리눅스 기반의 클러스터 VOD 서버와 내장형에 클라이언트의 구현

(Implementation of a Cluster VOD Server and an Embedded Client based on Linux)

서 동 만 † 방 철 석 †† 이 좌 형 †† 김 병 길 †† 정 인 범 †††

(Dongmahn Seo) (Cheolseok Bang) (Joahyoung Lee) (Byounggil Kim) (Inbum Jung)

요 약 VOD 시스템에서는 한정된 자원을 이용하여 많은 사용자에게 안정적인 QoS(Quality of Service)를 제공하는 것이 중요하다. 실제 구현 환경에서 QoS의 문제점들을 분석하기 위하여 본 연구에서는 소스 공개 플랫폼인 리눅스를 기반으로 하는 클러스터형 VOD 서버와 내장형 클라이언트 시스템을 구현하였다. 서버에서는 MPEG 미디어 데이타의 병렬 처리와 부하 균등, VCR 기능을 구현하였다. VOD 사용자에게 보다 친밀한 인터페이스를 제공하기 위하여 일반 TV를 단말로 사용하였고 VCR 기능들을 제공하기 위하여 내장형 보드를 사용하였다. 본 논문에서는 사용자들의 다양한 요구사항에서의 성능 측정을 바탕으로 VOD 시스템의 성능 한계 원인들을 평가한다. 또한, 분석된 자료를 근거로 VOD 시스템에서의 가용 메모리와 가용 네트워크 대역폭을 기준으로 하는 동적 진입 제어의 방법을 제안한다. 제안된 방법은 시스템 자원의 효율성을 증가시켜 더욱 많은 사용자들에게 QoS가 보장되는 미디어 스트림을 제공한다.

키워드 : 클러스터, 주문형 비디오, 내장형 시스템, 리눅스, MPEG-2, 진입 제어

Abstract For VOD systems, it is important to provide QoS to more users under the limited resources. To analyze QoS issues in real environment, we implement clustered VOD server and embedded client system based on the Linux open source platform. The parallel processing of MPEG data, load balancing for nodes and VCR like functions are implemented in the server side. To provide more user friendly interface, the general TV is used for a VOD client's terminal and the embedded board is used supporting for VCR functions. In this paper, we measure the performance of the implemented VOD system under the various user requirement features and evaluate the sources of performance limitations. From these analyses, we propose the dynamic admission control method based on the availability memory and network bandwidth. The proposed method enhances the utilization of the system resource for the more QoS media streams.

Key words : cluster, VOD, Embedded system, Linux, MPEG-2, admission control

1. 서 론

주문형 비디오 즉 VOD(Video On Demand) 서비스는 사용자의 요구에 따라 영화 데이타를 네트워크를 통하여 실시간으로 사용자에게 서비스하여 주는 것을 의미한다. VOD 서비스를 제공하기 위해서는 서비스를 제공하는 VOD 서버와 일반 사용자가 서비스를 받기 위해 사용하는 VOD 클라이언트 장치가 필요하다. VOD 서비스에서 이러한 서버와 클라이언트의 상호 관계와 특성을 고려하지 않고 VOD 서비스 시스템을 구현하는 경우 좋은 성능을 기대할 수 없다[1]. VOD 서비스 시스템의 성능은 비디오 스트림에 대한 일정한 QoS (Quality of Service)를 보장하는 동시 접속 사용자의

· 본 연구는 강원대학교 ITRC의 지원을 받아 수행하였음
· 본 연구는 한국과학재단 목적기초연구(R05-2003-000-12146-0)의 지원으로 수행되었음
· 이 논문은 강원대학교 두뇌한국21사업에 의해 지원되었음

† 비 회 원 : 강원대학교 컴퓨터정보통신공학과
dmseo@snslab.kangwon.ac.kr
†† 학생회원 : 강원대학교 컴퓨터정보통신공학과
csbang@snslab.kangwon.ac.kr
jhlee@snslab.kangwon.ac.kr
bgkim@snslab.kangwon.ac.kr
††† 정 회 원 : 강원대학교 전기전자정보통신공학부 교수
ibjung@kangwon.ac.kr

논문접수 : 2003년 7월 24일
심사완료 : 2004년 9월 13일

수로 나타낼 수 있다. QoS는 VOD 서버 특성, 저장 장치의 성능, 네트워크의 대역폭 그리고 클라이언트의 구조와 밀접한 관련이 있다.

본 논문에서는 클러스터 형태로 구성된 VOD 서버에서 다양한 사용자의 서로 다른 요구와 영화 데이타의 특징을 고려하여 보다 많은 사용자에게 안정적인 서비스를 제공하기 위한 연구 결과를 제시하고자 한다. 고성능 단일 VOD 서버는 성능을 향상시키기 위해 서버를 확장하기가 어렵다는 단점이 있다. 클러스터형 VOD 서버는 새로운 노드의 추가를 용이하게 함으로서 성능 향상을 기대할 수 있는 확장성을 갖추고 있고 가격 대 성능 비가 우수하다는 장점을 가지고 있다[1-5]. 본 논문에서는 MPEG 데이타를 각 서버 노드에 스트라이핑 하여 분산 저장하고, 사용자에게 제공함으로서 MPEG 데이타에 대한 병렬 저장 및 인출을 연구하였다. MPEG 데이타를 클러스터형 병렬 서버에서 분산 저장하므로 서버를 구성하는 모든 노드들에 동일한 부하가 분담되기 때문에 부하 분산 정책을 쉽게 유지할 수 있다.

