VIA 기반 PC 클러스터 시스템을 위한 무복사 파일 전송 메커니즘의 개발 및 성능분석

(Development and Performance Study of a Zero-Copy File Transfer Mechanism for VIA-based PC Cluster Systems)

박 세 진 † 정 상 화 †† 최 봉 식 ††† 김 상 문 †††
(Sejin Park) (Sang-Hwa Chung) (Bong-Sik Choi) (Sang-Moon Kim)

요 약 본 논문에서는 VIA(Virtual Interface Architecture) 기반 클러스터 시스템 상에서 효과적인 파일 전송을 위한 무복사 파일 전송 메커니즘의 개발 및 구현에 관하여 나타내었다. VIA는 클러스터 시스템을 위한 대표적인 사용자 수준 통신 방법이지만 파일 전송에 대한 라이브러리는 제공하지 않으며 파일 전송을 위해서는 커널 공간에서 사용자 공간으로 한번의 데이타 복사가 필요하다. 본 논문의 파일 전송 메커니즘은 파일시스템의 수정 없이 파일 전송 라이브러리만 제공함으로써, 네트워크 인터페이스 카드가 보내고자 하는 노드의 파일을 상대방 노드의 사용자 버퍼로 복사 없이 전송 가능케 하였다. 이를 위해 본 연구에서는 PCI 64bit/66MHz를 지원하고 물리적 네트워크로 기가비트 이더넷을 사용하는 VIA 기반의 네트워크 카드를 개발하였고, 이를 바탕으로 무복사 파일 전송 메커니즘을 구현하였다. 이러한 구현의 결과로 sender 측의 데이타 복사 횟수 및 문맥전환 시간을 줄였고, 기존의 VIA의 send/receive에 비해 CPU 사용률을 30%~40% 정도로 줄일 수 있었다. 본 논문에서는 TCP/IP에서 제공하는 무복사 파일 전송 및 VIA에서 사용되는 파일 전송 방법과의 비교 분석 실험을 통하여 본 논문에서 제시한 무복사 파일 전송 메커니즘의 성능을 보였다.

키워드 : PC 클러스터, VIA, 무복사 파일 전송, 기가비트 이더넷

Abstract This paper presents the development and implementation of a zero-copy file transfer mechanism that improves the efficiency of file transfers for PC cluster systems using hardware-based VIA(Virtual Interface Architecture) network adapters. VIA is one of the representative user-level communication interfaces, but because there is no library for file transfer, one copy occurs between kernel buffer and user buffers. Our mechanism presents a file transfer primitive that does not require the file system to be modified and allows the NIC to transfer data from the kernel buffer to the remote node directly without copying. To do this, we have developed a hardware-based VIA network adapter, which supports the PCI 64bit/66MHz bus and Gigabit Ethernet, as a NIC, and implemented a zero-copy file transfer mechanism. The experimental results show that the overhead of data coy and context switching in the sender is greatly reduced and the CPU utilization of the sender is reduced to 30% ~ 40% of the VIA send/receive mechanism. We demonstrate the performance of the zero-copy file transfer mechanism experimentally, and compare the results with those from existing file transfer mechanisms.

Key words : PC Cluster, VIA, Zero-copy file transfer, Gigabit Ethernet

· 이 논문은 부산대학교 자유과제 학술연구비(2년)에 의하여 연구되었음
† 비 회 원 : 삼성전자 정보통신총괄 무선사업부
sejnpark@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
shchung@pusan.ac.kr
††† 비 회 원 : 부산대학교 컴퓨터공학과
guyver3@pusan.ac.kr
listman@pusan.ac.kr
논문접수 : 2005년 3월 2일
심사완료 : 2005년 7월 25일

1. 서 론

최근 기가비트 이더넷을 비롯한 고속의 네트워크 환경이 제공되면서 네트워크 디바이스의 성능을 최대한 이끌어 내려는 노력들이 진행되고 있다. 커널 및 사용자 버퍼 사이의 데이타 복사 및 문맥 전환과 같은, 통신 시 발생하는 TCP/IP 프로토콜의 오버헤드를 최소화함으로써 서버나 클러스터 시스템의 성능을 극대화하고자 하

는 것이다. 메모리 전송에 있어서 이러한 오버헤드를 극복하고자 개발된 것이 U-Net[1], VIA[2], M-VIA[3], InfiniBand[4]과 같은 사용자 수준 통신 방법이다. 또한, 파일 전송에 있어서는 TCP/IP 기반의 sendfile 시스템 콜[5,6]과 같은 무복사 파일 전송방법이다. Sendfile 시스템 콜은 TCP/IP를 기반으로 하는 웹 서버, 클러스터 시스템 등에서 활용되고 있으며, Linux, UNIX, Solaris 등의 다양한 OS에서 제공하고 있다. 그러나 VIA를 기반으로 하는 클러스터 시스템에서는 VIA 프로토콜이 파일 관련 라이브러리에 대해서는 정의하고 있지 않다. 따라서 파일을 전송하기 위해서는 사용자 버퍼로 파일을 한번 복사하고 VI를 사용하여 전송하거나, 또는 파일 전송을 위한 소켓을 따로 사용하여 sendfile 시스템 콜을 호출하여야 한다.

한편, 로컬 노드의 파일을 전송하는 방법 외에 NAS (Network Attached Storage)를 대상으로 파일을 접근하기 위한 연구도 진행되어 왔다. 대표적인 구현 사례로는 TCP/IP 기반의 NFS (Network File System)[7]와 VIA 기반의 DAFS (Direct Access File System)이다[8]. 이러한 연구는 클라이언트 노드에게 라이브러리를 제공하며, 이를 통해 파일 서버의 파일을 접근하도록 해준다. 특히, DAFS는 VIA 프로토콜을 사용하여 파일서버를 접근할 때 발생하는 프로토콜 스택상의 데이타 복사 및 문맥 전환 등의 오버헤드를 줄임으로써 NFS보다 나은 성능을 제공하고 있다[9]. 그러나 DAFS는 전용의 파일 서버를 필요로 하며, 무복사 파일 전송을 위해서는 파일 시스템을 수정해야만 한다. 클러스터 시스템에서는 응용될 수 있는 프로그램에 따라 각각의 노드들이 클라이언트와 서버의 역할을 동시에 수행해야 하는 경우가 있다. 즉, 자신의 메모리를 전송하는 것과 같이 자신의 파일을 클러스터 시스템상의 노드들에 전송해야 하는 경우가 있다. 따라서 TCP/IP의 sendfile과 같이 VIA 기반 클러스터에서도 로컬 노드의 파일에 대하여 무복사 파일 전송을 제공하게 된다면 보다 효율적인 클러스터 시스템을 구축할 수 있게 된다.

