M-VIA를 이용한 병렬 VOD 서버 통신 모듈 설계 및 구현

*이 연구는 BK21 충남대학교 정보통신인력사업단의 지원을 받았음

유찬곤[0] 박의수 최현호 김형식 유관종
충남대학교 컴퓨터과학과
{jargon[0], uspark, hyuno, hskim, kjyoo}@cs.cnu.ac.kr

Design and Implementation of Communication Module for Parallel VOD Server using M-VIA

Yoo, Chan-gon[0] Park, Ui-Soo Choi, Hyeon-Ho Kim, Hyeong-Sik Yoo, Kwan-Jong
Dept. of Computer Science, Chungnam National University

요 약

병렬 VOD 서버는 효율적인 부하 분산을 위해 하나의 비디오 파일을 여러 노드에 나누어 저장하므로 VOD 서비스 제공을 위해서는 노드 간 멀티미디어 데이터 이동이 필수적이다. 그러므로 노드 간 데이터 이동을 지원하는 네트워크 대역폭이 전체 시스템 성능을 결정하는 요소가 된다. 본 논문에서는 사용자 수준 프로토콜 중 산업 표준 프로토콜인 VIA를 사용하여 병렬 VOD 서버를 위한 고속의 통신 모듈을 설계하고 구현한 결과를 제시한다.

1. 서 론

병렬 VOD 서버는, 사용자들의 서비스 요구가 특정 영화에 대해서 집중적으로 증가하는 경우에도 균등하게 부하를 분산시키기 위해서 하나의 영화 파일을 여러 노드에 나누어 저장하는 데이터 스트라이핑 방식을 사용한다[1]. 그러므로 병렬 VOD 서버에서는 노드 사이에 대용량의 멀티미디어 데이터 이동이 필수적이며 노드 간 데이터 이동 속도가 전체 시스템 성능을 좌우하는 중요한 요소가 된다. 노드 간 네트워크 대역폭을 증가시키기는 것은 높은 대역폭을 지원하는 네트워크 장비를 사용하는 것만으로 해결되는 것이 아니다. 그와 더불어 효율적인 네트워크 프로토콜의 사용이 요구된다. 높은 대역폭의 네트워크 장비를 사용한다 할 지라도 비효율적인 프로토콜의 사용은 전체 성능 저하의 중요한 원인이 되기 때문이다.

본 논문에서는 노드 간 네트워크 대역폭 향상을 위해 사용자 수준 프로토콜(User-level protocol) 중의 하나인 VIA(Virtual Interface Architecture)를 사용한다. 사용자 수준 프로토콜이란 사용자 프로세스와 네트워크 인터페이스 사이에 커널의 개입 없이 바로 데이터를 주고받는 프로토콜로서 무 복사(Zero-copy) 전송을 핵심 개념으로 한다. 무 복사 전송 프로토콜로는 FM, U-Net, AM, BIP[2,3,4,5] 등이 있으며 이러한 여러 사용자 수준 프로토콜들의 산업 표준이 VIA이다[6].

본 논문의 구성은 다음과 같다. 2장에서 VIA 명세의 한 구현 소프트웨어인 M-VIA에 대해서 소개하고 3장에서는 M-VIA를 이용한 병렬 VOD 통신 모듈의 구조를 설명한다. 4장에서는 TCP를 사용하여 만든 통신 모듈과 M-VIA를 사용하여 만든 통신 모듈의 성능 비교를 한 후 5장에서는 본 논문을 마무리한다.

2. 관련 연구

2.1 M-VIA

M-VIA(Modular-VIA)[7]는 리눅스 상에서 실행되도록 구현된 VIA 명세를 지원하는 소프트웨어 중의 하나로서, 아직은 몇몇 모델의 패스트이더넷 카드와 기가비트 카드만 지원한다[8]. 또한 비신뢰성(Unreliable) 전송 수준과 신뢰성(Reliable) 전송 수준의 2가지 신뢰성 수준만을 지원한다. M-VIA는 전송 시에는 완전한 무 복사를 지원하지만 수신 시에는 한번의 복사 작업이 일어난다.

