메시지 전송계층의 현황

한국과학기술원 손영철 · 김호중 · 맹승렬*

1. 서 론

클러스터 컴퓨팅 기술을 가능하게 한 요인은 저가의 PC 또는 웍스테이션의 빠른 성능 향상과 더불어 이들을 고성능으로 연결할 수 있는 새로운 통신망 하드웨어의 등장이다.

이러한 고성능 통신망 하드웨어로는 ATM, Myrinet, Gigabit Ethernet, Compaq Servernet 등이 있으며, 수 μsec의 전송 지연시간과 Gbps급의 대역폭을 제공하며 매우 빠른 성능 향상을 보이고 있다.

기존의 LAN과 WAN 환경에서는 통신망의 안정성(reliability)문제로 인하여 TCP/IP 와 같이 부하가 큰 통신 프로토콜을 사용하였으나, 통신망 하드웨어의 성능이 향상됨에 따라 통신 프로토콜의 부하가 통신 성능에 커다란 영향을 미치게 되었다. 현재의 통신망 하드웨어는 10μsec 이내의 지연시간을 가지지만 TCP/IP 프로토콜 스택의 부하는 약 100μsec 이상으로 매우 크므로 이러한 고성능 하드웨어에서 사용하기에는 적합하지 않다[5]. 특히 클러스터 시스템과 같이 제한된 환경에서의 통신의 경우, Myrinet과 같이 낮은 에러율을 보장하는 통신망 하드웨어를 사용할 수 있으므로 이에 적합한 통신 소프트웨어가 필요하다

메시지 전송계층은 프로세서와 통신망 인터페이스간의 메시지 전송을 담당하며, 궁극적인 목표는 노드간의 통신성능을 높이고 프로세서의 통신부하를 줄이는 것이다. 현재까지 AM[15], FM[17], BIP[11] 등의 다양한 메시지 전송계층이 발표되었으며 이러한 기술적 발전은 VIA[12]를 통하여 새로운 업계 표준으로 제안되었다. 본 논문에서는 현재까지의 메시지 전송계층들에서 제안된 기술을 종합적으로 살펴보고 앞으로의 발전방향을 서술한다.

2. 고성능 통신망 하드웨어

고성능 통신망 하드웨어 기술의 발전은 클러스터 컴퓨팅 기술을 가능하게 하였다. 대표적인 통신망 하드웨어인 Myricom사의 Myrinet은 양방향으로 최대 1.28Gbps의 통신 대역폭을 보장한다. 특히 Myrinet 통신망 인터페이스 카드는 LANai 프로세서와 이 프로세서에서 접근 가능한 SRAM을 가지고 있으며, LANai 프로세서를 사용자가 프로그램하여 메시지 전송에 필요한 다양한 기능을 수행할 수 있기 때문에 DMA를 위

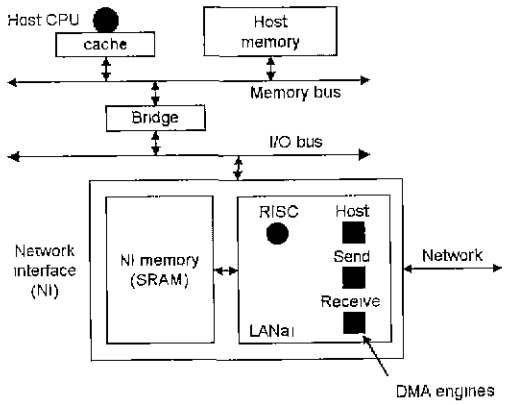

그림 1 Myrinet의 구조[1]

* 종신회원

한 주소변환 등의 작업을 호스트 프로세서의 부하 없이 수행할 수 있다. 이러한 기능은 새로운 메시지 전송계층을 위한 하드웨어 지원을 구현하기에 적합하기 때문에 대부분의 새로운 메시지 전송계층의 개발이 Myrinet에서 이루어졌다. 그림 1은 Myrinet의 통신망 인터페이스 구조이다.

3. 메시지 전송계층

본 절에서는 메시지 전송계층의 역할과 필요한 기능을 살펴보고, 현재까지 제안된 메시지 전송계층의 성능 향상 기법들과 통신 모델을 간략하게 살펴본다.

3.1 메시지 전송계층의 기능

메시지 전송계층은 다음과 같은 기능을 수행하여야 한다.

3.1.1 통신망 하드웨어의 다중화(multiplexing)

다중 프로그래밍 환경에서는 노드내의 단일 통신망 하드웨어를 여러 프로세스가 공유하므로, 하나의 프로세스가 통신을 수행하고 있는 중간에 다른 프로세스가 통신망 인터페이스에 접근할 때에 발생하는 문제를 해결하여야 한다. 이를 위하여 기존의 운영체제에서는 모든 메시지의 송신 및 수신을 커널을 통하여 수행함으로써 여러 프로세스가 동시에 통신망 인터페이스에 접근할 수 없도록 하였다.