본 논문에서는 지금까지 VIA 기반 클러스터 시스템에서 구현된 바 없는 무복사 파일 전송의 메커니즘을 제시하였다. 이를 위해 본 연구에서는 하드웨어 VIA 기반 네트워크 어댑터를 개발하고 및 이를 활용하여 무복사 파일 전송 방법의 실험 결과를 보였다. TCP/IP 기반의 read, mmap, sendfile 그리고 VIA에서의 전송방법과의 비교 실험을 통하여 본 논문에서 제시하는 무복사 파일 전송의 성능을 확인할 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 VIA의 개요에 대해 간략히 설명하고 3장에서는 무복사 파일 전송의 선행 연구로 TCP/IP 기반 및 VIA 기반의 파일 전송 방법을 설명한다. 4장에서는 본 연구에서 구현한 무복사 파일 전송에 대한 메커니즘 및 하드웨어에 대하여 상세히 설명한다. 5장에서는 구현된 무복사 파일 전송에 대한 실험 및 분석을 보이며, 6장에서 결론 및 향후 계획에 대하여 설명한다.

2. VIA 개요

VIA는 System Area Network (SAN) 상에서 높은 대역폭과 낮은 지연시간을 지원하고자 만들어진 사용자 수준 통신 방법이다. 그림 1은 개략적인 VI Architecture를 나타낸 것이다. VIA는 통신시에 발생하는 오버헤드를 줄이기 위해 Virtual Interface(VI)를 사용한다. 각 노드마다 만들어진 VI는 통신의 종단점 역할을 하며, 두 노드간에 만들어진 VI는 점대점으로 연결되어 가상의 통신 채널을 형성하게 된다. VI Kernel Agent는 두 노드 간의 VI 연결을 위해 필요한 연결 관리 그리고 사용자 영역의 메모리 할당 및 관리 등의 커널 서비스를 제공한다. VI User Agent는 데이타 전송, 연결 관리, Work Queue(WQ) 및 Completion Queue(CQ)의 관리, 메모리 할당 및 관리 그리고 에러 처리에 대한 라이브러리를 제공한다.

VI는 Work Queue(WQ)를 가지고, WQ는 Send Queue, Receive Queue로 구성된다. 사용자 프로세스는 WQ에 VI 디스크립터를 써넣음으로써 send/receive를 하게 되며, 도어벨을 통해 NIC(Network Interface Card)에게 알려준다. VI 디스크립터는 NIC에서 처리해야 할 전송에 필요한 제어 정보, 사용자 데이타의 주소 등의 정보를 담고 있다. 전송되어야 할 사용자 데이타는 커널의 간섭 또는 복사본 없이 바로 DMA를 통해 전송된다. 이때 VIA에서 사용자 데이타가 DMA를 통해 전송되기 위해서는 사용자 버퍼가 핀다운 되어야 하고, virtual-to-physical 주소 변환 과정이 필요하다. 그리고

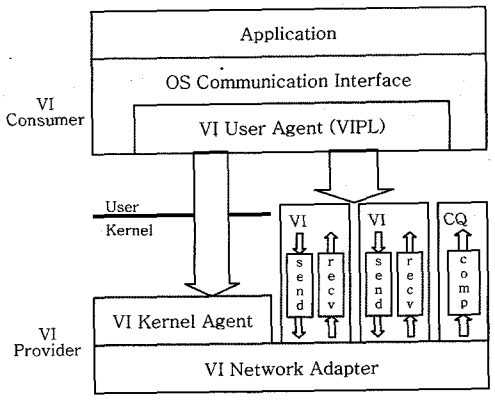

그림 1 VI Architecture

send/receive의 완료는 VI 디스크립터의 "Done bit" 및 Completion Queue (CQ)의 업데이트를 통하여 이루어진다.

3. 기존 파일 전송 방법

커널에서의 프로토콜 처리 관점에서 볼 때 파일 전송의 성능을 좌우하는 요소는 커널과 사용자 공간에서 발생하는 데이타 복사의 횟수와 프로세스 문맥 전환 시간이다. 따라서 이 두 가지 오버헤드를 줄이는 것이 보다 빨리 파일을 전송하는 방법이라 할 수 있다.

그림 2는 TCP/IP를 기반으로 하는 Linux 시스템에서 파일을 보내기 위한 3가지 방법을 나타낸 그림이고, 이를 데이타 복사 관점에서 요약한 것이 그림 3이다. 그림 2(a) *read* 방법은 파일의 내용을 커널에 존재하는 커널 버퍼로 읽어서 사용자 버퍼로 한번 복사를 하고, 네트워크로 전송하기 위해 사용자 버퍼의 내용을 소켓 버퍼로 다시 복사하게 된다(그림 3 ①→②). 이때, 커널의 프로세스와 사용자 프로세스 사이에 문맥 전환이 발생하게 된다. 그림 2(b) *mmap* 방식은 (a) *read*에서 발생하는 커널 버퍼와 사용자 버퍼 사이의 데이타 복사를 없애고 커널 버퍼에서 바로 소켓 버퍼로 복사가 일어나게 된다(그림 3 ③) 이 방법은 커널 버퍼에서 사용자 버퍼로 복사는 일어나지 않지만 사용자 프로세스에서 커널 버퍼를 공유하게 되므로 문맥 전환은 여전히 (a) *read*와 동일한 횟수로 일어난다.

그림 2(c) *sendfile* 시스템 콜은 네트워크를 통해 파일을 효과적으로 전송하기 위해 개발된 무복사 파일 전송 함수이다. Linux 커널 2.1의 *sendfile* 시스템 콜은 사용자 프로세스에서의 버퍼를 활용하지 않으므로 *read*나 *mmap*에 비해 문맥 전환 횟수가 적게 발생하게 되며, 그림 3 ③의 방향으로 한번의 데이타 복사가 일어나게 된다. 이때, *mmap*과 같이 데이타 복사는 같지만 사용자 버퍼를 사용하지 않으므로 문맥전환이 발생하지 않는다. 더 나아가 Linux 커널 2.4에서는 그림 3 ④같이 커널 버퍼의 데이타가 소켓 버퍼로 복사되지 않고, 단지 커널 버퍼의 디스크립터만 소켓에 첨부하여 NIC이 DMA를 통하여 데이타를 읽어가게 함으로써 무복사 파일 전송을 구현하고 있다.

