리눅스 기반 InfiniBand 장치 드라이버의 설계 및 구현

김선욱, 박찬호, 박경
한국전자통신연구원 컴퓨터시스템연구부

The Design and Implementation of InfiniBand Device Driver based on Linux

Sunwook Kim, Chanho Park, Kyoung Park
Computer System Research Department
Electronics and Telecommunications Research Institute
E-mail : swkim99@etri.re.kr

Abstract

InfiniBand 구조는 입출력 처리에 최적화된 입출력 연결망과 프로세서 상호간 통신에 최적화된 클러스터 연결망을 통합한 차세대 클러스터 연결망이다. 본 논문에서는 리눅스 운영체제를 기반으로 하는 InfiniBand 연결망용 호스트 채널 어댑터(HCA : Host Channel Adapter) 장치 드라이버의 소프트웨어 스택 및 구조, 그리고 메시지 전송 절차에 대해 설명하였고 장치 드라이버의 동작 검증을 위한 환경 및 테스트에 대해 논하였다.

I. 서론

최근 들어 인터넷 서버의 기능과 요구사항이 변화함에 따라 요구 기능에 적합한 서버구조가 제안되었고 새롭게 제안된 서버 구조는 새로운 시스템 연결망 구조의 필요성을 부각시켰고 급기야 InfiniBand 기술의 탄생을 촉발시켰다. InfiniBand 는 스위치 기반형 점대점 연결망이며 Compaq, Dell, IBM, HP, Intel, Microsoft, Sun Microsystems 등을 주축으로 지난 1999 년 8 월에 결성된 IBTA(InfiniBand Trade Association)에서 표준 규격 1.0 을 2000 년 10 월에 발표하였으며 2001 년 6 월 및 2002 년 11 월에 각각 표준 규격 1.0 을 수정 및 개선한 표준 규격 1.0a 및 1.1 이 발표되었다. InfiniBand 구조(IBA : InfiniBand Architecture)는 입출력 장치 구성 및 입출력 처리에 최적화된 입출력 연결망과 프로세서 상호간 통신에 최적화된 클러스터 연결망을 통합한 차세대 클러스터 연결망으로 통신 및 관리 메커니즘을 포함하고 있으며 소규모 컴퓨터 시스템에서 대규모 인터넷 서버 및 슈퍼컴퓨터에 이르기까지 다양한 영역에서 사용이 가능한 구조이다[1][2].

본 논문에서는 InfiniBand 연결망용 호스트 채널 어댑터(HCA : Host Channel Adapter)의 장치 드라이버 설계 및 구현에 대해 기술한다. 서론에 이어 2 장에서는 InfiniBand 연결망의 구조 및 구성 요소 들에 대해 기술하고 3 장에서는 리눅스 기반 InfiniBand 장치 드라이버의 구조 및 설계 내용에 대하여 기술한다. 4 장에서는 장치 드라이버의 동작에 대해 기술하고 5 장에서는 결론을 맺는다.

II. InfiniBand 연결망의 구조 및 구성 요소

IBA 연결망은 스위치 기반 비정형 연결망으로 종단 노드(프로세서 노드, 입출력 노드)가 연결되는 여러 개의 서브넷으로 구성된다. 서브넷은 종단 노드와 스위치, 라우터로 구성되며 서브넷 간 연결은 라우터를 통해서 이루어 진다. 각 종단 노드는 IBA 연결망 접속을 위한 채널 어댑터를 가지며, 프로세서 노드쪽에서는 HCA 를 사용하고 입출력 노드 및 입출력 장치쪽에서는 타겟 채널 어댑터(TCA: Target Channel Adapter)를 사용한다. HCA 는 다른 호스트 또는 입출력 장치와 통신을 위하여 호스트에게 채널 인터페이스를 제공하는 장치로 사용자 수준 프로그램에서 Verb 라는 소프트웨어 인터페이스를 사용하여 발생된 메시지 전송 요구를 해독하여 호스트 메모리에서 데이터를 읽어서 IBA 연결망으로 송출하고 또한 수신된 메시지를 해독하여 호스트 메모리에 직접 쓰는 작업을 수행한다. 또한 HCA 장치는 스위치와 TCA 모두에 직접 연결할 수 있으므로 서버

