고성능 PC 클러스터 시스템을 위한 VIA 기반 RDMA 메커니즘 구현

(A VIA-based RDMA Mechanism for High Performance PC Cluster Systems)

정 인 형 † 정 상 화 †† 박 세 진 †††
(In-Hyung Jung) (Sang-Hwa Chung) (Sejin Park)

요 약 PC 클러스터 상에서 기존의 TCP/IP와 같은 통신 프로토콜의 높은 소프트웨어 오버헤드를 제거하기 위한 노력으로 산업계 표준으로 Virtual Interface Architecture(VIA)가 제안되었다. VIA가 제공하는 통신 방식중, Remote Direct Memory Access(RDMA) 방식은 커널과 리모트 노드의 개입 없이 통신을 가능하게 함으로써 PC 클러스터 시스템에 효율적인 통신 방법을 제공한다. 본 논문에서는 VIA 기반 RDMA 메커니즘을 하드웨어로 구현하였다. 일반적인 송수신방식과 비교하여 본 논문에서 구현한 RDMA 메커니즘은 커널의 개입 없이 무복사 통신을 가능하게 하며, 또한 리모트 노드의 CPU의 사용 없이 통신을 수행할 수 있다. 실험결과, RDMA를 하드웨어 VIA 기반 네트워크 어댑터상에 구현함으로써 최소 12.5 μs의 지연시간, 최대 95.5MB/s의 대역폭을 얻을 수 있었다. 결과적으로 본 논문에서 구현한 VIA 기반 RDAM 메커니즘은 PC 클러스터 시스템에 효율적인 통신 방법을 제공한다.

키워드 : VIA, RDMA, 기가비트 이더넷, PC 클러스터링

Abstract The traditional communication protocols such as TCP/IP are not suitable for PC cluster systems because of their high software processing overhead. To eliminate this overhead, industry leaders have defined the Virtual Interface Architecture (VIA). VIA provides two different data transfer mechanisms, a traditional Send/Receive model and the Remote Direct Memory Access (RDMA) model. RDMA is extremely efficient way to reduce software overhead because it can bypass the OS and use the network interface controller (NIC) directly for communication, also bypass the CPU on the remote host. In this paper, we have implemented VIA-based RDMA mechanism in hardware. Compared to the traditional Send/Receive model, the RDMA mechanism improves latency and bandwidth. Our RDMA mechanism can also communicate without using remote CPU cycles. Our experimental results show a minimum latency of 12.5μs and a maximum bandwidth of 95.5MB/s. As a result, our RDMA mechanism allows PC cluster systems to have a high performance communication method.

Key words : VIA, RDMA, Gigabit Ethernet, PC Clustering

1. 서 론

최근의 기가빗 이더넷, Myrinet[1], SCI[2]등 고성능 네트워크의 등장과 고속 마이크로프로세서가 장착된 고성능 PC의 급속한 발전으로 강력한 클러스터 시스템의 구성이 가능해 졌다. PC 클러스터 상에서 응용프로그램의 성능은 각 노드의 성능과 각 노드를 연결하는 LAN 또는 SAN의 통신성능에 의존하게 된다. 패스트 이더넷 같은 범용의 연결망과 기존의 TCP/IP 프로토콜로는 높은 소프트웨어 오버헤드[3-5]와 부족한 네트워크 성능으로 인해 여러 가지 다양한 애플리케이션에 충분한 통신 성능을 제공할 수 없다. 특히 클러스터가 구성되는 System Area Network(SAN) 환경에서는 안정적인 데이타 통신이 가능해져 TCP/IP의 다양한 흐름 제어 및 에러 체크 서비스는 오히려 오버헤드로 작용하기도 한다. 따라서 Gbps 이상의 하드웨어 성능을 가지는 SAN 환경에서의 통신과정 중 소프트웨어 오버헤드를 제거하는 것

· 본 연구는 과학재단 지역대학우수과학자 지원연구(R05-2003-000-10726-0) 지원으로 수행되었음
† 비 회 원 : 삼성전자 무선사업부
inhyong77@hotmail.com
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
shchung@pusan.ac.kr
††† 비 회 원 : 부산대학교 컴퓨터공학과
sejnpark@pusan.ac.kr
논문접수 : 2004년 3월 11일
심사완료 : 2004년 7월 29일

은 아주 중요한 문제이며, 이러한 소프트웨어 오버헤드를 줄이고자 하는 노력이 진행되어 왔다. 이런 노력의 결과로 Active Message(AM)[6], U-Net[7], VMMC (Virtual Memory-Mapped Communication)[8], Fast Message(FM)[9] 등의 다양한 사용자 수준 통신 프로토콜이 제안되었다. 위의 다양한 프로토콜에 기초하여 낮은 지연시간, 높은 대역폭 SAN을 위한 업계 표준으로 Virtual Interface Architecture(VIA)가 제안되었다[10].

