효율적인 분산 VOD 서버를 위한 Channel Bonding 기반 M-VIA 및 인터벌 캐쉬의 활용

정 상 화[†] · 오 수 철[††] · 윤 원 주[†††] · 김 현 필[†††] · 최 영 인[†††]

요 약

본 논문에서는 분산 VOD 서버의 내부 통신망에 발생하는 부하를 줄이기 위해 channel bonding 기반 M-VIA 및 인터벌 캐쉬를 적용하는 방법을 제안한다. 분산 VOD 서버의 각 노드는 클러스터상에 분산 저장된 비디오 데이터를 서버 내부 통신망을 사용하여 전송받아 사용자에게 제공한다. 이 때, 대량의 비디오 데이터가 서버 내부 통신망을 통하여 전송됨으로 서버 내부 통신망에 부하가 증가한다. 본 논문에서는 서버 내부 통신망의 부하를 감소시키기 위해서 두 가지 기법을 적용하였다. 첫째, channel bonding을 지원하는 M-VIA를 개발하여 Gigabit Ethernet 기반 서버 내부 통신망에 적용하였다. M-VIA는 TCP/IP의 통신 오버헤드를 제거한 사용자 수준 통신 프로토콜로 통신에 소요되는 시간을 감소시켜준다. 이러한 M-VIA에 복수개의 네트워크 카드를 사용하여 통신이 가능하게 하는 channel bonding 기법을 적용함으로써 서버 내부 통신망 자체의 대역폭을 증가시켰다. 두번째, 인터벌 캐쉬 기법을 적용하여 원격 서버 노드에서 전송 받은 비디오 데이터를 지역 노드의 메인 메모리에 캐쉬함으로써, 서버 내부 통신망에 발생하는 통신량을 감소시켰다. 실험을 통하여 분산 VOD 서버의 성능을 측정하였으며, TCP/IP에 기반하고 인터벌 캐쉬를 지원하지 않는 기존의 분산 VOD 서버와 성능을 비교하였다. 실험결과, channel bonding 기반 M-VIA의 적용으로 약 20%의 성능 향상, 그리고 인터벌 캐쉬 기법을 적용하여 추가로 약 10%의 성능 향상이 생겨 총 30%의 성능 향상을 얻을 수 있었다.

키워드 : 주문형비디오, 채널 본딩, M-VIA, 캐쉬, 클러스터 시스템

Utilizing Channel Bonding-based M-VIA and Interval Cache on a Distributed VOD Server

Sang-Hwa Chung[†] · Soo-Cheol Oh[††] · Won-Ju Yoon[†††] · Hyun-Pil Kim[†††] · Young-In Choi[†††]

ABSTRACT

This paper presents a PC cluster-based distributed video on demand (VOD) server that minimizes the load of the interconnection network by adopting channel bonding-based MVIA and the interval cache algorithm. Video data is distributed to the disks of each server node of the distributed VOD server and each server node receives the data through the interconnection network and sends it to clients. The load of the interconnection network increases because of the large volume of video data transferred. We adopt two techniques to reduce the load of the interconnection network. First, an Msupporting channel bonding technique is adopted for the interconnection network. MVIA, which is a user-level communication protocol that reduces the overhead of the TCP/IP protocol in cluster systems, minimizes the time spent in communicating. We increase the bandwidth of the interconnection network using the channel bonding technique with MThe channel bonding technique expands the bandwidth by sending data concurrently through multiple network cards. Second, the interval cache reduces traffic on the interconnection network by caching the video data transferred from the remote disks in main memory. Experiments using the distributed VOD server of this paper showed a maximum performance improvement of 30% compared with a distributed VOD server without channel bonding-based MVIA and the interval cache, when used with a four-node PC cluster.

Key Words : Video On Demand, Channel Bonding, M-VIA, Cache, Cluster System

1. 서 론

최근에 인터넷의 급속한 보급 및 전송 속도 향상으로 인하여, 인터넷을 활용하는 다양한 형태의 멀티미디어 서비스가 급증하고 있다. 특히 가장 대표적인 멀티미디어 서비스인 VOD(Video On Demand : 주문형 비디오)는 방송국, 가상 캠퍼스 등의 여러 분야에서 수요가 급속히 증가하고 있으며, 앞으로는 가정용 비디오 대여 시장을 대체할 정도로 시장이 커질 것으로 예상된다.

VOD 서비스의 초기에는 사용자가 많지 않았으므로 단일 서버만을 사용하여 VOD 서비스를 하는 것이 가능하였다.

※ 본 연구는 한국학술진흥재단 지역대학우수과학자 지원사업 (과제번호:D00595, 접수번호:R05-2003-000-10726-0) 지원으로 수행되었음.
† 정 회 원 : 부산대학교 컴퓨터공학과 부교수
†† 정 회 원 : 한국전자통신연구원 선임연구원(교신저자)
††† 준 회 원 : 부산대학교 대학원 컴퓨터공학과
논문접수 : 2005년 9월 2일, 심사완료 : 2005년 10월 8일

그러나 VOD 서비스에 대한 요구가 증가하면서 최근에는 단일 서버만으로 증가된 사용자를 수용하는 것이 어렵게 되었다. 이에 대한 해결법은 클러스터 방식을 적용하여 분산 VOD 서버를 구축하는 것이다. 분산 VOD 서버는 각 서버 노드를 고속 네트워크로 연결하여 단일 VOD 서버 시스템을 구축하고 각 서버 노드에 분산 저장된 비디오 데이터를 각 서버 노드가 공유하는 형태이다. 대표적인 시스템으로 SeaChange사의 MediaCluster[1, 2], Kasenna의 Media Servers[3], University of Southern California의 Yima[4], European RDF project의 VoDKA[5]를 예로 들 수 있다. 이 방식은 비디오 데이터를 분산 VOD 서버의 디스크에 분산 저장한다. 그리고, 서비스 요청이 있을 때마다 해당 데이터가 저장되어 있는 노드의 디스크에서 서버 내부 통신망을 통하여 비디오 데이터를 전송받아서 사용자에게 제공한다. 이러한 분산 VOD 서버는 노드수를 증가시킴에 따라서 전체 분산 VOD 서버의 성능을 향상시킬 수 있다는 장점이 있다. 반면, 서버 내부의 통신망을 통하여 대량의 비디오 데이터를 전송하므로 서버 내부 통신망에 부하가 증가하여 전체 시스템의 성능이 저하될 가능성이 높다는 단점이 있다. 따라서 분산 VOD 서버의 성능을 향상시키기 위해서는 서버 내부 통신망에 발생하는 부하를 최소화하는 것이 중요하다.

서버 내부 통신망의 부하를 최소화하는 방안으로 두 가지를 들 수 있다. 첫째는 클러스터 시스템에서 사용 가능한 VIA[6]와 같은 사용자 수준 통신 프로토콜을 적용하는 것이다. 대표적인 통신 프로토콜인 TCP/IP는 통신에 소요되는 소프트웨어 오버헤드가 많기 때문에, 응용 프로그램은 네트워크의 물리적 대역폭을 충분히 활용할 수 없다. VIA는 TCP/IP의 오버헤드를 제거함으로써 사용자 프로그램이 사용 가능한 대역폭을 증가시킨 프로토콜이며, 이의 대표적인 소프트웨어 구현이 M-VIA[7]이다. 두 번째는 네트워크의 물리적 대역폭 자체를 증가시키는 방법으로 성능이 더 좋은 고속 네트워크로 장치를 교체하는 것을 들 수 있다. 대표적인 예로 SeaChange사의 MediaCluster에서 서버 내부 통신망으로 사용중인 Infiniband 가 있다. Infiniband의 대역폭은 최소 2.5Gbps며, 복수개의 링크를 사용한 환경에서는 최대 30Gbps까지 지원한다. 이것은 성능 향상은 쉬우나 고속 네트워크의 경우, 가격이 상당히 높기 때문에 비용 측면에서 쉬운 일이 아니다. 이에 대한 대안으로 고려할 수 있는 것이 channel bonding 기술이다. Channel bonding 기술은 복수개의 네트워크 카드를 장착하여 가상의 단일 고성능 네트워크 카드로 사용하는 기술로써, 고가의 장비를 적용하지 않고도 저렴한 비용으로 네트워크의 대역폭을 확장 시킬 수 있는 방법이다. 대역폭 확장이 필요할 경우 네트워크 카드를 추가적으로 설치해서 병렬적으로 활용함으로써 상대적으로 저렴한 비용으로 대역폭 확장이 가능해진다.