그림 4는 VIA에서의 일반적인 파일 전송 방법을 나타낸 그림이다. 서론에서 밝힌 바와 같이 VIA에서는 파일 전송 라이브러리를 제공하지 않는다. 따라서 일반적인 VIA 기반의 클러스터 시스템에서는 로컬 파일을 전

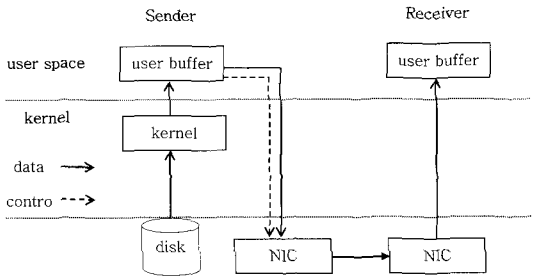

그림 4 VIA에서의 파일 전송 방법

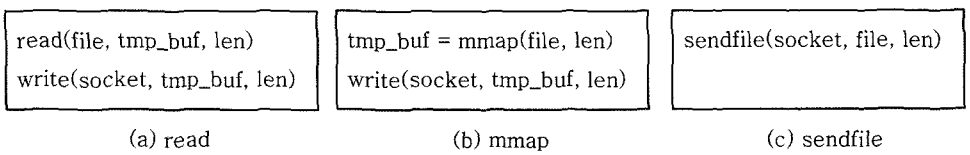

그림 2 TCP/IP에서의 파일 전송 방법

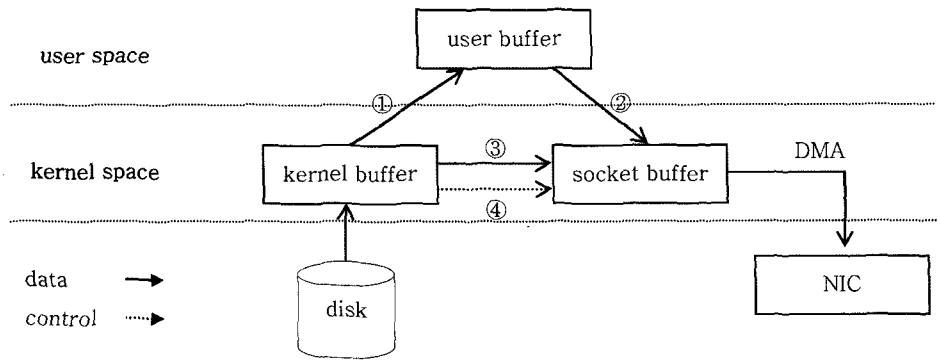

그림 3 TCP/IP에서의 파일 전송 방법에 따른 데이타 복사

송하기 위해서는 커널 버퍼의 내용을 사용자 버퍼로 한 번 복사하고, 이 버퍼에 대하여 VIA 라이브러리에서 제공하는 send/receive 또는 RDMA(Remote Direct Memory Access) write를 통하여 데이타를 전송하게 된다.

4. 무복사 파일 전송 메커니즘 구현

4.1 하드웨어 VIA 기반 네트워크 어댑터

그림 5는 본 연구에서 개발한 하드웨어 VIA 기반 네트워크 어댑터의 블록 다이어그램이다. 기존의 연구인 HVIA-GE[10]를 바탕으로 개발된 이 네트워크 어댑터는 PCI 64bit/66MHz를 지원하고 VIA 프로토콜 엔진을 위한 Xilinx사의 Virtex II Pro 30 FPGA[11]와 고속의 네트워크를 위하여 Intel사의 82544EI 기가비트 이더넷 칩[12]을 장착하고 있다. 특징으로는, 하드웨어적으로 가상주소를 물리주소로 변환하며, 도어벨, RDMA write, 그리고 send/receive 완료를 모두 커널의 간섭 없이 하드웨어로 처리할 수 있다. 특히, 주소변환테이블을 네트워크 어댑터의 SDRAM에 저장하여, VIA 프로토콜 엔진이 직접 주소변환테이블에 접근하여 효과적으로 주소변환을 할 수 있다[10,13].

그림 5의 오른쪽 부분은 하드웨어 VIA 기반 네트워크 어댑터의 핵심 모듈인 VIA 프로토콜 엔진과 기가비트 이더넷 컨트롤러를 나타낸 것이다. VIA 프로토콜 엔진은 Send/Receive FIFO, ATT(Address Translation Table) 매니저, 프로토콜 매니저, RDMA 엔진, 도어벨, 그리고 SDRAM 컨트롤러로 구성되어 있다. VIA 프로토콜 엔진은 PCI 버스를 통해 전달되는 VIA의 함수인 VIPL(Virtual Interface Provider Library)을 처리한다. 사용자 버퍼의 할당에 쓰이는 *VipRegisterMem* 함수를 처리할 때에는 사용자 버퍼의 가상주소와 물리주소 그리고 크기를 전달받아 이를 이용하여 ATT(주소변환테이블)에 저장하게 된다. 사용자로부터 send/receive 함수인 *VipPostSend/VipPostRecv*가 WQ(Work Queue)에 포스팅되면 도어벨을 통하여 VIA 프로토콜 엔진이 인식하게 된다. 그리고 나서 곧바로 VI 디스크립터를 DMA로 읽게 되며, 이를 이용하여 프로토콜 매니저는 ATT 매니저를 통해 사용자 버퍼의 물리주소를 구하게 된다. Send일 경우, 프로토콜 매니저는 호스트 메모리의 사용자 데이타를 DMA를 통해 읽어서 Send FIFO를 통하여 기가비트 이더넷 컨트롤러의 TX 버퍼로 옮기게 되며, receive의 경우에는 RX 버퍼를 통해 도착한 데이타를 Receive FIFO를 통하여 호스트 메모리의 사용자 버퍼 영역으로 DMA 하게 된다. 이러한 send/receive는 원격 노드 CPU의 간섭 없이 통신할 수 있는 RDMA로 구현될 수 있으며, 이때에는 RDMA 엔진이 처리하게 된다.