VIA는 사용자 수준의 무복사 전송을 제공하여 통신 프로토콜에서 소프트웨어 오버헤드를 줄여준다. 또한 VIA는 두 가지 방식의 통신 모델을 제공한다. 그 하나는 일반 송수신 방식의 통신이며, 다른 하나는 Remote Direct Memory Access(RDMA)방식이다. GM[11], VIA, Infiniband[12]등 많은 고성능 네트워크에서 낮은 지연시간과 높은 대역폭 전송을 위해 RDMA를 채택하고 있다. 또한 TCP/IP의 경우, 현재 RDMA를 지원하기 위해 초안이 작성 중이며 아직 실제 구현된 사례는 없다. 그러나 TCP/IP의 경우, RDMA를 지원하기 위해서는 RDMA Network Interface Card(RNIC)과 같은 하드웨어의 지원이 필수적이며, 실제 RDMA를 지원하기 위한 프로토콜이 아니므로 RDMA 관련 레이어의 추가 등 구현이 복잡해지고 이로 인해 성능 저하가 우려된다. 또한 VIA를 소프트웨어로 구현한 M-VIA[13], Berkeley VIA 프로젝트[14]의 경우 실제 NIC에서 RDMA를 처리할 수 없으므로 리모트 노드의 CPU 자원을 사용하게 되며, 완전한 RDMA를 구현할 수 없다.

VIA를 하드웨어로 구현한 사례로는 Emulex사의 cLAN[15]과 Tandem/Compaq사의 ServerNet II[16]를 들 수 있다. 두 사례 모두 구현에 관한 세부사항에 대해서는 보고된 바 없으며, 보드상에 메모리를 장착하고 있지 않으므로, 주소 변환표(ATT)는 하드웨어로 구현하지 않은 것으로 추측된다.

본 논문에서는 지금까지 구체적인 구현 내용이 보고된바 없는 VIA기반 RDMA 통신 모델의 하드웨어 구현 방법을 제시하고, 실제 구현을 통해 성능 향상을 보였다. 구현을 위해 기가빗 이더넷 기반의 하드웨어 VIA 카드인 HVIA-GE[17]를 이용하였으며, RDMA 관련 기능을 HVIA-GE내에 하드웨어로 구현함으로써, 최대한 네트워크의 물리적 성능에 근접한 성능을 제공한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 RDMA을 구현하기 위한 사용자 수준 통신 프로토콜인 VIA에 대해 설명한다. 3장에서 RDMA를 실제로 구현한 PCI 기반 네트워크 카드인 HVIA-GE에 관해서 살펴보고, 4장에서 VIA기반 RDMA구현에 관해 상세히 알아볼 것이다. 5장에서는 구현된 VIA기반 RDMA의 성능을 비교 분석하고, 마지막 6장에서 결론 및 향후 연구과제에 대하여 설명한다.

2. VIA 개요

VIA는 Compaq, Intel, Microsoft의 3개의 기업이 기존의 사용자 수준 통신 프로토콜(User-Level Communication Protocol)을 근간으로 클러스터 환경이나 SAN에서 사용할 고성능의 네트워크 프로토콜 표준의 필요성에 의해 제안되었다. VIA의 주된 목적은 SAN상에서 소프트웨어 오버헤드를 제거하여 낮은 지연시간과 높은 대역폭을 지원하는 통신 메커니즘을 제공하는 것이다.

VIA는 메모리 등록, VI 설정 등의 통신을 위한 초기화는 커널의 도움을 받아 이루어 지지만, 이후 통신 과정에서는 커널의 개입 없이 사용자 영역과 NIC간에 직접적인 접근으로 통신이 진행된다. VIA는 커널의 간섭 없이 사용자 영역과 NIC간의 직접적인 접근을 위해서 Virtual Interface(VI)라는 가상의 채널을 제공한다. VIA는 VI로 구성된 종단점간의 점대점 연결을 제공하는 연결형 통신 모델이며, VI는 WQ, CQ, doorbell로 구성된다. 그림 1은 VI Architecture 모델의 구성요소와 구성요소 간의 관계를 나타낸 그림이다.

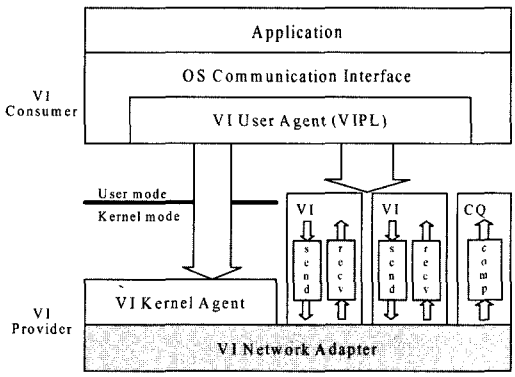


그림 1 Virtual Interface Architecture

3. HVIA-GE

그림 2는 본 논문에서 VIA기반 RDMA 통신 메커니즘 개발을 위해 사용한 PCI 카드인 HVIA-GE의 구조를 나타내고 있다. HVIA-GE는 부산대학교 컴퓨터 구조 및 시스템 연구실에서 개발된 32bit/33MHz PCI 버스 기반의 네트워크 어댑터 카드이다[17]. 물리적인 네트워크로는 기가빗 이더넷을 사용하고, 64MB SDRAM을 장착하고 있으며, FPGA에는 PCI 인터페이스 로직과 VIA 프로토콜 엔진 그리고 MAC을 직접 제어하는 기가빗 이더넷 컨트롤러(GEC)가 구현되어있다. PCI 인터페이스의 경우, PLX9054 등의 상용 칩을 사용하지

않고, DMA 초기화 시간을 최소화하기 위한 전용의 PCI 인터페이스 로직을 가지고 있으며, 기가빗 이더넷 칩은 현재 많은 기가빗 이더넷 카드에서 사용중인 National사의 DP83820(MAC), DP83861(PHY)를 채용하고 있다. 또한 GEC는 VIA 프로토콜 엔진과의 인터페이스를 제공하며, VIA 프로토콜 엔진에서 직접 통신 명령을 내릴 수 있다. 본 논문에서는 HVIA-GE의 64M SDRAM은 ATT를 저장 하는데 사용하였고, FPGA내에 VIA 프로토콜 엔진과 SDMAM 컨트롤러를 구현하였다.