구성시 서버의 규모에 따라 유연하게 시스템을 구성할 수 있다. TCA 는 입출력 장치와 연결망을 이어주는 장치로 디스크 콘트롤러, 네트워크 콘트롤러, RAID 콘트롤러 등과 같은 다양한 입출력 장치를 IBA 연결망에 정합할 수 있게 한다. 그림 1 은 IBA 구성요소들을 사용하여 나타낸 구성도이다. [3][4]

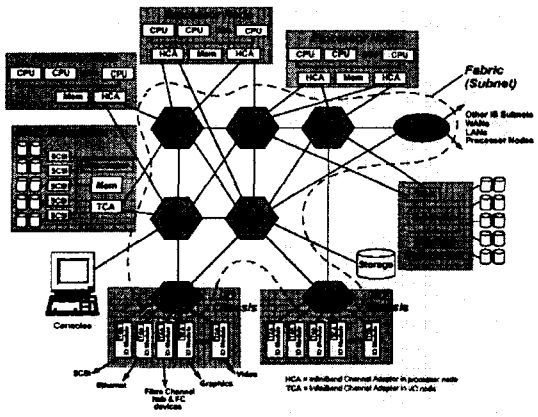

그림 1. InfiniBand Architecture 구성도

III. 리눅스 기반 InfiniBand 장치 드라이버의 구조 및 설계

HCA 장치를 시스템에서 구동시키기 위한 장치 드라이버는 호스트 시스템에서 수행되는 소프트웨어로서 사용자에게 InfiniBand 표준 규격 1.0a 에서 정의한 소프트웨어 전송 계층 기능을 만족하는 Verb 형태의 사용자 인터페이스(Verb API: Application Programming Interface)를 제공하고 도어벨 메커니즘(Doorbell Mechanism)을 통하여 사용자의 전송 요구를 HCA 하드웨어에 전달하며 HCA 하드웨어에 관련된 자원들을 관리 감독하는 기능을 수행한다. HCA 의 구성, 관리 및 동작 명령 전달을 위해서 정의되어진 Verb 는 사용자(응용 프로그램)의 메시지 전송 또는 데이터 서비스 요구를 HCA 하드웨어에 전송하기 위해서 다양한 기능을 정의하고 각 기능에서 사용되는 파라미터를 정의하고 있다. [4]

3.1 HCA 장치 드라이버 소프트웨어 스택

HCA 장치 드라이버는 사용자에게 그림 2 와 같은 소프트웨어 스택을 제공한다. 사용자는 Verb API (Application Programming Interface)를 사용하여 InfiniBand 전송 서비스를 사용하는 응용 프로그램을 작성하고, HCA 장치 드라이버는 Verb API 를 통해서 사용자에게 제공하는 소프트웨어 전송 인터페이스(Software Transport Interface) 함수를 제공한다. 소프트웨어 전송 인터페이스 함수는 커널 라이브러리 형태로 제공되어지며 전송 자원 관리 및 전송 서비스 명령을 생성하여 리눅스 운영체제의 커널 모드에서 실행되고 있는 드라이버의 하드웨어 추상화 계층(HAL: Hardware Abstraction Layer)에 전달하게 된다. 소프트웨어 전송 인터페이스로부터 명령을 받은 하드웨어 추상화 계층은 해당 명령을 도어벨 인터페이스를 통하여 HCA 장치에게 전달한다.