본 논문에서는 분산 VOD 서버를 구성하기 위해서 가격대 성능비가 높은 Gigabit Ethernet을 내부 통신망으로 사용한 PC 클러스터 시스템을 활용하였으며, 분산 VOD 서버의 내부 통신망에서 발생하는 부하를 감소시키고 대역폭을 증가시키기 위해서 앞에서 설명한 channel bonding과 M-VIA를 함께 적용하였다. 또한, 서버 내부 통신망에 발생하는 통신량 자체를 감소시키기 위해서 인터벌 캐쉬 [8] 기법을 적용하였다. VOD 서비스의 경우 신작 및 인기 비디오에 대한 서비스 요구가 집중될 것으로 예상되며, 이 경우 동일한 비디오 데이터가 네트워크 통하여 중복적으로 전송되기 때문에 네트워크에 부하가 많이 발생하게 된다. 이때 인터벌 캐쉬를 적용하여 지역 및 원격 디스크에서 전송받은 비디오 데이터를 효율적으로 캐쉬함으로써 통신망에 중복적으로 발생하는 통신량을 감소시켰다. 따라서, 본 논문에서는 channel bonding 기반 M-VIA와 인터벌 캐쉬 기법을 적용함으로써 분산 VOD 서버에서 서비스 가능한 동시 비디오 스트림의 수를 대폭 향상시켰다.

논문의 구성은 다음과 같다. 2장에서 배경연구를 소개하고, 3장에서 본 논문에서 제안하는 분산 VOD 서버에 대해서 설명한다. 4장에서는 channel bonding 기반 M-VIA 및 인터벌 캐쉬에 대해서 설명하고, 5장에서는 실험결과를 제시한다. 마지막으로 6장의 결론으로 끝을 맺는다.

2. 배경연구

분산 VOD 서버의 성능을 향상시키는 기법중의 하나는 고성능의 전용 하드웨어를 채택하는 것이다. SeaChange사의 MediaCluster는 디스크에서 읽혀진 비디오 데이터를 사용자에게 전송하는 부분의 성능 향상을 위해서, 이를 처리하는 전용 하드웨어를 장착하고 있다. Kasenna사의 Media Server 또한 MediaClustet와 유사한 기능을 하는 전용 하드웨어를 장착하고 있다. ETRI와 HICHEMTEC이 개발한 StreamXpert[9]는 무복사 개념을 적용하여 하드디스크에서 읽혀진 비디오 데이터가 메인 메모리를 거치지 않고 바로 네트워크 카드를 통하여 사용자에게 전송되도록 하는 전용 하드웨어를 개발함으로써, VOD 서버의 성능을 향상시켰다. 이와 같이 전용 하드웨어를 채택하는 방식은 VOD 서버의 성능을 많이 향상시킬 수 있다는 장점은 있으나, 고가의 전용 하드웨어를 채택함으로써 가격이 높다는 단점이 있다.

VOD 서버의 성능을 향상시키는 또 다른 방법으로 기존의 상용 하드웨어상에 소프트웨어적인 기법을 적용하는 것이 있으며, VOD 서버의 주요 자원인 디스크, 사용자 통신망, 내부 통신망의 성능향상에 관한 연구가 진행되어 왔다. 디스크의 성능 향상을 위한 연구로 스트라이핑 기법이 있다. 스트라이핑은 비디오 데이터를 디스크에 저장하기 위한 가장 일반적인 정책이며, 이의 성능 향상을 위한 것으로 가중치 스트라이핑 기법[10]이 있다. VOD 서버가 보유한 각 디스크들은 확장 및 유지 보수등의 이유로 각 디스크의 성능은 차이가 날 수 있다. 가중치 스트라이핑 기법은 성능이 좋은 디스크에는 가중치를 높게 부여하여 더 많은 비디오 데이터 블록을 저장하는 방식으로, 각 디스크의 성능을 고려한 부하균등화를 수행할 수 있기 때문에 디스크의 성능이 향상된다. 또한, 개선된 가중치 스트라이핑 기법[11]은 비디

오에 반영된 인기도를 디스크의 가중치를 계산하는데 추가하여 가중치 계산을 좀 더 정확히 함으로써 스트라이핑 기법의 성능을 개선시키려고 하였다.

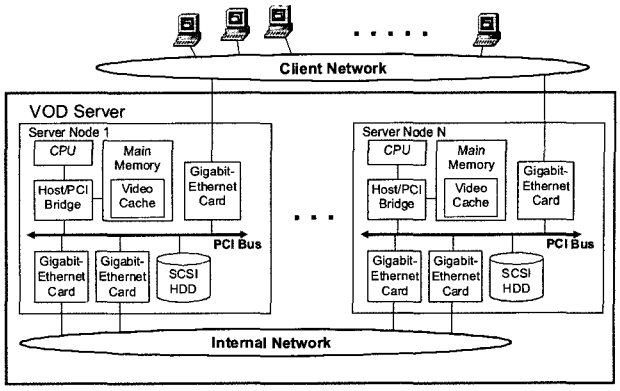
VOD 서버에서 사용자에게 비디오 데이터를 전송하는 사용자 통신망의 부하를 줄여주는 가장 기본적인 방법으로 broadcast/multicast[12, 13]가 있다.그러나, broadcast/multicast는 지연시간(비디오를 요청한 시간과 실제로 비디오 데이터가 전송되어 재생되는 시간차이)이 커서 VOD 서비스에 적절하지 못하며, 실제로 broadcast/multicast 기능을 지원하는 인터넷 환경도 거의 존재하지 않는다.

본 논문의 서버 내부 통신망에 적용할 channel bonding 기술은 Beowulf 클러스터[14]의 이더넷 channel bonding이나 네브라스카 링컨 대학의 Secure Distributed Information 그룹에서 개발한 MuniSocket(Multiple NIC Interface Socket)[15]처럼 기존의 TCP/IP 시스템 상에서 구현된 바가 있다. 또한, NCSA의 VMI[16]는 기존의 다양한 통신망 및 통신 프로토콜을 지원하는 미들웨어로써, TCP/IP, GM 및 VIA등의 다양한 통신 프로토콜을 통합하는 단일화된 통신 인터페이스를 제공하며 channel bonding 기능을 지원하고 있다. 기존의 channel bonding 방식은 동기종 네트워크상에 복수개의 네트워크 카드만을 지원하며, 통신 프로토콜 프로그램의 수정이 필요하다. 반면, VMI는 이기종 네트워크상에 복수개의 네트워크 카드를 활용한 channel bonding 기능까지 지원하고 있으며, 기존 통신 프로토콜의 상위 레벨에 구현되어 있으므로, 기존 통신 프로토콜 프로그램의 수정이 필요 없다. 이와 같이 channel bonding에 관한 다양한 연구가 수행되었으나, 이를 실제 응용 프로그램에 적용하여 그 효과를 입증하는 연구는 상대적으로 적었다.

