대역폭 및 낮은 전송지연시간을 지원하여 원격 노드와의 데이타 교환 시간을 최소화해야 하며, 공유 메모리

† 비 회 원 : 부산대학교 컴퓨터공학과
osc@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과
shchung@pusan.ac.kr
논문접수 : 2003년 12월 22일
심사완료 : 2004년 11월 19일

방식에 기반한 데이타 전송 방식을 제공하여 사용자의 프로그램 용이성을 지원해야 한다.

현재 PC 클러스터를 위한 대표적인 네트워크로 Fast/Gigabit Ethernet, Myrinet[1] 및 SCI(Scalable Coherent Interface : IEEE standard 1596-1992))[2]가 있으며, 물리적 대역폭은 각각 100Mbps/1Gbps, 2Gbps, 5.3Gbps이다. 전송 지연시간은 Gigabit Ethernet에 TCP/IP를 사용했을 때 45μs[3], Myrinet상에 GM을 사용한 경우는 6.3μs[1]이다. Dolphin사의 PCI-SCI 카드상에 공유메모리방식에 의한 데이타 전송시 SCI는 4.2 μs[4,5]의 전송지연시간을 가진다. 여기서, SCI가 대역폭 및 전송 지연 시간 측면에서 가장 우수한 성능을 보여준다. 특히, Fast/Gigabit Ethernet 및 Myrinet이 사용하는 데이타 전송 방식인 메시지 패싱 방식은 빈번한 데이타 복사와 OS 시스템 콜로 인한 소프트웨어 오버헤드가 크기 때문에 실제 사용자 프로그램 수준에서 사용 가능한 성능은 물리적 성능에 훨씬 못 미친다. 또한, 메시지 패싱 방식은 원격 노드에 존재하는 메모리에 접근하기 위해 라이브러리 형태의 명령어를 사용해야 함으로 프로그램이 어려운 단점이 있다. 반면, SCI는 SAN(System Area Network)을 지원하기 위해서 IEEE에서 개발한 interconnection의 표준으로 분산 공유 메모리 방식의 데이타 전송 방식을 지원한다. 분산 공유 메모리 방식은 메모리 트랜잭션을 사용하여 원격 노드에 존재하는 메모리를 접근할 수 있기 때문에 프로그램이 용이하다는 장점이 있다. 따라서, 네트워크의 성능 및 사용자 편의성을 고려했을 때, PC 클러스터 시스템을 위한 가장 적절한 네트워크는 SCI라고 판단된다.

SCI에 기반한 PC 클러스터 시스템으로는 Dolphin[6], LRR-TUM[7], TUC[8]에서 개발한 SCI 네트워크 카드를 사용한 시스템이 있으며, SCI 네트워크 카드상에 분산 공유 메모리에 기반한 가상 공유 메모리를 지원하는 SMiLE[9] 및 SciFS[10]와 같은 시스템 소프트웨어도 개발되었다. 또한, 이러한 시스템에서 제공하는 분산 공유 메모리 방식을 NUMA(Non-Uniform Memory Access) 방식이라 한다. SCI 네트워크 카드는 PC의 PCI 슬롯에 장착되며, PC 클러스터 시스템의 각 노드에 존재하는 메인 메모리를 공유 대상으로 한다. 이러한 분산 공유 메모리 방식은 네트워크 캐쉬를 지원함으로써 성능을 향상시킬 수 있다. 중대형 서버 시스템의 경우에는 네트워크 캐쉬를 지원하는 CC-NUMA(Cache Coherenet NUMA) 시스템이 등장하여 상업적인 성공을 거두었지만, PC 클러스터 시스템에는 이러한 방식의 적용이 불가능하다. 따라서, 본 논문에서는 SCI 기반 PC 클러스터 시스템을 위한 네트워크 캐쉬를 제안하고, 이를 활용한 NCC-NUMA 시스템을 개발하였다. NCC-NUMA 시스템은 공유 대상 메모리를 메인 메모리가 아닌 PCI 버스상에 위치시킴으로써 네트워크 지원이 가능하도록 하였으며, 이를 지원하는 NCC-NUMA 카드를 개발하였다. 또한, 이를 장착한 SCI 기반 PC 클러스터 시스템을 개발하고, 시스템 성능을 측정하였다.

2. 관련 연구

본 장에서는 기존의 시스템에서 네트워크 캐쉬를 지원하는 시스템에 대해서 살펴본다. PC 클러스터 시스템을 위한 네트워크 캐쉬는 소프트웨어와 하드웨어 방식으로 구현가능하다. 소프트웨어 방식은 Fast/Gigabit Ethernet 및 Myrinet과 같은 메시지 패싱 시스템상에 분산 공유 메모리를 지원하는 라이브러리를 개발하고, 성능 향상을 위해서 네트워크 캐쉬를 지원하는 방식이다. 대표적으로 Rice 대학에서 개발한 TreadMark[11]이 있다. 이러한 소프트웨어 방식은 메시지 패싱을 기반으로 하는 시스템상에 구현됨으로 구현 비용이 작다는 장점이 있으나, 지연시간이 큰 메시지 패싱 방식을 사용하고, 소프트웨어로 구현되어 있기 때문에 성능향상에 제약이 있다.

PC 클러스터 시스템을 위한 하드웨어 기반 네트워크 캐쉬에 관한 기존의 연구는 없지만, 중대형 서버를 위한 하드웨어 기반 네트워크 캐쉬에 관한 연구로 CC-NUMA 방식이 존재한다. 대표적인 시스템으로 Standford의 FLASH[12], MIT의 Alewife[13]등이 있으며, Data General의 Aviion[14], IBM의 NUMA-Q[15]는 상업적인 성공을 거두었다. CC-NUMA 방식은 메인 메모리를 공유 대상으로 하며, 시스템버스상에 CC-NUMA 모듈을 위치시킨다. 지역 CPU에서 원격 노드에 있는 메모리를 접근하고자 할때, 시스템 버스를 통하여 CC-NUMA 모듈로 메모리 접근 명령이 전달된다. CC-NUMA 모듈은 원격 노드의 CC-NUMA 모듈과 통신하면서 원격 노드의 메모리를 읽고 쓰며, 공유 메모리로 활용되는 메인 메모리와 네트워크 캐쉬사이의 캐쉬 일관성을 유지 시켜주는 역할을 담당한다. 이러한 시스템은 100MHz이상의 고속으로 동작하는 시스템 버스를 기반으로 함으로 전체 시스템이 고성능을 가지는 장점이 있으나, CC-NUMA 모듈의 개발 비용이 높다는 단점이 있다. 또한, PC 클러스터 시스템에서는 구현이 불가능하다.

