VIA기반의 통신 인터페이스 개발
이상기* 아름영** 서대희**
경북대 소프트웨어 공학과* 경북대 전자공학과**
{leesky, yylee}@palgong.knu.ac.kr, dwsen@ee.knu.ac.kr

Development of Communication Interface Based on VI Architecture
Sang-Ki Lee* Yoon-Young Lee** Dae-Wha Seo**
Dept. of Information and Communication*, Kyungpook National University
Dept. of Electronics**, Kyungpook National University

요약

hardt웨어와 소프트웨어의 발전과 함께 컨버터의 차이 배달 데터베이스가 크게 증가하고 있다. 클러스터 대내의 node들 사이에서 이런 대용량의 데이터를 보다 빠르게 전송하기 위해서 Lightwight Messaging 기법이 등장하였으며, 대표적으로 AM, FM, U-Net, VIA등이 있다. 이 중에서 VIA는 케일 수준에 구현된 TCP/IP를 대체해서 사용자 수준에서 커뮤니케이션을 저하시키고 네트워크 장애가 직접적으로 통신을 할 수 있게 하여 다양한 분야에서 사용되고 있으며, 새로운 프로토콜의 표준으로 자리잡아가는 추세이다. 그러나 이러한 장점에도 불구하고 프로그래밍 능력이 낮은 장점 때문에 제대로 코스터지하기까지는 많은 시간이 필요한 것이 사실이다. 이에 이 논문에서는 EIVL(Easy-to-use Virtual Interface Library)이라는 개발자의 편을 하여 쉽게 접근할 수 있는 라이브러리를 제안하였다. 그리고 EIVL, Native VIA, TCP/IP로 각각 같은 역할을 하는 프로그램을 작성하여 기존의 프로토콜들과 상용화를 비고하였다.

1. 서론

hardt웨어와 소프트웨어 기술 발전에 따라, 커뮤터 시스템에서 처리해야 할 데이터의 크기 및 그 수도 늘어나고 있다. 따라서 데이터의 전송을 추적하고 하는 데, 대용량의 데이터를 주로 사용하는 것이 필요하다. 특히 SAN(System Area Network)에서는 안전적인 데이터 전송에 가를 수배하는 TCP/IP의 다양한 투명 제어 메커니즘들이 오래 오래 해드로 작업하기 도 하였다[1][5].


이들 새로운 프로토콜들은 주로 주관적으로 패턴 전송 공중의 병목에 의한 문제를 줄이기 위해 (zero-copy) 메시지의 전송 속도를 높이는 것과 기존 프로토콜을 단순화시켜야 하는 것에 있다[5][10]. Lightweight Messaging 기법은 도입함으로써 수송 할 수 있는 분량은 병용 프로토콜에 비해 오래 해드가 작아서 전송 능력이 높아지며 프로토콜 처리 부하가 증가하지 않아서 문제 해결이 크게 개선되었다. 이런 장점들로 인해 앞에서 설명한 새로운 프로토콜들은 점점 그 활용도가 높아지고 있다.

그 중 VIA는 TCP/IP와의 프로토콜을 대체해 CPU가 단락

한 네트워크 통신 오버 해드를 제거해 줄 수 있고, 실제 기업

환경에 사용할 수 있는 시스템 수준의 고속 통신을 가능하게 하지만, 응용 프로그램 수준에서 CPU를 거치지 않고 네트워크 장애와 직접적으로 통신을 할 수 있도록 해줄 수 있다. 이런 점 때문에 현재 VIA가 널리 사용되고 있는 기존 프로토콜의 단점에 대비해 수용되기를 기대해 보고 한다.

2. Virtual Interface Architecture

VIA는 Compaq, Intel, Microsoft의 3개의 기업이 케이스터 환경이나 SAN에서 사용할 고성능의 네트워크 기반을 위한 인터페이스 표준의 필요에 따라서 제안한 것이다[5][8].

VIA의 주요점은 실제 기업 환경에 사용할 수 있는 시스템 수준의 고속 통신을 가능하게 하기 위한 것이며 이는 응용 프로토콜 수준에서 CPU를 거치지 않고 네트워크 장애와 직접적으로 통신을 할 수 있도록 해서 코스터로 사용되는 TCP/IP 등의 프로토콜에서 CPU가 단락한 네트워크 통신 오버 해드를 제거하려 한다.

