Virtual Interface Architecture의 구현

유정록[0] 이문상 맹승렬
한국과학기술원 전산학과
{jlyu, mslee, maeng}@calab.kaist.ac.kr

An Implementation of Virtual Interface Architecture

Jung-Lok Yu[0] Moon-Sang Lee Seung-Ryoul Maeng
Division of Computer Science Department of EECS, Korea Advanced Institute of Science and Technology

요 약

클러스터 컴퓨팅에서 성능 개선의 한계점으로 인식되는 것은 클러스터 내의 통신 서브시스템(communication subsystem)이다. 이러한 한계점을 해결하기 위해서 사용자 수준 통신 프로토콜이 제안되었고 Microsoft, Compaq, Intel 3사에 의해서 표준화되었다. 본 논문에서는 사용자 수준 통신의 산업계 표준으로 선정된 Virtual Interface Architecture(VIA)[1] 의 개요에 대해서 알아보고, VIA를 효과적으로 구현한 KVIA(Kaist-VIA)에 대해 기술한다.

1. 서론

최근 들어 고성능 PC나 워크스테이션의 가격이 하락하고 네트워크 기술(network technology)은 비약적으로 발전하였다. 고속의 값싼 PC나 워크스테이션을 고속의 네트워크으로 연결하여 고성능과 확장성을 제공하고자 하는 클러스터 컴퓨팅(cluster computing)에 대한 연구가 활발하게 진행중이다. 그러나 클러스터 컴퓨팅에서 성능개선의 제한이 되는 주요 원인은 클러스터 내의 통신 서브시스템(communication subsystem)이다. 기존의 TCP/IP와 같은 통신 프로토콜은 여러 개의 스택계층(stack layer)으로 구성되어 있어서 메시지를 송수신하는 과정에서 큰 자연시간을 가져온다. 또한 커널이 메시지 송수신을 위해 수행하는 데이터 복사(data copy) 및 시스템 콜(system call) 등은 큰 통신 오버헤드로 작용한다.

이러한 통신 오버헤드를 없애고 고성능의 통신 서브시스템을 제공하기 위해서 사용자 수준 통신 계층(user-level communication)[2]이 제안되었다. 사용자 수준 통신 계층은 메시지의 송수신과정에서 커널의 관여를 배제하고 프로세스(process)와 네트워크 장치(network device) 사이에 직접적으로 메시지를 송수신한다. 지금까지 VMMC, VMMC-II[3], GM, U-NET[4], BIP[5], FM[6], AM 와 같은 많은 사용자 수준 통신 프로토콜이 제안되었다. 이들 프로토콜들은 각각의 특성과 기능성을 가지고 있고 나타내는 성능 또한 다양하다.

이런 다양한 사용자 수준 통신 프로토콜들을 산업계 선두주자인 Microsoft, Compaq, Intel 의 3개 회사가 표준화한 것이 Virtual Interface Architecture(VIA)이다.

* 본 연구는 국가지정연구실사업 지원을 받아 수행되었음

본 연구에서는 사용자 수준 통신 프로토콜의 표준인 VIA를 Myrinet[7] 하드웨어 네트워크 장치를 사용하여 설계하고 구현하였다. 그리고 VIA의 또 다른 구현인 Berkeley-VIA와의 성능 비교를 수행하였다.

논문의 구성은 다음과 같다. 2절에서는 간략하게 VIA에 대해서 살펴보고 3절에서 KVIA(Kaist-VIA)의 상세한 설계 및 구현에 대해서 기술한다. 4절에서는 Berkeley-VIA[8]와 성능을 비교 분석하고 5절에서 결론을 맺는다.

2. Virtual Interface Architecture 개요

VIA는 크게 Virtual Interface(VI), VI 소비자, VI 제공자로 구성된다. Virtual Interface(VI)는 네트워크 하드웨어에 대한 안전하고 직접적인 접근을 제공하고, VI 소비자(consumer)는 VI를 사용하는 사용자 프로그램과 라이브러리(library)로 구성된다. 장치열기, 메모리 등록, 연결설정 등과 같이 고전적인 통신 프로토콜에서 운영체제가 담당하던 기능은 VI 제공자(provider)가 수행한다. 그림 1는 하나의 VI를 나타낸다.

