

리눅스 클러스터 파일 시스템을 위한 M-VIA기반 고속 통신 모듈의 설계 및 구현

박의수^{○*}, 최현호^{**}, 유찬곤^{***}, 유관종^{*}

충남대학교 컴퓨터과학과, 대전보건대학 컴퓨터정보처리과, 국방과학 연구소

Design and Implementation of The High-Speed Communication Module for a Linux Cluster File System Using M-VIA

Ui-Su Park*, Hyun-Ho Choi**, Chan-Gon Yoo***, Kwan-Jong Yoo*

*Dept. of Computer Science, ChungNam Univ.

**Dept. of Computer Information Processing, Daejeon health Science college

***Agency for Defense Development

E-mail: {uspark, kjyoo}@cs.cnu.ac.kr, hyuno@hit.ac.kr, jargon@hanmir.com

요 약

클러스터 파일 시스템은 데이터 입출력 대역폭을 극대화하여 효율성을 높이고 각 노드의 입출력 부담을 균등하게 부과하기 위하여 원본 파일을 여러 노드에 분산 저장한다. 이렇게 파일을 노드들에 분산 저장하기 위해 서는 효율적인 노드간 데이터 통신을 필요로 하며, 노드 내부에서도 클러스터 파일 시스템과 어플리케이션과의 효율적인 전용 데이터 교환 메커니즘을 지원해야 한다. 이를 위해 사용자 수준 통신 프로토콜인 VIA를 선정하여 운영체제(Operating System)의 간섭으로 인한 네트워크 계층간의 데이터 복사에 의한 병목현상을 줄이고자 하였다. 본 논문에서는 노드간 데이터 통신을 위해 M-VIA를 이용하여 통신모듈을 설계 및 구현하였다. 그리고 실제 성능테스트를 통하여 기존의 소켓 기반인 TCP/IP를 이용한 통신모듈과의 성능을 비교 평가하고 확인한다.

1. 서론

컴퓨터의 성능 향상과 대중화에 따른 가격의 하락 그리고 네트워크의 발달은 VOD(Video On Demand)서비스의 등장을 가능하게 하였다. 이러한 VOD서비스는 보다 많은 사용자에게 다양한 서비스를 안정적으로 제공하는 것을 목적으로 한다. VOD서비스에서 사용하는 비디오 데이터는 멀티미디어 데이터의 속성상 많은 저장 공간을 차지하며 높은 전송 대역폭을 요구한다. 원활한 서비스를 제공하기

본 논문은 한국과학재단이 지원한 지역협력센터(RRC)인 충남대학교 소프트웨어연구센터의 지원으로 수행된 과제의 결과입니다.

위해서는 저장장치(디스크)의 충분한 입출력 대역폭 확보와 추가되는 비디오 데이터에 대한 저장장치 및 네트워크 대역폭의 확장성 등이 요구된다. 일반적인 싱글 서버 모델로는 이러한 문제를 근본적으로 해결할 수 없기 때문에 클러스터 파일 시스템이 새로운 해결책으로 고려 되었다[1]. 클러스터 파일 시스템에서 하나의 파일은 각각의 노드로 스트라이핑 되어 분산 저장된다. 그러므로 클러스터 파일 시스템은 노드간 충분한 입출력 대역폭을 제공해야 한다. 그러나 고속 네트워크 장비만으로는 이러한 문제가 해결되

지 않는다. 그 이유는 VOD의 특성상 찾은 파일 복사가 이루어지는데, 메시지 전송 시 커널의 개입으로 찾은 복사와 컨텍스트 스위칭(context switching)이 발생하기 때문이다.