VOD 서비스는 영화 데이타들의 압축 기법으로 인해 VCR과는 달리 고속 재생과 역재생에 몇 가지 문제점들을 가지고 있다. 단지 일반 재생용 데이타를 빠르게 상영할 경우 네트워크에 과부하가 발생한다. 본 논문에서 사용하는 영화 데이타인 MPEG-2의 경우, 초당 24~30 프레임으로 구성되고, 고속 재생일 경우에 이보다 많은 프레임이 네트워크로 전송되어야 하기 때문이다. 본 논문에서는 MPEG-2 형태로 구성된 하나의 영화 데이타를 쪼개어서 서버 내부 각각의 노드에 병렬 저장하고, 사용자의 요청 시 병렬 인출하여 클라이언트에게 전달하도록 하였다. 병렬처리를 위한 성김도 단위를 MPEG-2 특성을 이용하기 위하여 GOP(Group Of Picture)로 하였다. GOP는 독립적으로 상영될 수 있으며 하나의 GOP는 동일한 시간 동안 상영되므로 QoS 제어가 수월하고, 고속 재생에서 사용되는 I 프레임들과 동기를 맞추기 용이하다는 장점이 있기 때문이다.

VOD 서버에서 일정한 QoS를 제공하면서 동시 접속 사용자 수를 증가시키기 위해서 진입 제어(Admission Control)가 필요하다. 진입 제어 기준을 마련하기 위해서는 VOD 서버의 성능 분석이 필수적이다. VOD 서버의 성능이 반영되지 않은 진입 제어 기준은 해당 서버의 여러 요소들을 반영하지 못하기 때문에 효과적인 진입 제어를 기대하기 어렵다. 이러한 이유들로 본 연구에서는 진입 제어에 영향을 미칠 수 있는 VOD 시스템 내부의 원인들을 분석하기 위하여 클러스터 서버 내부의 각각의 노드 상황을 살펴 볼 수 있는 모니터링 기능을 구현하여 사용자들의 요구 변화에 따른 메모리, 디스크 및 네트워크 장치들에서의 성능 변화량을 측정 할 수 있게 하였다.

VOD 서비스에서 서버가 전체적 성능에 중요한 부분을 차지하고 있지만, 사용자에게 서비스를 제공하는 것은 VOD 클라이언트이다. VOD 클라이언트는 사용자의 요구를 서버에게 전달하고, 사용자에게 편리한 인터페이스 환경을 제공하며 MPEG 데이타를 복호화 하여 사용자에게 보여주는 역할을 수행한다. 또한 MPEG 데이타의 복호화 이외에 서버로부터 전송되는 연속적인 미디어 스트리밍 데이타에 대한 적절한 버퍼링은 정상적인 서비스의 중요한 요인이 된다. 이러한 VOD 클라이언트의 경우 PC상에서 어플리케이션 프로그램을 통하여 서비스가 가능하지만, 본 연구에서는 VOD 서비스 사용자의 폭넓은 확대를 위해서는 컴퓨터에 친밀하지 않은 일반 사용자들도 쉽게 영화를 즐길 수 있으며, PC 보다 영화를 시청하기 좋은 플랫폼을 제공하는 TV를 VOD 클라이언트 단말로 사용하였다. 일반 TV를 VOD 서비스의 최종 단말로 사용하기 위해서는 다수의 사용자들로부터의 다양한 영화들에 대한 자유로운 선택 요구를 처리하는 기능, 즉 대화형(On-Demand) 기능이 있어야 한다. 본 연구에서는 이러한 사용자들의 대화형 요구를 처리하기 위하여 리눅스 내장형 보드를 사용하였다. 내장형 보드에 장착된 적외선 입출력 장치를 통하여 VOD 서비스 사용자들의 대화형 요구를 입력받아 서버에 전송후 처리 결과를 받도록 하였다.

구현된 VOD 시스템에서 다양한 사용자들의 동시 접속자 수를 계측 프로그램을 통하여 측정하였고, 이러한 성능 측정 결과를 분석하여 본 연구에서는 수시로 변화되는 VOD 시스템 내부의 가용 자원의 정보를 활용한 안정적인 QoS를 지원할 수 있는 동적 진입 제어 방법을 제안한다.

제안되는 진입제어 방식은 VOD 시스템에서의 가용 메모리와 가용 네트워크 대역폭을 기준으로 고안되었으며 시스템 자원의 효율성을 증가시켜 더욱 많은 사용자들에게 QoS가 보장되는 미디어 스트림을 제공할 수 있게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 MPEG-2 데이타의 스트라이핑과 VCR 기능에 관하여 설명한다. 3장에서는 클러스터 VOD 서버에 관하여 설명하고, 4장에서는 내장형 VOD 클라이언트 시스템을 설명한다. 5장에서는 구현된 시스템의 성능을 측정하고 결과를 분석한다. 6장에서는 가용 자원 기반의 동적 진입 제어 방법을 제안하고, 7장에서는 관련 연구들을 살펴본다. 마지막으로 8장에서는 본 논문의 결론을 맺고 향후 연구 계획을 설명한다.

2. MPEG-2의 스트라이핑과 VCR 기능

2.1 MPEG-2

MPEG-2의 PS(Program Stream)는 비디오 시퀀스(video sequence) 계층과 GOP(Group of Picture) 계층, 그리고 픽처(Picture) 계층으로 구성되어 있다. 비디오 시퀀스 계층은 일련의 같은 속성을 갖는 화면 그룹으로서 화면 크기, 화면 비율 등의 기본적인 정보를 가지고 있다. GOP 계층은 랜덤 엑세스의 단위가 되는 화면 그룹의 최소 단위로 영화를 상영하는 기본 단위이다. 픽처 계층은 화면 한 장의 공통된 속성으로 하나의 프레임과 동일하다.