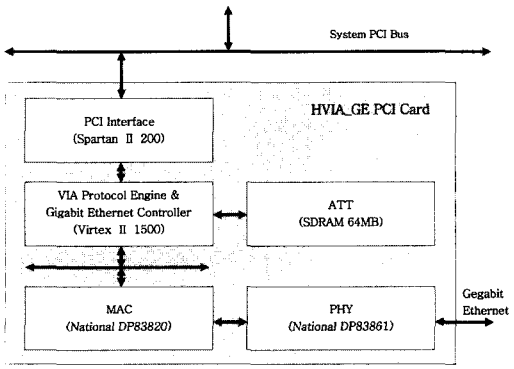


그림 2 HVIA-GE PCI 카드 구조

4. RDMA 메커니즘 구현

RDMA 통신모델은 RDMA 쓰기와 RDMA 읽기로 구성된다. RDMA 쓰기의 경우, 데이타 전송에 관해 수신측에 알리지 않고 송신측에서 수신측의 데이타 버퍼에 바로 데이타를 전송하게 된다. 이때 데이타 버퍼에 관한 정보(가상 주소와 메모리 핸들)가 전송 전에 송신측에 알려져야 한다. RDMA 읽기의 경우는 반대로 읽고자 하는 노드가 리모트 노드에 알리지 않고 리모트 버퍼 영역을 읽어온다. RDMA 읽기의 경우 RDMA 요구를 리모트 노드로 전송하고, RDMA 읽기 요구를 처리하기 위해서 리모트 노드에서는 DMA방식으로 로컬 영역의 데이타를 읽어와서 전송하게 된다. 데이타가 연결망을 통해 도착하게 되면 다시 DMA를 통해 사용자 영역으로 데이타를 복사하게 된다. 따라서 RDMA 쓰기와 비교하여 요구의 처리를 위해서는 두번의 통신이 필요하다. 즉 RDMA 읽기는 RDMA 쓰기에 비해 통신 오버헤드가 더 크며, 응답 시간에 큰 손실을 준다. 실제 RDMA 읽기는 프로그래머의 사용에 의해 RDMA 쓰기로 대체 가능하기 때문에 VIA 명세 1.0에서는 RDMA 쓰기는 기본 함수로 정의하고 있으며, RDMA 읽기는 옵션으로 정의하고 있다.

4.1 VIA기반의 RDMA 통신 과정

VIA 기반의 RDMA 쓰기의 동작 과정은 그림 3과 같다. 그림에서 ①과 같이 VI Architecture에서는 사용자 영역과 커널 영역 사이의 데이타 복사를 제거하기 위해서 사용자 영역의 통신용 버퍼를 등록해야 한다.

사용자 버퍼를 등록한 후, 그림에서 ②와 같이 노드2의 프로세스 B는 노드1의 프로세스 A에게 일반 송수신 방식으로 자신의 로컬 버퍼의 주소 정보를 알린다. 그

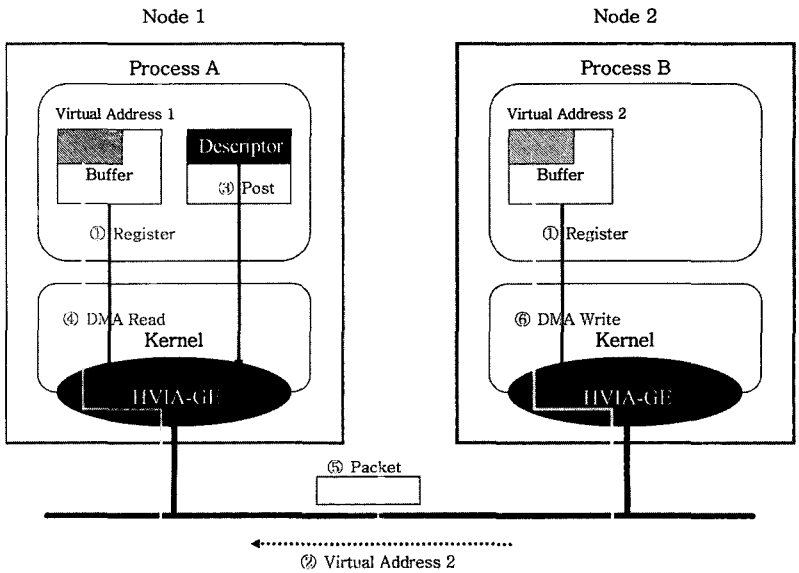


그림 3 RDMA 쓰기 수행 과정

후 노드1의 프로세스 A는 VI 디스크립터를 WQ에 포스팅(③)함으로써 RDMA 쓰기를 시작하게 되고, 사용자 버퍼의 데이타를 DMA(④)로 읽어와서 연결망을 통해 전송(⑤)하게 된다. 패킷이 노드2에 도착하게 되면 RDMA 엔진은 커널의 개입 없이 주소 변환 과정을 통해 DMA방식으로 데이타를 사용자 버퍼로 복사(⑥)하게 된다.

