실행주도 시뮬레이션에 의한 PC 클러스터 기반 CC-NUMA 시스템 성능분석

(Performance Analysis of PC Cluster-based CC-NUMA System using Execution-driven Simulation)

하 치 정 † 정 상 화 †† 오 수 철 †††
(Chi-Jung Ha) (Sang-Hwa Chung) (Soo-Cheol Oh)

요 약 본 논문에서는 PC 클러스터 기반 CC-NUMA 시스템을 제안하고, 시뮬레이션을 통하여 성능을 분석하였다. PC 클러스터 기반 CC-NUMA 시스템은 PC의 PCI slot에 CC-NUMA 카드를 장착함으로써 구현되며 공유메모리, 네트워크 캐쉬, 네트워크 제어 모듈을 포함한다. CC-NUMA 시스템은 PCI 버스상에 존재하는 메모리를 공유대상으로 하며, 공유메모리와 네트워크 캐쉬사이의 일관성은 IEEE SCI 표준에 의해 유지된다. CC- NUMA 시스템을 시뮬레이션 하기 위해 실행주도 시뮬레이터인 Limes를 수정하여 사용하였으며, 캐쉬 일관성 유지 알고리즘으로 SCI의 typical set을 구현하였다. 또한 기존 시스템과의 비교를 위해서 네트워크 캐쉬를 활용하지 않는 Dolphin사의 PCI-SCI 카드에 기반한 NUMA 시스템을 시뮬레이션 하였다. CC-NUMA 시스템의 성능을 측정하기 위하여 다양한 실험을 수행하였으며, 실험결과 CC-NUMA 시스템이 NUMA 시스템에 비해서 성능향상이 우수함을 알 수 있었다. 또한, CC-NUMA 시스템이 최적의 성능을 발휘하는 파라미터의 값을 도출하였으며, 이를 CC-NUMA 시스템의 실제 구현에 반영하였다.

Abstract This paper presents a PC cluster-based CC-NUMA system and evaluates the performance of the system using simulation. The CC-NUMA system is based on a CC-NUMA card which is plugged into PCI slot. The CC-NUMA card contains shared memory, network cache, and network control module. The CC-NUMA system shares memory on PCI bus. The coherence between the shared memory and the network cache is maintained by the IEEE SCI standard. To simulate the CC-NUMA system, Limes, an execution-driven simulator, is used with a modification. The cache coherence algorithm is based on the SCI's typical set. For comparison purpose, a NUMA system using the Dolphin's PCI-SCI card, which does not utilize network cache, is also simulated. Various experiments are performed to measure the performance of the CC-NUMA system. According to the experiments, the CC-NUMA system shows considerable improvements compared with the NUMA system. The parameters obtained during experiments are reflected by the real implementation of the CC-NUMA system.

1. 서 론

최근에 사회의 모든 분야에서 이루어지는 정보화와 인터넷의 급속한 보급으로 웹서버, 전자상거래서버, VOD 서버 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.

PC 기반 클러스터가 중대형 컴퓨터에 필적하는 단일 컴퓨팅 시스템으로서의 성공 여부는 PC를 연결하는 네

· 본 연구는 한국과학재단 목적기초연구(2000-2-30300-002-3) 지원으로 수행되었음.
† 비 회 원 : 삼성전자 연구원
chha@hyowon.pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
shchung@hyowon.cc.pusan.ac.kr
††† 비 회 원 : 부산대학교 컴퓨터공학과
osc@hyowon.pusan.ac.kr
논문접수 : 2000년 9월 20일
심사완료 : 2001년 3월 14일

트워크 성능과 지원소프트웨어에 달려 있다. 클러스터 시스템을 위한 네트워크로는 메시지 패싱에 기반한 Fast Ethernet, Myrinet 등과 분산공유메모리에 기반한 SCI를 들 수 있다. LAN용으로 개발된 Fast Ethernet은 100Mbps의 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 구성비용이 높다. 또한, Fast Ethernet과 Myrinet이 사용하는 메시지 패싱 방식은 빈번한 데이터 복사와 OS 시스템 call로 인한 소프트웨어 부하가 크기 때문에 네트워크의 물리적 성능을 저하시킨다. 이러한 소프트웨어 부하를 최소화하기 위해서 메시지 패싱 과정에서 OS의 개입을 배제한 사용자 수준 인터페이스가 개발되었다. 대표적인 시스템으로 AM[2], FM[3], U-Net[4]이 있으며, 최근에는 Myrinet을 위한 사용자 수준 인터페이스로 GM[5]이 개발되었다.