본 연구진은 본 논문에서 제안하는 NCC-NUMA 카드 개발의 사전연구로 NCC-NUMA 프로토타입 카드를 개발하였다[4,5]. PCI 버스에 장착되는 NCC-NUMA 프로토타입 카드는 공유 메모리 제어 모듈, 공유 메모리 및 네트워크 캐쉬를 포함한다. 네트워크 제어 모듈은 Dolphin사의 PCI-SCI 카드[6]를 활용하였다. 따라서,

NCC-NUMA 프로토타입 카드는 공유 메모리 제어 모듈의 캐쉬 일관성 유지 메카니즘 개발에 주력하였다. 지역 CPU에서 발생한 공유메모리 접근 명령은 PCI 버스를 통하여 NCC-NUMA 프로토타입 카드로 전송된다. 원격 노드의 데이타를 접근해야 하는 경우, NCC-NUMA 프로토타입 카드는 PCI 버스를 통하여 PCI-SCI 카드에게 원격 노드와의 데이타 전송을 요청한다. 따라서, 원격 메모리 접근은 PCI-SCI 카드 사용으로 인한 오버헤드를 포함한다. 본 논문에서는 공유메모리 제어 모듈과 네트워크 제어 모듈을 한 장의 카드로 통합함으로써, 이러한 오버헤드를 제거한 NCC-NUMA 카드를 개발하였다.

3. NCC-NUMA 기반 PC 클러스터 시스템

본 논문에서 제안하는 NCC-NUMA 카드에 기반한 PC 클러스터 시스템의 구조는 그림 1과 같다. 시스템의 노드로는 단일 CPU를 가진 PC를 기본으로 하되 복수개의 CPU를 탑재한 SMP 시스템을 사용할 수도 있다. PC는 메인보드상에서 시스템 버스 인터페이스를 제공하지 않음으로 NCC-NUMA 카드는 PCI 버스상에 위치한다. NCC-NUMA 카드는 공유메모리, 네트워크 캐쉬, 공유메모리 제어 모듈 및 네트워크 제어 모듈로 구성되고, 각 노드는 SCI 네트워크를 사용하여 연결된다. NCC-NUMA 카드의 공유메모리 제어 모듈은 지역 및 원격 CPU에서 발생한 공유메모리 read/write 명령을 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.

3.1 공유 메모리 위치

PC 클러스터 시스템에서 시스템 버스상의 메인 메모리를 공유한다고 가정할 때, 노드 1의 CPU가 노드 2의 NCC-NUMA 카드에 의해서 캐쉬중인 노드 1의 메인 메모리에 write 하는 과정을 설명하면 다음과 같다. 노드 1의 CPU에서 발생한 write는 시스템 버스를 통하여 메인 메모리에 전달된다. 이때 노드 1의 NCC-NUMA 카드는 시스템 버스상에 발생하는 write를 snoop하여 노드 2의 NCC-NUMA 카드에게 현재 캐쉬하고 있는 내용을 invalidation 시켜라는 명령을 전송해야 한다. 그러나, PCI specification 2.1에 규정된 캐쉬 지원 기능이 대부분의 PC 칩셋에서는 지원되지 않기 때문에, PCI 버스에 위치한 NCC-NUMA 카드는 시스템 버스를 snoop 할 수 없으며, 캐쉬 일관성을 유지시킬 수 없다. 이를 해결하기 위해서, 본 논문에서는 PCI 버스상에 공유 대상 메모리를 위치시킨다. 본 시스템에 존재하는 메모리는 지역메모리, 공유 메모리, 네트워크 캐쉬로 구성된다. 지역메모리는 각 노드의 시스템 버스에 장착되어 있는 메인 메모리로 지역 CPU의 L1, L2 캐쉬에 의해서 캐쉬되며, 원격 노드에서는 공유 및 캐쉬 되지 않는다. 공유 메모리는 지역 CPU의 L1, L2캐쉬에 캐쉬 되지 않으며, 원격 노드의 네트워크 캐쉬에 의해 캐쉬된다. 네트워크 캐쉬는 원격 노드의 공유 메모리를 캐쉬한다.

3.2 공유 메모리 접근

그림 2와 같이 각 노드에 존재하는 지역 메모리 및 공유 메모리는 물리/PCI 주소 공간에 존재한다. 이를 CPU에서 접근하기 위해서는 MMU(Memory Management Unit)에 의해서 CPU의 가상 주소 공간으로 매핑