픽처 계층에서 픽처는 기능적으로 서로 다른 I, P, B, D 네 종류의 타입을 가진다. I 프레임은 자신의 화면 정보만으로 부호화 되는 화면으로, 다른 프레임과는 상관없이 독립적으로 화면에 나타날 수 있어, 랜덤 엑세스를 위해서는 하나의 GOP 내에 최소 1개 이상의 I 프레임이 필요하고, 이때 I 프레임에서부터 영화가 상영된다. P 프레임은 I 프레임 또는 다른 P 프레임으로부터의 예측을 수행함에 따라 생기는 화면이다. 일반적으로 P 프레임에는 순방향 프레임간 예측 정보를 담고 있다. B 프레임은 MPEG의 특징인 쌍방향 예측에 의해 생기는 화면으로, 일반적으로 과거 영상으로부터 예측하는 순방향 프레임간 예측 정보와 미래로부터 예측하는 역방향 프레임간 예측 정보, 그리고 전후 양방향으로부터의 예측에 의한 정보를 담고 있다. D 프레임은 고속 재생(fast-forward), 역방향 고속 재생(fast-rewind) 등에 쓰이는 DC(Direct Current : 직류) 성분만을 갖는 화면으로 VCR 기능을 구현하는데 있어 중요한 프레임이지만, 실제에서는 거의 사용되지 않는다[6].

MPEG-2의 각 계층의 시작은 각각의 헤더로 시작되며, 각각의 헤더는 특정한 바이트 스트림으로 구분할 수 있다. 헤더의 시작 바이트 스트림은 [0x000001+각 헤더 시작 코드] 의 형태로 되어 있다. 비디오 시퀀스 헤더의 시작 코드는 B3, GOP의 시작 코드는 B5, 픽처의 시작 코드는 00이다. 이 헤더 정보를 이용하여 GOP 및 각각의 픽처(프레임)들을 분리하는 것이 가능하다.

2.2 스트라이핑

클러스터형 VOD 서버에서 각 노드 또는 노드 내의 각 디스크에 영화를 분산 저장하는 것을 스트라이핑이라고 한다. 스트라이핑은 각 노드간의 부하 분산과 영화 인기도, VCR 기능과 밀접한 관련이 있다. 한 영화 데이타를 스트라이핑 하는데 있어서 특정 노드에 많은 데이타를 저장한다면, 이 영화가 요청될 경우 노드들 사이의 부하에 균형을 맞출 수 없다. 따라서 영화를 분산 저장하는 스트라이핑 단계에서부터 영화를 적절히 배분하여 저장함으로서 별도의 부하 분산 없이 각 노드간에 균등한 부하를 줄 수 있는 방법을 사용해야 한다. 영화 데이타를 스트라이핑 하여 각 노드에 균등한 부하를 담당하도록 하는 방법으로 각 노드에 동일한 크기의 데이타를 저장하는 방법과 각 노드에 동일한 시간만큼 상영 할 수 있는 데이타를 저장하는 방법이 있다. 동일한 크기의 데이타를 저장하는 방법은 스트라이핑 단계에서 쉽게 구현이 가능하며, 각 노드의 디스크 저장 공간을 균등하게 사용할 수 있다. 반면에 동일한 시간만큼 상영 할 수 있는 데이타를 저장하는 방법은 각 노드에서 담당하는 영화 상영 시간을 동일하게 맞출 수 있으며, 영화 데이타의 특성을 살릴 수 있다는 장점이 있다. 본 연구에서는 다음에 설명할 VCR기능의 구현을 위하여 동일한 시간만큼 상영 할 수 있는 데이타를 스트라이핑 하는 방법을 사용하였다.

본 연구에서는 앞에서 설명한 MPEG-2의 특성을 이용하여 영화를 스트라이핑 하였다. 스트라이핑의 단위를 하나의 GOP 단위로 하여 각 노드는 순차적으로 하나의 GOP에 GOP 번호와 크기를 헤더로 첨부하여 저장한다. 따라서 영화 데이타가 초당 30프레임이고, 하나의 GOP에 15프레임의 픽처가 포함되어 있다면, 각 노드는 0.5초의 상영 시간을 갖도록 스트라이핑 된 것이다. 각 노드에서는 스트라이핑 된 데이타를 두 개의 디스크에 다시 순차적으로 스트라이핑 하였다. 따라서 한 노드에서 2개의 GOP를 전송하는 경우 하나의 디스크는 하나의 GOP 만을 전송하도록 하였다. 노드간, 디스크간의 스트라이핑 방식은 여러 가지 방식 중에서 라운드 로빈(Round-Robin) 방식[5,7]과 주사(SCAN)방식[7]을 선택적으로 사용하도록 하였다. 라운드 로빈 방식은 노드와 디스크의 순서에 따라 순차적으로 1 -> 2 -> 3... -> N 와 같은 방식으로 분배한다. 반면 주사 방식은 순차와 역순을 번갈아 가며 1 -> 2... -> N -> N-1 ... -> 1 과 같은 방식으로 분배한다.

2.3 VCR 기능

VOD 서버에서 VCR 기능을 지원하기 위한 방법으로 별도의 파일을 두는 방법과, 빠르게 데이타를 전송하는 방법, 스캔 방식 등이 있다[1]. 별도의 파일을 두는 방법은 원래 영화 데이타와 별개로 VCR 기능을 위한 별도의 파일을 두는 것으로, 처음 영화 등록 당시에 오버헤드가 존재하지만, 실제 서비스를 할 때에는 별다른 오버헤드가 없다는 장점이 있다. 빠르게 데이타를 전송하는 방법은 영화가 초당 30프레임으로 상영되고 있을 때 빨리 감기를 수행하면 이를 초당 60프레임으로 변경하는 것을 말한다. 이 방법은 영화 등록 당시의 오버헤드는 없는 반면 실제 서비스 단계에서 네트워크와 디스크에 많은 오버헤드를 주게 된다. 스캔 방식은 영화 데이타의

특정 프레임을 추출하여 전송하는 방식으로 등록 단계에서의 오버헤드는 적으나, 서비스 단계에서 영화 데이타를 검색하여 전송하기 때문에 서버에 오버헤드를 주게 된다.