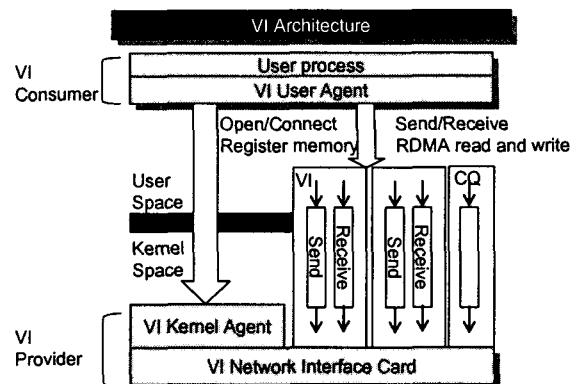
본 연구에서는 리눅스 클러스터의 핵심이 되는 클러스터 파일 시스템을 구성하는 각 노드간의 원활한 통신을 보장하기 위해 메시지 복사 중 커널의 개입이 이루어지지 않는 사용자 수준의 통신 프로토콜인 VIA를 이용하여 Fast Ethernet 환경 하에서 통신모듈의 구조를 정의하고 모듈간의 데이터 교환 과정을 설계한다. 따라서 본 논문에서는 노드간 통신에서 사용자 수준 프로토콜인 VIA계열의 M-VIA를 이용하여 클러스터 파일 시스템의 더 빠른 메시지 전송을 가능하게 하는 노드간 통신모듈(Inter-node communication module), 즉 고속 통신모듈 라이브러리를 개발한다. 그리고 기존의 소켓기반 TCP/IP를 이용한 통신 모듈과의 성능을 비교 평가한다.

2. 관련연구

2.1 VIA(Virtual Interface Architecture)

호스트 내부의 소프트웨어 구간이 노드간 통신의 새로운 병목 구간으로 등장함에 따라 이 문제를 해결하기 위하여 U-Net, Active Message, Fast Message, BIP와 같은 여러 사용자 수준 프로토콜(User-level Protocol)에 대한 연구가 행해졌다[2][3]. 이 연구들의 핵심적 개념은 사용자 프로세스에게 네트워크 인터페이스에 대한 직접적인 접근을 가능하게 하여 데이터 통신 경로 상에 커널의 개입을 없앤 것이다. 그리고 이러한 여러 사용자 수준 프로토콜의 표준을 정하기 위하여 Microsoft, Intel, Compaq 3사가 주도하여 컨소시엄을 구성하고 운영체제와 하드웨어 구조에 독립적인 사용자 수준 통신 프로토콜인 VIA(Virtual Interface Architecture)를 제안하였다[4].

VIA는 가상 인터페이스(virtual interface)를 통하여 각각의 사용자 프로세스가 시스템에 존재하는 네트워크 인터페이스를 독점적으로 사용하는 것처럼 보이게 해준다. VIA는 VI(Virtual Interface), VI 제공자(provider), VI 소비자(consumer), 완료큐(completion queue)의 4가지 주요 구성 요소로 이루어져 있다.



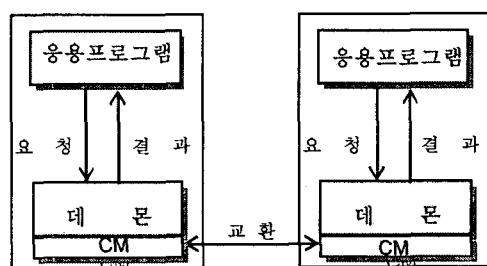
[그림 1] Virtual Interface Architecture 구성요소

2.2 M-VIA(Modular Virtual Interface Architecture)

M-VIA는 Lawrence Berkeley National Laboratory의 NERSC(National Energy Research Scientific Computing Center)에서 만든 리눅스 상에서 실행되도록 구현된 VIA이다[5]. M-VIA는 기존의 네트워크 프로토콜과 병존 할 수 있으며, 그에 따라 동일 네트워크 카드상에서 TCP/IP와도 같이 사용할 수 있다.

2.3 통신모듈(Communication Module)

클러스터 파일 시스템을 구성하는 각 노드에는 데몬이 있다. 데몬은 파일 혹은 디렉토리를 관리하고 파일을 다중 노드에 분산 저장하며 응용프로그램 혹은 노드간의 통신 서비스를 제공하기 위해 구현된 프로그램이다. 통신 모듈은 이러한 일을 하는 데몬에게 필요한 통신 서비스를 제공한다.