VIA의 특성은 다음과 같다.

- 각각의 VI는 통신상의 장애이며 이로인한 종결점은 점 테

턴 통신을 논리적으로 연결시켜 준다.

- 하나의 프로세스는 하나 이상의 VI를 가지고 있다.

- 네트워크 데터는 통신상의 종결점이 가상화하고, VI사이의 신호성 있는 데이터 전송을 담당한다.

- VI 구조는 기본적으로 가상 인터페이스 위에, VI 제공자, VI 소비자의 4개의 모듈로 구성된다[7].

그림 1은 VIA와 TCP/UDP/IP의 차이를 보여 주고
2001년도 한국정보과학회 가을 학술발표논문집 Vol. 28, No. 2

그림 1. TCP/IP와 VIA의 비교

반면, VIA의 사용자 프로세스는 VI를 통하여 운영 체제의 개방성이 없어 안전하게 네트워크 인터페이스에서 직접 접근해서 사용자가 개방형 시스템 비밀 사이의 데이터의 복사, 해킹 등의 패키지를 억제할 수 있기 때문에 보호 기능이 커널 수준에서 제공되었으나, 이러한 사용자 프로세스가 직접 다이렉토리 접근하는 경우에는 가장 주의를 해야할 부분의 보호 기능이 VIA가 제공하였다.

그림 2. EVIL 함수의 역할

OPEN은 채널을 생성하는 역할을 하며 via_open() 함수로 구현되었다. VIA 채널을 생성하는 단계에서는 NIC(Network Interface Card)에 접근하여 사용 권한을 확보하고, VI를 생성하는데 필요한 여러 속성을 설정하고, 응용 프로그램에서 필요한 메모리 영역을 동록하게 하며, 마지막으로 VI를 생성하도록 한다.

WAIT은 개방형 채널을 통해 다른 노드로부터의 연결을 대기, 수락하여 VipConnectWait()와 VipConnectAccept() 함수를 한 함수 내에서 처리하도록 via_wait() 함수로 구현하였다.

CONNECT는 via_connect() 함수로 구현되었는데 접속 대기 중인 상대방 허스트에 연결을 요청한다. 다른 노드에서 연결을 하기 위해 사용하는 함수로 내부적으로 VipConnectRequest()함수를 호출하여, 접속 대기 중인 상대방 허스트에 연결을 요청한다.

SEND와 RECV는 각각 via_send()와 via_recv() 함수로 구현되었고, VIA 채널을 통해 데이터를 송수신하고 응용 프로그램을 관리한다.

CLOSE는 개방형 VIA채널을닫는 역할을 하며 via_close() 함수를 사용하여 구현되었다. VIA채널을 닫기 위해서는 데이터의 전송이 끝난 후 VI를 통해 닫을 받는 자원들을 시스템에 돌려주는 과정이 필요하다. 먼저 복잡한 메모리를 반납해야 하며, 보호 테그(protoco)도 반납하여야 한다. VIA 채널을 통해 연결을 끊어내야 하기 때문에, VI가 파괴되고, NIC의 사용 권한도 반납하여야 한다. 이러한 작업들은 동시에 하나의 함수를 통해서 수행하도록 하였으며, 응용 프로그램의 작동이 한계적으로 이루어진다.

EVIL에서는 데이터 송수신을 위해 다중 비버를 탐색하고, 이를 라운드로 보낸 방식으로 via_send()는 응용 프로그램으로부터 송신 요청이 들어나면 데이터가 있는 메모리 공간을 읽은 후 이를 가리키는 디스크립트를 포스트 해주도록 작성하였다. via_recv()는 대기 중인 여러 개의 디스크립트 중 데이터 수신이 끝났을 때 서로 갈아짐에 따라서 메모리와 데이터를 다시 차례로 포스트하도록 한다. 페어의 복잡한 메모리 비버 영역(mem_buff)을 채널이 가질 수 있었다.