BIP, AM 1.0 등의 메시지 전송계층의 경우 통신망을 사용하는 프로세스의 개수를 노드당 하나로 제한하여 다중프로그래밍 환경을 지원하지 않는 방법으로 커널 수행으로 인한 부하를 감소시키는 방법을 사용한다. 현재의 대부분의 메시지 전송계층들은 메시지 버퍼를 다중화함으로써 송수신시에 발생하는 커널 수행 부하를 없애고 있다[5].

3.1.2 통신 프로토콜을 위한 처리

TCP/IP와 같은 통신 프로토콜에서는 여러 단계의 계층을 거치며 패킷을 생성하고 체크섬(checksum)을 계산하는 등의 프로토콜에 필요한 처리를 수행하여야 한다. 그러나 Myrinet과 같이 안정성이 우수하고 복잡한 패킷화가 필요하지 않은 통신망을 사용하는 클러스터 시스템의 경우, 복잡한 프로토콜 스택을 거칠 필요가 없이 흐름제어(flow control), 패킷 생성 등의 간단한 처리만을 수행하면 되므로 이러한 처리과정이 단순화될 수 있다[18].

3.1.3 프로세서와 통신망 인터페이스간의 데이터 전송

프로세스가 원하는 데이터를 통신망 인터페이스에 전달하는 과정으로 PIO(Programmed IO)와 DMA(Direct Memory Access) 방식이 있다. PIO는 CPU가 메모리에 존재하는 데이터를 직접 통신망 인터페이스에 전송하는 방식으로 write combining을 지원하지 않는 프로세서의 경우 워드단위로 데이터를 전송하게 되므로 전송 속도가 느리다. PIO의 경우 시작 지연시간(startup latency)이 없으므로 작은 메시지 전송시 통신 지연시간을 줄이는데 적합하다. DMA의 경우 CPU는 통신망 인터페이스에 DMA 레지스터를 셋업하고, 통신망 인터페이스가 이 정보에 의해 직접 메모리로부터 전송을 수행한다. 일반적으로 통신망 인터페이스와 메모리 간에는 대용량 전송(bulk transfer)이 가능하기 때문에 DMA를 사용하여 큰 메시지를 효율적으로 전송할 수 있다. 하지만 DMA를 수행하기 위해서는 메모리에 존재하는 페이지가 전송이 끝날 때까지 스웝 아웃(swap out)되지 않아야 하기 때문에 스웝이 불가능하도록 메모리를 고정(pinning)하는 작업이 필요하다.

메모리 고정 작업은 비교적 긴 시간을 필요로 하는 시스템 호출(system call)이기 때문에, 대부분의 메시지 전송계층이 DMA 영역을 미리 할당해 두고 이를 기본 버퍼로 사용한다. 즉 CPU가 메모리에서 DMA 영역으로 메시지를 복사한 후 통신망 인터페이스가 DMA 영역에서 데이터를 전송하는 방법을 사용한다. 최근에는 이러한 데이터 복사의 부하를 줄이기 위한 무복사(zero-copy) 통신 방법이 고안되고 있다.

3.1.4 제어신호의 전송

메시지가 통신망을 통해 전송되었을 때 메시지의 도착을 알리는 기능으로 인터럽트(interrupt)와 폴링(polling)이 있다 인터럽트는 일반적으로 처리시간이 길고 부하가 크기 때문에 메시지 전송이 많은 서버의 경우 인터럽트 처리로 인한

CPU 처리시간의 소모가 커지게 된다. 폴링은 CPU가 데이터의 유무를 직접 확인하는 방법으로 메시지가 많은 경우 인터럽트로 인한 부하를 줄일 수 있다. 직접 통신망 인터페이스에 접근함으로써 폴링을 수행할 경우 IO 버스를 거쳐야 하므로 폴링작업으로 인한 시스템 버스의 부하가 문제가 될 수 있다. 최근에 PCI 2.0 에서와 같이 coherent IO를 지원하는 IO 버스에서는 통신망 인터페이스가 특정 메모리 번지에 메시지의 유무를 저장하도록 하고 CPU 는 해당 메모리를 읽도록 하여 매 폴링이 캐쉬 적중(cache hit)되도록 함으로써 폴링의 부하를 줄이고 있다

3.2 메시지 전송계층의 기술발전

현재까지의 메시지 전송계층의 기술적인 발전을 나열하면 다음과 같다

3.2.1 사용자 수준(user-level) 메시지 전송 기법

다중 프로그래밍 환경을 지원하기 위하여 커널이 통신망 인터페이스를 다중화하는 방법은 메시지 송신 및 수신이 발생할 때마다 커널을 거쳐야 하므로 전송 지연시간이 늘어나고 문맥교환(context switch)으로 인한 프로세서의 부하가 발생하는 단점이 있다. 이를 해결하기 위하여 사용자 수준 메시지 전송 기법을 사용한다. 사용자 수준 메시지 전송 기법은 프로세스가 커널의 도움 없이 직접 통신망 인터페이스를 접근하면서 프로세스간의 프로텍션을 보장하는 기법이다.