SCI(Scalable Coherent Interface : ANSI/IEEE standard 1596-1992)[6]는 1μs 이하의 낮은 지연시간과 1GByte/sec의 대역폭을 가지는 네트워크로, point-to-point 연결에 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 가능하므로 클러스터 시스템에 가장 적합한 네트워크이다. SCI는 상용 CC-NUMA 서버인 IBM의 NUMA-Q[7], Data General의 Aviion[8]에 의해서 그 성능이 입증되었다.

SCI에 기반한 클러스터 시스템으로 Dolphin사의 PCI-SCI 카드[9]를 사용한 독일 Paderborn대학[10]의 PC 클러스터 시스템과 LRR- TUM[11]의 Smile 프로젝트에서 자체 개발한 PCI-SCI 카드를 사용한 PC 클러스터 시스템 등이 있다. 이들 시스템은 네트워크 캐쉬를 활용하지 않는 NUMA 방식을 채택하고 있으며, CC-NUMA 기능을 지원하지 않는다.

본 논문에서는 SCI를 사용하는 PC 클러스터 기반 CC-NUMA 시스템을 제안하고 시뮬레이션을 통하여 성능을 분석하였다. PC 클러스터 기반 CC-NUMA 시스템의 핵심적인 모듈은 PCI 버스에 plug-in 되는 CC-NUMA 카드이다. CC-NUMA 카드는 공유메모리, 네트워크 캐쉬, 네트워크 제어모듈로 구성되며 SCI의 캐쉬 일관성 유지 알고리즘을 사용한다. 시뮬레이션 시스템은 정확한 멀티프로세서 시뮬레이션을 수행할 수 있는 실행주도(execution-driven) 방식의 시뮬레이터인 Limes[12]를 사용하여 구현하였다. Limes 멀티프로세서 시뮬레이터는 노드를 구성하는 각각의 하드웨어 모듈을 소프트웨어 모듈로 프로그램하여 실제 시스템과 비슷한 시스템을 시뮬레이션 할 수 있다.

본 논문에서는 비교를 위해서 네트워크 캐쉬를 사용하지 않는 Dolphin사의 PCI-SCI 카드에 기반한 NUMA 시스템을 시뮬레이션 하였으며, 실험결과 CC-NUMA 시스템의 성능이 우수함을 확인하였다. 그리고 제안된 시스템에서 최적의 성능을 낼 수 있는 하드웨어 parameter를 찾는 실험도 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서는 PC 클러스터 기반 CC-NUMA 시스템에 대해서 설명한다. 3장에서는 실행주도 멀티프로세서 시뮬레이터인 Limes를 소개한다. 4장에서는 구현된 시뮬레이션 시스템과 실험을 통하여 제시한 CC-NUMA 시스템의 성능을 평가한다. 마지막 5장에서는 결론으로 구성한다.

2. PC 클러스터 기반 CC-NUMA 시스템

본 논문에서 제안하는 PC 클러스터 기반 CC-NUMA 시스템의 구조는 <그림 1>과 같다. CC-NUMA 카드는 PC의 PCI slot에 plug-in 되는 형태이며, 공유 메모리 제어 모듈, 공유메모리, 네트워크 캐쉬, 네트워크 제어 모듈 및 SCI 디렉토리로 구성된다. PC는 시스템 버스에 새로운 디바이스를 장착하는 것이 구조적으로 불가능하기 때문에 CC-NUMA 카드는 PCI 버스상에 위치한다. 또한, PCI 버스상에 위치한 CC-NUMA 카드는 시스템 버스를 snoop할 수 없으므로 지역 메모리(local memory)를 공유할 수 없다. 이를 해결하기 위해서 본 시스템은 CC-NUMA 카드상에 공유 메모리를 위치시킨다.

