■ 2000 정보과학 논문경진대회 수상작

Myrinet 상에서 VMMC를 기반으로 하는 효율적인 MPI 구현

(An Efficient Implementation of MPI over VMMC for Myrinet)

김 호 중 † 맹 승 렬 ††

(Ho-joong Kim) (Seung Ryoul Maeng)

요 약 클러스터 시스템의 성능을 향상시키기 위해서는 Myrinet과 같은 고성능 통신망 인터페이스가 필수적이다. 그러나 Myrinet에서 동작하는 저수준 통신 계층들은 각기 고유한 통신 방식을 사용하므로 호환성이 떨어진다. 따라서 MPI와 같은 통신 프로그래밍 표준을 효율적으로 구현하여 응용프로그램 수준에서 고성능과 호환성을 동시에 제공하여야 한다.

본 논문에서는 VMMC 통신 계층을 기반으로 MPI를 구현하였다. VMMC의 직접 저장 방식은 MPI의 Send/Recv 방식을 지원하기에 적합하지 않지만 본 논문에서는 두 가지 통신 방식을 변환하기 위한 송수신 큐 구조를 설계하고 늦은 위치 갱신, 선택적 무복사 전송 등의 최적화 기법을 적용함으로써 높은 전송 성능을 얻는다. MPI-VMMC의 최대 전송 대역폭은 90.7Mbytes/sec이며 이는 VMMC 통신 계층의 최대 전송 성능의 95%에 달한다.

Abstract Cluster systems employ high speed interconnection networks and use efficient communication layers to gain high performance and scalability. But the diversity in implementation mechanism among these communication layers causes lack of portability. A solution is to provide communication standard APIs such as MPI.

This paper introduces MPI-VMMC: an MPI implementation on VMMC. Though the direct deposit transfer mechanism used in VMMC is not suitable for Send/Recv mechanism used in MPI, the proposed sub-layer laid between MPI and VMMC efficiently translates from one mechanism to the other. We also use the lazy pointer and selective zero-copy transfer technique to gain high performance.

The peak performance of MPI-VMMC is 90.7Mbytes/sec, which is about 95% of the base communication layer's.

1. 서 론

클러스터 시스템(cluster system)은 PC나 워크스테이션과 같은 저가의 범용 컴퓨터들을 고속의 통신망(network)으로 연결한 시스템이다. 클러스터 시스템은 적은 비용으로 고성능의 시스템을 구성할 수 있으며 확장성이 뛰어나 병렬 처리 응용프로그램을 수행하기에 적합하다.

클러스터 시스템 상에서 수행되는 병렬 처리 응용프로그램은 일반적으로 다른 호스트 상의 프로세스들과 통신한다. 따라서 클러스터 시스템의 성능을 높이기 위해서는 호스트 프로세서의 성능 뿐 아니라 통신 성능을 높이는 일이 중요하다.

최근의 클러스터 시스템은 통신 성능을 높이기 위하여 Myrinet과 같은 고성능 통신망 하드웨어를 사용하며, 통신망 하드웨어의 성능을 최대한 활용할 수 있는 FM[1], BIP[2], VMMC[3] 등의 효율적인 통신 계층(communication layer)을 채택한다. 이러한 통신 계층

· 이 연구는 국가지정연구실 사업의 지원을 받는다.

† 비 회 원 : 한국과학기술원 전산학과
hjkim@camars.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전산학과 교수
maeng@cs.kaist.ac.kr

들은 공통적으로 커널(kernel)의 간섭을 배제한 사용자 수준 통신(user-level communication)을 제공하며, 메모리 복사의 회수를 줄이고 송수신 프로토콜을 최적화하여 메시지 전송 성능을 높인다.

그러나 이러한 통신 계층들은 비교적 저수준에서 동작하며, 통신 성능을 높이기 위하여 제각기 다른 통신 방식과 고유한 메커니즘(mechanism)을 사용하므로 응용프로그램 수준에서의 호환성이 떨어진다. 이 문제를 해결하기 위하여 MPI(Message Passing Interface)[4] 등의 통신 프로그래밍 표준을 상위 통신 계층에 구현하는 방법을 주로 사용한다. 이 때 상위 통신 계층의 오버헤드로 인하여 응용프로그램 수준에서 얻는 성능이 크게 감소할 수 있으므로, 클러스터 시스템에서 동작하는 응용프로그램이 높은 성능을 얻기 위해서는 고성능의 저수준 통신 계층을 기반으로 하여 통신 프로그래밍 표준을 효율적으로 구현하여야 한다.

본 논문에서는 VMMC 통신 계층을 기반으로 하여 MPI를 구현한다. VMMC는 직접 저장(direct deposit) 방식을 사용하여 통신함으로써 수신자 프로세서의 부하를 제거한 통신 계층이다. 따라서 VMMC는 Myrinet 상에서 동작하는 통신 계층 중 매우 우수한 메시지 전송 성능을 보인다. 그러나 직접 저장 방식은 MPI에서 사용하는 Send/Recv 방식과 다른 개념의 통신 방식이므로, VMMC 상에 MPI를 구현하기 위해서는 두 계층 사이에서 통신 방식을 서로 변환하여야 하는 오버헤드가 생긴다.