분산 VOD 서버의 성능에 영향을 미치는 중요한 자원은 사용자 네트워크, 서버 내부 네트워크 및 디스크이다. 기존의 분산 VOD 서버에 관한 연구는 위에서 설명한 바와 같이 대부분 사용자 통신망과 디스크의 성능향상에 집중되어 왔으며, 서버 내부 통신망의 성능 향상에 관한 연구는 상대적으로 적었다. 또한 서버 내부 통신망을 위한 기존의 연구는 디스크를 연결하는 SAN(Storage Area Network)이나 고가의 고성능 네트워크를 적용하는 방향으로 진행되어 왔다. 따라서, 본 연구에서는 상대적으로 연구가 적었던 서버 내부 통신망의 성능향상에 주력하였다. 또한, 가격대 성능비가 우수한 Gigabit Ethernet상에 소프트웨어적인 방법을 적용함으로써 고가의 하드웨어 없이도 VOD 서버의 성능을 향상시키려고 하였다.

3. 분산 VOD 시스템

본 논문에서 제안하는 분산 VOD 서버의 구조는 (그림 1)과 같다. 분산 VOD 서버는 PC 클러스터 시스템을 기반으로 하였으며, 각 노드를 연결하는 내부 통신망으로 가격대 성능비가 높은 Gigabit Ethernet을 사용하였다. 각 노드는 Pentium 기반 시스템을 사용하였으며, 메인 메모리, SCSI



(그림 1) 분산 VOD 서버

HDD, 서버 내부 통신망용 Gigabit Ethernet 카드, 사용자 통신망용 Gigabit Ethernet 카드로 구성된다. 분산 VOD 서버를 구성하는 각 서버 노드의 SCSI 디스크는 서비스할 비디오 데이터를 분산 저장한다. 각 서버 노드는 사용자에게 비디오 데이터를 전송할 때, 메인 메모리상의 비디오 캐쉬에 해당 비디오 데이터가 존재하면 디스크 접근 없이 비디오 캐쉬에서 데이터를 읽는다. 해당되는 비디오 데이터가 비디오 캐쉬에 없고 지역 노드의 디스크에 존재할 경우, 지역 디스크에서 비디오 데이터를 읽어온다. 비디오 데이터가 원격 노드의 디스크에 존재한다면, 서버 내부 통신망용 Gigabit Ethernet을 사용하여 원격 디스크에서 비디오 데이터를 읽어온다. 이 때, 본 논문에서는 channel bonding 기반 M-VIA를 적용함으로써 서버 내부 통신망의 성능을 향상시켰다. 각 서버 노드는 디스크 및 비디오 캐쉬에서 읽혀진 데이터를 sever-push 모델[17]을 사용하여 사용자에게 전송한다. Server-push 모델은 서버와 사용자의 연결이 설정되면, 서버는 사용자의 중지 요구가 올 때까지 정해진 전송률로 비디오 데이터를 전송한다.

본 논문에서는 위와 같이 channel bonding 기반 M-VIA 및 인터벌 캐쉬를 적용함으로써 서버 내부 통신망을 통하여 비디오 데이터를 전송하는 시간을 최소화하고, 서비스 가능한 비디오 스트림 수를 증가시키고자 하였다.

4. Channel bonding 기반 M-VIA 및 인터벌 캐쉬

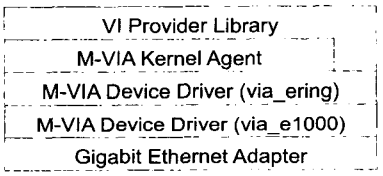
4.1 Channel bonding 기반 M-VIA

본 논문에서 제안하는 분산 VOD 서버에서는 각 서버 노드들간의 비디오 데이터를 channel bonding 기반 M-VIA를 사용하여 전송한다. Gigabit Ethernet을 위한 가장 대표적인 통신 프로토콜인 TCP/IP는 통신시 발생하는 많은 오버헤드 때문에 통신망의 물리적 성능을 최대한으로 발휘할 수 없다. 이러한 오버헤드는 첫째, 데이터가 사용자 프로그램에서 커널 및 네트워크 어댑터로 여러 번 복사되면서 발생하며, 둘째, 데이터를 전송할 때 사용자 프로그램에서 커널로 문맥이 전환(context switch) 될 때 발생한다. 이러한 문제점을 해결하기 위해서 등장한 것이 Compaq, Intel, Microsoft에서

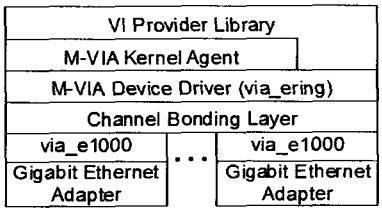
제안한 사용자 수준 통신 프로토콜의 표준인 VIA다. VIA는 통신을 커널이 아닌 사용자 수준에서 처리하게 하여 커널로의 문맥 전환 및 커널 내부에서 소요되는 시간을 제거하였으며, 통신 계층을 단순화시킴으로써 통신 중에 발생하는 데이터 복사 회수를 최소화시켰다. 이를 통하여 VIA는 통신망의 물리적 대역폭을 최대로 활용할 수 있게 하였다. M-VIA는 이러한 VIA를 Lawrence Berkeley Lab에서 리눅스상에서 구현한 것으로, VIA의 대표적인 소프트웨어 구현이다.

Channel bonding 기술은 복수 개의 네트워크 카드를 가상의 단일 네트워크 카드처럼 보이게 함으로써 쉽고 저렴하게 대역폭을 확장 시킬 수 있는 방법이다. 본 논문에서는 M-VIA와 channel bonding 기술을 동시에 적용함으로써 서버 내부 통신망의 대역폭을 향상시키고자 하였다.

(그림 2) ①은 M-VIA의 프로토콜 스택 구조를 보여준다. 최상위 계층인 VI Provider Library는 M-VIA를 위한 API (Application Programming Interface)이다. 원격 노드로 데이터를 보내기 위해서, M-VIA Kernel Agent가 두 노드사이의 연결을 설정한다. M-VIA Kernel Agent는 하드웨어 비종속적인 layer로써, 노드사이의 연결 설정에 필요한 커널 서비스를 제공한다. 일단 연결이 설정되면, 이후의 모든 통신은 M-VIA Kernel Agent를 거치지 않고 두개의 디바이스 드라이버인 via_ering과 via_e1000을 통하여 이루어진다. 이러한 방식은 커널 레벨에 존재하는 M-VIA Kernel Agent를 거치지 않음으로써, 통신수행시 커널로의 문맥 전환에 따른 오버헤드가 발생하지 않는다. via_ering은 M-VIA에서 Ethernet 기반 네트워크 카드를 위한 공통 기능을 처리하며, 네트워크 카드의 MTU (Maximum Transfer Unit) 크기로 전송할 데이터를 분할하는 역할을 한다. Ethernet 기반 네트워크 카드의 MTU는 1514 bytes이다. via_e1000은 특정 네트워크 카드에 특화된 기능을 지원하며, 본 실험에 사용된 네트워크 카드인 Intel 1000/Pro Gigabit Ethernet 카드를 위한 디바이스 드라이버이다. via_ering에서 분할된 데이터는 via_e1000 및 네트워크 어댑터를 통하여 원격 노드로 전송된다.