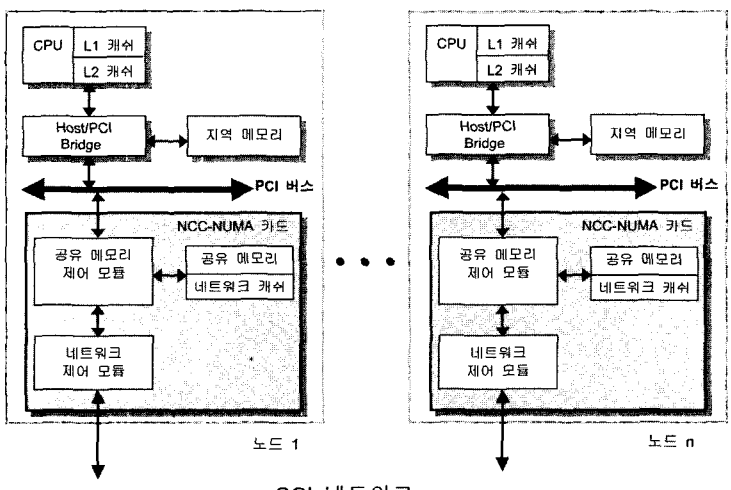


그림 1 NCC-NUMA 기반 PC 클러스터 시스템

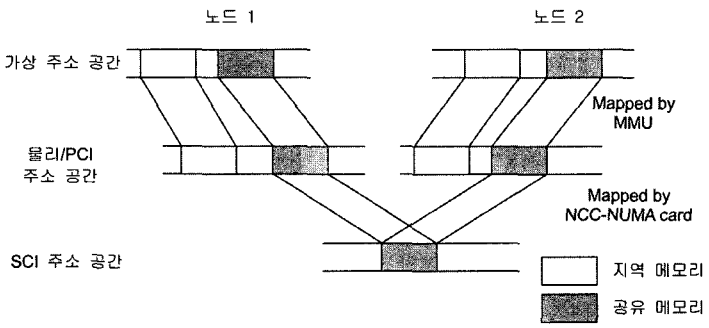


그림 2 지역 및 공유 메모리 매핑

한다. 또한, 물리/PCI 주소 공간은 NCC-NUMA 카드에 의해서 단일 SCI 주소 공간으로 매핑된다. 공유 메모리는 SCI 주소 공간에서 유일한 주소를 가지며, 모든 노드에서 동일한 SCI 주소를 사용하여 공유 메모리에 접근 가능하다. 지역 메모리는 원격 노드와 공유되지 않음으로 SCI 주소 공간으로 매핑되지 않는다.

NCC-NUMA 시스템의 노드 1이 공유 메모리를 읽는 과정을 설명하면 다음과 같다. CPU는 가상 주소를 사용하여 메모리 접근 명령을 발생시킨다. 가상 주소는 물리/PCI 주소 공간으로 변환되며, 물리/PCI 주소 공간이 지역 메모리 주소 공간에 속하면 지역 메모리를 대상으로 하는 메모리 접근 명령을 발생시킨다. 물리/PCI 주소 공간이 PCI 버스상에 위치한 공유 메모리에 속하면 PCI 버스를 통하여 NCC-NUMA 카드의 공유메모리 제어 모듈에 전달된다. 공유메모리 제어 모듈은 PCI 물리 주소 공간을 SCI 주소 공간으로 변환한다. 공유 메모리 및 네트워크 캐쉬에 유효한 데이타가 존재하면, 이를 읽어서 CPU로 전송한다. 유효한 데이타가 없으면, 네트워크 제어 모듈을 사용하여 데이타를 소유한 원격 노드로 메모리 read 명령어를 전송한다. 이를 수신한 원격 노드의 공유 메모리 제어 모듈은 유효한 데이타를 읽어서 노드 1로 전송한다. 노드 1의 공유 메모리 제어 모듈은 데이타를 네트워크 캐쉬에 저장하고, CPU로 전송한다.

NCC-NUMA 시스템은 공유 메모리가 PCI 버스상에 위치함으로 지역 CPU의 공유 메모리 접근 시간이 크다는 단점을 가지고 있다. 본 논문에서는 실험을 통하여 네트워크 캐쉬 사용으로 인한 이득이 공유 메모리가 PCI 버스상에 위치하는 단점을 극복하고, 더 나아가 성능을 향상시킬 수 있음을 보일 것이다.

4. NCC-NUMA 카드

본 논문에서는 그림 3과 같이 PCI interface FPGA, shared memory control module FPGA, LC3, SDRAM

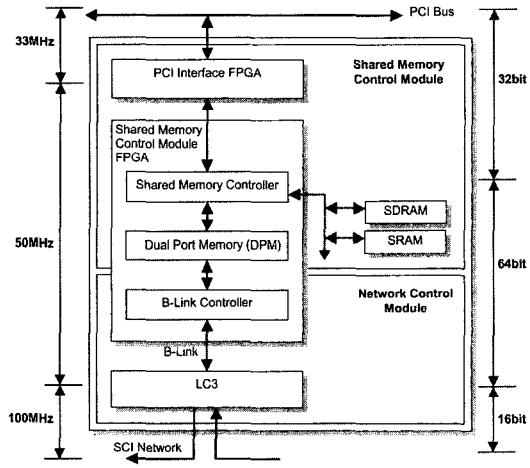


그림 3 NCC-NUMA 카드의 구조

및 SRAM으로 구성된 NCC-NUMA 카드를 개발하였다. 앞서 설명한 바와 같이 NCC-NUMA 프로토타입 카드의 경우, 원격 메모리 접근시 PCI 버스를 사용하는 오버헤드가 있었는데 이를 제거하기 위해서 공유 메모리 제어 모듈과 네트워크 제어 모듈을 한 장의 카드로 통합하였다. 그리고 데이타 처리 속도를 향상시키기 위해서, NCC-NUMA 카드의 데이타 버스를 64bit, 50MHz으로 확장하였다. NCC-NUMA 카드의 LC3, SRAM 및 SDRAM이 64bit 데이타 버스를 사용하기 때문에, NCC-NUMA 카드의 데이타 버스를 64bit로 함으로써 전체 시스템의 성능을 향상시킬 수 있다. 그러나, PCI interface FPGA는 PCI specification 2.1에 따라서 32bit, 33MHz로 동작한다.