그림 5에서 보는 바와 같이, 하드웨어 VIA 기반 네트워크 어댑터는 기가비트 이더넷을 위한 MAC(Medium Access Control)을 직접 제어하므로 기가비트 이더넷 컨트롤러는 MAC을 위한 디바이스 드라이버 역할을 하게 된다. 기가비트 이더넷 컨트롤러는 MAC의 초기화, 전송, MAC 관리 및 PCI 버스를 통한 인터페이스를 제공한다. MAC을 직접 하드웨어로 제어하는 기가비트 이더넷 컨트롤러는 복잡하지만 소프트웨어로 제어하는 방식에 비해 MAC의 초기화 및 실제 전송 시간에 있어 이점이 있다.

4.2 무복사 파일 전송 메커니즘

TCP/IP 기반의 *sendfile* 시스템 콜은 데이타 전송을 위해 소켓을 이용하고 있다. 그러나 많은 PC 클러스터 시스템은 TCP/IP의 오버헤드를 개선한 M-VIA와 같은 사용자 수준 통신 방식을 채택하고 있다. 이러한 PC 클러스터 시스템에서 파일을 보내기 위해서는 소켓을 사

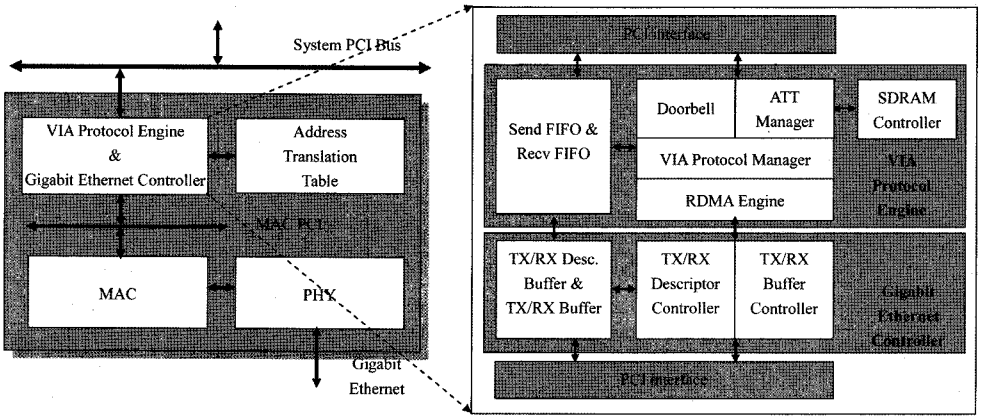

그림 5 하드웨어 VIA 기반 네트워크 어댑터

용하거나 또는 VI를 사용하여 사용자 버퍼로 파일을 한 번 복사하게 된다. 만약 소켓을 사용하지 않고 디스크에서 읽은 커널 버퍼의 내용을 바로 NIC이 읽어가게 된다면 보다 빠른 무복사 파일 전송을 할 수 있게 된다.

그림 6은 본 논문에서 제시하는 PC 클러스터 시스템을 위한 무복사 파일 전송 메커니즘을 나타내고 있다. 그림에서 보듯이, 최종적으로 NIC이 읽어갈 파일 데이타의 위치는 하드디스크에서 파일을 읽어오는 커널 버퍼다. 즉, 전송되어야 할 파일 데이타와 NIC 사이에는 파일을 커널 버퍼로 읽어오는 일만 수행할 뿐, 사용자 버퍼나 소켓 버퍼로의 데이타 복사, 그리고 소켓을 사용하기 위한 일련의 과정은 모두 사라지게 된다. 따라서 파일 데이타를 전송할 때 발생하는 문맥 전환 및 데이타 복사 횟수를 최소화하게 된다. 이와 같은 파일 전송 구현되기 위해서는 다음의 두 가지를 만족시켜야 한다. 첫째, NIC이 커널 버퍼를 접근할 수 있도록 물리주소를 알려주어 커널 버퍼에 대하여 DMA 읽기를 수행할 수 있도록 해야 하며, 둘째, NIC은 DMA 읽기 한 데이타를 네트워크를 통하여 다른 노드로 전송할 수 있는 프로토콜을 수행하여야 한다. 본 연구에서는 4.1장에서 설명한 하드웨어 VIA에 기반한 네트워크 어댑터를 개발하여 PC 클러스터 시스템에서 보다 개선된 파일 전송을 구현하였다.

4.3 세부 구현 내용

4.3.1 함수 인터페이스

그림 7은 본 논문에서 제시하는 무복사 파일 전송의 함수 인터페이스이다. VipSendFile은 하드디스크로부터 읽은 파일을 커널 버퍼로 복사한 뒤, HVIA-GE로 커널 버퍼의 주소와 크기 정보를 전달하는 함수이다. TCP/IP의 *sendfile* 시스템 콜과 유사하나 다른 노드로의 전송을 위해 소켓이 아닌 VI를 사용하게 된다. TCP/IP의 *sendfile*의 경우, NIC은 커널 버퍼의 주소 및 크기 정보, 그리고 원격 노드의 정보를 해당 소켓으로부터 알게 된다. 본 논문의 무복사 파일 전송의 경우에는 ViHandle을 이용하여 전송되어야 할 노드 정보를 알게 되며, 파일 디스크립터와 오프셋, 그리고 길이 정보를 *VipSendFle* 함수를 실행함으로써 알 수 있다. HVIA-GE는 이 주소를 사용하여 커널 버퍼의 데이타를 DMA 읽기하며, 주소와 크기 정보 및 ViHandle을 이용하여 전송되어야 할 노드로 파일을 전송하게 된다.

```
VIP_RETURN VipSendFile(
    VIP_VI_HANDLE ViHandle      //VI Handle
    int fd,                     //file descriptor
    int offset,                 //offset of the file
    int len)                    //length of the file
```

그림 7 VipSendFile

4.3.2 Sender

사용자는 VI를 사용하여 파일을 전송하기 위해 다음의 시나리오로 전개한다. 우선 전송하고자 하는 노드와의 연결을 위해 VI를 생성하고 상대방 노드와 연결을 설정한다. 일반 사용자 버퍼를 send/receive 하기 위해서는 메모리를 등록해야 하지만, sender 측에서는 사용자 버퍼를 사용하지 않으므로 메모리를 등록하는 과정은 생략되며, receiver 측에서는 전송되는 파일을 받아야 할 메모리를 등록한다. 그리고 사용자는 *VipSendFile* 함수를 호출하여 파일 전송을 한다.