사용자 수준의 메시지 전송을 지원하기 위하여 제안된 U-Net[10]은 통신하려는 프로세스간에 점대점의 end point를 프로세스의 주소영역에 할당한다. 프로세스는 처음 end point 생성 시에만 커널의 도움을 받는다. 그리고 메시지 전송 시에는 end point에만 접근하며 통신망 인터페이스가 end point로부터 전송할 데이터를 읽거나 쓴다. 따라서 프로세스가 직접 통신망 인터페이스의 레지스터를 접근하지 않으므로 다중화와 프로텍션의 문제가 해결된다. 이와 같이 사용자 수준의 메시지 전송은 메시지의 송신 및 수신에 필요한 메시지 큐(message queue)를 각 프로세스에게 할당하고, 통신망 인터페이스의 컨트롤러가 메시지의 유무를 판단하는 방법으로 구현될 수 있다. 이 때 메시지 큐는 메인 메모리 또는 통신망 인터페이스의 메모리에 위치할 수 있다[5].

사용자 수준 메시지 전달은 작은 메시지의 통신 성능을 높이는데 반드시 필요한 기능이다. 작은 크기의 메시지의 경우 커널 호출에 따른 메시지 당 송수신 오버헤드(per-message overhead)가 전체 통신 부하의 대부분을 차지하기 때문에 이러한 기법을 사용하여 부하를 줄여야 한다[4].

최근의 메시지 전송계층은 대부분 사용자 수준 메시지 전송기법을 사용하며[10, 12, 13, 14, 15, 16, 17] 일부는 다중 프로그래밍 환경을 제공하지 않음으로써[3, 11] 커널 수행으로 인한 부하를 감소시킨다.

3.2.2 무복사 전송

TCP/IP와 같은 통신 계층에서는 프로토콜 스택의 각 계층간에 메시지를 전달하기 위하여 복사가 일어난다. 이러한 메시지 복사는 전송 지연시간을 증가시킬 뿐 아니라 프로세서에 커다란 부하를 주기 때문에, 전송 성능을 저하시키는 중요한 요인이 된다. 예를 들어 Pentium III 500MHz 프로세서의 경우 1Mbytes의 메시지를 복사하는 데 걸리는 시간은 약 5ms로 매우 크다[9] 따라서 계층간의 메시지 복사를 줄이면 메시지 전송 성능을 향상시킬 수 있다.

메시지 복사가 발생하는 경우는 크게 두 가지이다. 첫째, TCP/IP 프로토콜과 같이 계층간에 메시지 전송에 필요한 정보들을 추가하거나 조작하는 과정에서 메시지 복사가 일어난다. 둘째, 사용자 프로세스에서 사용하는 최종 송수신 버퍼 영역과 통신 계층에서 사용하는 송수신 버퍼 영역의 주소가 서로 다름으로 인하여 사용자 프로세스 계층과 통신 계층 사이에 메시지 복사가 일어난다. 전자의 경우 메시지 복사를 줄이기 위해서는 불필요한 메시지 조작을 없애야 한다. 예를 들면 TCP/IP 프로토콜에서 체크섬 검사를 수행하는 작업을 하드웨어 체크섬으로 대체하여 복사를 줄일 수 있다. 후자의 경우에는 통신망 인터페이스가 사용자 프로세스의 최종 송수신 버퍼 영역에 직접 DMA 함으로써 메시지 복사를 제거할 수 있다.

이때 사용자는 송수신할 버퍼 영역의 가상 메모리 주소를 통신 계층에 전달하는 반면 DMA

엔진은 메시지 전송을 위하여 물리적 메모리 주소를 사용하게 된다. 따라서 통신 계층 내에서 가상 메모리 주소와 물리적 메모리 주소 사이의 주소 변환(address translation)이 일어나야 한다. 또한 DMA 도중에는 해당 메모리 영역을 고정하여야 한다. 이러한 주소 변환과 메모리 고정은 커널의 도움을 필요로 하므로 프로세서에 부담을 준다. 따라서 주소 변환과 메모리 고정의 효과적인 구현이 필요하다[5].

이러한 방법으로 프로세서에 의한 메모리 복사를 제거하는 것을 무복사 전송(zero-copy transfer)이라고 한다. 이런 무복사 전송을 위한 여러 가지 방법들에 대해 알아보기로 한다.

가. 주소 사상(address mapping) 변경 기법

일반적으로 운영체제에서는 사용자 프로세스와 커널 프로세스가 사용할 수 있는 메모리 영역을 분리함으로써 보호기능을 제공한다. 따라서 TCP/IP와 같이 커널에 존재하는 통신 계층의 경우 사용자 메모리에서 커널 메모리 영역으로 또는 그 반대로 메시지 복사가 일어난다. 그러나 사용자 메모리를 커널이 접근할 수 있게 함으로써 복사를 제거할 수 있다. Trapeze 통신 계층상에 구현된 TCP/IP 프로토콜인 Trapeze/IP[8]의 경우 하나의 메모리 영역을 사용자 영역과 커널 영역으로 번갈아 사용하여 복사를 하지 않는다. 사용자가 커널에 메시지 전송을 요청하면 커널에서는 커널 버퍼 영역을 사용자 버퍼 영역에 직접 사상(mapping)한다. 따라서 해당 메모리 페이지의 정보를 변경하는 것만으로 커널로의 복사와 동일한 효과를 얻을 수 있다. 마찬가지로 수신자 커널에서는 수신이 끝난 후 수신 버퍼를 사용자 메모리 영역으로 반환한다. 이 때, 페이지 단위로 사상하므로 송수신 버퍼가 페이지 단위로 정렬(align)되어 있어야 한다. 주소 사상 변경 기법에서 주소 변환과 메모리 고정은 송수신 요청 시에 이루어진다.