본 논문에서 구현한 MPI-VMMC는 송수신 큐 형태의 부계층을 이용하여 MPI의 Send/Recv 통신 방식과 VMMC의 직접 저장(direct deposit) 방식의 메시지를 서로 변환한다. 전송 성능을 개선하기 위하여 늦은 위치 갱신(lazy pointer) 방법[5]을 사용하여 송수신 큐 관리의 오버헤드를 줄이며, 메시지 크기에 따라 송신자와 수신자에서 무복사 전송(zero-copy transfer)을 선택적으로 수행한다. 이러한 최적화 기법들을 사용한 결과 MPI-VMMC는 최대 95.7Mbytes/sec의 전송 대역폭을 얻으며 이는 VMMC 통신 계층의 최대 전송 성능의 95%에 달한다.

2. 관련연구

2.1 Myrinet

Myrinet은 LAN 규모의 클러스터 시스템을 구성하는 스위치 방식의 고성능 통신망 하드웨어이다. Myrinet 통신망 인터페이스는 호스트와 통신망을 연결하는 역할을 담당하며 <그림 1>과 같이 RISC 프로세서와 DMA 엔진, SRAM 메모리로 구성된다. 통신망 인터페이스는 사용자가 프로그래밍 할 수 있도록 설계되어 있으므로, 사용자의 요구에 따라 다양한 메커니즘을 구현할 수 있다. Myrinet 상에서 동작하는 FM, BIP, VMMC 등의 통신 계층은 제각기 다른 통신 방식을 사용하여 구현되었으며, 따라서 다양한 성능을 나타낸다.

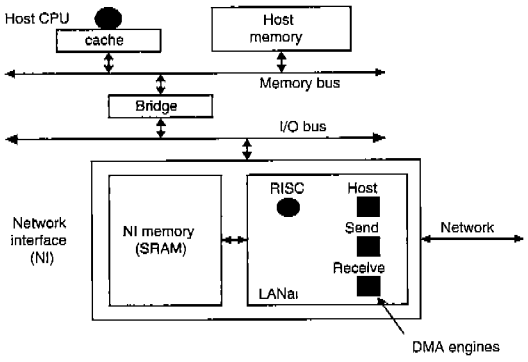

그림 1 Myrinet 통신망 인터페이스

2.2 저수준 통신 계층

기존의 TCP/IP와 같은 통신 계층은 프로토콜 자체의 부하가 매우 크며 프로토콜 내부에서 여러 차례 메시지를 복사함으로써 전송 성능이 매우 낮다[6, 7]. 따라서 Myrinet 과 같은 고성능 통신망 인터페이스 상에서 동작하는 통신 계층들은 가급적 메모리 복사의 수를 줄이고 커널의 간섭을 배제하며 간단하게 구현된 통신 프로토콜을 통신망 인터페이스에서 수행함으로써, 호스트 프로세서의 부하를 줄이고 통신망 하드웨어의 성능을 최대한 활용한다.

2.2.1 FM

FM(Fast Messages)은 작은 크기의 메시지를 효율적으로 전송하도록 설계된 통신 계층이다. FM은 PIO (programmed I/O)를 사용하여 메시지를 송신자 호스트로부터 통신망 인터페이스로 전달하므로 작은 메시지를 전송하는 경우 DMA 방식에 비하여 높은 성능을 얻는다. 그러나 PIO 방식은 호스트 프로세서의 간섭을 필요로 하므로 커다란 메시지를 전송하는 경우 DMA 방식에 비하여 최대 전송 성능이 낮다[8].

FM 2.x는 gather/scatter 기능을 지원하므로 MPI 헤더와 같이 작은 메시지를 효율적으로 보낼 수 있다. 따라서 FM 상에 구현된 MPI-FM은 FM 통신 계층의 기

본 성능에 근접하는 높은 성능을 얻는다.

2.2.2 BIP

BIP(Basic Interface for Parallelism)는 간단한 기능과 고성능을 목표로 개발된 통신 계층이다. BIP는 랑데뷰(rendezvous) 방식으로 통신하여 메시지를 무복사 전송한다. 또한 통신 계층을 간단하게 구현함으로써, Myrinet 상에서 동작하는 통신 계층 중 가장 높은 메시지 전송 성능을 얻는다.

그러나 메모리 보호(protection)를 제공하지 않는 등, BIP의 통신 환경은 상위 통신 계층 구현에 적합하지 않다. 따라서 MPI 구현시 성능 저하가 비교적 크다.

2.2.3 VMMC

VMMC(Virtual Memory-Mapped Communication)는 직접 저장 방식으로 메시지를 전송하는 통신 계층이다. 직접 저장 방식은 송신자가 수신자의 메모리 주소를 지정하여 메시지를 전송하는 방식이다. 수신자 내에서의 메시지 전송은 통신망 인터페이스가 담당하며 호스트 프로세서가 메시지 수신에 관여하지 않는다. 따라서 수신자 호스트에 부하가 걸리지 않으므로 통신 성능을 높일 수 있다.