이때 내부적으로 일어나는 과정은 다음과 같다. 이 함수는 보내야 할 파일을 커널 버퍼로 읽은 다음, 파일 전송을 알리는 도어벨을 울리고, NIC에게 커널 버퍼의 물리주소 및 보내야 할 파일의 크기 정보를 알려준다. 일반 send/receive에서는 WQ에 VI 디스크립터를 포스팅해주지만, 파일 전송의 경우엔 이러한 과정이 모두 생략

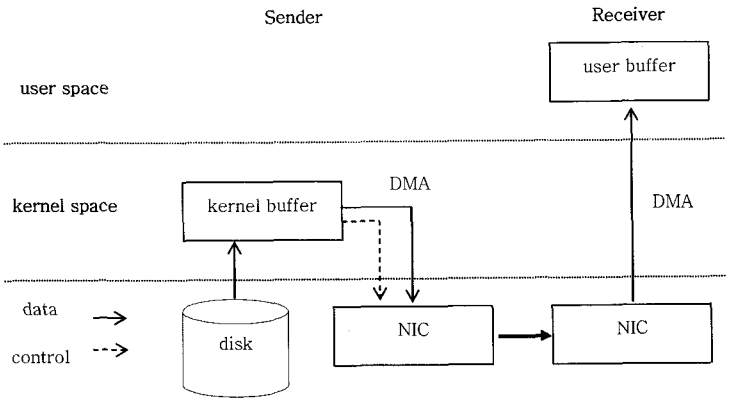

그림 6 무복사 파일 전송 메커니즘

되며, VI_handle, 커널 버퍼의 물리주소 및 크기 정보만을 네트워크 어댑터에게 알려준다. 네트워크 어댑터는 받은 물리주소를 내부 버퍼에 저장하고 커널 버퍼의 각 chunk(페이지 단위)에 대하여 DMA 읽기를 하게 된다. DMA 읽기 된 커널 버퍼의 내용은 이전에 형성된 VI 연결을 통하여 상대방 노드로 전송된다. 전송이 완료되면 NIC은 인터럽트를 통하여 완료를 알리게 된다.

4.3.3 Receiver

TCP/IP 기반 *sendfile*의 경우 receiver 노드에서 해야 할 작업에 대하여 정의하고 있지 않다. Receiver 노드에서 받은 파일 데이타는 파일 또는 사용자 버퍼로 다시 저장하게 되는데, 이러한 작업은 모두 소켓을 통해 받은 버퍼에서 이루어진다. 본 논문에서 구현한 *VipSendFile*의 경우에는 receiver 노드에서 VI를 사용하는 *VipPostRecv* 함수를 통해 파일 데이타를 사용자 버퍼로 직접 전송하게 되며, 만약 이 데이타를 파일로 저장할 경우에는 다른 복사 없이 곧바로 파일로 저장할 수 있게 된다. 따라서 receiver 노드에서는 다른 오버헤드 없이 VIA에서 제공하는 VIPL을 변경 없이 사용 가능하다.

4.3.4 완료

일반적으로 네트워크상의 send/receive 트랜잭션의 완료를 나타내기 위해서는 폴링과 인터럽트 두 가지 방법을 사용한다. 대부분의 네트워크 어댑터는 전송이 완료되면 인터럽트를 사용하여 알려주게 되며, 이 때 문맥 전환이 발생한다. VIA에서는 두 가지 방법을 모두 정의하고 있으며, M-VIA에서는 인터럽트를 사용하지만, 폴링 방식 또한 문맥 전환을 줄이는 효과가 있으므로 효율적이라 할 수 있다. 파일 전송의 경우 디스크에 접근하기 위한 초기 시간이 크다. 따라서 지연시간 보다는 CPU 사용률을 낮추는 것이 중요하다. 본 논문의 하드웨어 VIA 기반 네트워크 어댑터는 메모리 전송에 대하여 폴링 및 인터럽트 모두를 사용하고 있으나, 파일 I/O의 특성상 인터럽트를 사용하여 보다 효율적인 전송을 할 수 있도록 구현하였다.

5. 실험결과

본 논문에서 구현한 하드웨어 VIA 기반 네트워크 카드 및 무복사 파일 전송의 성능 측정을 위하여 두 대의 2GHz AMD Opteron 246 PC, 64bit/66MHz PCI 버스 기반의 클러스터에 본 연구에서 개발한 VIA 기반 네트워크 어댑터를 사용하였다. 운영체제로는 Linux 커널 2.4.22에 기반한 x86_64 Fedora core 1을 사용하였다. 비교 실험을 위하여 Intel PRO/1000 MT 기가비트 이더넷 카드를 사용하였다. 하드웨어 VIA 기반 네트워크 카드와 TCP/IP 그리고 M-VIA를 비교를 위해, 그리고 무복사 파일 전송 비교 실험에서 TCP/IP 기반의 *sendfile*과의 비교를 위해 이 기가비트 이더넷 카드를 사용하였다. 다만 무복사 파일 전송 비교실험에서는 2장에서 밝힌 세 가지 방법 중 *sendfile*의 성능이 가장 나으므로 *read, mmap* 방법의 성능은 제외하였다. 그리고 기존의 VIA 파일 전송 방법과 비교를 위하여, 위와 동일한 PC 클러스터 환경에서 성능을 측정하여 비교 분석하였다. 기본적인 성능 측정 프로그램으로 M-VIA에 포함된 *vnettest* 프로그램을 이용하였고, 파일 전송 및 TCP/IP에 적용하기 위해 필요한 수정을 하였다.