PCI interface FPGA는 Xilinx사의 15만 gate급 SpartanII(XC2S150-5-PG208, System Gates:150K)를 사용하여 PCI specification 2.1 기준에 의해 개발하였으며, 지역 CPU에서 발생한 공유 메모리 접근 명령어를 shared memory module FPGA로 전송한다. Shared

memory control module FPGA는 Xilinx사의 40만 gate급 VirtexE(XCV400E-6-BG432, System Gates: 596K)를 사용하여 개발하였으며, 지역 및 원격 노드에서 발생한 공유 메모리 접근 명령을 처리한다. 또한, SCI의 캐쉬 일관성 유지 알고리즘을 사용하여 캐쉬 일관성을 유지시키는 역할을 한다. LC3는 667Mbyte/s의 대역폭을 가지는 Dolphin사의 link controller로 원격 노드와의 SCI 트랜잭션 교환을 담당하며, 원격 노드와의 연결을 위해 16bit parallel data link를 사용한다. 또한, LC3는 FPGA와 같은 외부 device와의 인터페이스를 위해 64bit 버스인 B-Link를 제공한다. SRAM은 tag정보를 저장하며 4Mbyte의 Synchronous SRAM을 사용한다. SDRAM은 공유메모리 및 네트워크 캐쉬를 저장하며 상용 64Mbyte Synchronous DRAM 모듈을 사용한다. 각 노드에 위치하는 공유메모리의 크기는 32bit 시스템을 기준으로 할때, 4GB 주소공간안에서 클러스터 시스템을 구성하는 노드수에 따라 변경 가능하다. 예를 들어 8노드 클러스터 시스템에서 각 노드의 메인 메모리가 512MB일 경우, 각 노드의 공유 메모리 크기는 최대 437MB(=(4GB-500MB)/8)까지 확장 가능하다. 그리고, 클러스터 시스템의 각 노드가 64bit 기반 시스템으로 향상된다면, 공유 메모리의 크기는 64bit 주소공간까지 확장가능하다.

PCI interface FPGA는 SCI 캐쉬 일관성 유지 알고리즘의 효율적인 구현을 위해서 burst write 트랜잭션을 SCI 캐쉬 라인 크기단위로 분리하는 기능을 지원한다. 지역 CPU는 PCI 버스상에 존재하는 공유 메모리의 캐쉬 정책에 관한 정보가 없기 때문에 SCI 캐쉬 라인보다 큰 데이타를 가진 burst write 트랜잭션을 발생시킬 수 있다. 그러면, NCC-NUMA 카드는 burst write 트랜잭션을 SCI 캐쉬 크기에 맞게 여러개의 burst write 트랜잭션으로 분리해야 한다. PCI interface로 기존의 PLX9050[16]같은 상용 PCI interface 칩을 사용할 경우, PLX9050은 burst wirte 분리 기능을 지원하지 않기 때문에, shared memory controller에서 이 기능을 수행해야 한다. 그러나, shared memory controller에서 이 기능을 구현하기 위해서는 구조가 매우 복잡해진다. 반면, PCI interface FPGA에서는 단순한 알고리즘을 적용함으로써 이 기능을 수행할 수 있고, 동시에 shared memory controller의 구조를 단순화시킬 수 있다. 본 논문의 PCI interface FPGA는 SCI 캐쉬 라인 크기를 벗어나는 데이타를 retry하게 하는 PCI delayed write 트랜잭션을 사용함으로써 이것을 수행한다. 그림 4의 예를 보면, CPU는 burst write 1을 발생시킨다. PCI interface는 burst write 1-1을 정상 처리하고, burst write 1-2을 retry시킨다. Burst write 1-1의 종료후에,

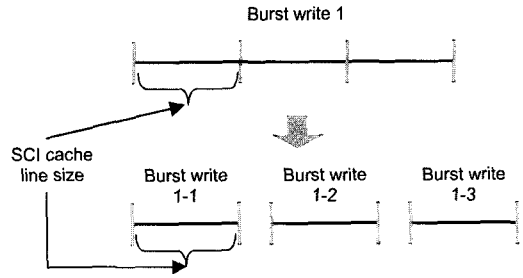


그림 4 Burst write 트랜잭션의 분리

burst write 1-2이 처리되고, burst write 1-3은 retry된다. Burst write 1-2가 종료되면, retry된 burst write 1-3이 처리된다.

CPU에서 원격 메모리 접근 명령이 발생하면 PCI interface FPGA를 통하여 shared memory control module FPGA내의 shared memory controller로 전달된다. Shared memory controller는 SRAM에 저장되어 있는 공유 메모리 및 네트워크 캐쉬에 관한 태그 정보를 읽어와서 유효한 데이타가 존재하는지 검사한다. 유효한 데이타가 존재하면 SDRAM에 저장된 데이타를 읽어서 CPU로 전송한다. 유효한 데이타가 없으면 데이타를 소유한 노드로 데이타의 전송을 요구하는 SCI 트랜잭션을 생성하여 dual port memory에 저장한다. Dual port memory는 원격 노드로 전송할 SCI 트랜잭션을 저장하는 outgoing queue와 원격 노드에서 전송된 SCI 트랜잭션을 저장하는 incoming queue로 구성되며, 각 queue는 총 16개의 SCI 트랜잭션을 저장할 수 있다. B-link controller는 dual port memory에 존재하는 SCI 트랜잭션을 LC3를 통하여 원격 노드로 전송하거나, 원격 노드에서 전송된 SCI 트랜잭션을 dual port memory에 저장한다. 원격 노드에서 전송된 SCI 트랜잭션은 dual port memory를 통하여 shared memory controller에 전달되어 처리된다.

Shared memory controller는 그림 5와 같은 구조를 가지며, 지역 CPU 및 원격 노드에서 발생한 공유메모리 접근 명령을 처리하고, 캐쉬 일관성을 유지시킨다. 지역 CPU에서 발생된 공유 메모리 접근 명령은 local access controller로 전달된다. Local access controller는 공유 메모리 및 네트워크 캐쉬에 유효한 데이타가 있으면 CPU로 전달하며, 유효한 데이타가 없을 경우, remote access controller로 공유 메모리 접근 명령을 전달한다. NCC-NUMA 시스템에서는 PCI 버스상에 위치하는 공유 메모리에 대한 접근 시간을 최소화해야 하며, 이를 위해서 local access controller를 hard wired logic으로 개발하였다. 또한, local access controller와 remote access controller가 동시에 memory controller

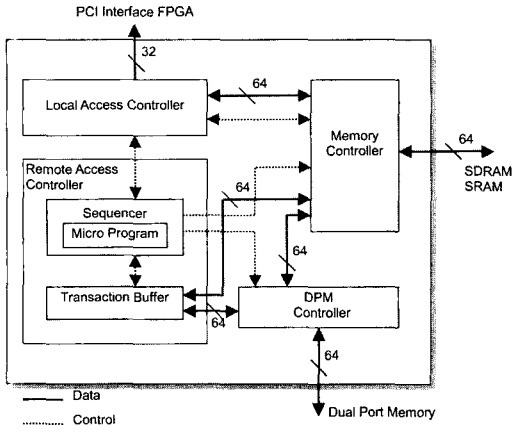


그림 5 Shared memory controller

를 사용하고자 할 때, local access controller에 우선 순위를 부여하였다.