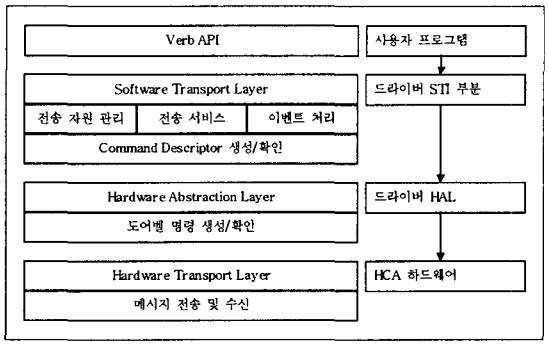

그림 2. HCA 장치 드라이버 소프트웨어 스택

3.2 HCA 장치 드라이버의 메시지 전송 처리

그림 3 은 사용자 프로그램이 HCA 드라이버를 거쳐서 메시지를 송수신 하는 절차를 나타낸 것으로 사용자 프로그램은 Verb API 를 통하여 전송에 필요한 자원을 확보한 후에 드라이버 HAL 에게 송신 또는 수신 명령을 Verb API 를 사용하여 의뢰한다. 송수신 명령을 받은 드라이버 HAL 에서는 해당 명령을 사용자가 지정한 전송 서비스에 맞도록 재구성하며 송신큐(SQ)와 수신큐(RQ)로 구성된 작업큐에 저장한다. 저장된 송수신 명령은 도어벨 인터페이스를 통하여 HCA 장치에게 전달되고 HCA 장치는 사용자 주소 공간에 존재하는 데이터를 DMA(Direct Memory Access) 엔진을 통하여 접근하여 해당 송수신 명령을 처리한다. 따라서 전송 명령에 해당하는 부분은 사용자의 의뢰에 따라 커널 주소 공간에서 관리되지만, 실제 전송 데이터는 사용자 주소 공간에서 관리됨으로 불필요한 데이터 복제 오버헤드를 제거할 수 있다.

3.3 Verb API 의 설계

Verb API 는 사용자가 InfiniBand 전송 서비스를 사용할 수 있도록 인터페이스를 제공한다. Verb 는 호스트

채널 인터페이스(Host Channel Interface)가 호스트에게 제공하는 기능을 추상화하여 정의한 것으로, InfiniBand Architecture Specification Version 1.0a 의 표준 규격 호환성 보장을 위하여 규격서에 명시된 Verb 의 기능 기술(Semantics)을 반드시 제공하여야 한다. Verb 사용자는 드라이버 STI 에서 제공하는 Verb API 라이브러리를 사용하여 응용 프로그램을 작성한다.

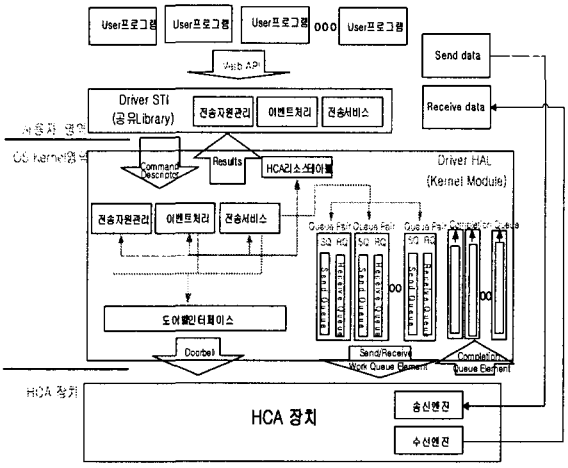

그림 3. 전송처리 개념도

Verb API 는 44 개가 제공되어지는데 대표적으로 HCA 에 접근하여 사용권한을 획득하고 초기화를 위해 Open_HCA_LIB()와 이와 반대로 HCA 에 대한 사용을 종료하고 할당된 리소스들을 해제하는 Close_HCA_LIB() , 송수신 명령 저장을 위한 송신큐와 수신큐를 생성하는 Create_QP_LIB()가 있으며 송수신 명령에 해당하는 Post_Send_LIB(), Post_Receive_LIB()가 있다.