일반 송수신의 경우에는 송신측과 수신측 모두 통신 요구에 따른 VI 디스크립터를 생성하고 양쪽 프로세스 모두에서 통신을 수행하게 된다. 통신에 사용되는 VI 디스크립터의 경우 일반 송수신방식과 RDMA 방식의 경우는 서로 다르다. 그림 4는 VI 디스크립터의 구조를 나타낸 것인데, 일반 송수신 방식의 경우 통신을 제어하기 위한 제어 세그먼트와 로컬 버퍼를 기술하는 데이타 세그먼트로만 이루어진다. 이와는 달리 RDMA 방식의 경우 송신측이 로컬 버퍼 영역과 리모트 버퍼 영역 모두를 기술해야 하기 때문에 그림에서와 같이 리모트 버퍼 영역을 기술하는 주소 세그먼트가 추가 된다.

실제 위의 과정이 가능하도록 VIA 프로토콜 엔진을 하드웨어로 구현하기 위해서는 크게 VipRegisterMem 함수를 통한 사용자 버퍼의 등록, HVIA-GE가 커널로의 복사 없이 데이타를 DMA로 읽고 쓰기 위한 주소 변환 기능과 전송을 시작하기 위한 VI 디스크립터, 전송을 제어하기 위한 VI 헤더등의 자료구조가 필요하다. 이제부터 실제로 하드웨어로 구현된 VIA 프로토콜 엔진의 각 구성요소와 RDMA를 지원하기 위한 기능들에 관해 자세히 알아보자.

4.2 VIA 프로토콜 엔진

그림 5는 HVIA-GE에서 VIA를 하드웨어로 처리하기 위한 코어 로직인 VIA 프로토콜 엔진과 기가빗 이더넷 컨트롤러(GEC)를 나타낸 그림이다. 그림과 같이 VIA 프로토콜 엔진은 새로운 트랜잭션이 있음을 알리

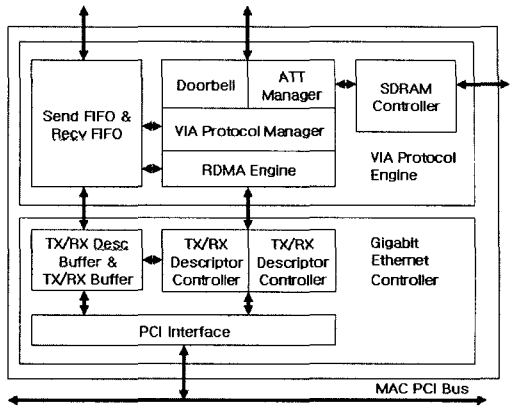


그림 5 VIA 프로토콜 엔진과 기가빗 이더넷 컨트롤러

는 doorbell, 사용자 버퍼의 등록과 주소 변환을 수행하는 ATT 매니저, RDMA를 처리하기 위한 RDMA 엔진, 기타 VI 연결설정 및 구성요소간의 동작을 제어하는 VI 프로토콜 매니저로 구성된다.

VIPL 함수의 파라미터들은 시스템 PCI 버스를 통해 직접 VIA 프로토콜 엔진으로 전달되며, 특히 사용자 버퍼를 등록하는 VIPL 함수인 VipRegisterMem의 경우에는 전송에 필요한 사용자 버퍼의 가상 주소, 물리적 주소, 크기 등을 ATT 매니저를 통해 SDRAM의 ATT에 저장하게 된다. 그리고 실제 송수신 트랜잭션이 있을 경우, 해당 VI 디스크립터를 생성하고 doorbell을 통해 VIA 프로토콜 엔진에게 새로운 트랜잭션이 있음을 알린다. 그러면 VIA 프로토콜 엔진은 하드웨어로 구현된 doorbell을 통하여 새로운 송수신 트랜잭션을 바로 알게 되며, ATT 매니저를 통해 전송에 필요한 VI 디스크립터 및 사용자 버퍼의 물리적 주소를 알아내어 DMA를 통해 읽은 다음, 필요한 정보를 GEC로 전달하게 된다. 이때 실제 전송되는 데이타는 Send/Recv FIFO 및

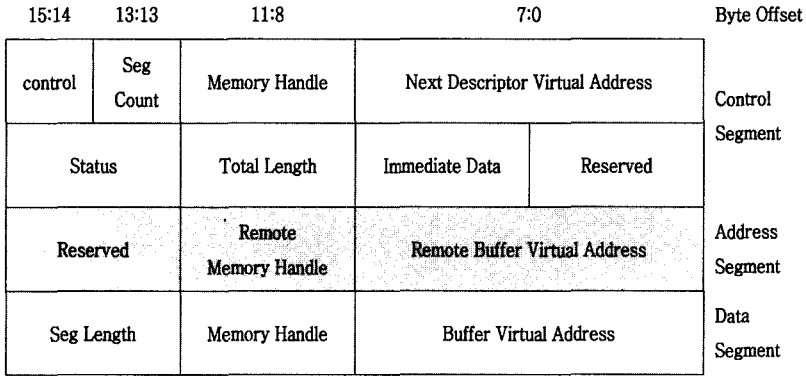


그림 4 VI 디스크립터

TX/RX 버퍼를 통하여 전달된다.