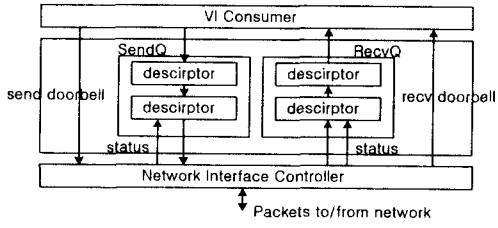

그림 1 Virtual Interface

VIA의 송수신은 기본 송수신(primitive send/recv)과 원격 메모리 쓰기/읽기(RDMA write/read)를 통해서 수행되고 사용자가 사용하는 메모리는 메시지 송수신전에 반드시 등록되어 있어야 한다. 메시지 송수신의 시작은 사용자 프로그램이 데이터에 대한 서술자(descriptor)를 작성하고 도어벨(doorbell)을 울림으로써 시작된다. 그리고 VI 제공자는 서술자내의 상태정보를 갱신하거나 완료 큐(completion queue)를 통하여 사용자에게 서비스 완료를 알리게 된다. VIA에서 제공하는 안전성 수준(reliability level)은 불완전한 전달(unreliable delivery), 완전한 전달(reliable delivery) 그리고 완전한 수령(reliable reception)이 있다.

3. KVIA의 설계 및 구현

KVIA는 Lanai7.2 PCI Myrinet 네트워크 인터페이스를 사용하여 리눅스 커널 2.2.13 환경에서 개발되었다. KVIA는 크게 VIA 표준에 의거한 사용자 수준 라이브러리와 VI 제공자 그리고 Myrinet 제어 프로그램(Myrinet Control Program: MCP) 으로 구성된다. KVIA 에서 사용한 Lanai7.2 인터페이스는 호스트의 메모리로부터 인터페이스내의 SRAM으로 직접 메모리 접근(Direct Memory Access)을 수행하는 DMA 엔진과 Myrinet 링크로 데이터를 업로드 또는 다운로드하는 두개의 DMA 엔진을 가지고 있고, 이들은 64bit 의 Myrinet LBUS를 통해 한 사이클에 최대 64bit 의 데이터를 전송한다.

3.1 도어벨(doorbell)의 설계 및 구현

사용자 프로그램에게 송수신 요구의 방법을 제공하는 것이 도어벨이다. KVIA에서는 이러한 도어벨을 Myrinet 네트워크 인터페이스에서 제공하는 하드웨어 도어벨 메커니즘을 사용하여 구현하였다. PCI 네트워크 인터페이스의 16MB 주소공간 중 8MB ~ 16MB 사이의 공간은 도어벨을 위해 할당되어 있다. 실제로 사용자 프로그램이 VI의 생성을 요구하였을 때 라이브러리에서는 VI 제공자의 도움을 받아 하드웨어 도어벨 영역과 자신의 가상메모리 영역사이의 메모리 사상을 수행한다. 메모리 사상은 프로세스간의 보호를 위해 하나의 프로세스당 페이지(page)크기까지 가능하다. 이렇게 사상된 메모리 영역에 도어벨이 울릴 경우 네트워크 인터페이스에서는 하드웨어적으로 이 도어벨 정보를 SRAM의 특정 영역으로 복사한다.

도어벨은 사용자 프로그램에서 작성한 서술자에 대한 메타정보이다. 따라서 도어벨은 송수신할 데이터에 대한 서술자를 얻기 위한 정보들로 구성되어야 한다. KVIA에서는 그림 2와 같은 32bit 도어벨을 사용하였다.

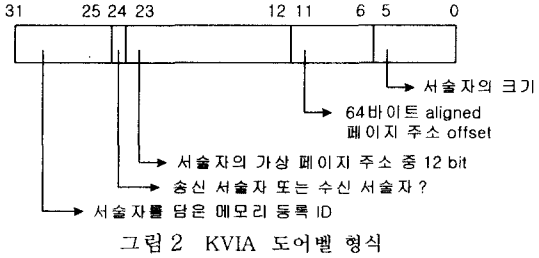

그림 2 KVIA 도어벨 형식

위의 도어벨을 사용하여 VI 제공자는 서술자에 대한 가상주소(virtual address)와 크기를 알 수 있다.