3.4 드라이버 STI 의 설계

드라이버 STI 는 커널 라이브러리 형태로 제공되어지며 공유 라이브러리로 형태로 구현된다. 드라이버 STI 에서는 Verb API 를 통해 요청된 명령을 커널에서 수행중인 드라이버 HAL 이 인식할 수 있는 Command Descriptor 로 변환하여 드라이버 HAL 에게 전송하게 되는데 Command Descriptor 의 전송은 리눅스 운영체제에서 제공하는 장치 파일을 통해서 이루어지며 형태는 장치 파일에서 제공하는 open, close, read, write, ioctl 등의 파일동작 구조체를 사용한다. HCA 장치 드라이버 구현 시에는 장치 파일에서 제공하는 파일 동작중 open, close, ioctl 만을 사용한다. 앞 절에서 언급한 Open_HCA_LIB() 는 open 동작 명령을 생성하고 Close_HCA_LIB()는 close 동작 명령을 생성한다. Create_QP_LIB()나 Post_Send_LIB()와 같은 나머지 Verb API 관련 함수들은 ioctl 동작 명령을 생성한다.

3.5 드라이버 HAL 의 설계

리눅스 운영체제의 모듈 프로그램 형태[5]로 구현되어 커널 모드에서 실행되는 드라이버 HAL 은 드라이버 STI 로부터 전송된 전송 자원 관리 및 전송 서비스 Command Descriptor 를 받아서 해당 명령과 관련된 자원 관리를 처리하며 전달된 명령을 도어벨 인터페이스를 통해서 HCA 장치에게 전달할 수 있도록 하드웨어 명령으로 변환하고 도어벨 인터페이스에게 HCA 장치 접근을 요청한다. 또한 도어벨 인터페이스를 통해서 명령처리 결과를 수신하여 그 결과를 해당 서비스 루틴에 전달하는 기능을 수행한다. Driver HAL 의 구조 및 구성요소는 그림 3 의 Driver HAL 부분과 같다.

Driver HAL 의 전송 자원 관리 서비스 루틴은 수신된 전송 자원 관리 명령과 HCA 제어 명령을 수신하여 처리한다. 처리되는 명령은 전송 자원 할당, 갱신, 검색, 삭제로 구성되며, 각 기능을 수행하기 위하여 HCA 리소스 테이블과 연동하여 동작한다. HCA 리소스 테이블은 InfiniBand Architecture Specification 1.0a 표준 규격에서 정의된 전송 리소스들을 관리하는 자료 구조로서 HCA 속성 및 작업큐등을 포함한다.

전송 서비스 루틴은 메시지 송수신을 위해서 제공하는 Verb API 를 통해서 수신된 사용자의 전송 요구를 서비스별로 구별하여 전송 자원을 참조한 후 전송 명령을 생성하고 작업큐에 입력한다.

이벤트 처리 서비스 루틴은 전송 명령에 대한 비동기 종료 이벤트 및 HCA 장치에서 발생하는 비동기 예외 상황을 처리하는 서비스 루틴이다.

도어벨 인터페이스는 HCA 장치와 정보 전달을 수행하는 주체로 각 서비스 루틴에서 생성된 하드웨어 명령을 HCA 장치 내부의 레지스터를 이용해 구성된 도어벨을 통하여 장치에 전달하는 기능을 수행한다. 또한 도어벨의 상태 필드를 검사하여 해당 명령의 처리 결과를 확인하고, 이를 해당 서비스 루틴에 전달한다. 도어벨은 두가지 형태로 나누어지며 전송 서비스 루틴에서 생성된 명령은 송신/수신 도어벨을 통하여 전달하고 전송 자원 관리 서비스 루틴에서 생성된 명령은 명령/상태 도어벨을 통해서 전달한다.

IV. 장치드라이버의 동작

본 논문에서 설계한 리눅스 기반 InfiniBand 장치 드라이버에 대한 동작을 검증하기위해 그림 5 와 같은

HCA 장치 기반 클러스터 시스템 형태의 테스트 베드를 구성하였다.

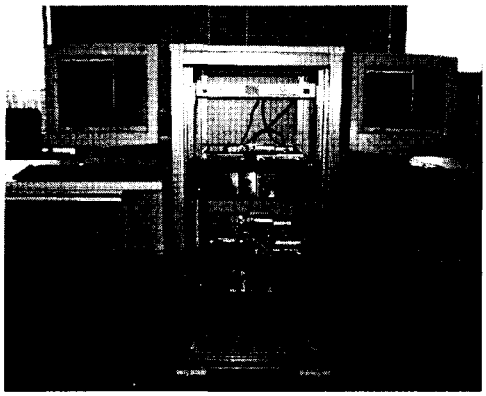

그림 5. 동작 검증을 위한 테스트 환경

노드의 개수는 2 개이며 각각의 PCI-X 슬롯에 HCA 장치가 설치되고 InfiniBand 케이블을 이용해연결한다. 각 노드는 Intel Xeon 1.4Ghz 프로세서를 가지는 IBM 의 xSeries 360 기종이며 운영체제는 리눅스 커널 2.4 를 기반으로 한다.