나. transfer redirection

VMMC-2[14]의 transfer redirection, AM상에 구현된 fast socket 등은 수신자 측의 메시지 복사를 제거하여 메시지 전송 성능을 높이는 방법이다. VMMC-2, AM과 같은 통신 계층들은 사용자 프로세스마다 기본 수신 버퍼 영역을 미리 지정하며, 모든 수신 메시지는 지정된 수신 버퍼에 저장된다. 이 방법은 수신 버퍼 지정 시에 한 차례만 메모리 고정 작업을 수행하므로 커널 호출로 인한 오버헤드를 줄일 수 있다. 또한 지정된 수신 버퍼로만 메시지를 전송할 수 있으므로 메모리 프로텍션을 제공한다.

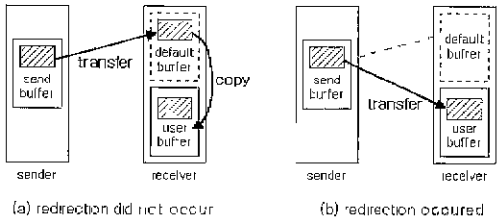

그림 2 VMMC-2의 transfer redirection

그러나 일반적으로 사용자 프로세스에서 사용하는 최종 수신 버퍼는 통신 계층에서 지정한 수신 버퍼와 다른 영역에 위치하므로 지정된 수신 버퍼로부터 최종 수신 버퍼로 한 차례 메시지 복사가 일어난다. redirection은 그림 2와 같이 수신자 측의 메시지 복사를 제거한다. 수신자 프로세스는 수신 명령 수행 시에 원하는 메시지가 이미 기본 수신 버퍼에 도착하였는가 검사한다. 만일 메시지가 아직 도착하지 않은 경우, 메시지를 받을 기본 수신 버퍼의 주소와 이를 대체할 최종 수신 버퍼 주소를 통신 계층에 전달한다. 통신 계층에서는 최종 수신 버퍼 영역의 가상 메모리 주소를 물리적 메모리 주소로 변환하고 메모리를 고정한 후, 메시지가 도착하였을 때 기본 버퍼를 거치지 않고 최종 수신 버퍼 영역으로 직접 DMA 전송한다. 이미 메시지가 도착한 경우에는 프로세서가 메시지를 최종 수신 버퍼 영역으로 한 차례 복사하여야 한다.

redirection을 수행하기 위해서는 메시지가 통신망 인터페이스에 전송되기 이전에 수신명령이 먼저 수행되어야 하며 최종 수신 버퍼 영역에 대해 주소를 변환하고 메모리를 고정해야 한다. 이처럼 redirection 명령 자체의 부하가 비교적 크므로, 작은 크기의 메시지를 수신하는 경우 무복사 전송임에도 불구하고 오히려 수신 성능이 저하될 수 있다[9].

다. 사용자 요청에 의한 메모리 고정

BIP, LFC[3] 등의 통신 계층은 사용자의 요청에 따라 메모리를 고정한다. BIP의 경우 메시지

를 전송할 메모리 영역을 미리 지정하여 커널에게 주소 변환 및 메모리 고정 요구를 보내야 한다. 커널은 메모리 영역 고정 요구에 대하여 물리적 메모리 주소를 사용자 프로세스에게 직접 알려준다. BIP의 무복사 전송은 수신자 기반의 랑데뷰(rendezvous) 방식으로서, 수신자 프로세스가 수신 메모리 영역을 고정한 후 송신자 프로세스에게 메시지 전송을 요청한다.

라. 자동 메모리 고정 (On-demand pinning)

수동 메모리 고정은 메시지를 전송하기 전에 메모리 영역을 사용자가 고정해야 하며, 고정되지 않은 영역으로는 메시지를 전송할 수 없다. 자동 메모리 고정은 이러한 문제를 해결한다.

자동 메모리 고정은 통신망 인터페이스가 필요에 따라 DMA할 메모리 영역을 고정하는 방법이다. VMMC, U-Net/MM, PM[13] 등은 메모리 고정을 위하여 필요한 주소 변환 정보를 소프트웨어 TLB의 형태로 관리한다. 이 방법은 주소 변환 시마다 커널을 호출할 필요가 없으므로 통신 지연시간을 줄일 수 있다.

이 때 사용 가능한 TLB entry의 수는 SRAM의 크기에 의해 제한된다. 이로 인하여 한 번에 고정 가능한 메모리의 크기가 제한되므로, 만일 양방향으로 동시에 대용량의 메시지 전송이 일어나면 서로 메모리를 고정하지 못하고 교착상태에 빠질 수 있다[13].