<그림 2>는 VMMC의 직접 전송 방식을 나타낸다. 응용프로그램의 관점에서는 수신자의 원격 메모리(remote memory) 영역이 송신자의 지역 메모리(local memory)에 사상(mapping)되어 있는 것으로 나타나며, 메시지 전송은 지역 메모리 영역으로부터 원격 메모리 영역으로의 메모리 복사와 같은 형태로 이루어진다. 그러나 실제로는 원격 메모리 영역은 송신자의 물리적 메모리(physical memory)에 존재하지 않으며, 이곳으로의 메시지 전달은 송신자의 통신망 인터페이스에 의해 번역되어 수신자의 통신망 인터페이스로 전달된다. 수신자 통신망 인터페이스에서는 받은 메시지를 메모리에 저장한다.

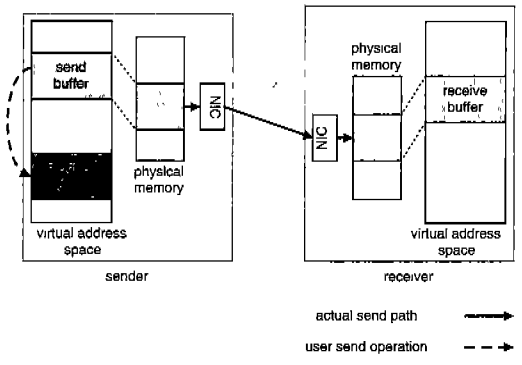

그림 2 VMMC에서의 메시지 전송

직접 저장 방식은 미리 지정된 수신 영역으로만 메시지를 전송하므로 MPI와 같은 일반적인 Send/Recv 통신 방식에는 적합하지 않다. VMMC에서 임의의 수신 영역으로 메시지를 전송하기 위해서는 우선 메시지를 지정된 수신 영역으로 보낸 후 일 회 복사하여 해당 영역으로 전달하여야 한다. 이러한 메모리 복사는 메시지 전송 성능을 떨어뜨리는 주된 요인이다. 또한 통신망 인터페이스의 DMA 엔진을 사용하여 전송하므로 송신 영역과 지정 수신 영역을 물리적 메모리 상에 고정(pin)하기 위한 작업이 필요하다.

VMMC-2는 무복사 Send/Recv 통신 방식을 제공하기 위하여 전송 방향 재지시(transfer redirection) 방법을 사용한다. 전송 방향 재지시는 메시지가 수신자 노드에 도착하기 전에 수신 명령이 먼저 수행(post)되는 경우 수신 버퍼 주소를 통신망 인터페이스에 미리 알려줌으로써 <그림 3>의 (b)와 같이 지정된 수신 영역으로의 메모리 복사를 제거하는 방법이다

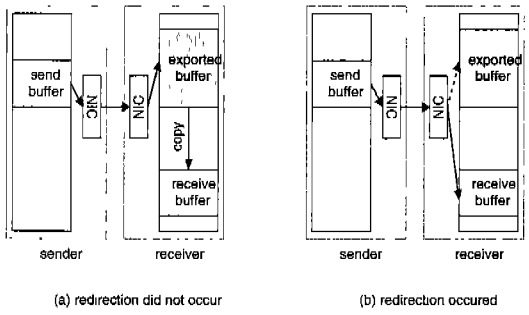

그림 3 전송 방향 재지시

그러나 전송 방향 재지시를 사용하여 메모리 복사를 제거하는 경우에도 전송 방향 재지시 명령 자체의 부하가 비교적 크다. 수신 버퍼 주소를 I/O 버스를 통해 호스트 프로세서로부터 통신망 인터페이스 프로세서로 알려주어야 하며, 수신 버퍼 영역이 메모리에 고정되지 않은 경우 이를 고정하기 위한 오버헤드가 발생한다. 이처럼 VMMC의 메시지 전송 방식은 MPI의 Send/Recv 통신 방식을 효율적으로 지원하기에 적합하지 않다.

3. 설계 및 구현

본 논문에서는 VMMC를 기반으로 MPI를 구현한다. MPI 1.1 표준을 지원하는 구현인 MPICH[9]의 NT 버전을 바탕으로 하여 기존의 TCP/IP로 구현된 채널 인터페이스(channel interface) 계층을 논문에서 제안하는

Send/Recv 부계층으로 교체한다.

MPI 1.1 표준은 네 가지 점대점 송신 방식을 정의하며 각각에 대하여 블록킹/논-블록킹의 두 가지 명령이 있다. 또한 점대점 수신 역시 블록킹/논-블록킹 방식의 두 가지 명령이 있다. 또한 이러한 열 개의 점대점 통신 명령 외에도 그룹 통신을 위한 다양한 명령들이 존재한다.

그러나 MPICH/NT를 비롯한 대부분의 실제 MPI 구현에서는 점대점 통신 명령들을 사용하여 그룹 통신 명령들을 구현한다. 또한 열 개의 점대점 통신 명령들 역시 하위 통신 계층의 논-블록킹 Send/Recv 명령을 공통적으로 사용하므로, 채널 인터페이스 계층을 구현하는 것만으로 모든 MPI 통신 명령을 지원할 수 있다.

앞서 언급한 바와 같이 VMMC는 Send/Recv 방식 지원에 어려움이 있다. 그러나 통신 계층 자체의 기본 성능이 우수하므로, Send/Recv 명령을 효율적으로 설계하여 오버헤드를 줄임으로써 높은 성능의 MPI를 구현할 수 있다.