동작 검증을 위한 테스트 프로그램의 플로우 차트는 그림 6 과 같으며 HCA 장치 를 통한 메시지 전송 과정을 나타낸다. 그림 7 은 테스트 프로그램의 실행화면을 나타낸다.

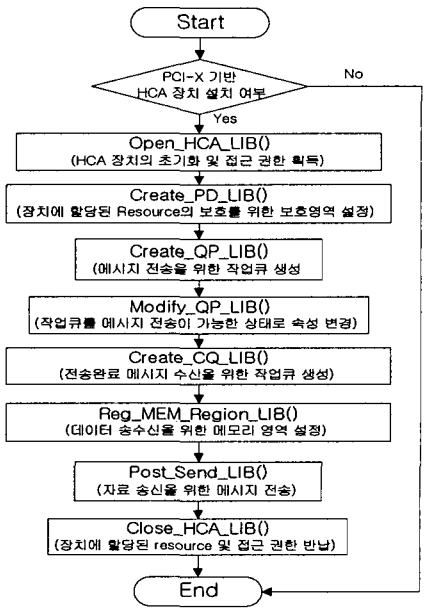

그림 6. 테스트 프로그램의 플로우 차트

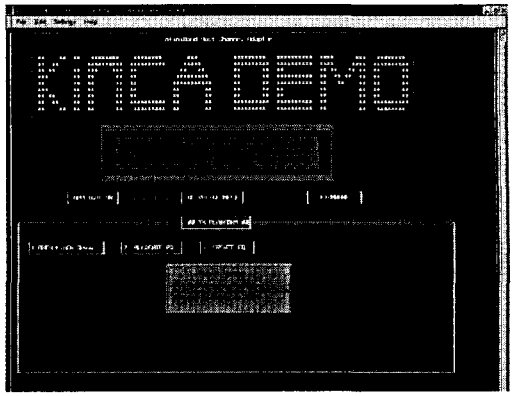

그림 7. 테스트 프로그램 실행화면

V. 결론

본 논문에서는 차세대 시스템 연결망인 InfiniBand 연결망 접속 장치인 HCA 장치에 대한 리눅스 기반의 장치 드라이버를 개발하였다. 개발된 장치 드라이버는 사용자에게 Verb API 및 드라이버 STI 를 제공하며 리눅스의 모듈 프로그램 형태로 구현된 드라이버 HAL 은 도어벨 인터페이스 통하여 HCA 장치에게 명령을 전달한다. 또한 HCA 장치는 DMA 를 통하여 사용자 영역의 전송데이터를 커널 영역으로의 복사없이 전송되므로 데이터 복제 오버헤드를 제거하였다. 개발된 장치 드라이버는 현재 성능 시험을 진행중이며 향후 차세대 클러스터 연결망 기술에 활용될 예정이다.

참고문헌

[1] 박경, 모상만, "InfiniBand: 차세대시스템연결망," 정보과학회지, 제19권 제3호, pp.43-51, 2001. 3.

[2] InfiniBand Trade Association(IBTA) Official Homepage, http://www.infinibandta.org/, Sep.2002.

[3] Sangman Moh, And etc. al.,"InfiniBand Technology: The Next-Generation System Interconnect," Proceedings of the 1st World Korean Business Convention, Oct.2002.

[4] 박경, 김성남, 한종석, 모상만, "듀얼프로세서 코어 기반 InfiniBand SoC," 반도체학술대회, 2002.

[5] Jonathan Corbet, Linux Device Drivers, 2nd Edition, Oreilly, Jun.2001.