4.3 RDMA를 위한 주소 변환

주소 변환표(ATT)는 RDMA 통신시 커널의 개입 없이 카드 내에서 DMA를 통해 등록된 사용자 버퍼로의 접근이 가능 하도록 필요한 정보들을 저장한다. ATT는 2 레벨로 구성되며, 24-bytes 크기를 가지는 레벨 1은 사용자 버퍼가 등록될 때 전체 버퍼에 대한 정보를 저장하며, 레벨 2는 사용자 버퍼가 구성된 물리적 페이지의 정보를 저장한다. ATT 레벨 1의 엔트리에는 사용자 버퍼의 물리적 페이지 개수, 첫 번째 페이지의 크기, 등록된 사용자 영역의 속성을 저장하는 메모리 속성, 시작 가상 주소 및 ATT 레벨 2 포인터를 저장한다. 그리고 ATT 레벨 2의 엔트리에는 실제 할당된 물리적 페이지의 물리적 주소를 저장한다. 아래의 그림 6은 등록된 사용자 버퍼 영역과 ATT의 관계를 보여준다.

위 그림은 실제 9 Kbytes의 사용자 영역을 할당하였을 경우 커널 영역에 메모리가 할당된 형태와 이 영역에 관한 ATT의 구조를 나타낸 그림이다. 그림에서와 같이 ATT 레벨 1의 엔트리에는 물리적 페이지의 개수(3), 첫번째 페이지의 크기(1K), 메모리 속성, 시작 가상 주소(1C00), ATT 레벨 2 포인터 정보가 저장된다. ATT 레벨 2의 엔트리는 각 물리적 페이지의 시작 물리적 주소(2C00, 5000, 8000)가 저장된다.

RDMA 수행시 RDMA 엔진에서 가상 주소와 메모리 핸들을 통해 해당 ATT 엔트리를 읽고, 각 엔트리의 정보를 이용하여 임의의 가상 주소를 물리적 주소로 변환한다. 이때 변환된 물리적 주소를 이용 하여 DMA 방식으로 데이타를 이동하게 된다.

한편, 본 논문에서의 구현 방법과 달리, NIC 상에 충분한 저장공간이 없거나, 기존의 범용 NIC을 이용한 소프트웨어 VIA 구현에서는 호스트 메모리를 활용하여 ATT를 구현하고 있다. 즉, 호스트 메모리의 특정 영역에 ATT를 저장하고, VIA 프로토콜 엔진이 DMA를 통해 ATT를 읽어와 주소 변환을 수행 한다. 이 방법은 RDMA 통신 수행시 ATT의 접근에 있어 비효율적이다.

본 논문에서 구현한 주소 변환 방법은 RDMA 통신시 주소 변환 과정을 커널의 개입 및 부가적인 호스트 메모리 접근을 없애고, 구현한 SDRAM 컨트롤러의 버스트 모드를 사용하여 ATT 접근을 개선하였다.

5. 실 험

본 논문에서 구현한 RDMA 통신의 성능 측정을 위하여 800MHz 펜티엄 III, 256MB의 100MHz SDRAM, 32bit/33MHz PCI 버스 기반의 PC 클러스터를 사용하였고, 운영체제로는 Linux 커널 2.4에 기반한 Redhat 7.2 배포판을 사용하였다. 또한 본 논문에서는 AceNIC의 AceNIC Tigon II 칩이 장착된 기가빗 이더넷 카드를 사용하여, 위와 동일한 PC 클러스터 환경에서 TCP/IP 및 M-VIA 성능을 측정하여 비교 분석 하였다. 이때 사용된 프로그램은 RDMA의 경우, VIPL을 사용하여 ping-pong 프로그램을 직접 제작하였으며, M-VIA의 성능은 배포판에 포함된 성능 측정 프로그램인 vnettest를 이용하였고, TCP/IP 소켓 라이브러리를 이용하여 vnettest를 수정하여 사용하였다. 실제 통신의 효율성은 CPU 사용 효율, 지연시간, 대역폭의 용어로 정의 된다. 따라서 본 논문에서는 이 세가지 성능 지표를 기준으로 실험을 진행하였다.

RDMA 방식의 통신에서 얻어지는 장점 중 리모트 노드에서 CPU 사용 효율 측면에서의 이점을 측정하기 위해서 일반 송수신 방식에서 수신측의 CPU 사용 효율

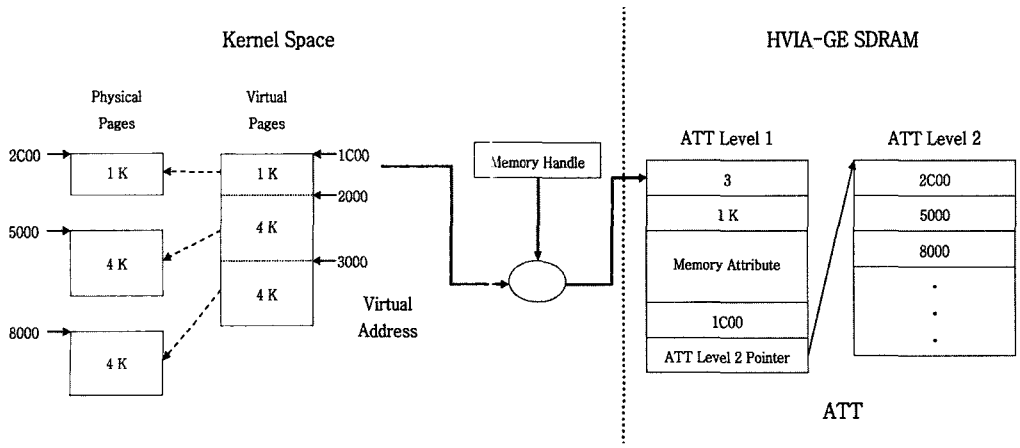


그림 6 주소 변환표

시간을 측정하였다. VipPostRecv 함수의 소프트웨어 처리시간은 gettimeofday 함수를 이용하여 측정하였고, 하드웨어 처리시간은 로직 분석기를 통해서 시간을 측정하였다.