본 논문에서는 별도의 파일을 두는 방식과 스캔 방식을 혼용한 방식을 사용한다. 이렇게 함으로서 영화 등록 단계에서 오버헤드는 존재하지만, 서비스 단계에서의 많은 오버헤드를 감소시켜 서버의 동시 접속 사용자 수를 늘려 전반적인 서버의 성능 향상을 꾀할 수 있다. 앞서 언급하였듯이, 임의 접근의 기본 단위인 GOP에서 영화 상영을 시작하기 위해서는 반드시 I 프레임을 찾아 상영을 시작한다. 이는 I 프레임은 독립적으로 화면에 보여질 수 있기 때문이다. 따라서 본 논문에서는 각 GOP에서 I 프레임을 추출하여 별도의 파일을 만들어 저장하였다. GOP 단위로 스트라이핑 될 때 이 파일 역시 I 프레임 단위로 동일하게 스트라이핑 되게 함으로서 별도의 동기화 없이 순서 정보만을 이용하여 일반 재생과 고속 재생, 역재생의 변환이 가능하도록 하였다.

3. VODCA 시스템의 구조

VODCA(Video-On-Demand on Clustering Architecture)는 본 논문에서 설계 구현한 클러스터 VOD 서버이다. VODCA는 그림 1에서 보는 바와 같이 HS(Head-end Server)와 MMS(Media Management Server)로 구성된다.

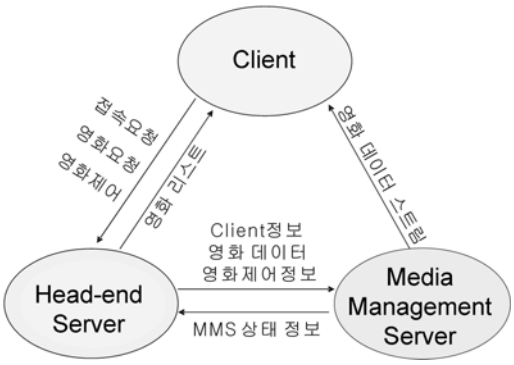

그림 1 VODCA의 구성

3.1 HS(Head-end Server) 구조

HS는 사용자의 접속 요청을 받아들이는 VODCA 시스템의 진입점으로서 일반적인 클러스터링 시스템에서의 Load Balancing 서버의 역할을 수행한다. 이를 위해 사용자의 시스템 진입 요청 및 영화 제어 요청을 처리하고, 각 MMS 노드들을 관리, 제어하는 역할을 수행한다. 또한 관리자에게 새로운 영화를 등록하기 위한 인터

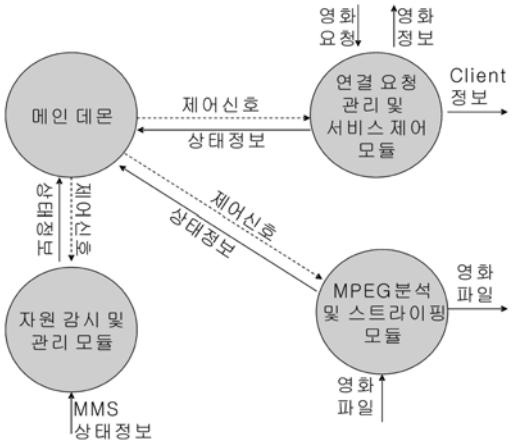

그림 2 HS의 모듈 구성

페이스를 제공하며, 등록 영화 데이타를 분석하여 각각의 MMS 노드에 분산 저장하는 역할을 수행한다. HS는 그림 2에서 보는 바와 같이 MPEG 분석 및 스트라이핑 모듈과 자원 감시 및 관리 모듈, 연결 요청 관리 및 서비스 제어 모듈, 그리고 메인 데몬(Main Daemon) 모듈로 구성된다.

3.1.1 MPEG 분석 및 스트라이핑 모듈

2.1절과 2.3절에서 설명한 MPEG2의 특성을 이용하여 MPEG-2의 영화 데이타를 파싱하여 각 헤더 정보를 읽고 분석하여, 데이타를 GOP와 I 프레임 별로 분할하고, 분할한 데이타에 순서를 표시하기 위한 시퀀스 번호와 해당 부분의 크기를 가지는 헤더를 첨부하여 각 MMS 노드에 전송한다. 2.2절에서 설명하는 바와 같이 전송 순서는 시스템 관리자의 선택에 따라 라운드 로빈 방식과 주사 방식으로 전송된다.

3.1.2 자원 감시 및 관리 모듈

이 모듈은 VODCA 시스템을 관리할 수 있는 환경을 제공한다. HS와 MMS의 CPU 사용량, 메모리 사용량, 네트워크 사용량 등의 자원 상태를 보여주며, MMS 노드의 추가, 삭제, 수정을 가능하게 한다. 모든 MMS의 자원 감시 모듈로부터 주기적으로 Heartbeat를 주고받아 각 노드의 상태 정보를 수집한다. 각 노드에 대한 모든 수집된 정보는 데이타베이스를 통해 관리된다.

3.1.3 연결 요청 관리 및 서비스 제어 모듈

클라이언트로부터 TCP 연결을 통해 접속 요청을 받으면 해당 클라이언트를 담당하기 위한 스레드를 생성한다. 이 스레드는 사용자의 영화 검색 요청에 따라 데이타베이스를 이용하여 영화 정보를 클라이언트에게 제공한다. 사용자가 영화 상영을 요청하면 MMS와 TCP 연결을 설정한 후에 사용자가 요청한 영화 코드를 해당 MMS에 전송하여 영화 전송을 개시하도록 한다. 영화

전송 개시 이후에 사용자로부터 영화 제어 요청이 수신되면 이를 해당 MMS들에게 전송하여 사용자의 요청에 맞는 영화를 전송하여 준다.