5.1 하드웨어 VIA 기반 네트워크 카드 성능분석

5.1.1 지연 시간 및 대역폭

그림 8과 그림 9는 HVIA-GE, M-VIA, 그리고 TCP/IP를 사용하였을 때 지연시간 및 대역폭을 나타낸 그래프이다. 이때 패킷의 크기는 이더넷 MTU (Maximum Transfer Unit) 크기인 1,514 바이트를 사용하였고, 완료 시 폴링방식을 사용하여 최소 지연시간을 나타내도록 하였다. 지연시간은 두 노드 간에 전송된 시간을 반으로 나눈 것이며, 대역폭은 이를 전송된 데이타 크기로 나눈 값이다. TCP/IP, M-VIA, 그리고 HVIA-GE의 최소 지연시간은 각각, 27 μs, 21.3 μs, 그리고 8.2 μs를 나타내었다. 따라서 HVIA-GE의 최소 지연시간은 TCP/IP 및 M-VIA에 비해 각각 3.3배 및 2.8배 빠른 것을 보여준다. 또한 1MB의 데이타를 전송하였을 때의

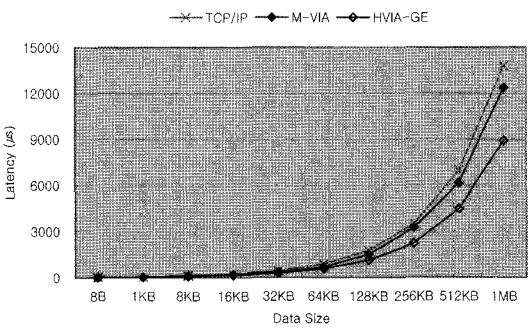

그림 8 지연시간 비교

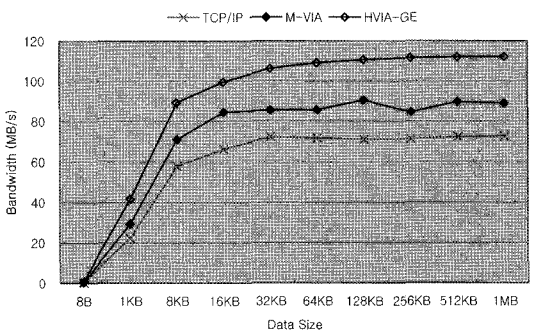

그림 9 대역폭 비교

최대 대역폭은 TCP/IP, M-VIA, 그리고 HVIA-GE 각각 72.5 MB/s, 90.5 MB/s, 그리고 112.1 MB/s를 나타내었으며, TCP/IP 및 M-VIA에 비해 HVIA-GE가 55% 및 24%의 증가를 나타내었다.

그림 10은 HVIA-GE의 최소 지연시간에 대한 분석을 나타낸 그림이다. 이 분석 그림은 두 노드 간에 send/receive를 실행한 것을 바탕으로 측정한 것이다. 우선 sender 및 receiver 공통적인 부분으로, 사용자 프로세스는 데이타를 전송하기 위해 send/receive 함수(*VipPostSend/VipPostRecv*)를 각각 실행하게 된다. 이때 가장 먼저 일어나는 일은 Send/Receive Queue에 VI 디스크립터를 포스팅하는 것으로, 이때 걸린 시간은 0.9 μs이다. 그런 다음 receiver는 sender로부터 데이타가 도착할 때까지 기다리게 된다. 그 다음 sender에서는 Send Queue로 포스팅된 디스크립터를 처리하여 receiver로 데이타를 보내게 되는데, 이때 HVIA-GE 카드에서 처리하는 시간은 2.4 μs이다. 이 기간에는 HVIA-GE 카드의 VIA 프로토콜 엔진이 VI 디스크립터를 읽어서, 보내야 할 사용자 버퍼의 물리주소를 계산하여 사용자 버퍼를 읽게 된다. 그리고 기가비트 이더넷 컨트롤러는 사용자 버퍼를 읽은 데이타를 TX 버퍼로 옮기고 MAC에게 데이타를 전송할 것을 알리게 되며, 동시에 VIA 프로토콜 엔진은 send의 완료를 수행한다. Sender의 MAC이 TX 버퍼를 읽어서 receiver의 RX 버퍼로 데이타를 옮기는 시간은 약 3 μs이며, 이 부분의 시간 측정은 두 노드의 HVIA-GE 카드에 동시에 로직 애널라이저를 장착하여 측정하였다.

Receiver에서는 VI 디스크립터의 처리 및 사용자 버퍼의 주소 변환 시간은 포함되지 않는데, 이 시간은 모두 sender에서 VI 디스크립터를 처리하고 주소 변환하는 시간과 동시 또는 그 이전에 발생하기 때문이다. 따라서 receiver에서 소요되는 시간은 RX 버퍼로 도착한 데이타를 기가비트 이더넷 컨트롤러 및 VIA 프로토콜 엔진을 거쳐 호스트 PCI 버스로 옮기는데 1 μs, 그리고 완료를 위한 시간 약 1 μs 이하의 시간이 걸리게 된다. 이러한 지연시간 분석에서도 알 수 있듯이, send/receive 시에 발생하는 대부분의 작업을 HVIA-GE 카드 내에서 처리하게 되며, 결과적으로 통신시 호스트에서 발생하는 시간을 최소화하여 낮은 지연시간 및 높은 대역폭을 얻을 수 있었다.

5.1.2 CPU 사용률

그림 11은 TCP/IP, M-VIA, 그리고 HVIA-GE의 CPU 사용률을 나타낸 그래프이다. 이 실험에서는 완료시까지 CPU가 계속 폴링하지 않도록 인터럽트 방식을 사용하였다. CPU 사용률에 영향을 끼치는 요소로는 데이타 복사, 문맥 전환, 시스템 콜 등 CPU에서 소요되는 작업들이다. TCP/IP는 데이타 크기가 커짐에 따라 CPU 사용률이 커지고 55%~60% 정도에서 수렴한다. M-VIA의 경우에는 페이지 하나 이하의 작은 크기의 메시지에서 지연시간을 줄이기 위하여 인터럽트 대신에 폴링을 이용하여 완료하게 되므로[14], 초기에는 거의 100%를 나타내다가 약 30% 정도로 수렴하게 된다. HVIA-GE의 경우 최대 14%에서 10KB 이상의 데이타