3.1 송수신 큐의 구조

본 논문에서는 MPI-VMMC를 구현하기 위하여 VMMC 계층과 MPI 계층 사이에서 메시지 전송 방식을 변환하는 부계층을 설계하였다. 이 부계층은 한 쌍의 송수신 큐와 형태로 존재한다.

<그림 4>는 MPI-VMMC 송수신 큐의 구조이다. n개의 노드로 구성된 클러스터 시스템의 각 노드는 n-1개의 송신 큐와 n-1개의 수신 큐를 가지며 각각의 큐는 점대점으로 대응된다. 수신 큐는 수신자의 물리적 메모리 영역에 존재하며 송신 큐는 반출(export)된 수신 큐를 송신자의 물리적 메모리 영역 외부에 반입(import)하여 만들어진다. MPI 송신 명령은 송신 큐로의 입력(enqueue) 명령으로 변환되며, 실제로는 송신 큐의 입력단(tail)이 가리키는 수신 큐의 물리적 메모리 주소로 VMMC 전송 명령을 수행한다. MPI 수신 명령은 수신 큐 출력단(head)에 해당하는 물리적 메모리 주소로부터 메시지를 읽어 온다(dequeue). 통신망 인터페이스에는 하나씩의 송신 및 수신 DMA 엔진이 존재하므로 하나의 큐 내에서의 모든 메시지 전송은 순차화(serialize)된다.

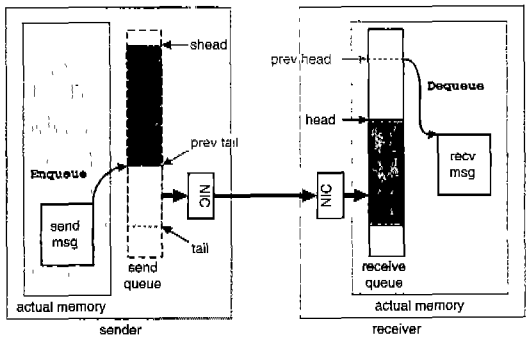

그림 4 MPI-VMMC 송수신 큐

VMMC에서는 메시지 도착을 알리는 방법으로서 폴링(polling)과 인터럽트(interrupt)를 사용할 수 있으나 하나의 수신 버퍼에 대하여 한 가지 방법만 사용할 수 있다. 메시지가 도착할 때마다 인터럽트가 발생하면 문맥 전환에 의한 부하가 매우 커지므로 본 구현에서는 폴링을 사용한다.

3.2 성능 향상 기법

3.2.1 큐 관리의 최적화

송수신 큐가 서로 다른 두 노드에 존재하므로 이들의 입력단/출력단 값의 일관성을 유지하여야 한다. 큐의 내용이 바뀔 때마다 상대방 노드의 큐 정보를 갱신하는 방법은 하나의 메시지 전송 시 두 개의 큐 관리 메시지를 추가로 발생시키므로 통신망의 성능을 크게 저하시킨다.

본 구현에서는 늦은 위치 갱신(lazy pointer) 방법을 사용하여 큐 관리 메시지의 수를 감소시킨다. 늦은 위치 갱신은 송신자 측에서 출력단의 값을 캐싱(caching)하는 방법이다. 메시지 송신 시에 송신자는 큐가 가득 찼는지 확인하기 위하여 입력단(tail)과 출력단(head)의 값을 비교해야 하는데, 출력단의 값은 수신자 노드가 갱신하므로 매번 이 값을 수신자로부터 얻어야 한다. 그러나 <그림 5>의

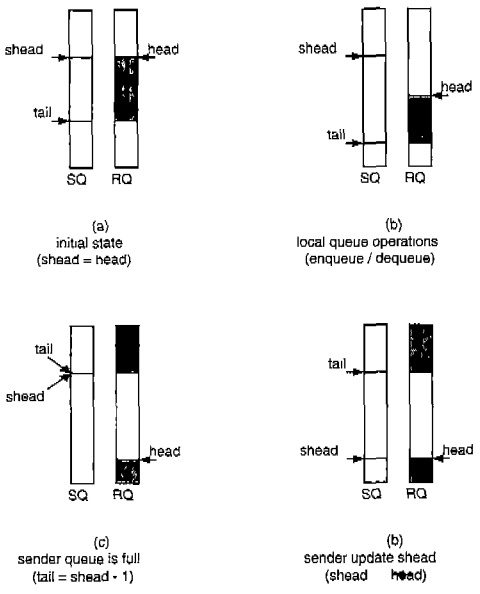

그림 5 늦은 위치 갱신

(a)에서 얻어낸 출력단 값을 가상 출력단(shead)에 캐싱하여 저장하고 송신자 내에서 (b)와 같이 큐 작업을 수행하다가 (c)처럼 송신 큐가 가득 차면 가상 출력단 값을 실제 출력단의 값으로 갱신하는 방법을 사용함으로서 메시지 교환 회수를 크게 줄일 수 있다.