① M-VIA without channel bonding



② M-VIA with channel bonding

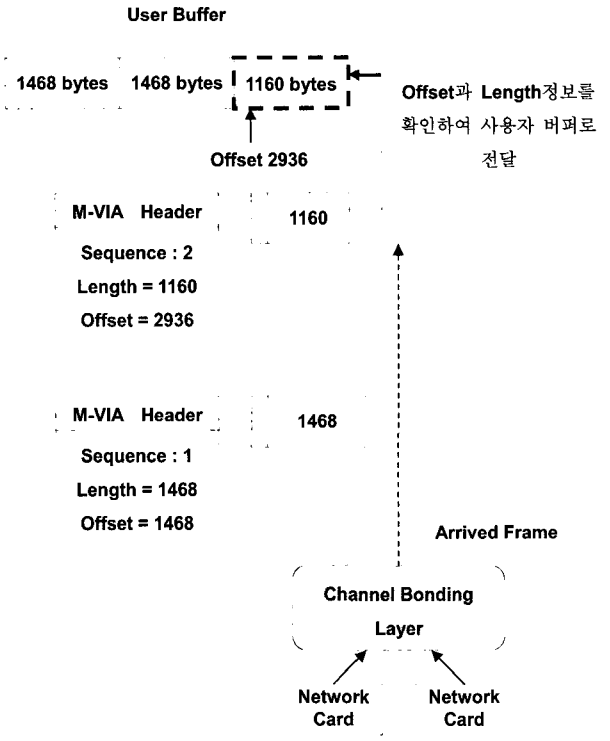
(그림 2) Channel bonding 기반 M-VIA의 프로토콜 스택

앞서 설명한 바와 같이 M-VIA는 VIA를 제공하기 위한 커널 서비스를 제공하는 M-VIA Kernel Agent 모듈, 이더넷 장치에 대한 공통적인 부분을 구현한 via_ering 모듈 및 실질적인 이더넷 장치의 디바이스 드라이버인 via_e1000으로 구성된다. Channel bonding layer는 복수 개의 이더넷 장치를 하나의 이더넷 장치로 보이게 하는 가상의 디바이스 드라이버라 할 수 있다. 따라서 channel bonding layer가 M-VIA 구조에 포함되기 위한 적절한 위치는 (그림 2) ②와 같이 via_ering 모듈과 실제의 이더넷 디바이스 드라이버인 via_e1000 사이가 될 것이다. Channel bonding layer는 복수 개의 Gigabit Ethernet 어댑터를 하나의 네트워크 어댑터처럼 관리하며, via_ering 모듈은 물리적인 네트워크 카드의 개수에 관계없이 마치 하나의 네트워크 카드가 있는 것처럼 동작할 수 있다.

Channel bonding layer에서 데이터 전송과 관련된 문제중의 하나는 복수개의 네트워크 카드중에서 어떤 것을 사용할지 결정하는 것이다. 본 논문에서는 리눅스의 channel bonding TCP/IP나 유사한 다른 업체들의 상업용 제품들에 선택되어 사용되고 있는 round-robin 방식을 적용하였다. Round robin 방식은 대부분의 경우에 있어서 공정한 패킷 분배를 보장하며, 알고리즘이 간단하기 때문에 오버헤드를 발생시키지 않는다. 선택된 Gigabit Ethernet 어댑터와 via_10000 layer는 분할된 데이터를 사용하여 패킷을 생성하고 이를 원격 노드로 전송한다.

패킷이 수신노드에 도착하면, via_e1000을 통하여 channel bonding layer로 전달된다. Channel bonding layer는 복수개의 Gigabit Ethernet 카드를 사용하여 수신된 데이터를 하나의 네트워크 어댑터를 사용하여 전송된것처럼 via_ering으로 전달한다. Channel bonding과 관련하여 수신측에서 발생할 수 있는 문제점은 (그림 3)과 같이 각 패킷이 복수개의 어댑터를 사용하여 전송됨으로, 송신측에서 먼저 전송된 패킷이 수신측에 나중에 도착하는 out of order delivery가 발생할 수 있다는 것이다. Channel bonding을 지원하지 않는 M-VIA의 경우, 각 패킷에 있는 sequence number를 조사하여 연속되지 않는 sequence number를 가진 패킷을 버리게 된다. 이러한 방식은 재전송을 발생시키게 되어 Gigabit Ethernet 어댑터의 성능을 저하시키게 된다. 이러한 문제를 해결하기 위해서, 본 논문에서는 M-VIA 패킷 헤더에 있는 offset field를 활용하였다. Offset field는 수신된 패킷 데이터가 수신 버퍼의 어떤 위치에 저장될지에 관한 정보를 가지고 있다. Channel bonding 기반 M-VIA는 연속되지 않는 sequence number를 가진 패킷을 버리지 않고, offset field를 사용하여 수신 데이터를 수신 버퍼에 바로 저장한다. 이러한 방식은 재전송을 포함한 추가적인 processing overhead를 발생시키지 않는다.

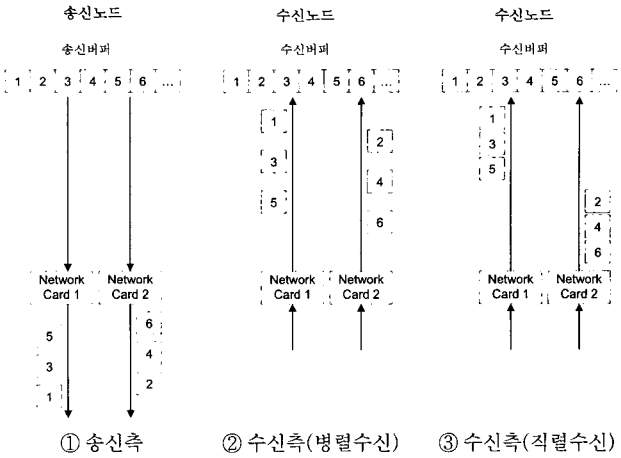
데이터 수신노드에서 발생하는 또 다른 문제중의 하나는 (그림 4)와 같다. 송신노드에서 round-robin 방식을 적용하여 두 개의 네트워크 카드를 사용하여 데이터를 전송했다고 가정하자. 이때 (그림 4) ①과 같이 송신측의 네트워크 카



(그림 3) 수신 데이터가 사용자 버퍼로 전달되는 과정

드 1은 패킷 1, 3, 5를, 네트워크 카드 2는 패킷 2, 4, 6을 동시에 전송하게 된다. 이러한 패킷들이 수신측에 도착했을 때 (그림 4) ②와 같이 송신측에서 전송한 순서대로 처리되어 수신버퍼로 복사되는 것이 복수개의 네트워크 카드를 사용한 병렬 전송을 최적화할 수 있는 방안이다. 수신측의 네트워크 카드에 패킷이 도착하면, 패킷이 도착한 사실을 인터럽트를 사용하여 호스트 CPU에 알려준다. 이 때, 기존 네트워크 카드의 경우, 인터럽트 발생시간을 감소시기 위해서 interrupt coalescing을 지원한다. Interrupt coalescing은 여러 개의 패킷이 도착한 후에 패킷의 도착 사실을 하나의 인터럽트를 사용하여 호스트 CPU에 알려줌으로써 인터럽트 발생시간을 감소시키는 것이다. 따라서, 복수개의 네트워크 카드를 사용한 경우, (그림 4) ③과 같이 네트워크 카드 1을 통하여 도착한 패킷 1, 3, 5가 한꺼번에 처리되는 동안, 네트워크 카드 2에 도착한 패킷 2, 4, 6은 기다리게 된다. 이와 같이 송신측에서 병렬로 전송한다고 하더라도 수신측의 네트워크 카드에서 이를 병렬로 처리하지 못함으로 전체 channel bonding 기반 M-VIA의 성능이 증가할 수 없다. 따라서, 이러한 문제를 해결하기 위해서 channel bonding 기반 M-VIA에서는 데이터의 수신과정을 (그림 4) ②와 같이 복수개의 네트워크 카드가 동시에 진행하도록 해야 한다.