VMMC-2는 소프트웨어 TLB를 개선하여 UTLB 기법을 제안하였다[14]. UTLB는 TLB entry를 메인 메모리에 저장하며 사용자 프로세스 별로 관리하므로 TLB 크기의 제한이 없다. 또한 통신망 인터페이스 메모리를 UTLB 캐쉬로 사용하여 검색 속도를 높인다. 주어진 가상 메모리 주소에 대한 TLB entry가 캐쉬에 있는지 검색하여 만일 캐쉬에 없으면 메인 메모리의 UTLB를 검색하여 캐쉬를 갱신한다. 메인 메모리 상에도 존재하지 않는 주소는 현재 물리적 메모리에 고정되어 있지 않으므로 커널을 호출하여 메모리를 고정하고 UTLB와 캐쉬를 갱신한다. UTLB를 사용하면 한 페이지에 대하여 한 번의 커널 호출만이 일어나므로 부하가 줄어든다. 그러나 캐쉬 미스가 발생하는 경우 메인 메모리를 검색해야 하므로 지연시간이 증가할 수 있다

3.2.3 기타 기술들

이 외에도, 메시지 전송계층의 최적화를 위하여 메시지의 도착을 프로세서에 알리는 방법인 인터럽트와 폴링의 장점을 결합하여 인터럽트의 발생 비율을 줄이고 폴링의 통신 지연시간 문제를 해결한 polling watchdog[2], 멀티캐스트 기법[3], 메시지의 전송 단계를 나누어 파이프라인(pipeline)형태로 전송하여 성능을 높이는 기법[8] 등이 제안되었다.

3.2.4 Virtual Interface Architecture(VIA)

VIA는 Microsoft, Compaq, Intel을 주축으로 제안된 클러스터를 위한 메시지 전송계층의 업계 표준이다. VIA는 지금까지 제안된 기술들을 표준화한 것으로, U-Net의 end point 구조와 유사한 VI(Virtual Interface)를 통하여 사용자 수준 메시지 전송을 지원하며 UTLB와 유사한 주소 변환을 제공함으로써 무복사 전송을 가능하게 한다. 그림 3은 VIA의 구조이다.

현재 VIA에 대한 다양한 연구가 진행되고 있다.

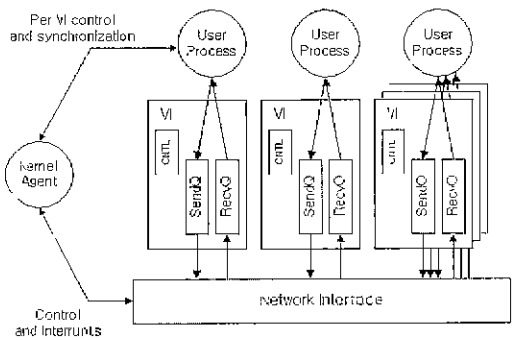

그림 3 Virtual Interface Architecture[4]

3.3 통신 모델

현재 제안된 메시지 전송계층들은 기능뿐만 아니라 지원하는 통신 모델 역시 다양하다. 각 메시지 전송계층이 지원하는 기본 통신 모델은 표 1에서와 같이 send/receive, RPC, direct deposit 등 다양하다. 응용프로그램의 작성 시 가장 좋은 성능을 얻기 위해서는 응용프로그램에 적합한 메시지 전송계층을 채택하고, 기본 통신 모델 및 API에 맞추어 작성하여야 한다.

표 1 메시지 전송 계층의 특징 비교[6]

	AM-II	BIP-0.92	FM-2.1	PM-1.2	VMMC-2
Comm Model	RPC	Send/Recv	Send/Recv	Send/Recv	Direct Deposit
Protection	Yes	No	Yes	Yes	Yes
Multi-programming	Yes	No	Yes	Yes	Yes
Buffer overflow	Prevented	Data Loss	Prevented	Tolerated	Impossible
Net Management	Dynamic /Full	Static	Static	Static	Dynamic/ Demand

예를 들어 SHMEM API를 사용하는 응용프로그램의 경우, VMMC와 같이 원격 메모리 접근을 기본적으로 지원하는 메시지 전송계층을 사용하여 효율적으로 작성할 수 있다. 하지만 VMMC 상에 MPI를 구현하는 경우 MPI를 위한 패킷헤더 생성 등의 작업으로 인하여 추가의 메시지 전송 또는 메모리 복사로 인한 성능 감소가 발생한다. MPI를 사용하는 경우에는 send/receive 형태의 통신 모델을 지원하는 메시지 전송 계층을 선택하는 것이 유리하다. 그 중에도 FM 의 gather/scatter 기능은 MPI와 같은 상위 계층을 지원하기에 적합하기 때문에 FM 상의 MPI에서 응용프로그램을 수행하는 것이 좋다[9].