5.1 RDMA, M-VIA 그리고 TCP/IP의 성능 비교

그림 7과 그림 8은 RDMA, M-VIA, 그리고 TCP/IP의 지연시간 및 대역폭 성능을 비교하여 나타낸 그림이다. 세 경우 모두 1,514bytes MTU를 사용하여 성능을 측정하였다. 측정된 지연시간은 왕복시간의 반이며, 대역폭은 전송된 양을 지연시간으로 나눈 값이다.

그림 7이 보여주는 것과 같이 VIA기반의 RDMA 통신이 M-VIA와 TCP/P보다 전체 데이타 범위에서 우수한 성능을 보여주고 있다. RDMA, M-VIA 및 TCP/IP의 성능을 살펴보면, 최소 지연시간의 경우 4bytes 전송시 RDMA가 12.5㎲, M-VIA가 57.6㎲, TCP/IP가 117.9㎲를 보인다. 즉 클러스터 환경에서의 통신 응답시간이 M-VIA에 비해서는 약 4.6배, TCP/IP에 비해서는 약 9.4배의 성능 향상을 보인다. 이러한 성능의 향상은 VIA기반 RDMA가 TCP/IP의 통신시 발생하는 문맥전환, 데이타 복사, 인터럽트 처리 등의 오버헤드를 제거하고, M-VIA와 달리 VIA를 HVIA-GE 상에서 하드웨어로 구현한 것에 기인한다.

최대 대역폭의 경우 HVIA-GE의 RDMA 방식이 95.5 MB/s, M-VIA가 63.5 MB/s, TCP/IP가 60.0MB/s를 보여주고 있다. 즉, 최대 대역폭의 경우, HVIA-GE상에서의 RDMA가 M-VIA보다 약 1.5배, TCP/IP보다 약 1.6배 우수하다. 이러한 성능 향상은 HVIA-GE가 시스템 및 MAC의 PCI 버스에 대해 512 bytes DMA 버스트 전송을 제공하여 각각 최대 120MB/s 및 117MB/s를 전송할 수 있는 전용 PCI 컨트롤러를 탑재하고 있는 것과 VIA 프로토콜 엔진과 GEC 간의 데이타 전송을 매 클록마다 파이프라인 방식으로 내부 FIFO 및 버퍼를 통하여 이동 가능하게 구현함에 기인한다.

5.2 HVIA-GE 상에서 일반 송수신 방식과 RDMA의 비교분석

5.2.1 지연시간과 대역폭

아래 그림 9와 그림 10은 HVIA-GE상에 구현된 VIA의 통신 방식 중 일반 송수신 방식과 RDMA 방식의 지연시간과 대역폭의 성능 개선의 정도를 나타낸 그림이다.

RDMA 방식은 일반 송수신 방식에 비해 지연시간 측면에서는 데이타 크기에 무관하게 평균 2.9㎲의 성능 향상을 보여주고, 대역폭 측면에서는 데이타 크기에 따라 1.4MB/s에서 0.1MB/s의 성능향상을 보여준다. 이 성능개선의 원인은 RDMA 방식의 통신의 경우, RDMA 쓰기가 되는 노드에서 VI 디스크립터를 생성하고 WQ에 포스팅 하는 비용이 사라지기 때문이며, 이에 따라 완료를 위해 VI 디스크립터의 상태 필드에 "done bit"을 쓰는 것도 필요 없게 된다. 데이타 크기가 증가