[그림 2] 통신모듈의 동작

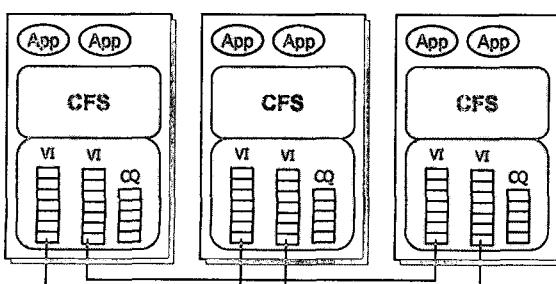
3. M-VIA 기반 통신모듈의 설계 및 구현

3.1 설계 고려 사항

- CFS는 응용프로그램 개발자들에게 단일 시스템 형상을 제공해야 한다.
- 통신 모듈은 노드들간의 프로세스 뿐만 아니라 한 노드 안에서의 프로세스들간에도 효율적인 데이터 교환 메커니즘을 제공해야 한다.
- 멀티미디어 데이터 등과 같은 대규모 데이터의 처리 성능을 극대화하며 텍스트 등의 작은 데이터 처리 성능 향상은 고려하지 않는다.
- Intra-node communication module은 노드 내에서만 운영 체제 프로세스 사이의 데이터 교환을 담당하고 교환되는 데이터가 대용량의 멀티미디어 데이터이므로 리눅스상의 가장 속도가 빠른 방법을 사용해야 한다.
- 서비스 요청 및 수락, 처리 결과 대기 시 바쁜 대기가 발생하지 않아야 한다.
- 데드락이 발생하지 않아야 한다.
- Inter-node communication module은 네트워크 초기화 및 작동 시에 다른 노드 대온 실행 여부에 영향을 받지 않아야 한다.

3.2 VI 연결

데온간 VI의 연결은 1:1 방식으로 이루어 지므로, n개의 노드가 있는 경우 각 노드는 n-1개의 VI 연결이 필요하다. 예를 들면, [그림 3]과 같이 3개의 노드로 클러스터 파일 시스템이 구성되어 있다고 가정하면, 각 노드는 2개의 VI연결을 가지게 된다. 이러한 연결 구조는 n개의 노드로 확대되어도 그대로 적용된다.



[그림 3] VI 연결 구조

3.3 클래스 설계

VIA를 이용한 통신은 로컬 및 원격 주소 설정, 디스크립터와 데이터 버퍼를 위한 메모리 할당 및 등록, VI생성과 상대 프로세스의 VI와 연결 설정, 송수신을 위한 디스크립터의 처리 등의 작업을 필요로 한다. 이러한 일련의 작업들을 효율적으로 관리하기 위하여 본 논문에서는 VI사용자를 위한 클래스 CVI와 완료큐(Completion queue)사용을 위한 CCq를 정의하여 VI와 CQ를 객체로 다룰 수 있게 하였다. CVI클래스는, 데이터 송수신 시 메모리 할당 및 등록으로 인한 자연시간을 줄이기 위해 클래스 생성자(Constructor)에서 데이터 송수신이 일어나기 전에 미리 필요한 디스크립터 및 데이터 저장을 위한 메모리를 등록한다. CVI와 CCq클래스의 주요 멤버 함수는 [표 1][표 2]와 같다.

[표 1] CVI 클래스 멤버 함수

멤버 함수	파라미터	함수 설명
CVI()	IN char *devicename	VI 생성
	IN	
	VIP_RELIABILITY_LEVEL level	
	IN VIP_CQ_HANDLE cqHandle	
~CVI();	void	등록된 메모리 해제와 VI 삭제
DescriptorSetting()	void	VI 연결을 위한 디스크립터 설정
ConnectTo()	IN char *hostname	Hostname의 연결시도
	IN unsigned char discriminator	
ConnectFrom()	IN char *hostname	Hostname으로 부터 연결요청 수락
	IN unsigned char discriminator	
ConnectClose()	void	VI 연결 해제
SendPacket()	IN VIPACKET vi_pkt	패킷 송신
RecvPacket()	VIP_VI_HANDLE viHandle	패킷 수신
	OUT VIPACKET& vi_pkt	

[표 2] CCq 클래스 멤버 함수

멤버 함수	파라미터	함수 설명
CCq()	IN char *devicename;	Device name 의 장치에 완료큐 생성
	IN VIP ULONG EntryNumber;	
-CCq()	void	완료큐 삭제
getCQHand()	OUT VIP_CQ_HANDLE *cqHand	완료큐 핸들
cqWait()	OUT VIP_VI_HANDLE *viHand	VI 의 핸들을 리턴

[표 3]은 클러스터 파일 시스템을 구성하는 노드의 하드웨어와 소프트웨어 구성에 대한 표이다.