한편 수신자는 메시지 수신 시에 큐가 비어 있는지 검사하기 위해서 입력단의 값을 필요로 한다. 따라서 입력단 값 없이도 큐가 비어 있는지 검사할 수 있다면 입력단 갱신 메시지가 필요하지 않다. 수신자의 경우 메시지 번호와 태그를 이용하여 현재 메모리에 저장된 메시지의 타당성(validity)을 검사한다. 현재 출력단이 가리키는 메시지가 타당하지 않다면 큐가 비어있는 상태이다.

3.2.2 전송 방향 재지시의 선택적 사용

직접 저장 방식은 수신자의 지정된 기본 버퍼로만 메시지를 전송할 수 있다. VMMC에서 제공하는 전송 방향 재지시는 이러한 한계를 해결하고 최종 수신 영역으로 메시지를 직접 전송할 수 있도록 한다. 그러나 전송 방향 재지시는 메시지가 도착하기 전에 사용하여야 성능 향상을 얻을 수 있다. 또한 명령 자체의 부하가 크다. 측정 결과 전송 방향 재지시 수행에 소요되는 시간은 약 9.1μsec정도이며 이는 본 실험 환경에서 약 2300bytes 정도의 메모리를 복사하는 데 걸리는 시간과 같다. 따라서 2300bytes보다 작은 메시지를 전송하는 경우 기본 버퍼로부터 일 회 복사하는 것보다 전송 방향 재지시를 사용하여 무복사 전송할 때 더 많은 시간이 소요된다. 본 논문에서는 메시지 크기에 따라 전송 방향 재지시를 선택적으로 사용하여 수신 성능을 높인다.

3.2.3 MPI 헤더 전송 방식

MPI 헤더는 비교적 작은 크기의 메시지로서 데이타 메시지와 함께 전송되어야 한다. 그런데 일반적으로 MPI 헤더는 데이타와 서로 연속되지 않는 메모리 영역에 위치한다. gather/scatter를 지원하는 FM과 달리 VMMC는 메모리 복사 형태의 메시지 전송 방식을 취하므로 하나의 메시지로는 하나의 연속된 메모리 영역만을 보낼 수 있다. 따라서 VMMC에서는 MPI 헤더를 별도로 전송해야 한다. 더군다나 VMMC는 작은 메시지의 전송 성능이 낮으므로 이러한 메시지 개수의 증가로 인하여 전송 성능이 크게 감소한다. 따라서 작은 크기의 데이타 메시지는 헤더와 함께 일 회 복사하여 하나의 메시지로 포장하여 전송하는 것이 유리하다. 반면에 메시지의 크기가 커지는 경우 메모리 복사로 인한 오버헤드가 증가하므로 헤더를 별도의 메시지로 보내는 무복사 전송이 유리하다.

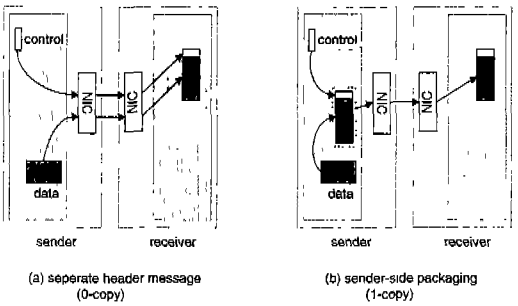

그림 6 MPI 헤더 전송 방식

4. 성능 평가

4.1 구현 환경

본 논문에서는 두 대의 Pentium III 450MHz 컴퓨터를 LANai 4.3 프로세서를 사용하는 Myrinet 통신망으로 연결하여 구성한 Windows NT 4.0 기반의 클러스터 시스템 상에서 MPI-VMMC의 성능을 측정하였다.

4.2 최적화 기법에 의한 성능 향상

제안한 각 최적화 기법으로 인한 성능 향상을 평가하기 위하여 <표 1>의 다섯 가지 모델의 성능을 비교한다.

표 1 MPI-VMMC 성능 평가 모델

	효율적인 큐 관리	송신자 측 무복사	수신자 측 무복사	비고
(1)	NO	NO	NO	최적화 없음
(2)	YES	NO	NO	큐 관리로 인한 성능 향상
(3)	YES	NO	YES	수신자 측 무복사에 의한 성능 향상
(4)	YES	YES	NO	송신자 측 무복사에 의한 성능 향상
(5)	YES	선택적	선택적	모든 최적화 기법을 적용한 MPI-VMMC

(1)은 최적화 기법들을 적용하지 않은 경우이다. (2)는 늦은 위치 갱신, 수신자 측 메시지 타당성 검사 등의 효율적인 큐 관리를 통하여 불필요한 메시지의 전송을 줄인 경우이다. (1)과 (2)의 경우 송신자와 수신자에서 각각 한 차례씩 메모리 복사가 일어난다. (3)은 전송 방향 재지시를 사용하여 무복사 수신하는 경우이다. (4)는 MPI 헤더 메시지를 별도의 메시지로 전송함으로서 무복사 송신하는 경우이다. 마지막으로 (5)는 제안한 모든 최적화 기법들을 사용한 경우이다.