3.2 주소 변환(address translation)

VI 제공자 중 네트워크 인터페이스에서는 물리적 주소(physical address)를 사용하여 서술자와 데이터의 DMA를 수행한다. 따라서 효율적인 주소변환 메커니즘이 필요하다. KVIA에서는 여러 주소변환 메커니즘[9] 중에서 interrupt-driven 주소 변환 메커니즘을 사용하였다. Myrinet 제어 프로그램에서는 서술자나 데이터의 가상 주소에 대한 물리주소를 얻기 위해 호스트 프로세서로 인터럽트를 발생시킨다. 그리고 이러한 요구는 VI 제공자의 인터럽트 서비스 루틴을 통해서 요구된 물리주소를 MCP에게 통보한다. 네트워크 인터페이스에서는 변환된 물리주소를 사용하여 DMA를 수행하게 된다. 가상주소에 해당하는 물리주소를 적당한 태그와 함께 변환 보호 테이블(Translation and Protection Table: TPT)에 저장함으로써 이후의 요구되는 주소변환에 사용한다. KVIA 에서는 여러 프로세스가 공유하는 하나의 TPT 테이블을 사용하고 주소변환이 요구되는 가상주소의 [24:12]bit를 사용하여 8K개의 엔트리를 인덱싱한다.

3.3 데이터 송수신 메커니즘

VI 제공자인 네트워크 인터페이스에서 도어벨을 인지하였을 때 송신 도어벨인 경우 KVIA에서는 곧바로 송신과정을 수행한다. 송신 과정은 서술자의 크기에 무관하게 DMA 동작을 수행하도록 하였다. 또한 서술자가 기술하는 데이터도 4바이트를 제외한 모든 크기의 데이터에 대해서 역시 DMA 동작을 수행한다. 그러나 수신 도어벨인 경우에는 도어벨 로깅(logging)기법을 사용하였다. 즉 수신 도어벨이 인지됐을 경우에 수신할 데이터가 없다면 수신 서술자에 대한 정보를 네트워크 인터페이스에서 유지하고, 이 후에 데이터가 수신되었을 때 가장 먼저 인지된 수신 도어벨을 바탕으로 수신 서술자를 얻어 수신된 데이터를 사용자 프로그램에게 전달하게 된다.

3.4 완료 큐(completion queue)의 설계 및 구현

완료 큐는 하나의 프로세스내에서 요구된 서비스들의 순서를 보장하기 위한 것이다. KVIA에서는 완료 큐를 설계 할 때 가능한 한 네트워크 인터페이스 내의 메모리를 효과적으로 사용하기 위해 호스트 메모리를 사용하도록 하였다. 어떤 프로세스가 완료 큐의 생성을 요구하였을 때 KVIA 라이브러리에서는 완료 큐에 대한 메모리를 할당받고 이것을 VI 제공자에게 등록한다. 네트워크 인터페이스에서는 요구된 서비스가 완료되었을 때 요구한 VI ID, 송수신 플래그(flag)등의 정보를 DMA를 사용하여 전송한다. 따라서 사용자 프로세스는 지역 메모리를 폴링(polling)하게 되어 CPU 시간의 낭비를 막고, 지연시간을 줄인다.

3.5 KVIA 연결 설정 메커니즘

VIA 표준[1]에서는 연결(connection)의 종점(endpoint) 역할을 하는 VI들간의 연결을 클라이언트-서버모델로 명시하고 있다. 따라서 KVIA에서는 VIA 표준에 따라 전체 연결 설정 메커니즘으로 양방향(two-way) 클라이언트 서버 모델을 사용하였다. 즉 연결 서버는 연결 요구를 기다리고 있고 클라이언트는 연결을 요청한다. 연결 요청을 받은 서버는 적절한 조건 검사를 통해 연결을 수락(accept)할 것인지 거절(reject)할 것인지를 결정한다. 수락이 되면 서버는 클라이언트에게 수락 메시지를 보내고 연결이 확립된다.