[표 3] 하드웨어와 소프트웨어 구성

하드웨어	CPU	PentiumIII 600MHz(Katmai)
	Main Memory	128MB
	Hard Disk	EIDE Western Digital 60GB
	Network	Fast Ethernet(3Com)
소프트웨어	OS	Linux (Kernel 2.4.2-3)
	M-VIA	1.2b2
	Compiler	g++ 2.96

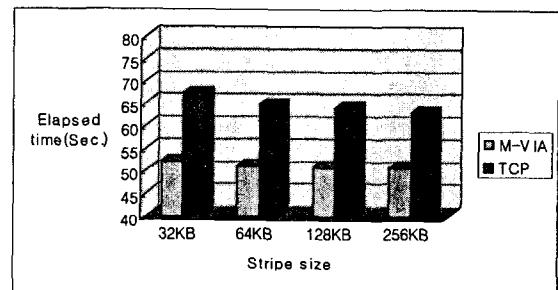
3.4 패킷 송수신

VIA는 연결 지향형 프로토콜로서, 비 신뢰성 전송(Unreliable Delivery), 신뢰성 전송(Reliable Delivery), 신뢰성 수신(Reliable Reception)의 3 가지 신뢰성 수준을 규정한다[6]. 모든 VI NIC은 비 신뢰성 전송(Unreliable Delivery) 수준을 지원해야 하며, 신뢰성 전송(Reliable Delivery) 수준과 신뢰성 수신(Reliable Reception)은 선택적이다. 신뢰성 수준은 VI의 속성이며 같은 신뢰성 수준을 지닌 VI 사이에서만 연결이 설정된다. M-VIA 1.2는 비 신뢰성 전송(Unreliable Delivery)과 신뢰성 전송(Reliable Delivery)의 2 가지 신뢰성 수준을 지원하므로, VI의 속성을 신뢰성 전송으로 지정하여 사용하면, 패킷 오염 감지 및 패킷 중복 수신과 누락 감지, 그에 따른 패킷 재 전송 요청 및 수신 패킷 재정렬 등의 작업을 어플리케이션이 처리하지 않아도 되므로 응용프로그램 입장에서는 많은 부담을 줄일 수 있다. 그러나, VI의 수신 큐에 디스크립터가 준비되어 있지 않음으로써 발생되는 경쟁 상태(race condition) 문제는 피할 수 없다. 신뢰성 있는 데이터 전송을 위해서는 수신 측의 디스크립터에 항상 수신을 위한 디스크립터가 준비되어 있어야 하므로, 이에 본 논문에서는 디스크립터 순환 삽입 방식을 사용하여 VI의 수신 큐에 항상 수신을 위한 디스크립터가 준비 되도록 하였다.

4.2 실험 결과

구현된 M-VIA 통신모듈과 기존의 소켓기반 TCP/IP 통신 모듈의 성능 비교를 위해 500Mbytes의 멀티미디어 파일에 대해서 cfs_read() 읽기 명령에 대한 대역폭과 지연시간을 측정하였다. 왜냐하면, VOD서비스에서 읽기 수행이 가장 빈번하고 큰 비중을 차지하기 때문이다. 실험 패킷 사이즈는 32KB, 64KB, 128KB, 그리고 256KB이다.

[그림 4]는 하나의 노드 안에서 한 개의 프로세스가 500Mbytes의 파일 읽은 시간을 보여주고 있다. M-VIA와 TCP 통신 모듈이 나은 결과를 보여주고 있다.