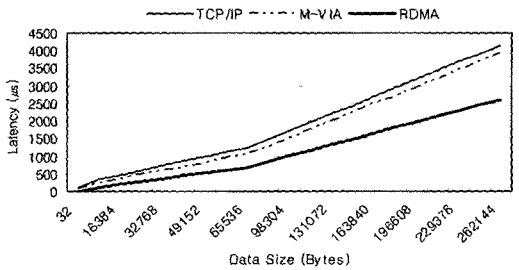


그림 7 지연시간 : RDMA, M-VIA, TCP/IP

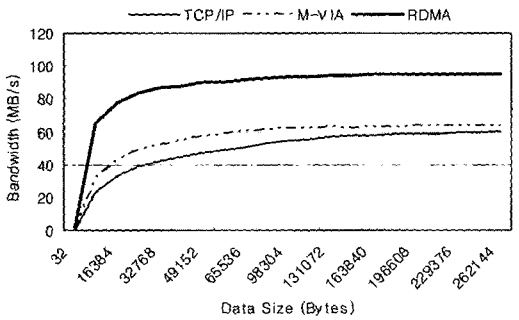


그림 8 대역폭 : RDMA, M-VIA, TCP/IP

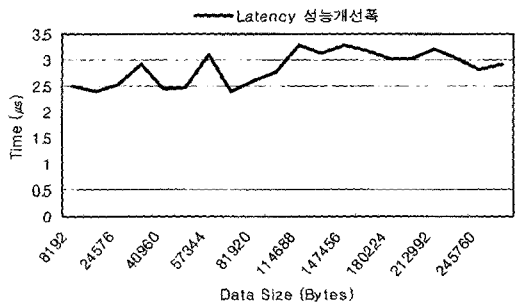


그림 9 일반 송수신과 RDMA의 지연시간 성능 차이

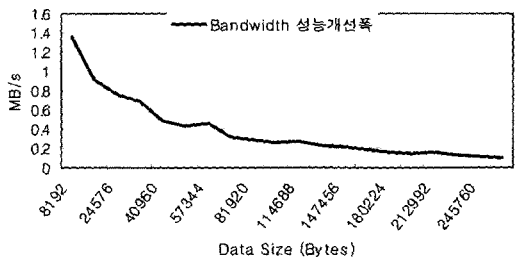


그림 10 일반 송수신과 RDMA의 대역폭 성능 차이

하는 경우는 성능개선의 폭이 상대적으로 줄어들지만, 10~20KB 이하의 메시지로 노드간 통신이 일어나는 응용 프로그램에서는 성능개선 효과가 있을 것으로 보인다.

5.2.2 CPU 사용 효율

하드웨어로 구현된 본 논문의 VIA 프로토콜 구현에서는 일반 송수신 방식과 RDMA 방식의 차이는 송신측의 경우는 수행과정이 동일하며, 앞에서 설명한 것처럼 VI 디스크립터의 내용이 달라질 뿐이다. 실제 두 가지 방식의 차이는 수신측에서 일어나며, RDMA의 경우 VipPostRecv 함수의 처리부분과 VipRecvWait 함수 등의 완료에 관계된 처리부분이 없어진다. 이번 실험에서는 이런 RDMA의 장점 중 하나인 수신측의 CPU 자원 사용의 최소화를 보이기 위해 각 함수의 CPU 처리시간을 측정하였다.

그림 11은 일반 송수신시의 VipPostRecv 함수의 CPU 수행시간을 측정한 것이다. 그림 11에서 CPU 시간은 프로세스에서 VI 디스크립터를 작성하고, 만들어진 VI 디스크립터를 해당 VI의 WQ에 포스팅하고 doorbell에 적은 후 리턴될 때까지의 과정을 말한다. 일반 송수신 방식의 통신에서 수신측의 VipPostRecv 함수를 수행하는 데 걸리는 CPU 시간은 데이타 크기와 상관없이 평균 6㎲이다. RDMA는 VipPostRecv 함수를 수행하지 않으므로 수행하지 않으므로 0㎲가 되며, 이 시간 동안 수신 프로세스는 자신의 작업을 계속 수행할 수 있게 된다.

그림 12는 통신의 완료를 처리하기 위한 VipRecvWait 함수의 처리시간을 데이타 크기에 따라 측정한 그림이다. 그림에서와 같이 완료를 처리하기 위한 함수는 데이타 크기에 따라 비례하여 증가하고 있다. 이 수치는 또한 연결망의 지연시간은 제외한 것으로 연결망의 상태에 따라 처리시간이 증가한다. 이처럼 CPU 사용 효율의 측면에서는 일반 송수신 방식은 수신측 사용자 프로세스가 통신과정에 참여하며, 그 동안 자신의 작업을 진행할 수 없다. 하지만 RDMA의 경우 노드의 프로세스는 통신에 관여할 필요가 없으며, HVIA-GE에서

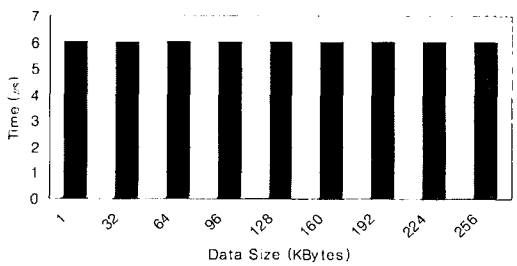


그림 11 VipPostRecv 함수의 CPU 시간

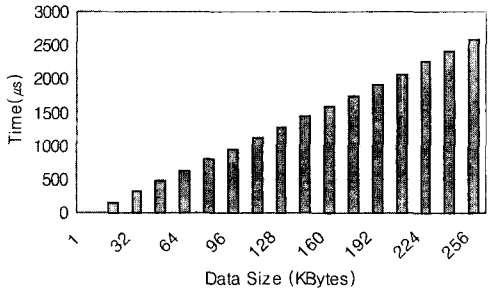


그림 12 VipRecvWait 함수의 처리시간

CPU 자원의 소비 없이 통신을 처리한다.

이상의 실험을 통해 본 논문에서 구현한 VIA기반의 RDMA 통신방식은 TCP/IP와 소프트웨어 VIA인 M-VIA에 비해 지연시간과 대역폭에서 우수성을 보여주고 있으며, 또한 HVIA-GE상에서 일반적인 송수신방식보다 지연시간과 대역폭에서의 성능개선과 CPU 사용 효율에서의 이점을 보여주고 있다.