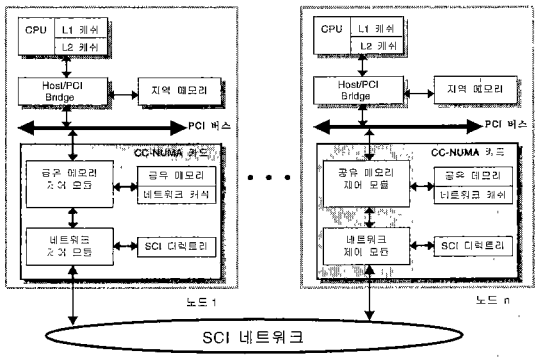

그림 1 PC 클러스터 기반 CC-NUMA 시스템

본 시스템의 지역메모리는 지역 CPU에서만 캐쉬 가능하며 다른 노드와 공유하지 않는다. CC-NUMA 카드에 있는 공유메모리는 PCI 버스의 특성 때문에 지역 CPU에서 캐쉬하지 않으며, 원격 노드의 CC-NUMA 카드에 있는 네트워크 캐쉬에 의해서 캐쉬된다. 각 노드에 존재하는 공유메모리들은 하나의 PCI global address space로 매핑되며 각 노드는 동일한 PCI address를 가지고 공유메모리에 접근한다. SCI 디렉토리는 SCI에 기반한 공유리스트 정보를 저장한다. 공유메모리 제어 모듈은 지역 및 원격 CPU에서 전달된 공유메모리에 대한 read/write transaction을 처리하며, SCI의 캐쉬 일관성 유지 알고리즘을 사용하여 공유메모리와 네트워크 캐쉬 사이의 캐쉬 일관성을 유지시키는 역할을 한다. 또한, 캐쉬 일관성 유지 작업시 필요한 원격 노드와의 transaction 교환은 네트워크 제어 모듈을 통하여 이루어진다.

네트워크 제어 모듈은 SCI 디렉토리를 관리하며, 공유메모리 제어 모듈 및 원격 노드에서 전송된 read/write/invalidation transaction을 처리한다. 예를 들어, CPU가 PCI 버스상에 위치한 공유메모리에 대한 read transaction을 발생시키면, Host/PCI bridge는 PCI 버스상에 read transaction을 발생시킨다. CC-NUMA 카드는 PCI read transaction의 주소가 공유메모리 주소로 판단되면, CC-NUMA 카드에서 데이터를 읽어서 응답을 하게 된다. 이때, 공유메모리 혹은 네트워크 캐쉬상에 유효한 데이터가 있으면 read transaction에 바로 응답을 한다. 유효한 데이터가 없다면, 네트워크 제어 모듈을 통하여 원격 노드로 데이터의 전송을 요청한다. 요청한 데이터가 도착하면 공유메모리, 네트워크 캐쉬, SCI 디렉토리정보를 갱신하고 read transaction에 응답을 한다.

3. Limes 멀티프로세서 시뮬레이터

멀티프로세서 시뮬레이터는 프로세서의 역할을 하는 참조 생성기(reference generator)와 메모리를 시뮬레이션 하는 메모리 시스템 시뮬레이터(memory system generator)의 연결 방식에 따라서 기록주도(tracedriven), 프로그램주도(program-driven), 실행주도(execution-driven)방식으로 구분할 수 있다. 기록주도 시뮬레이션은 참조 생성기가 목적 애플리케이션을 수행시켜 메모리 접근 기록을 생성한 후, 그 기록을 메모리 시스템 시뮬레이터의 입력으로 사용하여 시뮬레이션을 수행한다. 프로그램주도 시뮬레이션은 참조 생성기가 목적 애플리케이션 수행 중에 메모리 접근이 발생하면 직접 메모리 시스템 시뮬레이터를 수행시켜 시뮬레이션을 수행한다. 따라서, 프로그램주도 시뮬레이션은 기록주도 시뮬레이션에 비해 좀더 실제 시스템에 가까운 시뮬레이션을 할 수 있다. 실행주도 시뮬레이션은 프로그램주도 시뮬레이션과 유사하며, 애플리케이션 프로그램을 수정하여 메모리 접근이 발생하는 부분에 메모리 시스템 시뮬레이터를 수행시키는 명령어를 추가하는 것이 차이점이다.