2.2 VIA 프로그래밍 시 주의사항

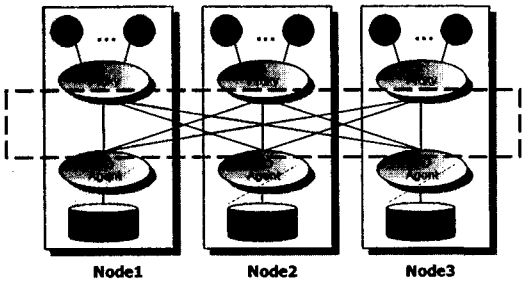
VIA는 디스크립터를 기반으로 하여 송수신 작업을 행한다. 데이터를 송신하는 측에서는 송신을 위한 디스크립터를 송신 큐에 넣으며, 수신 측에서는 수신을 위한 디스크립터를 수신 큐에 넣고 데이터 수신을 기다린다. 문제는 수신 큐에 수신 디스크립터가 없는 경우 발생한다. 이런 경우, 수신 측에서는 패킷을 제대로 받을 수 없으므로 VIA 프로그래밍 시에는 수신 큐에 항상 충분한 디스크립터가 존재하도록 해야 한다. 또한, 큐에 존재할 수 있는 디스크립터의 개수도 VIA 소프트웨어마다 다르고, 디스크립터 처리에 따르는 오버헤드가 크므로 크기가 작은 패킷이 자주 이동하는 경우일수록 성능이 낮아진다. 따라서 위와 같은 사항에 유의하여 시스템을 설계하여야 한다.

3. 통신 모듈 설계

3.1 병렬 VOD 서버 구조

본 논문에서 제시하는 병렬 VOD 서버 구조는 [그림1]과 같다. 각 노드는 분산 저장되어 있는 스트라이프 유닛들을

Proxy와 I/O Agent를 통해서 주고받는다.



[그림 1] 병렬 VOD 서버 구조

각 노드의 Proxy와 I/O Agent는 [표1]에 제시된 패킷 구조를 사용한다. PX2IO 패킷은 Proxy가 I/O Agent에게 필요한 데이터를 요청하기 위해 사용하고 IO2PX 패킷은 I/O Agent가 Proxy에게 요청한 스트라이프 유닛을 전달하기 위해 사용한다.

[표 1] PX2IO와 IO2PX의 패킷 구조

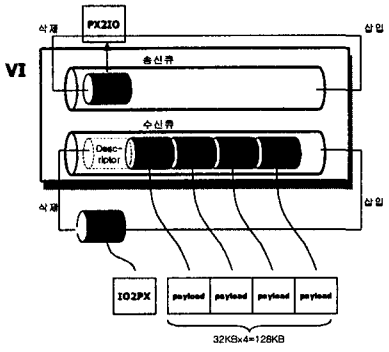
PX2IO	node_id	페이로드를 요청하는 Proxy의 노드 ID
	thread_id	Thread 번호(CS의 ID로 사용)
	movie_name	영화 이름
	offset	스프라프 파일 상에서의 오프셋
	stripe_num	스트라이프 유닛 번호
IO2PX	node_id	페이로드를 수신할 Proxy의 노드 ID
	thread_id	Thread 번호(CS의 ID로 사용)
	data_length	Payload 버퍼에 있는 실제 데이터의 길이
	stripe_num	스트라이프 유닛 번호
	payload	영화 파일 데이터

3.2 CVI, CCq 클래스 설계

VIA를 이용한 통신은 로컬 및 원격 주소 설정, 디스크립터와 데이터 버퍼를 위한 메모리 할당 및 등록, VI 생성과 상대 프로세스의 VI와 연결 설정, 송수신을 위한 디스크립터의 처리 등의 작업을 필요로 한다. 이러한 일련의 작업들을 효율적으로 관리하기 위하여 본 논문에서는 VI 사용을 위한 클래스 CVi와 완료 큐(Completion queue) 사용을 위한 CCq를 정의하여 VI와 CQ를 객체로 다룰 수 있게 하였다. CVi 클래스는, 데이터 송수신 시 메모리 할당 및 등록으로 인한 지연 시간을 줄이기 위해 클래스 생성자(Constructor)에서 데이터 송수신이 일어나기 전에 미리 필요한 디스크립터 및 데이터 저장을 위한 메모리를 등록한다.