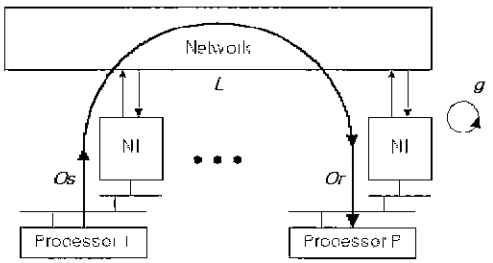

그림 4 LogP 모델[7]

메시지 전송계층은 통신망과 응용프로그램 사이에서 통신에 필요한 기본적인 primitive 를 제공하는 역할을 한다. 따라서 메시지 전송계층은 기존에 사용해 왔던 Socket, MPI 등의 다양한 프로그래밍 모델을 효율적으로 지원하고, 통신 신뢰성 보장, 멀티캐스트 등의 응용프로그램 요구를 수용할 수 있어야 한다. UCSD의 HPVM과 같은 클러스터 시스템의 경우 FM 상에 SHMEM, MPI, Global Array와 같은 다양한 프로그래밍 모델을 지원함으로써 클러스터의 가용성을 높였다 이처럼 클러스터를 위한 메시지 전송계층은 다양한 모델을 효율적으로 지원하여야 한다. 또한 메시지 전송계층의 성능 평가도 동일한 프로그래밍 모델 상에서 이루어져야 한다.

4. 성능 평가

본 장에서는 메시지 전송계층의 성능 평가 방법중 하나인 LogP 모델을 설명하고, GM[16]과 BIP 상에 MPI 계층을 구현한 MPI-GM과 MPI-BIP의 성능을 비교한다.

Linux 환경에서 MPI를 지원하는 메시지 전송계층인 GM과 BIP는 매우 우수한 성능을 보이는 것으로 알려져 있다. GM은 Myrinet을 개발한 Myricom에서 제공하는 메시지 전송계층으로 다양한 기능을 제공한다. BIP는 최소한의 기능으로 Myrinet 상의 메시지 전송계층 중 가장 작은 통신 지연시간과 높은 대역폭을 얻는다.

GM과 BIP는 특성이 크게 다르므로 MPI와 같은 상위 통신계층을 구현할 때 추가되는 오버헤드를 고려하지 않고 하위 통신계층에서 성능을 직접 비교하는 것은 바람직하지 않다. 따라서 본 논문에서는 동일한 API를 지원하는 MPI-GM과 MPI-BIP를 비교하였다.

4.1 LogP 모델

LogP 모델[6, 7]은 기존의 통신 지연시간과 대역폭의 개념을 통신 하드웨어 지연시간(Latency), 프로세서의 오버헤드(Overhead), 메시지 전송 파이프라인의 병목(Gap)으로 나눈 개념이다. 일반적으로 통신 지연시간이라 함은 두 노드간의 점대점 통신에 소요되는 시간을 말하는데, LogP 모델에서는 이를 통신망 하드웨어에서 지연된 시간과 프로세서에서 통신망 하드웨어로 전송하는 시간으로 나눔으로써 메시지 전송계층의 오버헤드를 세분하여 나타낸다. L은 메시지 전송계층이 통신망을 얼마나 효율적으로 사용하는가를 나타내는 지표이며, O는 메시지 전송에 드는 프로세서의 계산 손실을 측정하는 지표이다.

하지만 최근의 메시지 전송계층들은 다양한 시

스템 자원을 사용하기 때문에 LogP와 같은 마이크로벤치마크(microbenchmark)만으로 성능을 비교하는 것은 문제가 될 수 있다. 따라서 이러한 성능 평가 결과가 실제 응용프로그램을 수행할 때의 클러스터 시스템의 성능과 어떤 관련이 있는지에 대한 연구가 필요하다[6].

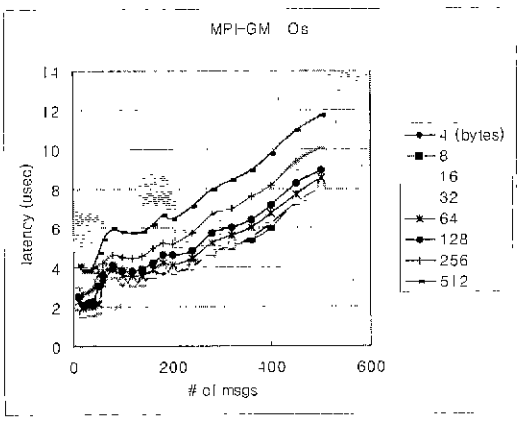

그림 5(a) MPI-GM의 Os

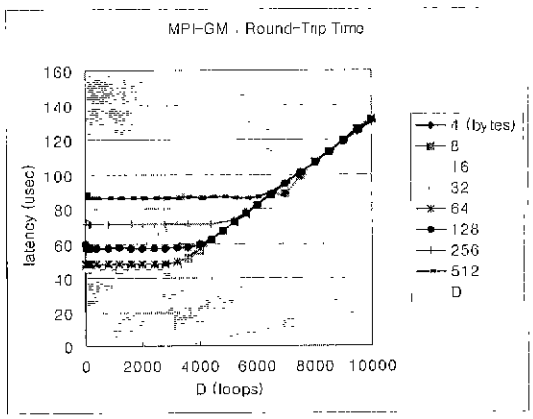

그림 5(b) MPI-GM의 RTT

4.2 MPI 성능 평가

그림 5와 그림 6은 각각 GM과 BIP 상에 구현된 MPI의 성능을 LogP 모델로 측정한 결과이다. 실험 환경으로서 Linux 2.2.13을 기반으로 하는 두 대의 2-way Pentium III 500MHz SMP 노드를 32bit 33MHz LANai 4.3 프로세서를 장착한 Myrinet으로 연결하였다.