3. 사용자 편의성을 고려한 VIA 라이브러리

3.1 EVIL의 특성

VIA의 특성과 VI의 특성은 VIA 클라이언트가 제공하는 VI의 API(Virtual Interface Programming Library API)를 사용하여 프로그램을 작성하게 되고, VIA의 특성을 자유롭게 조절할 수 있도록 VIA 연구진 개발자들이 VIA를 이용한 고성능의 응용 프로그램을 작성할 수 있는 API를 정의하고 이를 개발하였다.

3.2 EVIL API의 구현

EVIL API를 구현하여 사용자 편의성을 높이는 VI의 고성능 통신에 필요한 여러 단계들을 직접적이고 단계로 묶고 단단한 함수 메트로 정의하였다.

이러한 EVIL을 구현하는 함수들 각 함수들의 역할을 간단히 나열할 것이다.
4. 실험 및 결과

4.1 실험 환경

실험을 위하여 4대의 Intel Pentium III-600Mhz 시스템을 사용하였다. 실험에 사용된 리눅스는 키널 2.2.14버전으로 VIA를 제공하는 환경으로는 Gigant 샤이 cLAN을 사용하였고, Gigant 샤이 cLAN은 cLAN5000 NIC상에서 동작하는 것으로 하드웨어적으로 VI를 지원하는 NIC이다.

4.2 성능 평가

성능 평가를 위하여 EVIL과 VIPL, LANELAN Emulation[31]을 이용하여 각각 동일한 크기의 메시지를 전송한 경우를 반복하여 전송하여 그 성능을 측정하는 프로그램을 작성하였다.

그림 3에 나타난 결과는 전송하는 데이터의 총량을 1MB에서 500MB로 늘어가면서 평균 전송 대역폭을 측정하였다. 이 때 전송시간은 메시지의 크기 16KB로 고정하였다. EVIL을 사용하였을 때 Native VIA를 사용한 경우보다 약 5~10%의 대역폭 손실이 있지만 다른 밸런스 트래픽 송수신에 IP를 경유하는 LAN을 이용한 경우보다는 29~62%의 성능 개선이 있음을 확인할 수 있었다. 그리고 전송하는 데이터의 양이 커지더라도 일정 수준의 성능을 나타내고 있었다.

그림 4는 20MB의 메시지를 전송하는 데, 반면에 전송하는 메시지의 크기를 변화시키면서 그 성능의 변화를 살펴보았다. 메시지의 크기가 클수록, 성능이 향상되는 것을 확인할 수 있는데, 이것은 메시지의 크기가 클수록 동일한 크기의 메시지를 보내고 있는 환경이 많아지기 때문이다. 반면에 전송하는 메시지의 크기를 변화시키면서 결과 약 7~17% 정도의 손실이 발생하였다.

5. 결론

본 논문에서는 SAN환경에서 Lightweight Messaging 기법의 일종인 VIA를 이용하는 동물 프로그램의 작성을 용이하게 하기 위해 라이브러리, 즉 EVIL을 구현하고 그 성능을 평가하였다.

EVIL를 사용하였을 때 Native VIA의 API 세트를 VIA를 이용하여 작성한 프로그램의 약 1/3에 해당하는 코드를 단순화하는 프로그램을 작성할 수 있었고, 그 성능 또한 크게 차이가 나지 않았다. 이는 그만큼 동물 프로그램의 개발 속도가 빨라짐을 의미하며 더이상이 용이하게 하였음을 의미한다.

 특히 EVIL의 경우는 매우 간단하면서도 효과적인 방법을 이용함으로써, VIA를 이용한 프로그램을 작성하는 것이 매우 간편해졌다. 간편함을 추구하기 때문에 성능 저하가 우려되었지만, 실험을 통해 확인한 결과 Gigant 샤이에서 제공하는 예제 프로그램에 비해 성능이 크게 떨어졌지 않았음을 나타낸 Lightnet Message로의 기능을 중분히 하고 있음을 확인할 수 있었다.

특히는 파일 전송을 위한 보다 다양한 허브 장치의 방법들을 내부적으로 적용하여 작은 크기의 메시지의 전송에도 높은 성능을 발휘하며, 전체적으로 더 안정적이며, 높은 처리율을 나타내는 연구가 계속될 것이다.

참고 문헌