프로그램주도 시뮬레이터의 예로는 Rochester 대학의 MINT[13]가 있으며, 실행주도 시뮬레이터의 예로는 Stanford대학의 TangoLite[14], 그리고, MIT의 Proteus[15]가 있다. 그러나, 이러한 시뮬레이터는 MIPS나 Sparc 시스템을 기본환경으로 구현되어 본 연구에서 수행하고자 하는 PC 클러스터 시스템의 기반이 되는 Pentium 시스템의 시뮬레이션에는 적합하지 않다. PC 환경을 기반으로 하는 실행주도 시뮬레이터로는 Illinois 대학의 Augmint[16]나 Belgrade대학의 Limes[12]가 있으며, 본 논문에서는 하드웨어 구조와 유사하게 시스템을 시뮬레이션 할 수 있는 Limes를 사용하였다.

Limes는 단일 프로세서 상에서 i486/P5+에 기반한 멀티프로세서 시스템을 시뮬레이션 할 수 있다. 시뮬레이션을 구동할 수 있는 기본 애플리케이션으로 SPLASH-2 suite[17]을 지원하며, parallelism을 표현하는 ANL (Argonne National Lab) 매크로를 사용하는 모든 애플리케이션을 수행할 수 있다.

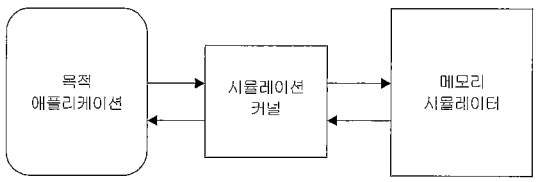

그림 2 Limes 시뮬레이터 구조

Limes는 목적 애플리케이션, 커널, 메모리 시뮬레이터의 세 가지 모듈로 구성된다. <그림 2>에서는 목적 애플리케이션, 커널, 메모리 시뮬레이터의 관계를 보여준다. 목적 애플리케이션은 Limes에서 수행되는 병렬 애플리케이션 프로그램이다. 메모리 시뮬레이터는 캐쉬, 버스, 메인 메모리 등 시스템에 존재하는 메모리와 관련된 모듈을 시뮬레이션 한다. Limes 메모리 시뮬레이터는 기본으로 SMP 방식의 메모리 모델을 시뮬레이션 한다. 커널은 시스템에 있는 CPU를 시뮬레이션 한다. 커널은 목적 애플리케이션에서 실행될 코드를 읽어서 수행하며, 메모리 접근이 발생하는 경우 메모리 시뮬레이

터를 수행하여 시뮬레이션을 진행한다

4. 시뮬레이션

4.1 시뮬레이션 대상 시스템

본 논문에서 제안한 PC 클러스터 기반 CC-NUMA 시스템을 시뮬레이션하기 위해서 각 노드가 <그림 3>의 형태를 가지는 시뮬레이션 모델을 설정하였다. Limes의 메모리 시뮬레이터는 SMP 형태의 메모리 모델을 지원하고 있기 때문에 본 논문에서 제안하는 CC-NUMA 시스템을 시뮬레이션 하기 위해서 메모리 시뮬레이터를 수정하였다. 이를 위해서 기존 메모리 시뮬레이터의 L1 캐쉬 및 지역 메모리 모듈을 활용하였으며, SMP를 지원하는 시스템 버스 모듈을 수정하여 단일 CPU를 장착한 PC 노드의 시스템 버스 모듈로 사용하였다. 또한, 기존의 메모리 시뮬레이터에는 존재하지 않지만 PC 시스템에는 존재하는 L2 캐쉬, PCI/Host bridge, PCI 버스 및 CC-NUMA 시스템의 핵심 부분인 CC-NUMA 카드 모듈을 추가하였다.