(a)는 송신자의 오버헤드(Os: send overhead)를 측정한 결과로서, 한 번에 보내는 메시지의 수가 증가함에 따라 프로세서의 오버헤드가 증가한다. 통신망 인터페이스에서 단위 시간에 처리할 수 있는 메시지의 수에는 한계가 있으므로 프로세서에서 통신망 인터페이스로 메시지를 전달하기 위해 기다리는 시간이 길어지기 때문이다.

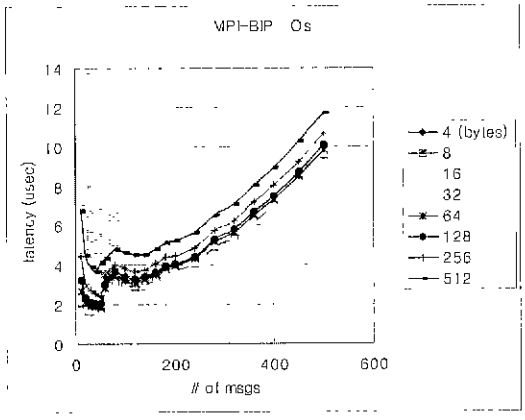

그림 6(a) MPI-BIP의 Os

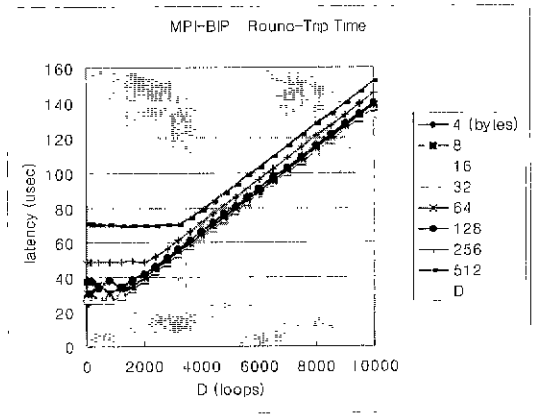

그림 6(b) MPI-BIP의 RTT

(b)는 추가 지연시간(D)의 변화에 따른 왕복 지연시간(RTT: round-trip time)을 측정하여 송신자의 오버헤드와 수신자의 오버헤드(Or: receive overhead)의 합을 계산한 결과이다. 왕복 지연시간은 RTT=2(Os+L+Or)로 구성되는데, 메시지를 송신한 후 충분한 추가 지연시간 D를 주면 그 동안 상대방 노드로부터 답신 메시지가 전송되므로 이 경우 전체 왕복 지연시간은

RTT'=Os+D+Or이다 이 때 RTT'와 D로부터 Os+Or을 구할 수 있으며, (a)에서 구한 Os를 이용하여 Or을 측정한다. (b)의 그림에서 D가 충분히 클 때 RTT'의 그래프는 D와 평행하게 증가하며 Os+Or은 RTT'와 D의 차이다. 또한 RTT와 Os, Or로부터 L의 값을 얻을 수 있다.

그림 7에서 MPI-BIP는 MPI-GM에 비하여 L이 매우 작은 반면 Os+Or이 대단히 크다. 게다가 메시지의 크기가 증가할수록 Os-Or이 점차 증가한다. BIP는 기본 통신 계층을 매우 간단하게 설계하여 L을 최소한으로 줄였지만 MPI와 같은 상위 통신계층을 지원하기 위해서는 추가 구현이 필요하므로 이러한 오버헤드가 발생한다. 반면 GM은 다양한 기능을 지원하므로 MPI 수준에서 추가되는 오버헤드가 작다. 따라서 메시지의 통신 지연시간이 중요한 프로그램은 MPI-BIP를 사용하는 것이 유리하고, 프로세서의 계산량이 많고 통신 지연시간이 크게 중요하지 않은 프로그램은 O가 작은 MPI-GM을 사용하는 것이 유리하다.

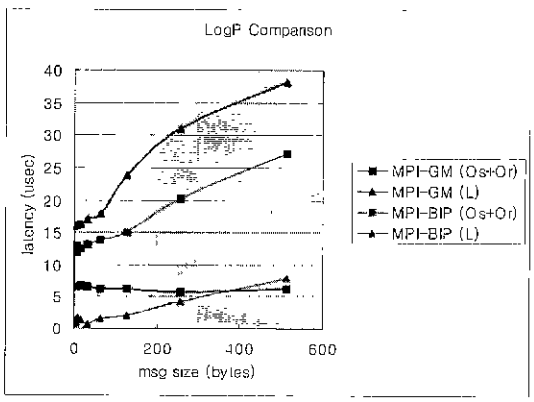

그림 7 MPI-GM, MPI-BIP의 O, L 비교

5. 결 론

최근의 메시지 전송 계층의 기술 발전으로 메시지 전송계층의 부하가 10μsec 이내로 감소하였으며 통신망 하드웨어 한계에 근접하는 통신 대역폭을 얻게 되었다.