4.2.1 효율적인 큐 관리에 의한 성능 향상

<그림 7>은 늦은 위치 갱신, 수신자 측 메시지 타당성 검사 등의 효율적인 큐 관리 기법을 사용한 경우의 성능 향상을 보여 준다. 메시지 하나 당 필요한 두 개의 큐 관리 메시지를 줄임으로서 전체 메시지 전송 회수를 ⅓로 줄인 결과, 1000bytes 미만의 작은 메시지에서 전송 대역폭이 3배로 증가하였다. 그러나 메시지의 크기가 커질수록 큐 관리 메시지로 인한 오버헤드가 전체 메시지 전송 시간에서 차지하는 비중이 작아지므로 성능 차이가 줄어든다.

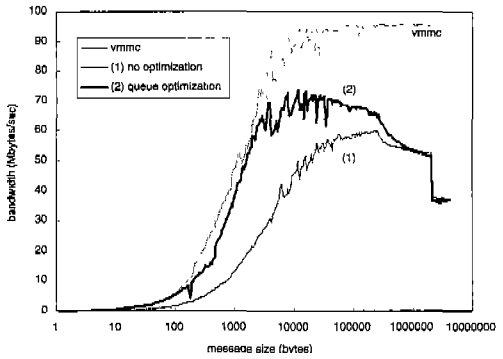

그림 7 효율적인 큐 관리에 의한 성능 향상

4.2.2 전송 방향 재지시에 의한 성능 향상

<그림 8>은 전송 방향 재지시를 사용하는 경우의 성능 향상을 나타낸다. 8Kbytes 이하 크기의 메시지를 전송하는 경우 전송 방향 재지시 자체의 오버헤드로 인하여 전송 대역폭이 오히려 낮아진다. 실제 오버헤드는 약 2300bytes 정도의 메모리 복사에 해당하는 시간이지만 실제로는 전송 방향 재지시를 사용하여도 이미 기본 버퍼로 수신되어 한 차례 복사하여야 하는 경우가 생긴다. 기준 메시지의 크기는 I/O 버스 및 메모리 버스 등의 성능에 따라 달라질 수 있다.

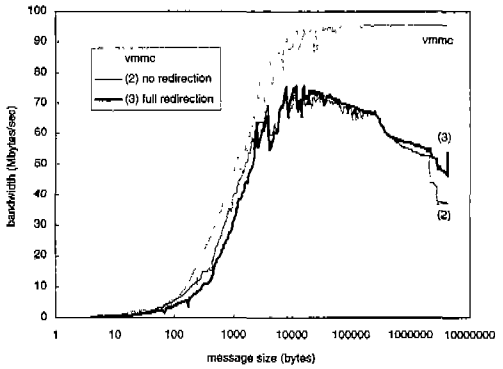

그림 8 전송 방향 재지시에 의한 성능 향상

한편 커다란 메시지를 전송하는 경우에도 전송 방향 재지시로 인한 성능 이득은 그다지 높지 않게 나타난다. 전송 방향 재지시를 사용하면 수신자 호스트에 부하를 주지 않으므로 전송 성능이 송신자에 의하여 결정되기 때문이다. 커다란 메시지를 전송하는 경우 송신자 측에서 메시지를 한 차례 복사하여 보내는 오버헤드가 커지므로 전송 지연 시간의 감소로 인한 성능 향상 효과를 기대할 수 없다.

4.2.3 무복사 송신에 의한 성능 향상

<그림 9>는 MPI 헤더를 전송하기 위해 별도의 메시지로 보내는 경우의 성능 향상을 나타낸다. 작은 메시지를 전송할 때는 통신망 인터페이스에서 메시지를 생성하기 위한 오버헤드가 전송 시간의 대부분을 차지한다. 따라서 작은 메시지 전송 시에는 헤더와 데이타를 하나의 메시지로 포장(package)하여 전송하는 것이 전체 메시지 개수를 줄일 수 있으므로 유리하다. 반면에 커다란 메시지를 전송하는 경우 메모리 복사 오버헤드가 커지므로 헤더를 별도의 메시지로 보내는 무복사 전송이 유리하다. 실험 결과 8bytes의 헤더 메시지를 전송하는 데에 걸리는 시간은 약 8Kbytes의 메모리를 복사하는 시간과 같으며, 이로 인하여 16Kbytes 이하의 메시지 전송 시에는 헤더와 데이타를 한 차례 복사하여 전송하는 경우에 더 높은 성능을 얻을 수 있다.

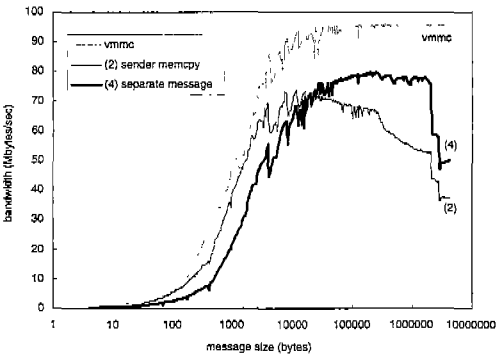

그림 9 무복사 송신에 의한 성능 향상

4.2.4 선택적 무복사 전송에 의한 성능 향상

앞서 살펴본 바와 같이 MPI-VMMC에서 무복사 전송이 항상 우수한 성능을 나타내지는 않는다. 따라서 측정 결과를 바탕으로 <표 2>와 같이 메시지 크기에 따

라 선택적으로 무복사 전송을 수행하도록 하였다.