기존의 네트워크 카드가 interrupt coalescing을 지원하는 방식은 크게 2가지가 있다. 첫번째 방법은 첫번째 패킷이 도착할 때, timer를 설정하는 것이다. 이 후, timer가 expire 되면, 이때까지 도착한 수신 패킷들을 한꺼번에 처리하는 방식이다. 두번째 방법은 하나의 인터럽트로 처리할 패킷의 수를 설정하는 방식이다. 예를 들어 패킷의 수가 5로 설정



(그림 4) 복수개의 네트워크 카드에서 수신 데이터가 처리되는 순서

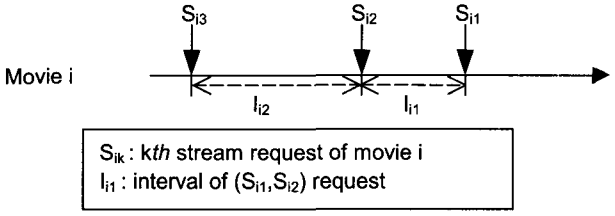
되어 있다면, 5개의 패킷이 도착했을 때 이를 인터럽트를 사용하여 처리하는 방식이다. 본 논문의 실험에서 채택한 Intel 1000/Pro Gigabit Ethernet 카드는 timer 방식을 채택하고 있다. 본 논문에서는 수신된 패킷을 (그림 4) ②와 같이 처리하기 위해서 timer 값을 0으로 세팅함으로써 interrupt coalescing 기능을 off 시켰다

4.2 인터벌 캐쉬

본 논문에서는 원격 디스크에서 읽은 데이터를 지역 노드의 메인메모리에 존재하는 비디오 캐쉬에 저장함으로써, 서버 내부 통신망에 발생하는 통신량을 감소시키려고 하였다. 비디오 데이터는 크기가 크고 순차적으로 접근되는 특징을 가지고 있기 때문에, 일반적인 데이터를 위한 캐쉬 기법을 적용하기 어렵다. 본 논문에서는 비디오 데이터를 캐쉬하는 알고리즘중에서 성능이 우수한 인터벌 캐쉬[8]를 채택하였다. 인터벌 캐쉬는 같은 비디오에 대한 스트림 요구가 연속적으로 발생할 때, 연속한 두 스트림 요구의 시간 간격에 해당하는 비디오 데이터를 캐쉬하는 정책이다. 이 정책에서는 스트림의 요구 간격이 짧을수록 자주 요구되는 데이터로 간주하여 이를 캐쉬함으로써 캐쉬의 hit ratio를 높일 수 있다. 또한, 스트림의 요구 간격이 짧은 데이터는 비디오 캐쉬 요구량도 작기 때문에, 좀 더 많은 사용자 요구에 해당하는 비디오 데이터를 캐쉬할 수 있어서 캐쉬의 hit ratio는 더욱 높아진다.

(그림 5)와 같이 동일 비디오 M_i에 대해서 스트림 요구 S_{i1}, S_{i2}, 그리고 S_{i3} 가 연속적으로 발생한다고 가정하자. S_{i1}이 서비스되고 있을 때 S_{i2}의 요구가 발생하면, 연속된 스트림 요구인 S_{i1}과 S_{i2}가 하나의 인터벌인 I_{i1}을 생성하며, 비슷한 방법으로 S_{i2}와 S_{i3}도 인터벌 I_{i2}를 생성한다. 이때, S_{i1}을 서비스하기 위해서 디스크에서 읽은 비디오 데이터는 S_{i2}를 위해서 비디오 캐쉬에 저장되며, S_{i2}는 디스크가 아닌 비디오 캐쉬에서 비디오 데이터를 읽어온다. 또한, S_{i2}도 S_{i3}을 위해서 자신이 사용한 비디오 데이터를 계속 비디오 캐쉬에 유지한다. S_{i1}이 S_{i2}를 위해서 비디오 데이터를 캐쉬할 때 필

요한 비디오 데이터의 크기는 S_{i1}과 S_{i2}의 인터벌인 I_{i1}의 시간간격에 해당하는 비디오 블록 개수이다. 예를 들어 I_{i1}에 해당하는 시간간격이 3초이고 3초동안 비디오를 재생하는데 필요한 비디오 블록수가 6개라면, 6개의 블록을 캐쉬한다.이때, 인터벌에 해당하는 비디오 블록은 circular queue형태로 관리되며, S_{i1}은 circular queue에 비디오 데이터를 쓰고, S_{i2}는 circular queue에서 비디오 데이터를 읽어가는 형태가 된다.



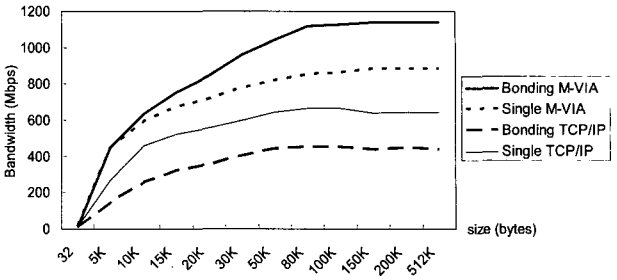
(그림 5) 인터벌 캐쉬

5. 실 험

본 논문에서는 4대의 Pentium-III 시스템을 Gigabit Ethernet으로 연결한 클러스터 시스템을 사용하여 분산 VOD 서버를 구축하였다. 각 서버는 1.4GHz Pentium-III 프로세서, 1GB 의 메인 메모리, 16GB의 SCSI 하드디스크 및 66MHz/64bit PCI 버스를 가지고 있으며, 운영체제로는 Linux 커널 2.4.7을 사용하였다. 또한, 각 서버 노드를 연결하기 위한 서버 내부 통신망으로 Intel사의 1000/Pro Gigabit Ethernet 카드와 3Com사의 SuperStack3 4900 Gigabit Ethernet 스위치를 사용하였다. 본 실험에서 사용하는 비디오 데이터는 초당 0.88Mbps의 평균 전송률을 가지는 MPEG-4 비디오이다. 모든 스트림 요구는 4초 간격으로 발생한다.

5.1 Channel bonding 기반 M-VIA의 대역폭

본 절에서는 2대의 서버를 연결하여 channel bonding 기반 M-VIA와 TCP/IP의 대역폭을 (그림 6)과 같이 측정하였다. TCP/IP를 위한 channel bonding 모듈은 Linux에 구현된 것을 사용하였다. 본 실험에서는 기존의 M-VIA, channel bonding기반 M-VIA, 기존의 TCP/IP, channel bonding기반 TCP/IP를 각각 single M-VIA, bonding M-VIA, single TCP/IP, bonding TCP/IP로 부르겠다. Bonding M-VIA의 대역폭은 1.15Gbps로 single M-VIA 의 대역폭인 0.88Gbps보다 30% 향상되었으며, 0.66Gbps의 대역폭을 가지는 single TCP/IP보다 74% 향상되었다. 반면에 bonding TCP/IP의 대역폭은 single TCP/IP보다 성능이 나빠진다. Fast Ethernet 상에서 수행된 기존의 연구[18]를 살펴보면 bonding TCP/IP의 성능은 약 105Mbps로 single TCP/IP의 대역폭인 60Mbps보다 약 75% 향상된 것을 보여준다. 4.1절에서 설명한 바와 같이 복수개의 네트워크 카드를 사용할 경우, 송신측에서 먼저 보낸 패킷이 수신측에 뒤에 도착할 수 있다. Bonding TCP/IP는 sequence number를 체크하여 연속된



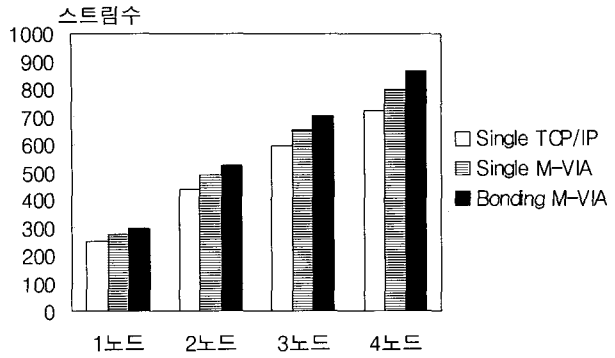
(그림 6) Channel bonding을 지원하는 M-VIA와 TCP/IP 의 대역폭

sequence number를 가지지 않는 패킷을 rejection 이나 버퍼링을 사용하여 처리한다. Rejection의 경우, 패킷의 재전송이 발생하게 됨으로 추가적으로 소요되는 시간이 상당히 증가한다. 버퍼링의 경우, 연속된 sequence number를 가지지 않은 패킷을 임시로 저장하고 있다가 저장된 패킷이 수신될 순서가 되었을 때 이를 수신 버퍼로 전송하는 방식이다. 이 방식은 시간이 많이 소요되는 데이터 복사를 발생시킨다. 이러한 오버헤드에도 불구하고 Fast Ethernet은 물리적 대역폭이 100Mbps로 매우 낮기 때문에 channel bonding을 적용함으로써 성능이 향상되었다. 그러나 Gigabit Ethernet의 경우 물리적 대역폭이 1Gbps로 상당히 높기 때문에 현재의 CPU 성능으로는 복잡한 TCP/IP 프로토콜 처리에 많은 부담이 있으며, channel bonding을 적용할 경우, TCP/IP 프로토콜을 처리하는 부담은 더욱 증가한다. 또한, rejection 혹은 버퍼링에 기반한 out of order delivery 처리로 인하여 데이터 수신에 많은 시간이 추가가 소요됨으로 bonding TCP/IP의 성능이 감소되는 것이다. 그러나 bonding M-VIA는 M-VIA 헤더의 offset 정보를 활용하여 out of order delivery 문제를 처리함으로써 추가적인 오버헤드를 발생시키지 않으며, 성능 향상을 이룰 수 있게 해준다.