3.3 디스크립터 관리 방법

VIA를 이용하여 프로그래밍 할 때에는 수신 큐에 디스크립터가 준비되어 있지 않음으로써 발생되는 경쟁 상태(race condition)의 문제를 막아야 한다. 본 논문에서 Proxy와 I/O Agent는 클라이언트 풀(client-pull) 방식으로 동기화 한다. Proxy는 I/O Agent에게 PX2IO 형태의 데이터 패킷을 보내고 I/O Agent로부터 IO2PX 형태의 16바이트 데이터 패킷과 고정된 크기의 페이로드(*payload*)를 받는다. 본 논문에서는 페이로드의 크기를 32KB, 64KB, 128KB, 256KB의 4 가지로 고정하였다. M-VIA1.2의 MTS(Maximum Transfer Size)는 32KB이므로 256KB의 데이터를 수신하는 경우에는 8번에 걸쳐서 수신작업이 이루어져야 한다. 즉, Proxy는 I/O Agent로부터 IO2PX 패킷과 32KB씩 8번의 페이로드를 받을 수 있도록 항상 수신 큐에 디스크립터를 준비해 놓아야 한다. Proxy가 I/O Agent로부터 IO2PX 패킷과 페이로드 패킷을 받은 후에는 다시 PX2IO 패킷을 I/O Agent에게 보내 다시 IO2PX 패킷과 페이로드 패킷을 받는다. 결론적으로 Proxy의 수신 큐에 최소한 존재해야 하는 디스크립터의 수는, 페이로드의 크기가128KB인 경우 IO2PX패킷을 받기 위한 디스크립터 1개와 페이로드 수신을 위한 4개의 디스크립터, 합하여 모두5개이고, 이 들 패킷들은 반복해서 순서대로 Proxy 측 VI의 수신 큐에 도착한다. 그러므로 이들 디스크립터를 [그림2]와 같이 수신 큐 내에서 순환하도록 하면, 5개의 디스크립터만으로 경쟁 상태가 발생하지 않도록 할 수 있다. I/O Agent는 Proxy로부터 PX2IO 패킷을 받을 수 있도록 수신 큐에 디스크립터를 대기 시켜야 한다. 이도 역시, IO2PX 패킷과 페이로드 패킷을 보내고 Proxy에서 수신이 완료되어야만 다시 PX2IO 패킷을 보내므로, I/O Agent에서는 PX2IO 패킷을 수신 후, 바로 수신 큐에 해당 디스크립터를 넣음으로써 경쟁 상태 발생을 막을 수 있다.