표 2 메시지 크기에 따른 전송 방식

메시지 크기	송신자측 무복사	수신자측 무복사	효율적인 큐 관리
8Kbytes 이하	1회 복사	1회 복사	YES
8Kbytes ~16Kbytes	1회 복사	무복사	YES
16Kbytes 이상	무복사	무복사	YES

이처럼 세 가지의 최적화 기법을 모두 적용하는 경우 <그림 10>에서 보듯이 송신자 또는 수신자 중 한 쪽에서만 무복사 전송하는 경우에 비하여 커다란 메시지 전송 시 월등히 높은 성능을 나타낸다. 앞서 실험한 결과들은 송신자 또는 수신자 측에서의 메모리 복사 오버헤드로 인하여 VMMC의 성능을 충분히 활용하지 못하였으나, 선택적 무복사 전송 시에는 MPI 헤더 메시지를 보내는 외에 거의 오버헤드가 생기지 않기 때문이다.

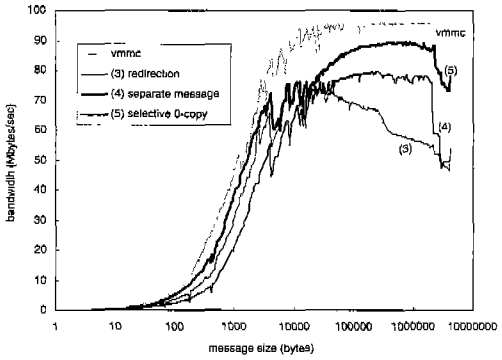

그림 10 선택적 무복사 전송에 의한 성능 향상

한편 <그림 10>에서 2Mbytes 이상 크기의 메시지 전송 시에는 성능이 떨어진다. 이는 본 구현에서 기본 수신 버퍼의 최대 크기를 4Mbytes로 제한하기 때문이다. 따라서 2Mbytes 이상 크기의 메시지를 전송하는 경우 동시에 두 개 이상의 메시지를 전송할 수 없다. 또한 순환 큐(circular queue) 방식으로 큐를 관리하므로 큐의 끝에 걸리는 경우 메시지가 두 개로 나뉘어 전송될 수 있는데 비하여 전송 방향 재지시는 하나의 수신 버퍼에 대하여 하나의 연속된 메모리 영역에 대해서만 수행할 수 있다. 그 결과 하나의 메시지를 전송한 후 수신자 노드에서 전송 방향 재지시가 수행되지 않은 부분을 복사할 때까지 송신자가 기다려야 하므로 전송 성능이 떨어진다. 이 문제를 해결하기 위하여 전송할 메시지가 큐의 끝에 걸려 두 개로 나누어지는 경우 수신 큐에 사용 가능한 연속된 공간이 확보될 때까지 송신자 측에서 메시지 전송을 지연한다.

4.3 MPI-VMMC의 성능 평가

<그림 11>에서 MPI-VMMC의 최대 전송 대역폭은 90.7Mbytes/sec로서, VMMC 통신 계층의 최고 성능인 95.7Mbytes/sec의 약 95%에 달한다. 그러나 중간 크기의 메시지를 전송하는 경우 메모리 복사 및 헤더 전송 오버헤드로 인하여 성능이 낮아진다.

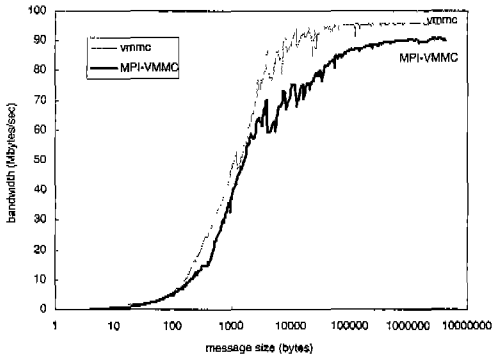

그림 11 MPI-VMMC의 성능 평가

4.4 다른 MPI 구현과의 성능 비교

<그림 12>는 MPI-VMMC와 MPI-FM의 성능을 비교한 것이다. FM은 작은 메시지 전송에 최적화된 통신 계층이므로 작은 메시지를 전송하는 경우 FM이

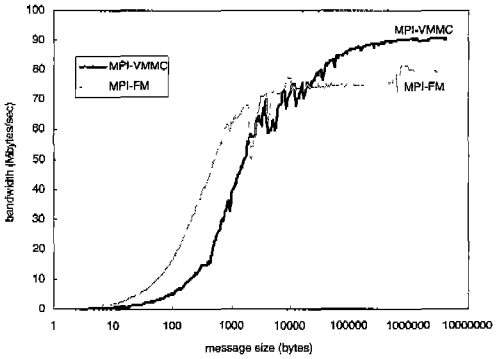

그림 12 MPI-VMMC와 MPI-FM 성능 비교

VMMC보다 높은 성능을 나타낸다. 또한 MPI-FM은 FM의 gather/scatter 기능을 사용하여 MPI 헤더를 효율적으로 전송하므로 MPI-VMMC에 비하여 오버헤드가 적다. 따라서 작은 메시지를 전송하는 경우 MPI-FM을 사용하는 것이 유리하다. 그러나 VMMC의 최대 전송 대역폭이 FM보다 높으며 MPI 계층에서 최소한의 오버헤드로 무복사 전송을 하므로, 16KBytes 이상의 큰 메시지를 전송하는 경우 MPI-VMMC가 MPI-FM에 비하여 우수한 성능을 얻는다.