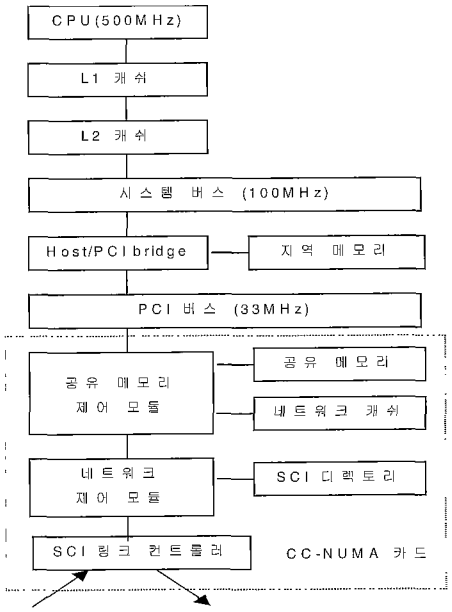

그림 3 PC 클러스터 기반 CC-NUMA 시스템

또한, 본 논문에서는 CC-NUMA 시스템과의 성능 비교를 위해 <그림 4>와 같이 Dolphin사의 PCI-SCI 카드에 기반한 NUMA 시스템을 시뮬레이션으로 구현하였다. 이를 위해서 CC-NUMA 시스템에서 CC-NUMA 카드 모듈을 제거하고, PCI-SCI 카드 모듈을 추가하였다. PCI-SCI 카드는 Dolphin사의 PCI-SCI 카드를 기준으로 시뮬레이션 모델을 작성하였으며, PCI/SCI bridge와 SCI 링크 인터페이스로 구성된다. PCI/SCI bridge는 지역 CPU의 원격 메모리 접근 요구 및 원격 CPU의 지역 메모리 접근 요구를 처리한다. SCI 링크 인터페이스는 CC-NUMA와 동일하다.

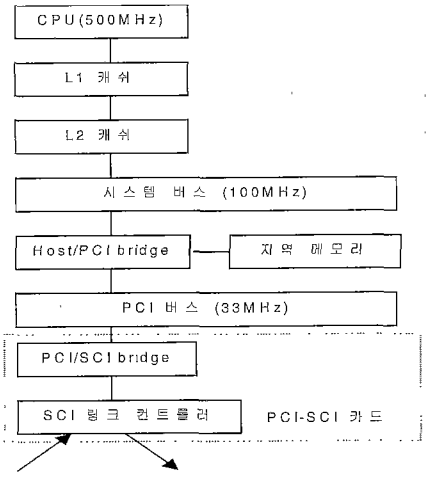

그림 4 NUMA 시스템

4.2 시뮬레이션 parameter

시뮬레이터에서 사용한 parameter는 <표 1>과 같다. PC 시스템과 관련된 CPU, 캐쉬, 지역 메모리, 버스 등에 관한 parameter는 Pentium 기반 PC를 기준으로 설정하였다. CC-NUMA 카드와 관련된 parameter는 본 연구진이 현재 구현중인 CC-NUMA를 기준으로 하였으며, PCI-SCI 카드와 관련된 parameter는 Dolphin사의 PCI-SCI 카드를 기준으로 설정하였다. L1 캐쉬는 2 way set-associative 방식이고, 크기는 16KBytes이며, 캐쉬 line size는 32bytes 이다. L2 캐쉬는 4 way set-associative 방식이고, 크기는 256KBytes이며, 캐쉬 line size는 32bytes 이다. 각 노드의 공유 메모리의 크기는 32MBytes이고, 네트워크 캐쉬의 크기는 32KBytes이다. SCI point-to-point link delay는 SCI 네트워크에 연결된 노드수에 따라서 링크 컨트롤러에서 발생할 수 있는 충돌을 고려하였다.

시뮬레이션 대상 프로그램은 SPLASH-2 suite[17]의 OCEAN, FFT, LU, RADIX, WATER-SPATIAL을 사용하였으며, 공유메모리는 각 노드에 같은 크기로 분

표 1 시뮬레이션에서 사용한 parameter

Parameter	Value
Process speed	500MHz
System bus speed	100MHz
PCI bus speed	33MHz
L1 cache access time	4ns
L2 cache access time	18ns
Local memory access time	50ns
Shared memory access time in local CC-NUMA card	1.2μs
Shared memory access time in remote CC-NUMA card	8.0μs
Shared memory access time through Dolphin's PCI-SCI card	4.02μs
SCI point-to-point link delay (Single ring)	400ns(2node) 600ns(4node) 700ns(8node) 750ns(16node)
SCI point-to-point link delay (Dual ring)	400ns(2node) 500ns(4node) 550ns(8node) 575ns(16node)

할하여 분산시켰다. CC-NUMA 시스템의 경우, 지역 메모리(local memory), 공유 메모리(shared memory), 원격 메모리(remote memory)에 대한 접근이 발생한다. 이때, 다른 노드와 공유가 전혀 발생하지 않는 데이터는 공유 메모리가 아닌 지역 메모리에 위치시킨다. 이러한 상황을 고려하기 위해 지역 메모리와 공유 메모리의 접근 비율을 1:1로 하였다. 이 비율은 메모리 접근에서 발생할 수 있는 지역 메모리의 접근 비율을 최소화시킨 것이다.