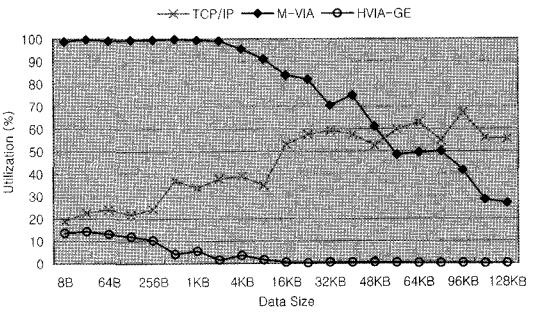

그림 11 CPU 사용률

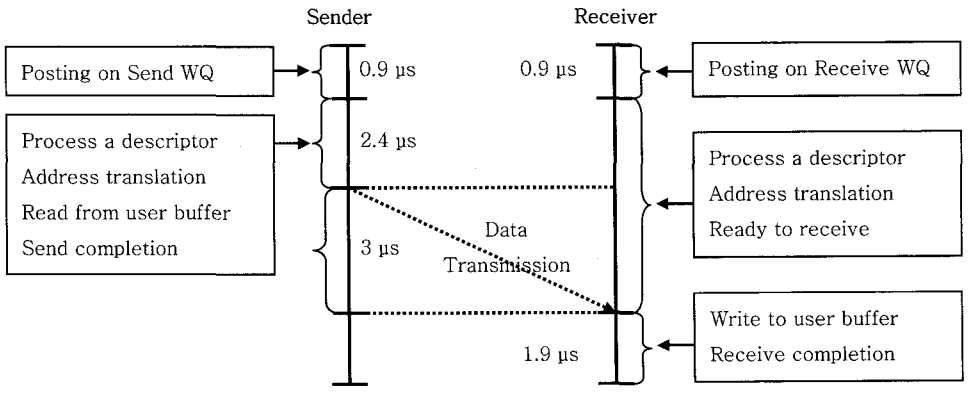

그림 10 지연시간 분석

에서는 1% 이하의 CPU 사용률을 나타낸다. 이는 위에 언급한 CPU 작업들이 최소화되어 HVIA-GE에서 대부분 처리하고 있음을 나타낸다.

5.2 무복사 파일 전송 메커니즘 성능 분석

5.2.1 Sender에서의 지연시간

그림 12는 기존의 TCP/IP 및 VIA에서의 방법과 본 논문에서 구현한 *VipSendFile*에 대하여 sender에서의 지연시간을 나타낸 그림이며, 그림 13은 이때 CPU에서 걸린 시간을 측정한 그림이다. TCP/IP-sendfile은 TCP/IP 기반의 *sendfile* 시스템 콜 방법을 나타내며, HVIA-send/receive는 일반적인 VIA의 전송 방법을, 그리고 HVIA-VipSendFile은 본 연구에서 개발한 무복사 파일 전송 방법을 나타낸다. 여기서 측정된 지연시간은 파일을 읽기 시작한 시점부터 NIC에서 전송완료 메시지가 도착한 시점까지의 시간을 측정한 것이며, 각 방법에 따라 커널 버퍼 및 사용자 버퍼의 데이타 복사 및 문맥 전환 시간, 그리고 완료 시 인터럽트 처리 시간이 포함되어 있다. 이 실험에서 고려되어야 할 사항은 파일을 하드디스크에서 읽을 때 초기 접근시간이 아주 크다는 것이다. 따라서 파일 전송 방법 자체의 성능 분석을 위해서는, 하드디스크에서 커널 버퍼로 데이타를 읽는 초기 시간을 무시할 수 있도록 해야 하며, 이를 위해 본 실험에서는 100,000번 반복하였다.

그림 12에서 TCP/IP 기반의 *sendfile* 시스템 콜은 데이타 크기가 커짐에 따라 VIA 기반의 방식보다 더 많은 시간 차이를 나타내고 있다. 본 논문에서 구현한 *VipSendFile*이 *sendfile* 시스템 콜에 비해 47%~52% 정도 성능 향상을 나타냈다. 이는 *sendfile* 시스템 콜이 무복사 프로토콜이기는 하지만 TCP/IP 프로토콜을 처리하는데 걸리는 시간이 많음을 나타낸다. 또한 그림 13에서처럼 거의 같은 비율로 CPU에서 시간이 걸림을 알 수 있다. 그림 12에서 VIA 기반의 두 방법을 비교하면 HVIA-VipSendFile이 HVIA-send/receive에 비해 약 3% 정도의 성능향상을 나타내고 있다. CPU 시간을 비교해보면 HVIA-VipSendFile이 HVIA-send/receive에

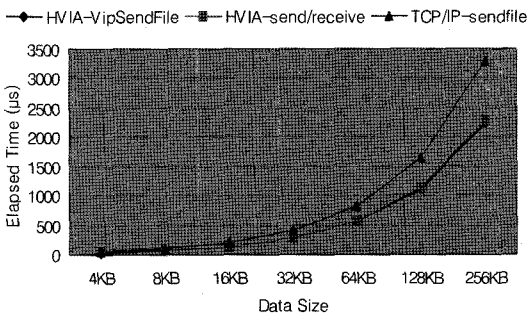

그림 12 Sender에서의 지연시간

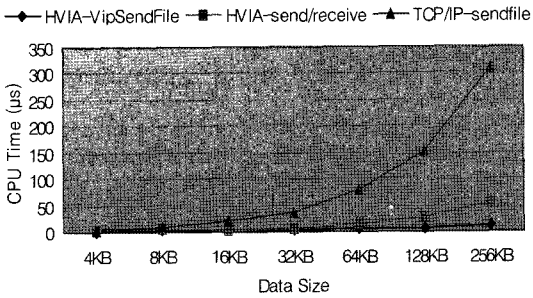

그림 13 Sender에서의 CPU 시간

비해 256KB의 데이타에 대해 45 μs 정도 빠르게 수행되고 있는데, 이와 같이 수행시간의 차이는 크지 않지만 CPU 시간은 중요한 의미를 가진다. 이는 그림 13에서 기존의 VIA 방식은 커널 영역에서 사용자 영역으로 복사 및 문맥 전환이 발생하여 이에 따라 CPU 시간이 증가하는 것을 의미하며, *VipSendFile*은 복사 및 문맥전환 오버헤드를 줄임으로써 데이타 크기가 증가하더라도 CPU 시간의 증가폭이 극히 미미하다.