3.6 KVIA 안전성 수준

KVIA에서는 3가지의 안전성 수준 중에서 불완전한 전달을 제공한다. 불완전한 전달이 의미하는 것은 송신된 데이터가 수신측의 사용자 프로그램에게 전달되지 않을 수 있음을 의미한다. 그리고 이러한 불완전한 전달이 발생했을 때 VI 제공자는 VI들간의 연결을 종료한다거나 재전송을 요구하지는 않는다. 다만 불완전한 전달이 발생했음을 사용자 프로그램에게 알리는 정도의 안전성을 제공한다. KVIA 안전성 수준이 불완전한 전달만을 제공하기 때문에 원격 호스트의 메모리 읽기를 수행하는 원격 메모리 읽기(RDMA read)는 KVIA에서 제공하지 않는다.

3.7 KVIA의 한계점

현재 KVIA 에서는 VIA 표준[1]에 명시된 모든 인터페이스를 구현하지 않았다. 아직까지 VIA 컴포넌트 - 네트워크 인터페이스 컨트롤러, 등록된 메모리, VI, 완료 큐 등 - 에 관련하여 질의(query)와 설정(set) 등의 인터페이스는 구현되지 않았다. 그리고 Myrinet 제어 프로그램은 네트워크 인터페이스 하드웨어에서 제공하는 전체 자원(resource)을 병렬적으로 이용하지 않고 직렬적인 제어 흐름에 따라 사용하고 있다. 이것은 각각의 독립적인 동작(operation)별로 비용(cost)의 분석이 쉽고 구현이 용이하지만 성능면에서는 큰 손실을 보게 되는 주요 원인이다. 따라서 더 나은 성능을 얻기 위해서는 Myrinet 제어 프로그램에서 자원 사용의 병렬화가 도입되어야 한다. 또한 전체 KVIA 라이브러리나, VI제공자 등에서 최적화 기법을 사용하여 코드를 갱신할 필요가 있다.

4. 성능 비교

이 절에서는 VIA 의 또 다른 구현인 Berkeley -VIA와의 성능 비교를 수행하였다. 성능 비교 수단(metric)으로는 지역 사용자 프로그램과 원격 사용자 프로그램 사이의 Round Trip 지연시간과 단방향(one-way) 대역폭을 사용하였다.

Berkeley-VIA는 현재 배포된 버전이 Myrinet Lanai4.x 의 네트워크 인터페이스를 사용한 것이기 때문에 성능 실험에서 Lanai4.3 의 하드웨어를 사용하였다. Lanai4.x 는 하드웨어에서 도어벨 메커니즘을 제공하지 않기 때문에 Berkeley-VIA는 단순히 네트워크 인터페이스 내의 SRAM 메모리 사상(mapping)과 폴링(polling)을 사용하여 도어벨을 구현하였고, 주소변환은 물리주소를 DMA 동작을 통하여 얻어오는 방법을 사용한다. 또한 모든 데이터와 서술자를 크기에 관계없이 DMA 동작으로 송수신을 하고 불완전한 전달의 안전성 수준을 제공한다.

성능 실험 환경으로는 Pentium-III 500MHz Dual 노드 2대를 사용하였고, 노드 내에는 256MB 의 메모리와 32bit PCI 버스를 갖추고 있다. 그리고 네트워크 인터페이스로는 2MB의 SRAM, 32bit 로컬 버스, 32bit DMA 엔진을 가진 Lanai4.3과 4MB 의 SRAM, 64bit 로컬 버스, 64bit DMA 엔진을 가진 Lanai7.2 를 사용하였다. 두 노드는 웜-홀(worm hole) 라우팅을 수행하는 Myrinet 스위치를 사용하여 연결되어 있다.

4.1 Round Trip 지연시간

그림 3 는 두 VIA 구현 사이의 지연시간을 비교한 것이다.

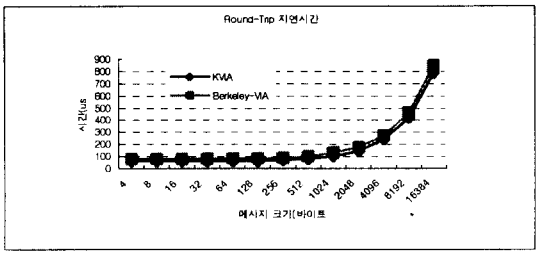

그림 3 Round Trip 지연시간

전체적으로 모든 메시지의 크기에 대해서 KVIA 가 17 ~ 30 us의 지연시간 단축을 보여주었다. 4 바이트의 메시지에 대해서는 KVIA 가 59us, Berkeley -VIA가 77us 이다. 이는 KVIA에서 하드웨어 도어벨 메커니즘을 사용하고 실제로 Lanai7.2에서의 로컬 버스 및 DMA 엔진이 64bit로 확장되었기 때문이다.