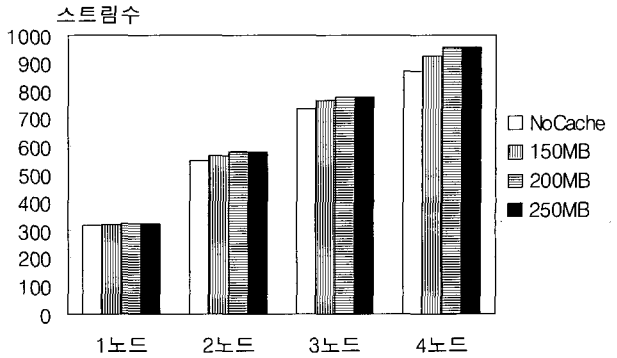
5.2 Bonding M-VIA를 적용한 분산 VOD 서버의 성능

본 실험에서는 분산 VOD 서버의 내부 통신망을 위한 통신 프로토콜로 bonding M-VIA를 사용했을 때의 성능을 측정하였으며, 비교 대상으로 single M-VIA 및 single TCP/IP를 적용한 시스템을 함께 실험하였다. 스트라이핑 블록 크기는 512KB로 하였으며, 비디오 캐쉬는 동작하지 않았다.

(그림 7)은 bonding M-VIA, single M-VIA 및 single TCP/IP를 사용했을 때, 분산 VOD 서버가 동시에 서비스 가능한 최대 스트림 수를 보여주며, bonding M-VIA를 사용한 분산 VOD 서버가 single M-VIA 및 single TCP/IP를 사용한 경우보다 더 많은 스트림을 사용자에게 서비스함을 알 수 있다. 4 노드 VOD 서버를 기준으로 했을 때 bonding M-VIA를 적용한 분산 VOD 서버의 최대 스트림수는 864개로, single M-VIA를 적용한 경우인 801개보다 8%의 성능 향상이 이루어지고 있으며, 720개의 스트림을 서비스하는 single TCP/IP 보다 20%의 성능향상이 이루어짐을 알 수 있다. 이것은 M-VIA 사용으로 인하여 서버 내부 통신망에



(그림 7) Bonding-VIA를 사용한 최대 스트림수



(그림 8) 인터벌 캐쉬를 적용한 최대 스트림 수

서 발생하는 통신 오버헤드가 많이 감소하여 데이터 전송시간이 감소했기 때문이다. 또한, channel bonding 기법의 적용으로 인하여 서버 내부 통신망 자체의 대역폭이 증가했기 때문이다.

5.1절의 실험결과에 따르면 bonding M-VIA 의 대역폭은 single M-VIA 에 비해서 30%, single TCP/IP보다 74% 성능이 향상되었으나, 본 절에서 이를 적용한 전체 VOD 서버의 성능은 각각 8%와 20% 향상된 것을 알 수 있다. 이것은 분산 VOD 서버가 bonding M-VIA를 채택한 서버 내부 통신망뿐만 아니라 디스크 및 사용자 통신망도 함께 사용하여 VOD 서비스를 수행하기 때문이다.

5.3 인터벌 캐쉬의 성능

본 실험에서는 분산 VOD 서버에 인터벌 캐쉬 기법을 적용했을 때의 성능을 측정하였다. 비디오 캐쉬의 할당량은 150MB, 200MB, 250MB로 변화시켰으며, 서버 내부 통신 프로토콜은 bonding M-VIA를 사용하였고, 스트라이핑 블록 크기는 512KB이다. 비디오 캐쉬가 동작하기 위해서는 동일 비디오 데이터에 대한 연속된 스트림 요구가 발생해야 하며, 이를 모델링하기 위해서 각 비디오에 인기도를 부여하였다. 실제 VOD 서비스에서는 인기 비디오에 대한 사용자의 요청 빈도가 더 많을 것이므로 이와 유사한 형태의 사용자 요청을 만들기 위하여 Zipf 의 법칙[19]에 따라 비디오 데이터에 인기도를 부여하였다. 본 실험에서는 20개의 비디오 데이터를 대상으로 하였다.

(그림 8)과 <표 1>은 인터벌 캐쉬 기법을 적용했을 때, 분산 VOD 서버가 동시에 서비스 가능한 최대 스트림 수 및 비디오 캐쉬의 hit ratio를 보여준다. (그림 8)을 보면 비디오 캐쉬의 크기가 증가할수록 서비스 가능한 스트림 수가 증가하며, 비디오 캐쉬의 크기가 200M가 되면서 성능향상이 포화된다. 또한, 노드수가 증가할수록 비디오 캐쉬의 성능이 향상되며, 4 노드 분산 VOD 서버를 기준을 했을 때 비디오 캐쉬가 200MB인 경우는 총 956개의 스트림을 서비스하여 비디오 캐쉬를 사용하지 않는 경우인 864개의 스트림보다 약 10%의 성능향상이 발생한다. 이것은 비디오 캐쉬의 크기와 분산 VOD 서버의 노드수가 증가할수록 <표 1>과 같이 비디오 캐쉬의 hit ratio가 증가하여, 원격 디스크에서 비디오 데이터를 전송받는 시간이 감소하기 때문이다. 따라서 디스크에서 비디오 데이터를 읽는 시간이 감소한 만큼 더 많은 비디오 데이터를 사용자에게 전송할 수 있게 된다.

<표 1> 인터벌 캐쉬 적용에 따른 비디오 캐쉬의 hit ratio

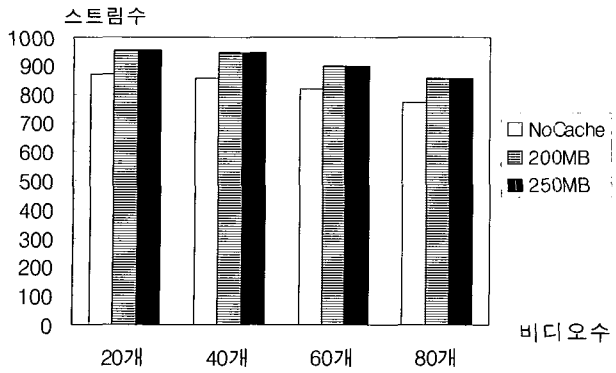
	1 노드	2 노드	3 노드	4 노드
150MB	0.43	0.68	0.68	0.67
200MB	0.68	0.75	0.78	0.80
250MB	0.68	0.75	0.78	0.80

5.4 서비스 대상 비디오 수의 변화에 따른 성능

5.2와 5.3절의 실험에서는 비디오 캐쉬의 성능을 측정하기 위해서 서비스 대상 비디오로 20개를 사용하였다. 본 절에서는 4 노드상에서 서비스 대상 비디오 수를 80개까지 변화시키면서 실험을 수행하였다.