5.2.2 CPU 사용률

그림 14는 3가지 방법에 대한 CPU 사용률을 나타낸 그래프이다. TCP/IP 기반의 *sendfile* 시스템 콜의 경우 데이타 크기와는 상관없이 8%~10% 정도 나타낸다. HVIA-send/receive의 경우 데이타 크기가 6KB 이하일 때 CPU 사용률이 TCP/IP-sendfile 보다 높게 나타난다. HVIA-send/receive의 지연시간이 TCP/IP-sendfile 보다 낮게 나타나는데도 불구하고 이러한 현상이 나타나는 것은 데이타 크기가 6KB 이하일 때에는 커널 버퍼에서 사용자 버퍼로 복사되는 CPU 시간이 상대적으로 크기 때문이다. 그러나 HVIA-send/receive와 HVIA-VipSendFile은 데이타가 커짐에 따라 CPU 사용률이 점점 낮아지고 있으며, 이는 그림 11의 경향과 비슷한 것을 알 수 있다. 이는 본 연구의 HVIA-GE는 데이타 전송이 하드웨어로 이루어지므로 CPU 사용률이 TCP/IP에 비해 현저히 낮게 나타난다. 더구나 HVIA-

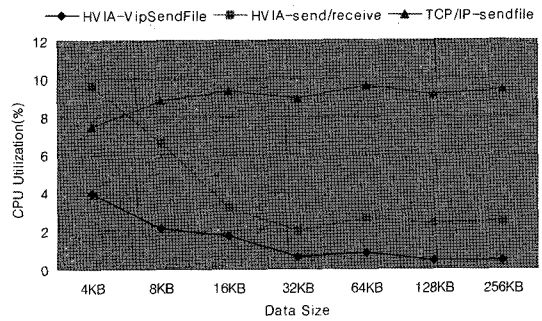

그림 14 CPU 사용률

VipSendFile의 CPU 사용률은 HVIA-send/receive 보다 약 30%~40%의 측정값을 나타내었고, 이는 두 번의 문맥전환과 한 번의 데이타 복사를 줄였기 때문이다.

6. 결론 및 향후 계획

본 논문에서는 VIA 기반 클러스터 시스템을 위한 하드웨어 VIA 기반 네트워크 카드와 이를 기반으로 무복사 파일 전송 메커니즘을 구현하였다. 본 논문의 하드웨어 VIA 기반 네트워크 어댑터는 PCI 64bit/66MHz 및 기가비트 이더넷을 지원하고 VIA 프로토콜을 하드웨어로 구현함으로써 8.2 μs의 지연시간, 121 MB/s의 대역폭, 그리고 아주 낮은 CPU 사용률을 보여준다. 또한 본 논문의 파일 전송 메커니즘은 파일시스템의 수정 없이 파일 전송 라이브러리만 제공함으로써, 네트워크 인터페이스 카드가 보내고자 하는 노드의 파일을 상대방 노드의 사용자 버퍼로 복사 없이 전송 가능케 하였다. 이러한 구현의 결과로, sender 측의 데이타 복사 및 문맥전환 오버헤드를 줄여 TCP/IP의 *sendfile*에 비해 47%~52%의 성능향상을 가져왔으며, VIA 기반 send/receive 방식에 비해 30%~40%의 CPU 사용률을 나타내었다. 향후 3대 이상의 노드를 대상으로 VOD 스트리밍 서비스와 같은 다양한 파일 관련 응용프로그램에 적용해 봄으로써 기가비트 이더넷 기반의 클러스터 시스템에서 보다 나은 성능을 제공할 수 있을 것이다.

참 고 문 헌

[1] 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.

[2] Virtual Interface Architecture Specification 1.0, http://www.viarch.org/

[3] P. Bozeman and B. Saphir, "A Modular High Performance implementation of the Virtual Interface Architecture," Proc. Of the 2nd Extreme Linux Workshop, June 1999.

[4] InfiniBandTM Architecture, http://www.infinibandta. org/

[5] Dragan Stancevic, "Zero Copy I: User-Mode Perspective," Linux Journal, Volume 2003 Issue 105, January 2003.

[6] Jeff Tranter, "Exploring The Sendfile System Call," Linux Gazette, Issue91, June 2003.

[7] Shepler, S., et. al. NFS version 4 Protocol, Internet Engineering Task Force RFC3010, December 2000.

[8] DAFS Collaborabive, Direct Access File System Protocol, Version 1.0, September 2001, http://www.dafscollaborative.org

[9] A. Feforova, M. Seltzer, K. Magoutis, S. Addetia, "Application performance on the Direct Access File System," ACM SIGSOFT Software Engineering Notes, Proc. of the 4th international workshop on Software and Performance, vol 29, January 2004.

[10] S. Park, S.H. Chung, I.S. Yoon, I.H. Jung, S.M. Lee, B. Lee, "HVIA-GE: A Hardware Implementation of Virtual Interface Architecture Based on Gigabit Ethernet," Lecture Notes on Computer Science 3280, October 2004.

[11] Xilinx Virtex II Pro 30, http://www.xilinx.com

[12] Intel Corporation, 82544 Gigabit Ethernet Controllers with Integrated PHY, http://www.intel.com/design/network/products/lan/controllers/82544.htm

[13] 박세진, 정상화, 윤인수, "HVIA-GE: 기가비트 이더넷에 기반한 Virtual Interface Architecture의 하드웨어 구현", 한국정보과학회논문지: 시스템 및 이론 vol. 31, no. 5.6, June 2004.

[14] M-VIA Core Release 1.2, http://old-www.nersc.gov/research/FTG/via/

박 세 진

1998년 부산대학교 컴퓨터공학과 학사 2000년 부산대학교 컴퓨터공학과 석사 2000년~2005년 부산대학교 컴퓨터공학과 박사. 2005년~현재 삼성전자 정보통신총괄 무선사업부 책임연구원. 관심분야는 클러스터시스템, 병렬처리, VIA, WiBro, HSDPA

정 상 화

정보과학회논문지 : 시스템 및 이론 제 32 권 제 6 호 참조

최 봉 식

2004년 부산대학교 전자전기정보컴퓨터공학부 학사. 2000년~현재 부산대학교 컴퓨터공학과 석사과정. 관심분야는 클러스터 시스템, VIA, RDMA

김 상 문

2004년 부산대학교 전자전기정보컴퓨터공학부 학사. 2000년~현재 부산대학교 컴퓨터공학과 석사과정. 관심분야는 클러스터 시스템, VIA, VOD