이러한 메시지 전송계층의 기술 발전은 Microsoft, Compaq, Intel의 주축으로 제안된 VIA의 기본 구조에 많은 영향을 주었으며, VIA는 많은 통신망 하드웨어 업체의 지원을 받으며 앞으로의 클러스터 시스템에 사용될 것으로 예측된다. VIA는 현재까지의 기술 발전을 토대로 메시지 전송 계층의 기본 틀을 표준화 한 것으로, 앞으로도 VIA의 구현과 이에 따르는 다양한 설계 이슈들에 대한 연구가 계속 되어질 것이다.

한편, 통신 계층의 기술 발전과 더불어 메시지 전송 계층과 클러스터 응용프로그램간의 효율적인 연계 및 실제 응용프로그램을 통한 성능평가가 이루어져야 할 것이다.

참고문헌

[1] R. Bhoedjang et al., "User-Level Network Interface Protocols", IEEE Computer, Vol. 31, No. 11, Nov 1998, pp.53-60.

[2] O. Maquelin et al., "Polling Watchdog: Combining Polling and Interrupts for Efficient Message Handling," ISCA '96, pp. 179-188.

[3] K. Verstoep, K.G. Langendoen, and H.E. Bal, "Efficient Reliable Multicast on Myrinet," ICPP'96, Vol. III, pp 156-165.

[4] D. Dunning et al., "The Virtual Interface Architecture", IEEE Micro, Vol. 18, No. 2, Mar/Apr 1998, pp.66-76.

[5] S. Mukherjee and M. D. Hill,"A Survey of User-Level Network Interfaces for System Area Networks", UWCS TR @1340, University of Wisconsin-Madison, Feb 1997.

[6] S. Araki et al.," User-Space Communication: A Quantitative Study", ICS '98.

[7] D. Culler, et all., "Assessing Fast Network Interfaces", IEEE Micro, Vol. 16, No. 1, Feb 1996, pp.35-43.

[8] A. Gallatin et al.,"Trapeze/IP: TCP/IP at Near-Gigabit Speeds", In Proc. of the 1999 USENIX Technical Conference (Freenix Track), Jun 1999.

[9] 김호중, "Myrinet 상에 VMMC를 기반으로 하는 효율적인 MPI 구현", 석사 논문, 한국과학기술원, 2000.

[10] T. Eicken et al.,"U-Net: A User-Level

Network Interface for Parallel and Distributed Computing", In Proc of the 15th Annual ACM Symposium on Sperating System Principles, Dec 1995.

[11] L. Prylli et al.,"Protocol Design for High Performance Networking: a Myrinet Experience", Research Report 97-22, LIP-ENS Lyons, France, 1997.

[12] Compaq Computer Corp. and Intel Corp and Microsoft Corp. "Virtual Interface Architecture Specification Version 1.0", December 1997.

[13] H. Tezuka et al., "Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication", In Proc. of the 12nd Annual Int'l Parallel Processing Symp., March 1998.

[14] C. Dubnicki et al., "VMMC-2: Efficient Support for Reliable, Connection-Oriented Communication", In Hot Interconnects V, August 1997.

[15] B. N. Chun et al., "Virtual Network Transport Protocols for Myrinet", In Hot Interconnects V, August 1997.

[16] http://www.myri.com/GM/doc, "The GM Message Passing System", October 1999.

[17] M. Lauria et al., "Efficient Layering for High Speed Communication: Fast Messages 2.x", HPDC7, Jul 1998

[18] A. Barak et al., "Performance of the Communication Layers of TCP/IP with the Myrinet Gigabit LAN", Computer Communications, Vol. 22, No. 11, Jul 1999.

손 영 철

1994.2 한국과학기술원 전산학과 학사
1996.2 한국과학기술원 전산학과 석사
1996.3~현재 한국과학기술원 전자전산학과 전산학전공 박사과정 재학 중
관심분야:병렬처리, 분산공유메모리 시스템, 상호연결망, 클러스터 컴퓨팅
E-mail:ycsohn@camars.kaist.ac.kr

김 호 중

1998.2 한국과학기술원 전산학과 학사
2000.2 한국과학기술원 전산학과 석사
2000.3~현재 한국과학기술원 전자전산학과 전산학전공 박사과정 재학 중
관심분야:상호연결망, 클러스터 컴퓨팅
E-mail:hjkim@camars.kaist.ac.kr

맹 승 렬

1977.2 서울대학교 공과대학 전자공학과 학사
1979.2 한국과학기술원 전산학과 석사
1984.2 한국과학기술원 전산학과 박사
1984.3~현재 한국과학기술원 전자전산학과 전산학전공 교수
관심분야:컴퓨터구조, 병렬처리, 클러스터 컴퓨팅
E-mail:maeng@cs.kaist.ac.kr