6. 결론 및 향후 과제

본 논문에서는 VIA기반 RDMA 통신 메커니즘의 하드웨어 구현방법과 성능을 제시하였다. RDMA를 HVIA-GE 상에 하드웨어로 구현함으로써 최소 12.5㎲의 지연시간, 최대 95.5MB/s의 대역폭을 얻을 수 있었다. 이는 TCP/IP와 M-VIA에 비해 큰 폭의 성능 향상을 보여준다. 또한 HVIA-GE의 SDRAM을 이용하여 ATT를 구현하고 주소 변환과정을 하드웨어로 처리함으로써 낮은 지연시간, 높은 대역폭의 통신을 지원하며, RDMA 통신시 리모트 노드의 CPU 자원의 사용 없이 통신을 수행할 수 있다. 이로 인해 RDMA 통신시 리모트 노드의 CPU 시간이 없어지며, 이 시간 동안 리모트 노드의 프로세스는 송수신 트랜잭션 처리로부터 자유로워지며 CPU 사용 효율 측면에서 큰 이득을 볼 수 있다.

향후 과제로는 좀더 RDMA 성능을 개선하기 위해 VIA 프로토콜의 분석과 수정이 필요할 것으로 생각된다. VIA 프로토콜에는 실제 RDMA 통신 전에 리모트 노드가 자신의 로컬 버퍼의 주소를 RDMA 하고자 하는 노드에게 알리는 방식을 취하고 있다. 이는 작은 크기의 데이타를 일회적으로 통신하는 경우 상당한 오버헤드가 된다. 따라서 RDMA 통신을 위한 주소 교환방식을 개선할 필요가 있으며, VIA 프로토콜의 수정을 통해 VI를 연결하는 초기화 과정 동안 추가의 데이타 교환 없이 주소 교환을 가능하게 구현할 수 있을 것으로 보인다. 그리고 구현된 VIA기반 RDMA 메커니즘을 비디오 스트리밍 서버와 같은 다양한 응용 프로그램에 적

용하면 좋은 성능을 보일 것으로 기대된다.

참 고 문 헌

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, W. Su, "Myrinet - A Gigabit per second Local Area Network," IEEE Micro, 1995.

[2] IEEE: Standard for Scalable Coherent Interface (SCI) IEEE Std.1596-1992, IEEE Computer Society, Aug. 1993.

[3] D. D. Clark, V. Jacobson, J. Romkey, H. Salwen, "An Analysis of TCP Processing Overhead," IEEE Communications Magazine , pp. 23-29, June 1989.

[4] J. Kay and J. Pasquale, "Profiling and Reducing Processing Overheads in TCP/IP," IEEE/ACM Transactions on Networking, Vol. 4, No. 6, pp. 817-828, Dec. 1996.

[5] R. A.F. Bhoedjang, T. Ruhl, and H. E. Bal, "User-Level Network Interface Protocols," IEEE Computer, Vol. 31, No. 11, pp. 53-60, Nov. 1998.

[6] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser, "Active Messages: A Mechanism for Integrated Communication and Computation," 19th International Symposium on Computer Architecture, May 1992.

[7] T. von Eicken, A. Basu, V. Buch, and W. Vogels. "U-Net: A User-level Network Interface for Parallel and Distributed Computing," Proc, of the 15th ACM Symposium on Operating Systems Principles (SOSP), Colorado, December 3-6, 1995.

[8] C. Dubnicki, A. Bilas, K. Li, and J. Philbin, "Design and Implementation of Virtual Memory-Mapped communication on Myrinet," presented at Proceedings of the International Parallel Processing Symposium, pp. 388-396, 1997.

[9] S. Pakin, M. Lauria, and A. Chien. "High Performance Messaging on Workstations: Illinois Fast Messages(FM) for Myrinet," Proc. of the Supercomputing'95, December 3-8, 1995.

[10] Virtual Interface Architecture Specification. http://www.viarch.org/

[11] Myricom, The GM Message Passing System, 10/16/1999.

[12] Various. Infiniband tutorials. In Proceedings of the I/O Technology Forum and Expo and Server I/O 2000, Monterey, CA, February 2000. http://www.sresearch.com

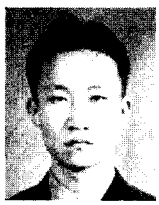
[13] http://www.nersc.gov/research/FTG/via

[14] http://www.millennium.berkeley.edu/via.php3, P. Buonadonna, A. Begel, D. Gay, and D. Culler, "An Analysis of VI Architecture Primitives in Support of Parallel and Distributed Communication," Apr. 2000.

[15] Emulex Corporation, Hardware-based (ASIC) implementation of the Virtual Interface standard, http://www.emulex.com/products/legacy/vi/clan1000.html

[16] ftp://ftp.compaq.com/pub/supportinformation/papers/tc000602wp.pdf

[17] "고성능 클러스터 시스템을 위한 VIA 기반 네트워크 카드의 구현", 박세진, 정상화, 윤인수, 정인형, 이소명, 한국정보과학회 병렬처리시스템연구회, 2003. 11.



정 인 형

2002년 부산대학교 컴퓨터공학과 학사 2004년 부산대학교 컴퓨터공학과 석사 2004년~현재 삼성전자 무선사업부. 관심분야는 병렬처리, 클러스터 시스템, VIA, RDMA

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, 컴퓨터및정보통신연구소 연구원. 관심분야는 클러스터 시스템, 병렬처리, VIA, VOD, TOE, RDMA

박 세 진

1998년 부산대학교 컴퓨터공학과 학사 2000년 부산대학교 컴퓨터공학과 석사 2000년~현재 부산대학교 컴퓨터공학과 박사과정. 관심분야는 클러스터 시스템, 병렬처리, VIA, RDMA