3.1.4 메인 데몬(Main Daemon) 모듈

이 모듈은 VODCA 시스템 관리자에게 전반적인 시스템 관리를 위한 인터페이스 환경을 제공하고, 다른 HS의 모듈을 제어한다. 관리자와의 모든 인터페이스는 이 메인 데몬 모듈을 통하며, 관리자의 설정에 따라 해당 모듈을 동작하도록 한다.

3.2 MMS(Media Management Server) 구조

MMS는 HS의 제어에 따라 사용자에게 영화 데이타를 전송하여 주는 서버로서 스트리밍 서버 또는 스토리지 서버의 역할을 수행한다. 2초 간격으로 자신의 시스템 자원 상태 정보를 HS에 보고하고, HS로부터 제어 명령을 수신하여 명령을 수행한다. MMS는 그림 3과 같이 영화 관리 모듈과 MMS 자원 감시 모듈, 영화 서비스 모듈, 그리고 MMS 메인 데몬 모듈로 구성된다.

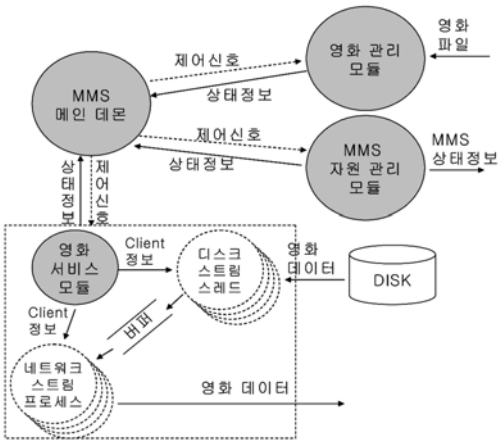

그림 3 MMS의 모듈 구성

3.2.1 영화 관리 모듈

3.1.1의 MPEG 분석 및 스트라이핑 모듈에서 영화 데이타가 분석되고 스트라이핑 된 후 MMS에 전송되면 데이타를 자신의 디스크에 분산 저장한다. HS의 명령에 따라 영화를 저장, 삭제, 수정의 작업을 수행한다.

3.2.2 MMS 자원 관리 모듈

HS에게 Heartbeat를 통하여 2초 간격으로 자신의 상태를 검사하여 보고한다. 리눅스의 /proc 파일 시스템을 이용하여 메모리 사용 상태, 스왑 영역 사용 상태, CPU 사용 상태, 네트워크 사용 상태를 수집한다. Heartbeat는 32Bytes의 구조체로 2초 간격으로 전송되기 때문에 128bps의 네트워크 대역폭을 사용하므로 시스템의 성능에는 큰 영향을 미치지 않는다.

3.2.3 영화 서비스 모듈

두 개의 버퍼를 사용하여 클라이언트에게 영화를 전송하는 모듈로 네트워크 스트림 프로세스와 디스크 스트림 스레드로 구성된다. 클라이언트의 영화 상영 요청이 HS를 통하여 수신되면, 네트워크 스트림 프로세스를 생성하여 클라이언트와의 연결을 설정하고, 디스크 스트림 스레드를 생성한 후에 버퍼에서 영화 데이타를 읽어 설정된 연결을 통해 클라이언트에게 전송하는 역할을 수행한다. 생성된 스레드는 디스크로부터 해당 영화 데이타를 클라이언트의 제어 명령에 따라 읽어 버퍼에 쓰는 역할을 수행한다.

3.2.4 MMS 메인 데몬 모듈

HS로부터 명령을 수신하여, 명령에 따라 MMS의 다른 모듈을 동작시키고 명령을 전달한다. 이 모듈을 통하여 다른 모듈들의 동작이 관리된다.

3.3 서버 개발 환경 및 구현

VODCA는 1대의 HS와 6대의 MMS로 구성하였으며, 각 노드의 사양은 표 1과 같이 일반적인 하드웨어와 소프트웨어를 사용하였다. 각 노드는 리눅스를 기반으로 하여 개발되었으며 관리자용 인터페이스를 위하여 Qt 라이브러리를 사용하였고, 그 외 부분들은 C와 C++를 사용하여 개발하였고, gcc 2.96 버전을 이용하여 컴파일 하였다. 데이타베이스는 MySQL을 사용하였다.

표 1 VODCA의 하드웨어 구성

CPU	Intel Pentium4 1.6GHz
메모리	256MB DDR
디스크	Segate Barracuda ATA Ⅳ 40GB 7200RPM × 2
운영체제	RedHat 7.3 (Kernel 2.4.18)
네트워크	100Mbps Fast Ethernet

4. 내장형 VOD 클라이언트 구조

4.1 클라이언트 구조

그림 4에서 보는 바와 같이 내장형 클라이언트는 사용자 인터페이스 모듈과 네트워크 수신 모듈, 재조합 모듈, 영화 재생 모듈로 구성되어 있다.

4.1.1 사용자 인터페이스 모듈

HS에 연결 요청을 하고 연결을 유지한다. 사용자의 요구에 따라 HS에 명령 코드를 전송하고, HS로부터 전송 받은 각종 영화 정보를 사용자에게 보여주는 역할을 수행한다. 사용자의 요청은 RS-232C 시리얼 통신을 이용하여 적외선 방식의 리모트 콘트롤러를 통해 입력받도록 하였다.

4.1.2 네트워크 수신 모듈

MMS 노드들로부터 전송되어진 영화 데이타를 수신

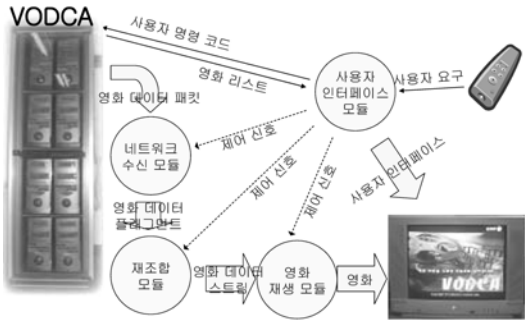

그림 4 내장형 클라이언트의 구성

하는 역할을 수행한다. 하나의 스레드가 다수의 MMS 노드들과 연결하여 데이타를 전송 받아 클라이언트 시스템의 공유 메모리 영역에 서버에 스트라이핑 된 크기 단위로 저장한다.