4.2 단방향 대역폭

그림 4는 KVIA와 Berkeley-VIA사이의 대역폭을 비교한 그림이다.

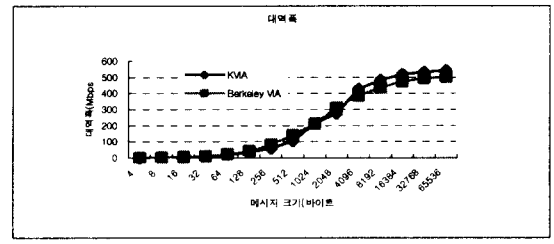

그림 4 단방향 대역폭

KVIA는 메시지 크기가 커질수록 Berkeley -VIA보다 더 나은 성능을 보이고 있다. 이것은 Lanai7.2 네트워크 인터페이스에서 로컬 버스와 DMA 엔진에의 단위 클럭(clock)당 전송되는 데이터의 양이 더 많기 때문이다. 그러나 성능의 차이가 그렇게 크지 않은 것은 KVIA 의 Myrinet 제어 프로그램에서 자원 사용의 병렬화와 최적화 기법이 사용되지 않았기 때문이고 이는 앞으로 개선해야 할 과제이기도 하다. Lanai7.x 의 네트워크 인터페이스를 사용한 Berkeley -VIA가 배포되면 더 확실한 성능 비교를 수행할 수 있을 것이다.

5. 결론 및 향후 계획

이 논문에서는 산업계 표준으로 알려진 VIA 를 설계하고 구현하였다. Ber keley-VIA와의 성능 비교에서 지연시간과 대역폭에서 우수한 성능을 보여주었다. 현재 KVIA 는 지속적인 개선 과정에 있고 성능 분석도 계속되고 있다. 앞으로 VIA 표준 모두를 만족하는 KVIA의 구현과 Myrinet 제어 프로그램의 병렬화 및 최적화 기법에 관한 연구가 필요하다.

참고 문헌

[1] " Virtual Interface Architecture Specification. Version 1.0", Compaq, Intel and Microsoft Corporations, Dec 16, 1997, available at http://www.viarch.org

[2] Raoul A.F. Bhoedjang, Tim R Ühl and Henri E. Bal, " User-Level Network Interface Protocol " , IEEE Computer Nov. 1998, page. 53-60

[3] C. Dubnicki, L. Ift ode, E. W. Felten, Kai Li. Software support for virtual memory-mapped communication, Proceedings of IPPS '96, the 10th International Parallel Processing Symposium, Honolulu, HI, USA, 15 -19 April 1996. Los Alamitos, CA, USA: IEEE Comput. Soc. Press, 1996, p.372-381.

[4] L. Prylli and B. Tourancheau. BIP: a New Protocol Designed for High Performance Networking on Myrinet., *Workshop PC-NOW, IPPS/SPDP98*, Orlando, USA, 1998.

[5] T. von Eicken, Anindya Basu, Vineet Buch and Werner Vogels, U-Net: A User -level Net work Interface of parallel and Distributed Computing, Proc. of the 15th ACM Symposium of Operating Systems Principles, vol. 29 Dec 1995, pp. 40-53

[6] S. Pakin, V. Karamcheti, A. A. Chien . Fast messages: efficient, portable communication for workstation cl usters and MPPs. IEEE Concurrency, vol.5, (no.2), April-June 1997. p.60-72.

[7] Myricom Inc. http://www.myri.com

[8] Philip Buonadonna, Andrew Geweke, David Culler, "An Implementation and Analysis of the Virtual Interfa ce Architecture", Proceedings of SC98, Orlando, Florida, November 1998

[9] Incoporating Memory Management into User -level Network Interfaces, Anindya Basu, Matt Welsh, Thorsten von Eicken, Presented at Hot Interconnects V, August 1997, Stanford University