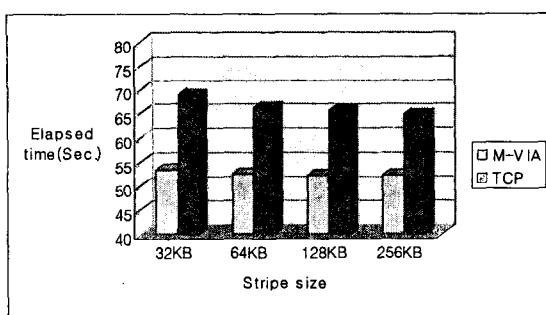


[그림 4] one process in one node (total 1 process)

[그림 5]는 같은 파일을 각각의 노드에서 1개의 프로세스가 읽은 결과를 보여준다.

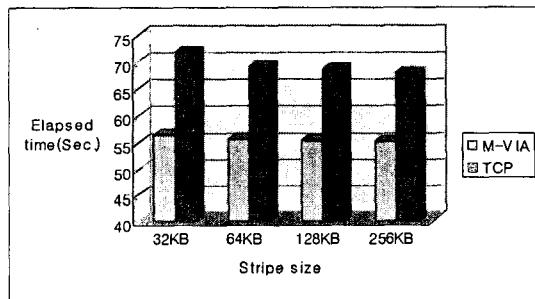
4. 실험결과 및 분석

4.1 실험 환경



[그림 5] one process in each node (total 3 processes)

[그림 6] 역시 M-VIA기반 통신 모듈이 TCP기반 통신 모듈보다 월등함을 보인다.



[그림 6] five processes in each node (total 15processes)

실험결과 M-VIA기반 통신 모듈이 기존의 TCP기반 통신 모듈보다 더 나은 성능을 보임을 확인 하였다. 또한 스트라이프 유닛 사이즈도 통신 모듈의 전송 속도에 작은 영향을 미침을 알 수 있다.

5. 결론

본 논문에서는 클러스터 파일 시스템의 기존 소켓기반의 TCP/IP를 이용한 통신모듈이 지니고 있는 문제점인 대용량의 데이터에 대한 노드간의 전송속도 문제점을 해결하기 위해 클러스터 파일 시스템에 적합한 사용자 수준의 통신 프로토콜인 M-VIA를 선정하였고, M-VIA가 클러스터 파일 시스템에 적합한 프로토콜인지 타당성을 제시하였다. 그리고 M-VIA를 이용하여 통신모듈의 기능을 효과적으로 구현하는 방안을 제시한 후에 Fast Ethernet 상에서 고속 통신

모듈의 모델을 제시하고 구현하였다. 본 연구 결과를 토대로 좀더 다양한 네트워크 환경 하에서의 통신 성능 측정이 필요하다. 그리고 통신 모듈의 최적화의 요건이 되는 다양한 패킷사이즈에 대한 테스트와 함께 최고의 성능을 발휘할 수 있는 효율적인 디스크 스캐폴링 알고리즘을 통해 성능 향상을 추구하여 성능향상을 시도해 보아야 할 것이다.

[참고문헌]

- [1] Rajkumar Buyya, High Performance Cluster Computing Programming and applications, Volume 2, 1999
- [2] S.Pakin, V.Karamcheti, and A.Chien,"Fast Messages: efficient, portable communication for workstation clusters and MPPs," IEEE Concurrency, Apr.1997
- [3] T. Von Eicken and al. " active Messages: A Mechanism for Integrated Communication and Computation," IN Proceedings of the 19th Symp. Computer Architecture, Gold Coast, And. Australia, May 1992
- [4] D. Dunning, G.Regnier, G.McAlpine, D. Cameron, B.Shubert, F. Berry, A.M.Merritt, E. Gronke, and C. Dodd. " The Virtual Interface Architecture," IEEE Micro, 18(2), Mar. 1998
- [5] National Energy Research Scientific Computer Center. M-VIA: A high performance Modular VIA for Linux. <http://www.nersc.gov/research/FTG/via/>
- [6] "Virtual Interface Architecture Specification," Version 1.0, December 1997, http://developer.intel.com/design/servers/vi/developer/ia_imp_guide.htm