4.1.3 재조합 모듈

공유 메모리 영역에 저장되어 있는 영화 데이타의 헤더를 읽어서 영화 재생 모듈에서 요구하는 순서에 맞는 시퀀스 번호를 가진 데이타를 검색하여 파이프 메커니즘을 사용해 영화 재생기에게 전송하는 역할을 수행한다.

4.1.4 영화 재생 모듈

재조합 모듈과 연결된 파이프로부터 영화 데이타를 받아 디코딩 작업을 수행한다. 디코딩 된 영화 데이타는 TV 화면을 통해 사용자에게 보여준다. 고속 상영 및 역재생 동작 시에는 인코딩 과정에서 사운드 데이타를 출력하지 않도록 하여 영상 데이타만 사용자에게 출력되도록 한다.

4.2 내장형 VOD 클라이언트 개발 환경 및 구현

내장형 클라이언트의 사양은 표 2와 같다. 클라이언트는 디지털 아날로그 컨버터를 이용하여 일반 가정용 TV와 연결하여 서비스를 제공하도록 하였다. 내장형 클라이언트의 구현 과정은 크게 커널과 파일 시스템 구축, 사용자 인터페이스를 위한 환경 구축, 영화 재생기로 나누어 볼 수 있다.

표 2 클라이언트 개발 환경

Board	IB790
CPU	Intel Pentium III 800Mhz
메모리	64MB SDRAM
디스크	128MB Flash Disk
운영체제	Linux Kernel 2.4.18
입력장치	RS-232C 시리얼 적외선 수신 장치, 적외선 리모트 컨트롤러
GUI	Embedded Qt 3.0.6
Movie Player	mplayer 0.18
개발 환경	RedHat 7.3, P4 1.6GHz, 256MB

4.2.1 커널 및 파일 시스템

RedHat 7.3이 설치된 일반 PC를 개발 환경으로 하여 Filesystem Hierarchy Standard에 맞게 디렉토리 구조를 생성하여 구현 시 사용되는 패키지 및 커널을 컴파일 하여 독립적으로 부팅 가능한 상태로 만들었다[8]. 커널 컴파일 단계에서 프레임 버퍼를 활성화하도록 설정하여 GUI를 지원 할 수 있도록 하였다. 그 후에 타깃 디스크를 ext3 타입으로 포맷한 후 부팅에 필요한 최소한의 파일들을 선별하여 별도의 파티션에서 타깃 디스크로 복사하였다[8]. 그 상태에서 리눅스 설정에 필요한 파일 및 최소한의 바이너리 파일, 라이브러리 파일들을 복사하였다. 필수 유틸리티 외에 개발과 디버깅을 위해 ftp, sshd와 같은 별도의 유틸리티를 추가하였다.

4.2.2 사용자 인터페이스

클라이언트에서는 GUI 환경의 사용자 인터페이스를 제공하기 위하여 리눅스 프레임 버퍼 상에서 동작하는 윈도우 개발 환경인 Embedded Qt 3.0을 사용하였다[9]. 개발용 PC에 Qt/Embedded 3.0.6 라이브러리를 설치한 환경에서 애플리케이션을 개발하였다. VODCA 서버는 C를 기반으로 제작되었으나 Qt는 C++ 기반이기 때문에 상호간의 통신에 문제점이 발생한다. 따라서 Qt의 라이브러리를 최소한으로 사용하고 대부분의 라이브러리를 표준 C 라이브러리를 사용하도록 함으로서 문제를 해결하였다. 특히 Qt에서는 사용자 입력 이벤트 처리를 하기 위한 함수들(signal(), connect())이 다른 기능의 함수들로 재정의 되어 있어 시그널 처리가 용이하지 않기 때문에 타이머 함수를 이용하여 0.5초마다 사용자의 입력을 검사하는 방식으로 구현하였다. 또한 애플리케이션에 사용된 라이브러리를 타깃 디스크에 복사하여 클라이언트에 이식하였다.

4.2.3 영화 재생기

영화 재생기는 Open Source Project로 개발 중인 mplayer 0.18 버전의 소스를 이용하였다[10]. mplayer에서 시그널 핸들러를 등록하여 고속 재생과 역재생 시에 오디오를 재생하지 않도록 수정하였다. mplayer에서 제공되지 않는 고속 재생과 역재생을 구현하기 위해 리모트 콘트롤러를 통한 VCR 기능 입력을 시그널로 전환하여 전달하면 VCR 모드에 맞게 데이타를 선별하도록 각각의 VCR 동작 모드에 따른 시그널 핸들러를 등록하였다. mplayer의 기존 HTTP 연결 기능을 수정하여 여러 MMS 노드로부터 데이타를 수신하여 데이타를 재조합 하는 기능을 새로 추가하였다. 따라서 기존의 2단계 버퍼링 기능(HTTP 수신 부분과 영화 상영 부분)을 3단계 버퍼링(재조합 부분의 버퍼 추가)으로 강화하였다.

5. 성능 측정 및 결과 분석

5.1 성능 측정기

성능 측정을 위하여 계측 프로그램(Yardstick Program)을 만들었다[11]. 계측 프로그램은 가상 부하 서버, 가상 부하 클라이언트, 내장형 클라이언트로 이루어져 있다.