Remote access controller는 local access controller에서 전달된 공유메모리 접근 명령과 태그의 상태에 따라 적절한 SCI 트랜잭션을 생성하고 dual port memory를 통하여 원격 노드로 전송한다. 또한, 원격 노드에서 발생한 공유메모리 접근 명령을 처리하여, 처리 결과를 공유 메모리 접근 명령이 발생한 노드로 전송한다. Remote access controller는 캐쉬 일관성 유지를 위해 다양한 SCI 트랜잭션을 처리해야 함으로, micro program 방식을 적용하여 개발하였다. Sequencer는 SCI 트랜잭션을 처리하는 전체 과정을 제어하며, micro program은 각각의 SCI 트랜잭션에 대한 처리 순서를 저장한다. Transaction buffer는 현재 처리중인 SCI 트랜잭션에 관한 정보를 저장하며, 총 16개의 SCI 트랜잭션을 저장할 수 있다.

5. 실험 및 분석

본 논문에서는 실험을 통하여 NCC-NUMA 카드에 기반한 PC 클러스터 시스템(NCC-NUMA 시스템)의 성능을 측정하였으며, 비교 대상으로 네트워크 캐쉬를 지원하지 않는 NUMA 시스템의 성능을 함께 측정하였다. NUMA 시스템은 Dolphin사의 PCI-SCI 카드에 기반한 PC 클러스터 시스템으로, 지역 메모리를 공유메모리로 사용한다. 실험을 위해서 4노드 PC 클러스터 시스템을 구축하였으며, 각 노드는 단일 인텔 펜티엄III 800MHz를 사용한 PC이고, 운영체제로는 리눅스(Kernel version 2.2.13)를 사용하였다. 각 노드의 메인 메모리의 크기는 256MB이고, NCC-NUMA 카드의 공유 메모리 및 네트워크 캐쉬의 크기는 각각 48MB, 12MB이다. SCI 네트워크는 링 구조를 사용하였다.

5.1 지역 메모리, 지역 공유 메모리, 원격 공유 메모리 read 시간

2 노드 PC 클러스터상에서 시스템 버스에 존재하는 지역 메모리, PCI 버스에 존재하는 지역 공유 메모리, 원격노드의 PCI 버스에 존재하는 원격 공유 메모리에 대한 메모리 접근 시간을 측정한 결과는 표 1과 같다. NCC-NUMA 시스템과 NUMA 시스템의 지역 메모리 접근 시간은 10ns로 동일하다. 그러나, NCC-NUMA 시스템의 지역 공유 메모리 read 시간은 NUMA 시스템보다 매우 크다. 이것은 NCC-NUMA 시스템의 지역 공유 메모리가 PCI 버스상에 존재하기 때문이다. NCC-NUMA 시스템의 원격 공유 메모리 read 시간은 캐쉬 일관성 유지를 위해 소요되는 시간 때문에 NUMA 시스템 보다 조금 크지만, 네트워크 캐쉬의 효과를 고려할 경우, NUMA 시스템보다 작아질 것이다. 따라서, NCC-NUMA 시스템의 성능이 NUMA 시스템보다 우수하기 위해서는 네트워크 캐쉬 사용으로 발생한 이득이 지역 공유 메모리가 PCI 버스상에 존재하는 단점를 보상할 수 있어야 한다. 이것을 측정하기 위해서 다음의 실험을 수행하였다.

표 1 2 노드에서 메모리 read 시간

	NCC-NUMA 시스템	NUMA 시스템
지역메모리	10ns	10ns
지역 공유 메모리	0.9μs	10ns
원격 공유 메모리	5.38μs	4.02μs

5.2 SPLASH-2 벤치마크 실험을 통한 성능 측정

NCC-NUMA 시스템의 성능을 측정하기 위해서 2, 4 노드 클러스터 시스템을 구축하고 SPLASH-2[17] 벤치마크를 수행하였다. SPLASH-2는 Standford 대학에서 개발된 공유 메모리 시스템을 위한 대표적인 벤치마크이다. 실험에 사용된 SPLASH-2 프로그램의 problem size를 표 2에 정리하였다.

SPLASH-2를 수행하기 위해서 공유 데이타를 노드 수에 따라 같은 크기로 분할하여 각 노드의 공유 메모리에 할당하였다. 또한, 원격 노드에서 접근되지 않고 지역 노드에서만 접근되는 공유 메모리 부분은 지역 메모리에 할당하여, 메모리 접근에 사용되는 시간을 최소

표 2 SPLASH-2 프로그램의 problem size

프로그램	Problem size
LU	256 x 256 matrix 16 x 16 blocks
Barnes	4K particles
Ocean	66 x 66 ocean
FFT	1024 points

화하였다. Ocean과 FFT를 분석해 보면 원격 노드에서 접근되지 않고 지역 노드에서만 접근되는 공유 메모리 부분이 일정한 패턴을 가지고 존재한다. 따라서, 이 부분을 지역 메모리에 할당하였다. LU와 Barnes에서는 지역 노드에서만 접근되는 메모리 부분이 일정한 패턴을 가지고 있지 않기 때문에, 모든 메모리를 공유 메모리에 할당하였다. 5.2.1절과 5.2.2절에서는 실험을 통하여 NCC-NUMA 시스템에 가장 적합한 캐쉬 라인 크기 및 캐쉬 블록 할당 정책을 찾아내고, 5.2.3절에서는 NCC-NUMA 시스템과 NUMA시스템의 성능을 비교 분석한다.