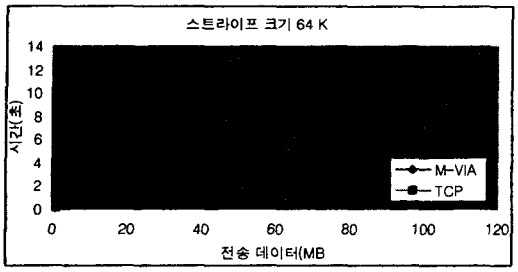
[그림 2]128KB의 스트라이프 유닛 수신을 위한 디스크립터

4. 성능 비교

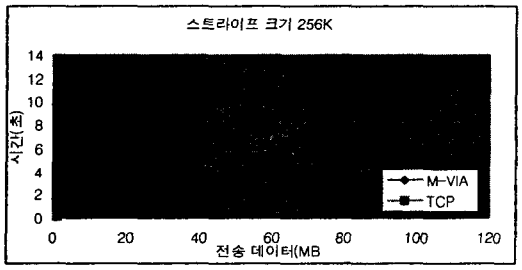
본 장에서는 노드 간 통신 프로토콜로 TCP를 사용했을 때와 M-VIA를 사용했을 때의 성능을 비교한다. 실험 환경은 [표2] 와 같으며, 100MB의 데이터를 각 스트라이프 유닛 크기 단위로 가져오는 시간을 측정하였다. 실험 결과는 [그림3]과 [그림4]에서 보는 바와 같이 M-VIA로 구현된 통신 모듈이 TCP로 구현된 통신 모듈보다 전반적으로 우수한 결과가 나왔으나 스트라이프 유닛의 크기가 성능 상에 미치는 영향은 미미한 것으로 나타났다.

[표2] 테스트 환경

하드웨어	CPU	PentiumIII 600MHz(Katmai)
	Main Memory	128MB
	하드디스크	Fast Ethenet(3Com)
	네트워크	Fast Ethenet(3Com)
소프트웨어	운영 체제	LINUX (Kernel 2.4.2-3)
	M-VIA	1.2b2
	컴파일러	g++ 2.96



[그림 3] 스트라이프 유닛 크기 64KB 일 때



[그림 4] 스트라이프 유닛 크기 256KB 일 때

5. 결론 및 향후 과제

본 논문에서는 M-VIA를 사용한 병렬 VOD 서버의 동작 모델을 제시한 후, 통신 모듈의 설계와 구현 과정을 설명하고 TCP를 사용한 통신 모듈과의 성능 비교 테스트를 실시하였다. 실험 결과에 따르면 M-VIA를 사용한 통신 모듈이 TCP를 사용한 통신 모듈보다 성능 상의 우위를 보였으며 스트라이프 유닛 크기에 따른 성능 상의 차이는 거의 없었다. 이는 패스트이더넷 상에서의 테스트 결과이며 좀더 낮은 성능의 네트워크 카드 상에서 이 차이는 더욱 커질 것으로 예상된다. 또한, 성능 상의 장점 외에도 본 논문에서 구현한 통신 모듈은 VIA 명세를 따르는 M-VIA의 VIPL(Virtual Interface Provider Library)을 사용하기 때문에 다른 VIA환경에서도 소스의 수정 없이 사용이 가능하다는 장점이 있다.

향후 과제로, 좀더 다양한 VIA 환경 및 사용자 수준 프로토콜 상에서의 성능 실험 및 비교 분석 작업이 요구된다. 성능 테스트는 테스트 환경에 따라서 매우 다양한 결과를 보이며 어느 한 환경에서의 테스트만을 가지고 일반적인 결론을 이끌어 내기에는 어려움이 있기 때문이다.

6. 참고 문헌

[1] Jack Y.B.Lee Parallel Video Servers:A Tutorial, IEEE Multimedia, Vol 5, No. 2, 1998[1]

[2] T. Von Eicken, A. Basu, V. Buch, and W. Vogels, "U-Net:a User-Level Network Interface for Parallel and Distributed Computing," In In Proceedings for the 15th SOSP, Copper Mountain, CO, Dec. 1995

[3] S. Pakin, V. Karamcheti, and A. Chien, "Fast Messages: efficient, portable communication for workstation clusters and MPPs," IEEE Concurrency, Apr. 1997

[4] L. Prylli and B. Tourancheau. "BIP: A New Protocol Designed for High Performance Networking on Myrinet. Lecture Notes in Computer Science," 1388:472-485, Mar. 1998.

[5] T. Von Eicken and al. "Active Messages: A Mechanism for Integrated Communication and Computation," In In Proceedings of the 19th Symp. Computer Architecture, Gold Coast, Qnd. Australia, May 1992

[6]"Virtual Interface Architecture Specification," Version 1.0, December 1997, http://developer.intel.com/design/servers/vi/developer/ia_imp_guide.htm

[7] National Energy Research Scientific Computer Center. M-VIA: A high performance Modular VIA for Linux. http://www.nersc.gov/research/FTG/via/

[8] Hardware Requirements http://www.nersc.gov/research/FTG/via/doc/install_guide.html#install_guide