가상 부하(Load) 서버는 HS 노드에서 동작하는 성능 측정용 프로그램으로 가상 부하 클라이언트들에게 영화 전송을 MMS에 요청하도록 하는 제어 명령을 전송한다. 가상 클라이언트로부터 오류를 전송 받으면 현재 스트림의 수를 출력하고, 모든 가상 클라이언트에게 스트림 연결을 해제하도록 한다. 가상 부하 클라이언트는 가상 부하 서버의 제어에 따라 새로운 접속 요청을 HS에게 보내고, MMS로부터 비디오 데이타 1.5Mbits를 수신한 시간을 측정하고, 1초에서 그 시간을 제외한 만큼 usleep() 함수를 이용하여 대기하도록 함으로서 영화의 대역폭 요구량에 맞게 비디오 데이타를 소모하여 실제 클라이언트와 동일한 부하를 서버에 준다. 만약 1.5 Mbps를 만족하지 못하는 스트림이 발생하면 가상 부하 서버에게 오류를 전송한다. 가상 부하 클라이언트는 리눅스 PC를 이용해 구현되었다. PC 하나가 30개의 클라이언트만큼의 부하를 서버에 준다. 총 14대의 PC를 사용하여 420개의 가상 클라이언트를 시험하였다. 내장형 클라이언트는 VOD 서버로부터 비디오 데이타를 받아 영화를 재생하여 VOD 서버의 정상 동작 상태를 감시한다. 성능 측정기를 이용한 실험의 네트워크 구성은 그림 5와 같다.

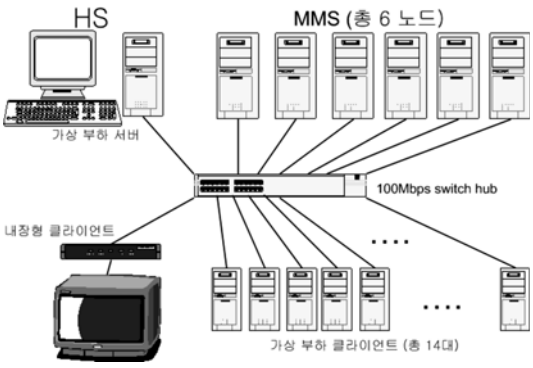

그림 5 성능 측정기의 Network topology

5.2 서버 성능 측정 결과

5.2.1 영화 데이타

사용된 영화 데이타는 표 3과 같다. 가상 부하 서버는 사용자들의 영화 요청이 λ=0.25의 포아송 분포를 따른다고 가정하고 요청을 발생시켰다[3,12]. 고속 재생과 역방향 재생은 I 프레임 만을 클라이언트에게 전송하므로, I 프레임의 크기와 초당 프레임 수를 고려하여 계산한 결과 약 5Mbps의 대역폭을 필요로 함을 알 수 있었다. 성능 측정은 각 영화의 대역폭 요구량을 만족시키는 최대의 클라이언트 수를 측정한다. 모든 측정값은 5회 측정한 값의 평균이다.

각 영화의 GOP는 I-B-B-P-B-B-P-B-B-P-B-B의 순서로 총 12개의 프레임을 포함한다. 고속 재생에는 I 프레임만을 전송하므로 12배속으로 상영하는 것과 동일하다.

5.2.2 일반 재생만 서비스되는 경우

그림 6은 영화를 분산 저장하는 노드의 수에 따른 성능의 변화를 나타낸다. MMS 노드의 수가 4일 때까지는 성능이 향상되지만, 그 이상으로 노드의 수가 많아져도 성능 향상의 정도가 떨어지는 것을 볼 수 있다. 성능 측정 당시 각 노드의 CPU 사용량을 측정해 본 결과 최대 10%를 넘지 않았다. 따라서 네트워크의 대역폭, 각 노드의 디스크 대역폭, 메모리 사용량을 그 원인으로 생각해 볼 수 있다.

네트워크 대역폭에 따른 문제점을 분석하기 위해, 이론상 100Mbps의 네트워크 대역폭을 성능 측정 영화 데이타의 전송 대역폭 요구량 1.5Mbps로 나누어 보면 약 66개의 클라이언트를 지원할 수 있다. 따라서 6개 노드에서 약 396개의 클라이언트를 지원 할 수 있어야 하지만 영화 2의 경우 230개의 클라이언트를 지원하는 것으

표 3 사용 영화 정보

구 분	영화 1	영화 2	비고
영화 제목	John Q	Ice Age	
크 기	352 × 288	352 × 288	
초당 프레임 수	25	25	
대역폭 요구량	179.7 Kbytes/s	179.7 Kbytes/s	
프레임 시간	0.04 sec	0.04 sec	
영화 상영 시간	110 분	85 분	
GOP 크기 (최대-평균-최소)	252.9-124.1-7.4	281.7-120.8-20.8	KB
I-frame 크기 (최대-평균-최소)	43.6-25.8-6.9	50.2-22.8-6.6	KB
Header 크기	4910 Bytes	2558 Bytes	

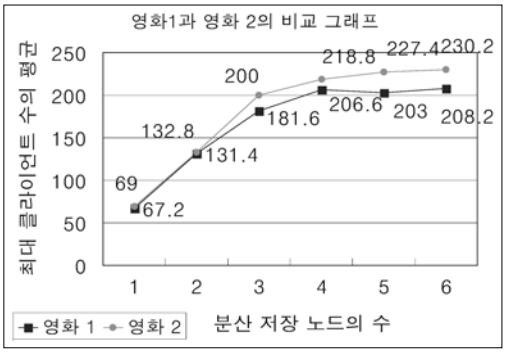

그림 6 분산 저장 노드의 수에 따른 성능 측정 결과

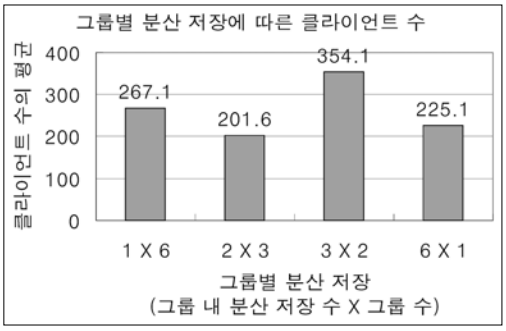

그림 7 그룹별 분산 저장에 따른 성능 측정 결과

로 측정되었다.