5.2.1 캐쉬 라인 크기에 따른 speedup

본 시스템에 가장 적합한 캐쉬 라인 크기를 찾기 위해서 캐쉬 라인 크기를 32 byte, 64 byte로 변경하면서 성능을 측정하였다. 표 3을 보면 캐쉬 라인 크기로 64 byte를 사용한 것이 32 byte를 사용한 것보다 speedup이 더 좋다. 이것은 캐쉬 라인 크기가 64 byte인 경우, 한 번의 원격 메모리 접근으로 더 많은 데이타를 읽어올 수 있어서, 원격 공유메모리 접근시간이 캐쉬 라인 크기가 32 byte인 경우보다 작아지기 때문이다.

캐쉬 라인 크기를 128 byte로 할 경우에 대한 분석은 다음과 같다. NCC-NUMA 카드의 LC3는 SCI specification을 따라서 64 byte의 데이타 패킷을 지원하고 있다. 따라서, 본 시스템에서 128 byte의 캐쉬 라인 크기를 지원하기 위해서는 1회의 원격 공유 메모리 접근 명령을 2개의 SCI 트랜잭션을 사용해서 처리해야 한다. 또한, 2개의 SCI 트랜잭션이 하나의 트랜잭션처럼 처리되도록 보장해주는 프로토콜을 구현해야 한다. 따라서, 캐쉬 라인 크기를 128 byte로 확장할 경우, 오버헤드가 많이 발생하여 성능향상이 매우 작거나, 감소할 것으로 예상된다. 따라서, 본 시스템에서 가장 적당한 캐쉬 라인 크기는 64 byte로 판단된다.

5.2.2 캐쉬 블록 할당 정책에 따른 speedup

본 절에서는 캐쉬 블록 할당 정책을 direct-mapped, 2-way set associative, 4-way set associative로 변경하면서 성능을 측정하였다. 표 4를 보면 4-way set associative, 2-way set associative, direct mapped 방식의 순으로 성능이 좋은 것을 알 수 있다. Set associative에서는 direct-mapped보다 캐쉬 블록에서 conflict가 발생할 확률이 작기 때문에 set associative 방식이 높은 캐쉬 hit ratio를 가지며, 공유 메모리 접근 시간이 감소한다. 또한, 실험결과에서 direct-mapped, 2-way set associative, 4-way set associative의 순서대로 speedup의 차이는 좁아지고 있으며, 2-way set associative와 4-way set associative의 성능차는 그리 크지 않다. 그리고, 8-way set associative의 경우, 4-way set associative에 비해 성능 향상은 그리 크지 않을 것이나, 구현되는 하드웨어는 훨씬 복잡해질 것으로 예상된다. 따라서, 본 시스템에 가장 적당한 캐쉬 블록 할당 정책으로 4-way set associative를 선택하였다.

5.2.3 NCC-NUMA 시스템과 NUMA 시스템 비교

본 연구의 NCC-NUMA 시스템과 Dolphin사의 PCI-SCI 카드에 기반한 NUMA 시스템의 성능을 비교하였으며, 결과는 표 5, 표 6 그리고 그림 6에 나와 있다. NCC-NUMA 시스템의 캐쉬 라인 크기는 64 byte이며, 캐쉬 블록 할당 정책은 4-way set associative이다. 표 5의 실험결과를 보면, LU와 Barnes에서는 NCC-NUMA 시스템이 NUMA 시스템보다 훨씬 높은 성능향상을 보여주며, Ocean과 FFT의 경우에도 크지는 않지만 NCC-NUMA 시스템이 NUMA 시스템보다 높은 성능향상을 보여주고 있다.

LU에서 NCC-NUMA 시스템의 speedup은 노드수가 증가할수록 증가한다. 그러나, 2노드 NUMA 시스템은 1노드 NUMA 시스템보다 느리며, 4노드 NUMA 시스템은 1노드 NUMA 시스템과 거의 비슷한 성능을 보여준다. 이것은 그림 6에서 보는 바와 같이 노드수가 증가할수록 원격 공유 메모리 접근 시간이 급격히 증가하며, 2, 4노드에서 전체 수행 시간에 대한 원격 공유 메모리

표 3 캐쉬 라인 크기에 따른 speedup

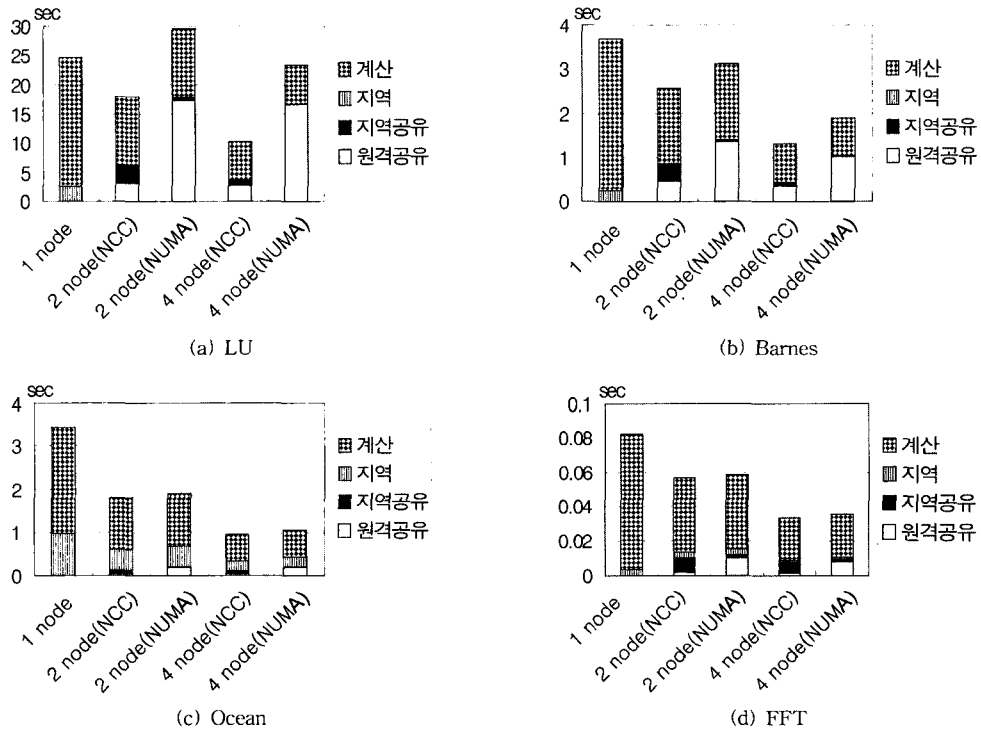
	LU		Barnes		Ocean		FFT	
노드수	2	4	2	4	2	4	2	4
32 byte	1.15	1.98	1.11	2.13	1.29	2.39	1.39	2.33
64 byte	1.33	2.31	1.29	2.50	1.38	2.56	1.57	2.62