4.5 MPI-VMMC의 성능 개선 방안

MPI-VMMC는 VMMC 상에 효율적으로 Send/Recv 방식의 통신 계층을 구현함으로써 높은 성능을 얻는다. 그러나 메모리 복사 및 헤더 메시지 전송 오버헤드가 비교적 크다. 이는 VMMC 통신 계층의 특성에 기인한다.

첫째, 작은 메시지의 전송 성능이 낮다. VMMC는 비교적 복잡한 내부 메커니즘을 사용하므로 메시지 당 오버헤드가 크기 때문이다. 따라서 상위 계층에서 메시지 개수를 줄이기 위한 작업을 수행해야 한다.

둘째, VMMC는 메모리 복사 형태의 전송 명령을 제공하므로 MPI 헤더 메시지를 함께 전송하기에 부적절하다. 통신망 인터페이스에서 gather/scatter를 제공하면 별도의 복사 없이 헤더와 데이타를 하나의 메시지로 전송하므로 전송 성능을 개선할 수 있다.

셋째, 전송 방향 재지시 명령의 오버헤드가 크다. 통신망 인터페이스 제어 프로그램을 개선하여 전송 방향 재지시의 수행 시간을 단축하여야 한다.

넷째, 하나의 수신 버퍼에 대하여 동시에 하나의 전송 방향 재지시 만을 수행할 수 있다. 따라서 논-블록킹 수신 명령을 여러 개 수행하는 경우 처음 수행되는 수신 명령 외에는 무복사 수신이 불가능하다.

이러한 VMMC의 단점들이 개선되면 MPI 수준에서 보다 높은 성능을 얻을 수 있게 될 것이다.

5. 결 론

본 논문에서 구현한 MPI-VMMC는 기존의 MPICH/NT에서 사용하는 TCP/IP 기반의 Send/Recv 통신 계층을 VMMC를 기반의 통신 계층으로 대체한 것이다.

VMMC는 전송 성능이 뛰어나지만 MPI의 Send/Recv 통신 방식을 효율적으로 지원하기에는 부적합한 통신 계층이다. 본 논문에서는 늦은 위치 갱신 방법을 사용하여 효율적으로 큐를 관리하고 메시지 크기에 따라 선택적으로 무복사 전송을 함으로써 MPI 수준에서 추가되는 오버헤드를 최소화하였다.

MPI-VMMC는 최대 90.7Mbytes/sec의 전송 대역폭을 얻으며, 이는 VMMC의 최대 전송 대역폭의 95%를 활용하는 효율적인 MPI 구현이다.

참 고 문 헌

[1] M. Lauria, S. Pakin, and A. A. Chien, "Efficient Layering for High Speed Communication: the MPI over Fast Messages (FM) Experience," Cluster Computing, HPDC7 special issue, 1999.

[2] L. Prylli and B. Tourancheau, "Protocol Design for High Performance Networking: a Myrinet Experience," Research Report 97-22, LIP-ENS Lyons, France 1997.

[3] C. Dubnicki, A. Bilas, Y. Chen, S. N. Damianakis, and K. Li, "VMMC-2: Efficient Support for Reliable, Connection-Oriented Communication," In Hot Interconnects V, August 1997.

[4] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard," June 1995.

[5] S. Mukherjee, B. Falsafi, M. D. Hill, and D. A. Wood, "Coherent Network Interfaces for Fine-Grain Communication," In Proc. of the 23rd Int'l Symp. on Computer Architecture, May 1996.

[6] S. Mukherjee and M. D. Hill, "A Survey of User-Level Network Interfaces for System Area Networks," Technical Report UWCS TR @1340, Universitry of Wisconsin-Madison, February 1997.

[7] A. Barak, I. Gilderman, and I. Metrik, "Performance of the Communication Layers of TCP/IP with the Myrinet Gigabit LAN," Computer Communications, Vol.22, No. 11, July 1999.

[8] R. A. F. Bhoedjang, T. Ruhl, and H. E. Bal, "User-Level Network Interface Protocols," IEEE Computer, Vol. 47, No. 11, pp. 53-60, November 1998.

[9] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard," Technical Report, Argonne National Laboratory and Mississippi State University, 1995.

김 호 중

1998년 2월 한국과학기술원 전산학과 학사. 2000년 2월 한국과학기술원 전산학과 석사. 2000년 3월 ~ 현재 한국과학기술원 전자전산학과 전산학전공 박사과정 재학중. 관심분야는 상호연결망, 클러스터 컴퓨팅

맹 승 렬

1977년 2월 서울대학교 공과대학 전자공학과 학사. 1979년 2월 한국과학기술원 전산학과 석사. 1984년 2원 한국과학기술원 전산학과 박사. 1984년 3월 ~ 현재 한국과학기술원 전자전산학과 전산학전공 교수. 관심분야는 컴퓨터구조, 병렬처리, 클러스터 컴퓨팅