4.3 실험 결과

<표 1>에서 제시한 parameter를 사용하여 CC-NUMA 시스템과 NUMA 시스템을 시뮬레이션 하였다. 실험에 사용된 SPLASH-2 suite 애플리케이션의 problem size를 정리하면 <표 2>와 같다.

표 2 SPLASH-2 suite 애플리케이션의 problem size

Appli-cation	Type	Problem size
Ocean	Study of ocean movements	34*34 ocean grid
FFT	FFT computation	16K complex doubles
Radix	Radix sort	16K integer keys, radix 64
LU	Blocked dense linear algebra	64*64 matrix, 16*16 blocks
Water-Spatial	Study of forces and potentials of water molecules in a 3-D grid	64 molecules

4.3.1 네트워크 캐쉬 line size에 따른 성능

표 3 Single ring에서 네트워크 캐쉬 line size에 따른 speed-up

Node \	Ocean		FFT		Radix		LU		Water-Sp	
	64 byte	128 byte	64 byte	128 byte	64 byte	128 byte	64 byte	128 byte	64 byte	128 byte
2	3.5%	3.4%	2.0%	1.9%	2.0%	1.5%	1.8%	1.9%	1.4%	1.5%
4	2.8%	2.6%	2.1%	1.9%	2.4%	2.0%	2.3%	2.7%	1.6%	2.0%
8	2.9%	2.9%	1.4%	1.1%	3.6%	3.0%	3.8%	4.1%	1.5%	1.9%
16	2.9%	2.5%	1.2%	1.1%	2.8%	2.2%	2.6%	2.9%	0.8%	0.9%

<표 3>은 네트워크 캐쉬 line size가 32byte인 경우를 기준으로 하여 64byte, 그리고 128byte 일 때의 speed-up을 보여준다.

네트워크 캐쉬 line size가 32byte에서 64byte로 증가할 때와 32byte에서 128byte로 증가할 때의 speed-up은 거의 유사한 것을 볼 수 있다. 이것은 네트워크 캐쉬 line size가 64byte에서 128byte로 증가할 때 성능 향상이 saturation되는 것을 보여준다. 그 이유는 SCI 네트워크의 전송 frame은 64byte와 256byte를 지원하지만, 128byte를 보내기 위해서는 256byte의 frame을 사용해야 함으로 부하가 발생되기 때문이다.

따라서, 본 CC-NUMA 시스템에서 최적의 네트워크 캐쉬 line size는 64byte임을 알 수 있다.