표 4 캐쉬 블록 할당 정책에 따른 speedup

	LU		Barner		Ocean		FFT	
노드수	2	4	2	4	2	4	2	4
direct-mapped	1.33	2.31	1.29	2.5	1.38	2.56	1.57	2.62
2-way	1.36	2.38	1.31	2.59	1.39	2.57	1.58	2.63
4-way	1.37	2.42	1.33	2.62	1.39	2.58	1.58	2.63

표 5 NCC-NUMA 시스템과 NUMA 시스템의 speedup

	LU		Barnes		Ocean		FFT	
노드수	2	4	2	4	2	4	2	4
NCC NUMA	1.37	2.42	1.33	2.62	1.35	2.58	1.57	2.63
NUMA	0.83	1.06	1.10	1.81	1.30	2.35	1.53	2.51
NUMA에 대한 NCC NUMA의 성능향상	39 %	56 %	18 %	31 %	4 %	9 %	3 %	5 %



(a) LU (b) Barnes

(c) Ocean (d) FFT

그림 6 계산 시간과 메모리 접근 시간

접근 시간의 비율이 각각 58%, 71%를 차지하기 때문이다. Barnes의 경우도, LU와 유사하게 2, 4노드에서 전체 수행 시간에 대한 원격 공유 메모리 접근 시간의 비율이 각각 43%, 54%를 차지하기 때문에, 노드수 증가에 따른 NUMA 시스템의 성능향상이 크지 않다. 반면에 NCC-NUMA 시스템의 원격 공유 메모리 접근 시간은 네트워크 캐쉬사용으로 인하여 많이 증가하지 않는다. 또한, NCC-NUMA 시스템에서 지역 공유 메모리를 접근할 때 발생하는 오버헤드는 네트워크 캐쉬사용으로 인한 이득보다 훨씬 작다. 이것때문에 NCC-NUMA 시스템의 성능이 NUMA 시스템보다 많이 좋아진다.

Ocean과 FFT에서도 NCC-NUMA 시스템의 원격 공유 메모리 접근시간은 NUMA 시스템보다 많이 향상된 것을 알 수 있다. 그러나, NCC-NUMA 시스템의 전체 수행 시간은 NUMA 시스템보다 조금 좋다. 이것은, 그림 6과 같이 Ocean과 FFT는 LU나 Barnes처럼 노드수가 증가하여도 NUMA 시스템의 원격 공유 메모리 접근시간이 급격히 증가하지 않기 때문이다. 또한, NCC-NUMA 시스템에서 지역 공유 메모리를 접근할 때 발생하는 오버헤드가 네트워크 캐쉬 사용으로 인한 이득을 많이 상쇄시키기 때문에 NCC-NUMA 시스템의 성능향상이 그리 크지는 않다.

NCC-NUMA 시스템에서는 원격 공유 메모리에 대한 접근 회수가 많을수록 성능이 향상되며, 지역 공유 메모리에 대한 접근 회수가 많을수록 성능 향상 폭은 줄어든다. 따라서, NCC-NUMA 시스템의 성능이 NUMA 시스템에 비해서 얼마나 많은 성능 향상이 이루어지는지는 지역 공유 메모리와 원격 공유 메모리에 대한 접근 비율에 의해서 결정된다. 표 6은 지역 공유 메모리 접근 회수의 총합을 1로 했을 때, 원격 공유 메모리 접근 회수의 비율을 나타낸다. LU와 Barnes의 경우, 원격 공유 메모리 접근 회수의 비율이 Ocean과 FFT보다 큰

표 6 지역 공유 메모리 대 원격 공유 메모리 접근 회수의 비율

	LU		Barnes		Ocean		FFT	
노드수	2	4	2	4	2	4	2	4
지역공유	1	1	1	1	1	1	1	1
원격공유	0.96	3	0.96	2.84	0.25	0.33	0.22	0.35

것을 알 수 있다. 따라서, LU와 Barnes에서는 네트워크 캐쉬 사용으로 인한 이득이 매우 크며, 이것은 전체 프로그램의 성능향상을 유도하고 있다. 반면에 Ocean과 FFT에서는 원격 공유 메모리 접근 비율이 낮기 때문에 네트워크 캐쉬 사용으로 인한 이득이 지역 공유 메모리가 PCI 버스상에 위치하는 단점보다 많이 크지 않으며, 성능 향상 폭이 많이 크지 않다.

요약하면, NCC-NUMA 시스템은 LU와 Barnes처럼 원격 공유 메모리 접근 회수가 많은 프로그램에서는 매우 우수한 성능을 보이며, Ocean과 FFT처럼 원격 공유 메모리 접근 회수가 작은 프로그램에서도 NUMA 시스템에 비해서는 좋은 성능을 보여주고 있다. 이것은, 네트워크 캐쉬사용으로 인한 이득이 지역 공유 메모리가 PCI 버스상에 위치하는 단점을 극복하고 시스템의 성능향상을 이끌고 있기 때문이다. 그리고 표 6에서와 같이 노드수가 증가할수록 원격 공유 메모리 접근 비율이 증가하고 있기때문에, 노드수가 8이상이면 NCC-NUMA 시스템의 성능은 NUMA 시스템보다 훨씬 향상될 것이다.