(그림 9)에서 보는 바와 같이, 서비스 대상 비디오 수가 증가하게 되면 분산 VOD 서버가 서비스 할 수 있는 최대 스트림 수는 감소한다. 이때, 비디오 캐쉬의 크기를 250MB까지 증가시켜도 분산 VOD 서버의 성능은 더 이상 향상되지 않으며, 80개의 비디오수를 가지는 경우 855개의 스트림을 서비스 한다. 서비스 대상 비디오 수가 증가하면, Zipf 법칙에 따라서 각 비디오에 부여되는 인기도가 감소하며, 비디오 캐쉬의 대상인 동일 비디오에 대한 연속된 사용자 요구가 감소한다. 따라서 <표 2>와 같이 비디오 캐쉬의 hit ratio도 감소한다. 여기서, 비디오 캐쉬의 크기를 250MB까지 확장하더라도, 비디오 캐쉬의 대상인 연속된 사용자 요구는 증가하지 않기 때문에 비디오 캐쉬의 hit ratio도 증가하지 않으며, 더 이상 분산 VOD 서버의 성능도 증가하지 않는다.

(그림 9)를 보면, 비디오 파일 수의 증가에 따라서 비디오 캐쉬를 적용한 분산 VOD 서버의 성능도 감소하지만, 비디오 캐쉬를 사용하지 않는 경우에도 분산 VOD 서버의 성능이 감소되는 것을 볼 수 있다. Linux에서는 디스크 캐쉬를 지원한다. 비디오 파일 수의 증가에 따라서 동일 비디오에 대한 사용자의 연속된 요구가 감소하면, 디스크에 존재하는 동일 비디오 데이터에 대한 접근 요구가 감소한다. 따라서, 디스크의 캐쉬이 성능이 감소하고 전체 VOD 서버의 성능도 감소한다. 결론적으로, 비디오 캐쉬를 적용했을 때의 성



(그림 9) 서비스 대상 비디오 수의 변화에 따른 최대 스트림 수

〈표 2〉 서비스 대상 비디오 수의 변화에 따른 비디오 캐쉬의 hit ratio

	20개	40개	60개	80개
200MB	0.80	0.79	0.77	0.74
250MB	0.80	0.79	0.77	0.74

능을 비디오 캐쉬를 적용하지 않았을 때의 성능과 비교해 보면, 계속 약 10%의 성능 향상을 보이는 것을 알 수 있으며, 서비스 대상 스트림의 수가 증가하더라도 비디오 캐쉬의 효과는 유효할 것으로 판단된다.

본 절의 실험 결과에 따르면 비디오수가 80개인 경우 분산 VOD 서버는 동시에 최대 855개 스트림을 서비스 할 수 있다. 비디오 한편의 상영시간이 2시간이고, 2 시간동안 855개의 사용자 요구가 균일하게 발생한다고 가정하면, 각 스트림 요구의 평균 시간 간격은 약 8.4초(=7200초/855스트림)이다. 이때, 사용자의 스트림 요구가 발생하기 시작하여 2시간 후에는 분산 VOD 서버가 서비스할 수 있는 최대 스트림 수에 도달한다. 이후, 스트림 요구가 평균 8.4초 간격으로 발생하면, 분산 VOD 서버는 항상 자신이 서비스할 수 있는 최대 스트림을 서비스하게 된다. 스트림 요구가 8.4초 보다 작은 경우에도, 분산 VOD 서버는 항상 최대 스트림 수를 서비스한다. 그러나 스트림 요구가 분산 VOD 서버의 성능을 초과해서 발생함으로, 새로운 사용자 요구는 현재 서비스 중인 사용자 요구가 끝날 때까지 지연된다. 스트림 요구 간격이 8.4초보다 크다면, 분산 VOD 서버는 자신의 성능보다 작은 스트림 수를 서비스하게 된다.

5.3절과 5.4절의 실험결과를 종합해 보면, 비디오 캐쉬의 크기를 200MB 이상으로 증가시켜도 분산 VOD 서버의 성능은 더 이상 증가하지 않는 것을 알 수 있다. 따라서, 본 논문에서 제안하는 분산 VOD 서버가 최대 성능을 발휘하는 비디오 캐쉬의 크기는 약 200MB라고 판단된다.

5.5 분산 VOD 서버의 수행 시간 분석

<표 3>은 5.4절의 실험과 동일한 환경인 channel bonding 기반 M-VIA, 200MB 비디오 캐쉬 및 80개의 비디오를 사용했을 때, 분산 VOD 서버의 각 부분에서 소요되는 시간을 측정한 것이다. <표 3>에서 TOE(TCP/IP Offload Engine) 적용전을 보면 분산 VOD 서버의 전체 수행시간 중에서 TCP/IP를 사용하여 사용자에게 데이터를 전송하는 시간인 send_client가 전체 수행시간의 50%를 차지하고 있다. 이것은 channel bonding 기반 M-VIA 및 인터벌 캐쉬의 적용으로 인하여 내부 통신망의 영향을 받는 read_local, read_remote 및 send_remote의 성능은 향상된 반면에 send_client는 본 논문에서 최적화하지 않았기 때문에 상대적으로 수행시간 비율이 상승한 것이다. 따라서 분산 VOD 서버의 성능을 추가로 향상시키기 위해서는 send_client에서 소요되는 시간을 최적화할 필요가 있다. 그러나 인터넷에 기반한 VOD 서비스에서 사용자 네트워크는 TCP/IP 이외의 통신 프로토콜을 적용할 수가 없다. TCP/IP에 기반한 통신망에서 통신 부하를 감소시켜 성능을 향상시킬 수 있는 방법으로 TOE 를 고려할 수 있다. TOE는 통신 시간이 많은 TCP/IP 프로토콜의 처리를 호스트 CPU가 아닌 네트워크 카드상에서 수행함으로써 호스트 CPU의 부하를 줄여주는 기술이다. 대표적으로 Alacritech사의 SLIC[20] 및 Adaptec사의 NAC-7211 [21] 등이 있으며, 실험 결과에 따르면 호스트 CPU에는 약 20%의 부하만 발생시키는 것으로 알려져 있다.

따라서 본 논문에서 구현된 분산 VOD 서버에 TOE를 적용할 경우, send_client에 소요되는 시간은 50%에서 20%로 감소할 것이다. 이 경우, 감소된 30%의 시간은 read_local, read_remote, send_remote 처리에 추가로 사용되어 전체 수행 시간 비율은 <표 3>의 TOE적용후와 같이 될 것이다. 이것은 read_local, read_remote 및 send_remote 처리에 소요되는 시간이 60% 증가된 것이다. 따라서, 5.4절에서 855개의 스트림을 처리하던 분산 VOD 서버는 사용자 네트워크에 TOE를 적용함으로 인하여 60% 증가된 1368개 스트림을 서비스 할 수 있을 것으로 예상된다. 4노드를 기준으로 1368개의 스트림을 동시 처리할 경우, 각 노드는 사용자 네트워크 및 서버 내부 통신망 각각으로 301Mbps(=1368스트림 X 0.88Mbps/4노드)의 데이터를 전송해야 한다. 0.88Mbps는 MPEG4의 평균 전송률이다. 301Mbps의 데이터는 서버 내부 통신망 및 사용자 통신망에서 충분히 수용할 수 있는 데이터 양이다.

전체 VOD 서비스 가입자중에서 약 10%가 동시에 VOD 서비스를 받는다고 가정하면 본 논문에서 개발한 VOD 서버는 TOE 적용시 13680 세대(=1368스트림 * 10배)를 동시에 지원할 수 있는 능력을 가지게 된다. 13680세대는 하나의 소도시에 해당하는 인구로써, 보 논문에서 개발된 4노드 분산 VOD 서버가 소도시 하나를 서비스할 수 있을 것이며, 노드수를 증가시킬 경우 서비스 능력은 더 증가할 것으로 판단된다.