4.3.2 Single ring과 dual ring의 비교

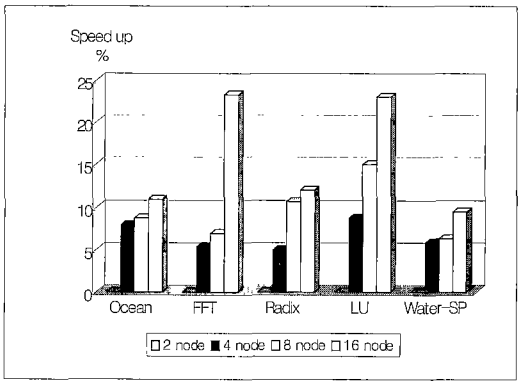

그림 5 Single ring에 대한 dual ring의 speed-up

<그림 5>는 SCI 네트워크가 single ring인 경우를 기준으로 하여 dual ring인 경우의 speed-up을 보여준다. 네트워크 캐쉬 line size는 앞장의 시뮬레이션 결과에 따라 64byte를 사용했다. 2노드 시스템에서는 single ring과 dual ring이 transaction 전송에 필요한 평균 hop수가 동일하기 때문에 성능은 거의 유사하다. 그러나, 노드수가 증가하면 single ring에 대한 dual ring의 평균 hop수의 비율이 감소하기 때문에 single ring에 대한 dual ring의 speed- up이 증가된다. 노드수가 8개 이하인 경우는 dual ring 적용으로 인한 성능 향상이 15%를 넘지 않으므로 가격 대 성능비를 고려한다면 single ring을 적용하는 것이 좋으며, 16노드 이상이 되면 성능 향상이 커지므로 dual ring을 적용하는 것이 유리하다고 판단된다.

4.3.3 CC-NUMA 시스템과 NUMA 시스템의 성능 비교

CC-NUMA 시스템과 NUMA 시스템의 성능을 비교 분석하였다. CC-NUMA 시스템의 네트워크 캐쉬 line size는 최적의 캐쉬 line size인 64byte를 사용했으며, SCI 네트워크는 single ring을 사용하였다.

표 4 CC-NUMA 시스템과 NUMA 시스템의 수행시간 (단위 : ms)

Node	Ocean		FFT		RADIX		LU		Water-SP	
	CC-NUMA	NUMA	CC-NUMA	NUMA	CC-NUMA	NUMA	CC-NUMA	NUMA	CC-NUMA	NUMA
2	906	1014	1001	1116	1369	1429	783	974	795	1057
4	535	577	605	680	778	909	572	701	581	730
8	517	546	431	473	625	705	618	725	579	703
16	545	564	458	496	664	736	674	762	596	702

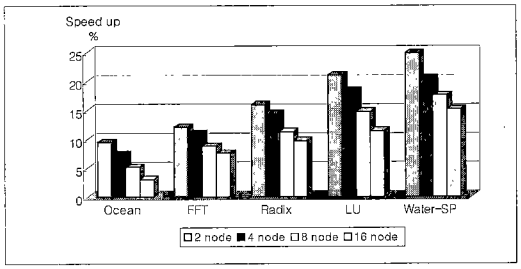

그림 6 NUMA 시스템에 대한 CC-NUMA 시스템의 speed-up

표 5 네트워크 캐쉬 hit ratio

Node	Ocean	FFT	Radix	LU	Water-SP
2	95.4%	96.6%	98.2%	99.2%	99.90%
4	95.3%	96.6%	97.5%	99.0%	99.85%
8	95.1%	96.6%	96.8%	98.6%	99.80%
16	94.9%	96.3%	96.5%	98.3%	99.77%

<표 4>는 각 애플리케이션의 수행시간을 나타내며, <그림 6>은 이를 기준으로 하여 NUMA 시스템에 대한 CC-NUMA 시스템의 성능향상을 나타낸다. <표 5>는 CC-NUMA 시스템의 네트워크 캐쉬 hit ratio를 보여준다. <그림 6>에서 보면 CC-NUMA 시스템이 NUMA 시스템에 비해 전체적으로 좋은 성능을 보이고 있음을 알 수 있다. NUMA 시스템에서는 네트워크 캐쉬를 지원하지 않음으로 매번 새로운 원격 메모리 접근 시 네트워크 traffic을 발생시킨다. 이와 달리 CC-NUMA 시스템은 캐쉬 일관성을 유지하는 부하가 있지만, 네트워크 캐쉬가 적당한 hit ratio를 유지하면 네트워크 traffic을 대폭 줄일 수 있기 때문에 전체 시스템의 성능이 향상된다. 또한, 공유메모리가 PCI 버스 상에 위치함으로써 발생되는 부하를 네트워크 캐쉬를 통하여 극복할 수 있다는 것을 보여준다. 노드수가 증가함에 따라 성능 향상 폭이 줄어드는 이유는 네트워크 캐쉬의 hit ratio가 줄어드는 것과, 캐쉬 일관성 유지를 위한 부하가 증가되는 결과로 볼 수 있다. Ocean, FFT, Radix, LU, Water-SP의 순서대로 성능향상이 높은 것은 <표 5>에서 나타난 바와 같이 네트워크 캐쉬 hit ratio가 높은 순서이다.