6. 결론 및 향후 연구 과제

고성능 PC 클러스터 시스템을 구축하기 위해서는 네트워크 접근 시간을 최소화하는 것이 중요하다. 본 논문에서는 SCI 기반 PC 클러스터 시스템상에서 네트워크 접근 시간을 최소화하기 위한 네트워크 캐쉬를 제안하고 이에 기반한 NCC-NUMA 카드를 개발하였다. NCC-NUMA 카드는 각 노드의 PCI 슬롯에 plug-in되는 형태이며, 공유메모리, 네트워크 캐쉬, 공유 메모리 제어 모듈 및 네트워크 제어 모듈을 포함한다. 네트워크 캐쉬는 PCI 버스에 존재하는 공유메모리를 캐쉬하며, IEEE SCI 표준에 기반한 캐쉬 일관성 유지 알고리즘을 사용한다.

SPLASH-2 벤치마크를 사용한 시스템 성능 측정 결과, 네트워크 캐쉬 사용으로 인한 이득이 공유메모리가 PCI 버스상에 위치함으로서 발생하는 단점을 극복할 수 있다는 것을 확인하였고, NCC-NUMA 시스템이 NUMA 시스템보다 우수한 성능을 가짐을 알 수 있었다. NCC-NUMA 시스템에서 공유 메모리가 위치하는 PCI 버스는 앞으로 PCIX 혹은 PCI Express로 대체될 것이다. PCIX 혹은 PCI Express는 고속 데이타 전송을 지원함으로 CPU에서 NCC-NUMA 카드까지의 명령 전달 시간이 감소할 것으로 되기 때문에, NCC-NUMA 시스템의 성능은 더욱 높아질 것이다. 또한, NCC-NUMA 카드에 캐쉬 기능을 지원하지 않는 NUMA 카드로 동작 가능하게 하는 옵션을 제공함으로써, 프로그램 특성에 따라 유연하게 사용할 수 있게 하는 것이 필요하다.

참 고 문 헌

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

[2] IEEE Standard for Scalable Coherent Interface (SCI), *IEEE Computer Society*, August 1993.

[3] H. Ong and P. A. Farrell, "Performance Comparison of LAM/MPI, MPICH, and MVICH on a Linux Cluster connected by a Gigabit Ethernet Network," *Proceedings of the 4th Annual Linux Showcase & Conference*, Atlanta, Georgia, USA, October 2000.

[4] Sang-Hwa Chung, Soo-Cheol Oh, Se-Jin Park, Han-Kook Jang, Chi-Jung Ha, "A CC-NUMA Prototype Card for SCI-Based PC Clustering," *Proceedings of IEEE International Conference on Cluster Computing*, Nov. 2000.

[5] Sang-Hwa Chung, Soo-Cheol Oh, "An SCI-Based PC Cluster Utilizing Coherent Network Cache," *Cluster Computing*, Vol. 6, Issue. 2, pp. 153-159, Apr. 2003.

[6] http://www.dolphinics.no/dolphin2/interconnect/index.html

[7] Georg Acher, Wolfgang Karl, and Markus Leberecht, "The TUM PCI/SCI Adapter," *Scalable Coherent Interface/SCI, Architecture and Software for High-Performance Compute Clusters, LNCS State-of-the-Art Survey*, October 1999.

[8] 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.

[9] M. Schulz, J. Tao, C. Trinitis, and W. Karl, "SMiLE: An Integrated, Multi--Paradigm Software Infrastructure for SCI-based Clusters," *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, Berlin, Germany , May, 2002.

[10] Emmanuel Cecchet, "Memory Mapped Networks: A New Deal for Distributed Shared Memories? The SciFS Experience," *IEEE International Con-*

ference on Cluster Computing (CLUSTER'02), September, 2002.

[11] P. Keleher, S. Dwarkadas, A.L. Cox, and W. Zwaenepoel, "TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems," *Proceedings of the Winter 94 Usenix Conference*, pp. 115-131, January 1994.

[12] Jeffrey Kuskin, David Ofelt, Mark Heinrich, John Heinlein, Richard Simoni, Kourosh Gharachorloo, John Chapin, David Nakahira, Joel Baxter, Mark Horowitz, Anoop Gupta, Mendel Rosenblum, and John Hennessy, "The Stanford FLASH Multiprocessor," *Proceedings of the 21st Annual International Symposium on Computer Architecture*, 1994.

[13] Anant Agarwal, Ricardo Bianchini, David Chaiken, Kirk L. Johnson, David Kranz, John Kubiatowicz, Beng-Hong Lim, Kenneth Machkenize, and Donald Yeung, "The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor," *MIT/LCS Memo TM-454, Messachusetts Institute of Technology*, 1991.

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

[15] http://www-1.ibm.com/servers/eserver/xseries/numa/index.html

[16] http://www.plxtech.com

[17] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. "The SPLASH-2 Programs: Characterization and Methodological Considerations," *In Proceedings of the 22nd International Symposium on Computer Architecture*, pp. 24-36, Santa Margherita Ligure, Italy, June 1995.

오 수 철

1995년 부산대학교 컴퓨터공학과 학사 1997년 부산대학교 컴퓨터공학과 석사 1997년~1998 LG전자 멀티미디어 연구소 연구원. 1998년~2003 부산대학교 컴퓨터공학과 박사. 2003년~2004년 (주)아이온테크 연구원. 2004년~현재 부산대학교 BK21 사업단 기금교수. 관심분야는 클러스터 시스템, 병렬처리, CC-NUMA, VOD

정 상 화

정보과학회논문지 : 시스템 및 이론
제 31 권 제 6 호 참조