본 절에서는 전체 VOD 서버의 추가적인 성능 향상을 위해서 사용자 통신망에 TOE를 적용하는 것을 제안하였다. 이와 같이 서버 내부 통신망의 경우도 성능 향상을 위해서 TOE의 적용을 고려할 수 있으나, 기존의 Gigabit Ethernet 카드상에 channel bonding 기반 M-VIA를 적용하는 것이 가격대 성능비가 더 높은 것으로 판단된다. 따라서 본 논문

〈표 3〉 분산 VOD 서버의 수행 시간 분석

항 목	시간 비율		설 명
	TOE적용전	TOE적용후	
read_local	10%	16%	지역 디스크에서 비디오 데이터를 읽어오는 데 소요되는 시간
read_remote	25%	40%	원격 노드의 디스크에서 네트워크를 사용하여 데이터를 읽어오는 데 소요되는 시간
send_remote	15%	24%	원격 노드의 요청에 의해서 지역 디스크에서 비디오 데이터를 읽어서 원격 노드로 전송하는 시간
send_client	50%	20%	사용자 네트워크를 통하여 사용자로 데이터를 전송하는 시간
합	100%	100%	

에서 제안한 분산 VOD 서버에서는 서버내부 통신망에는 channel bonding 기반 M-VIA를, 사용자 통신망에는 TOE를 적용하는 것이 최적화된 시스템 구성인 것으로 판단된다.

6. 결 론

분산 VOD 서버에서는 서버 내부 통신망에 발생하는 비디오 데이터 전송 부하를 최소화시키는 것이 중요하다. 본 논문에서는 서버 내부 통신망에 발생하는 부하를 감소시킨 분산 VOD 서버를 제안하였다. 첫째, channel bonding을 지원하는 M-VIA를 개발하여 서버 내부 통신망의 대역폭을 증가시켰다. 두 번째, 인터벌 캐쉬 기법을 적용하여 원격 서버 노드에서 전송 받은 비디오 데이터를 지역 노드의 메인 메모리에 캐쉬함으로써, 서버 내부 통신망에 발생하는 통신량을 감소시켰다. 4 노드 시스템을 기준으로 한 실험을 통하여 분산 VOD 서버의 성능을 측정하였으며, TCP/IP에 기반하고 인터벌 캐쉬를 지원하지 않는 기존의 분산 VOD 서버와 성능을 비교하였다. 실험결과, channel bonding 기반 M-VIA 적용으로 약 20%의 성능 향상, 그리고 인터벌 캐쉬 기법을 적용하여 추가로 약 10%의 성능 향상이 생겨 총 30%의 성능 향상을 얻을 수 있었다. 또한, 분산 VOD 서버가 최적의 성능을 발휘하기 위해 필요한 비디오 캐쉬의 크기는 200MB로 전체 비디오 데이터의 크기에 비해 상대적으로 작음을 알 수 있었다. 향후 과제로 5.5절에서 제안한 바와 같이 사용자 통신망에 TOE를 적용하여 분산 VOD 서버의 성능을 향상시키는 방안에 대해서 연구를 진행할 것이다.

참 고 문 헌

[1] http://www.schange.com/Products/MediaCluster.asp

[2] "VOD over IP, SeaChange Technology Overview," White paper, SeaChange, 2003.

[3] "Kasenna Media Servers", Datasheet, Kasenna.

[4] Cyrus Shahabi, Roger Zimmermann, Kun Fu, and Shu-Yuen Didi Yao, "Yima: a second-generation continuous media server," Computer, Vol.35, Issue 6, pp.56-62, June, 2002.

[5] M. Barreiro, V. M. Gulias, J. L. Freire, and J. J. Sanchez. "An Erlang-based hierarchical distributed VoD," 7th International Erlang/OTP User Conference(EUC2001), Ericsson Utvecklings AB, September, 2001.

[6] D. Dunning et al., "The Virtual Interface Architecture," IEEE Micro, Vol.18, No.2, pp.66-76, 1998.

[7] P. Bozeman and B. Saphir, A Modular High Performance implementation of the Virtual Interface Architecture, Proc. of the 2nd Extreme Linux Workshop, 1999.

[8] http://www.movain.com/product/StreamXpert.htm

[9] Yuewei Wang, David H. C. Du, "Weighted Striping in Multimedia Servers," IEEE multimedia systems, pp.102-109, June, 1997.

[10] You-Jung Ahn; Jong-Hoon Kim; Yoo-Hun Won, "A placement policy to maximize the number of concurrent video streams supported in clustered video-on-demand servers," Proceedings of the IEEE Region 10 Conference, pp.333-336, Sept., 1999.

[11] K.A. Hua and S.Sheu, "Skyscraper broadcasting: A new boradcasting scheme for metropolitan video-on-demand systems," In Proc. of ACM SIGCOMM, Sep., 1997.

[12] L-S. Juhn and L-M. Tseng, "Harmonic broadcasting for video-on-demand service," IEEE Transaction on Broadcasting, Sept., 1997.

[13] http://www.beowulf.org/intro.html

[14] N. Mohamed, J. Al-Jaroodi, H. Jiang, and D. Swanson, "A Middleware-Level Parallel Transfer Technique over Multiple Network Interfaces," in Proceeding IPDPS 2003, Workshop on Advances in Parallel and Distributed Computational Models, Nice France, April, 2003.

[15] S. Pakin and A. Pant, "VMI 2.0: A Dynamically Reconfigurable Messaging Layer for Availability, Usability, and Management", HPCA-8, Feb., 2002.

[16] Jack Y.B. Lee, "Parallel Video Servers: A Tutorial," IEEE Multimedia, Vol.5, No.2, April/June, 1998.

[17] A. Dan, D. Dias, R. Mukherjee, D. Sitaram and R. Tewari, "Buffering and Caching in Large-Scale Video Server," In Proceeding COMPCON. IEEE, 1995.

[18] T. Sterling, D. Becker, D. Savarese, M. Berry, C. Reschke, "Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation," Proceedings of IPPS '96, pp.104-108, 1996.

[19] R. L. Axtell, "Zipf Distribution of U.S. Firm Sizes," Science, Vol.293, pp.1818-1820, Sept., 7, 2001.

[20] "Alacritech 1000x1 Single-Port Server and Storage

Accelerator:Chariot 4.0 Performance Testing," White paper, Alacritech.
[21] "Increase Performance of Network-Intensive Application with TCP/IP Offload Engines," White paper, Adaptec.

정 상 화

e-mail : shchung@pusan.ac.kr
1985년 서울대학교 전기공학과(학사)
1998년 Iowa State University 전기 및 컴퓨터공학과(석사)
1993년 University of Southern California 전기 및 컴퓨터공학과(박사)
1994년~2000년 부산대학교 컴퓨터공학과 조교수
2000년~현재 부산대학교 컴퓨터공학과 부교수
관심분야 : 컴퓨터구조, 클러스터 시스템, TOE, RFID, VOD

오 수 철

e-mail : ponylife@etri.re.kr
1995년 부산대학교 컴퓨터공학과(학사)
1997년 부산대학교 컴퓨터공학과(석사)
2003년 부산대학교 컴퓨터공학과(박사)
1997년~1998년 LG전자 멀티미디어 연구소 연구원
2003년~2004년 (주)아이온테크 연구원
2004년~2005년 부산대학교 BK21 사업단 기금교수
2005년~현재 한국전자통신연구원 선임연구원
관심분야 : 클러스터 시스템, TOE, VOD

윤 원 주

e-mail : anospirit@pusan.ac.kr
2002년 부산대학교 컴퓨터공학과(학사)
2004년 부산대학교 컴퓨터공학과(석사)
2004년~현재 부산대학교 컴퓨터공학과 박사과정
관심분야 : TOE, RFID, VOD

김 현 필

e-mail : handfeel@pusan.ac.kr
2005년 부산대학교 컴퓨터공학과(학사)
2005년~현재 부산대학교 컴퓨터공학과 석사과정
관심분야 : VOD, RFID

최 영 인

e-mail : chotbull@pusan.ac.kr
2005년 부산대학교 컴퓨터공학과(학사)
2005년~현재 부산대학교 컴퓨터공학과 석사과정
관심분야 : VOD, TOE