CC-NUMA 시스템의 경우, 지역 메모리, 공유 메모리, 원격 메모리에 대한 접근이 발생하는데, 다른 노드와 공유가 전혀 발생하지 않는 데이터는 공유 메모리가 아닌 지역 메모리에 위치시켜야 한다. 이러한 상황을 고려하기 위해 지역 메모리와 공유 메모리의 접근 비율을 1:1로 하였지만 실제는 지역 메모리 접근 비율이 더 높다고 예상된다. 이 비율이 높으면 실제 CC-NUMA 시스템의 speed-up이 증가되는 요인이 된다. 따라서, 본 논문에서 제안한 CC-NUMA 시스템을 실제로 구현한다면 시뮬레이션 결과보다 더 좋은 성능을 나타낼 것으로 예상한다.

5. 결 론

본 논문에서는 PC 클러스터 기반 CC-NUMA 시스템을 제안하고, 시뮬레이션을 통하여 성능을 분석하였

다. CC-NUMA 카드는 각 노드의 PCI slot 에 plug-in 되는 형태이며, 공유메모리, 네트워크 캐쉬, 네트워크 제어 모듈을 포함한다. 네트워크 캐쉬는 PCI 버스에 존재하는 공유메모리를 캐쉬하며, IEEE SCI 표준에 기반한 캐쉬 일관성 유지 알고리즘을 사용한다. 제안된 CC-NUMA 시스템의 성능 분석을 위하여 Limes의 메모리 시뮬레이터를 수정하여 시뮬레이션을 수행하였다. 실험결과 CC-NUMA 시스템이 네트워크 캐쉬를 활용하지 않는 NUMA 시스템에 비해서 전체적으로 좋은 성능을 가짐을 알 수 있었으며, 공유메모리가 PCI 버스 상에 위치함으로서 발생하는 부하를 네트워크 캐쉬를 통하여 극복할 수 있다는 것을 확인하였다. 또한, 파라미터를 변경시키며 실험을 수행한 결과, 8노드 이하의 SCI 네트워크를 형성할 때, 네트워크 캐쉬 line size가 64byte이고 single ring을 사용할 경우, CC-NUMA 시스템이 최적의 성능을 가짐을 알 수 있었다. 본 연구진은 캐쉬 일관성 유지 알고리즘으로 SCI의 typical set을 사용하고 실험에서 도출된 파라미터를 적용한 CC-NUMA 시스템을 현재 구현중에 있다.

참 고 문 헌

[1] http://www.myrinet.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. 60-73, 1997.

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

[5] Myricom, Inc. The GM API. White Paper. Myricom, Inc., 1997. http://www.myri.com/GM/doc/gm_toc.html

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

[7] http://www.sequent.com/whitepapers/numa_ arch. html

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

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

[10] http://www.uni-paderborn.de/pc2/

[11] Wolfgang Karl, Markus Leberecht, 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.

[12] Davor Magdic, "Limes: A Multiprocessor Simulation Environment for PC Platforms," IEEE TCCA Newsletter, March 1997.

[13] Jack E. Veenstra, Robert J. Fowler, "MINT Tutorial and User Manual, Technical Report 452, The University of Rochester, Computer Science Department, August 1994.

[14] Herrod, S.A., "TangoLite: Introduction and User's Guide," technical report, Stanford University, Stanford USA, November 1993

[15] Brewer, Eric A. Proteus: a high-performance parallel-architecture, MIT. Lab for Computer Science MIT/LCS/TR 516, September 1991.

[16] A-T. Nguyen, M. Michael, A. Sharma, J. Torrellas, "The Augmint Multiprocessor Simulation Toolkit for Intel x86 Architectures," Proceedings of 1996 International Conference on Computer Design, October 1996.

[17] Woo S. C., Ohara M., Torrie E., Pal Singh J., Gupta A., "The SPLASH-2 Programs: Characterization and Methodological Considerations," Proceedings of the 22nd ISCA, pp. 24-36, June 1995.

하 치 정

1999년 부산대학교 컴퓨터공학과 학사. 2001년 부산대학교 컴퓨터공학과 석사. 2001년 ~ 현재 삼성전자 연구원. 관심분야는 클러스터 시스템, 병렬처리, SCI, CC-NUMA

정 상 화

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

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