그림 7은 그룹별 분산 저장에 따른 클라이언트의 수를 보여 주고 있다. 3개 노드에 하나의 영화를 분산 저장하고 총 두 개의 그룹으로 MMS를 구성하였을 때 354.1개로 가장 높은 성능을 나타내었다. 이는 이론상 수치인 396개에 근접하는 수치로 그림 6에서 분산 저장 노드의 수가 6개일 경우의 230개보다 월등히 높은 성능을 가진다. 따라서 그림 6에서 분산 저장 노드의 수가 4개 이상일 경우의 성능 확장성 감소의 원인이 네트워크라고 하기 어렵다.

각 노드의 디스크 전송 속도는 최대 100MB/s이고, 지속 전송 속도는 22~41MB/s이다[13]. 이는 네트워크의 대역폭을 능가하는 속도이므로 디스크의 대역폭은 성능 확장성 감소 원인이 아니라고 할 수 있다.

모든 상황을 종합해 볼 때 성능 확장성 감소의 원인은 각 노드의 메모리라고 생각해 볼 수 있다. MMS 노드에서는 스트림 하나 당 최소 두 개의 버퍼를 가진다. 버퍼의 크기는 분산 저장 단위인 GOP의 크기와 같다. 따라서 영화 2의 GOP 크기를 버퍼의 크기라고 생각하면 스트림 하나 당 평균 240KB, 최대 563.4KB의 메모리를 필요로 하게 된다. 실제로 성능 측정 당시 메모리 사용량을 측정해 본 결과 스트림 하나 당 1300~1500KB정도의 메모리를 사용하고 있었다. 따라서 MMS의 노드의 메모리가 256MB임을 감안한다면 187~201개 이상의 스트림을 지원하는 경우 메모리가 부족하게 되어 메모리 스왑(swap)이 일어나고, 이는 결국 빈번한 페이지 폴트(page fault)를 발생하는 쓰레싱 상태로 빠지므로 시스템의 확장성 있는 성능을 저해하는 요인이 되고 있다. 그림 6에서 200개의 클라이언트를 넘어서면서부터 성능 향상의 정도가 확연하게 떨어지는 이유는 각 노드의 메모리 때문임을 확인 할 수 있었다.

5.2.3 일반 재생 및 고속 재생이 동시에 서비스되는 경우

고속 재생 비율에 따른 성능 측정 결과는 표 4와 같다. 좌측의 %로 나타낸 수치는 전체 클라이언트 중 고속 재생을 요청한 클라이언트의 비율을 나타낸다. 고속으로 표현된 것은 일반 재생과 고속 재생을 모두 포함한 수치이다. 일반 재생만으로 성능 측정을 했을 때 보다 고속 재생이 일정 비율로 포함되는 경우 보다 좋은 성능을 나타내고 있다. 이런 결과는 전송하는 단위에 따라 MMS 서버 노드에서 사용하는 버퍼의 크기가 결정되기 때문이다. 일반 재생의 경우 전송 단위가 GOP이고 고속 재생의 경우 단위가 I 프레임이어서 상대적으로 고속 재생 일 때가 메모리를 적게 소모하기 때문이다. 표 4에서 영화 2의 평균 GOP크기는 120KB이지만 평균 I 프레임의 크기는 22.8KB로 약 1/6작다. 일반 재생일 경우 스트림 하나 당 평균 240KB의 버퍼가 필요 하지만 고속 재생일 경우 스트림 하나 당 평균 45KB의 버퍼가 필요하다. 그러므로 스트림 하나 당 사용하는 메모리의 크기가 줄어들어 보다 많은 스트림 개수를 제공하게 되는 것으로 관측되었다.

표 4 고속 재생 비율에 다른 성능 측정 결과

	노드 6개 × 1개 그룹		노드 3개 × 2개 그룹	
	고속	일반	고속	일반
10%	239	231.8	378.2	354.1
20%	275.4		389	
30%	279.2		319	

5.3 결과 분석

앞의 결과에서 관찰되는 것처럼 분산 저장 노드의 수와 일반 재생 및 고속 재생의 비율에 따른 각각의 성능을 측정하였다. 분산 저장 노드의 수에 따라 점차 성능이 향상되는 것을 관찰할 수 있었으나, 노드가 4개 이상이 되어 클라이언트의 수가 200개가 넘을 경우 성능 확장성이 현저하게 줄어드는 것을 볼 수 있었다. 또한 일반 재생만으로 서비스하는 것 보다 고속 재생이 일정 비율 포함되면 보다 많은 클라이언트를 서비스 할 수

있음을 측정하였다. 고속 재생의 경우 I 프레임만을 클라이언트에 전송하기 때문에 최대 5Mbps의 대역폭이 필요하다. 그러나 일반 재생 영상이 아닌 영상의 검색에 대한 서비스이므로 대역폭을 충족시키지 못하여도 서비스의 의미가 있기 때문에 네트워크에 의해 지연되어도 충분히 서비스가 가능하였다. 따라서 상대적으로 메모리를 적게 사용하는 고속 재생의 비율이 높아짐에 따라 전체 스트림의 개수도 증가된 것으로 분석되었다.

5.4 클라이언트의 성능 측정

클라이언트에서 VODCA 서버에 접속하여 영화를 상영할 경우 메모리는 적게 소모하지만 CPU 자원의 대부분 사용하는 것으로 측정되었다. 클라이언트에서 일반 재생에서 고속 재생이나 역재생으로 변환할 경우 1초 정도의 지연 시간이 발생되었다. 고속 재생에서 역재생으로 전환할 경우에는 최대 3초의 지연 시간이 발생하였다. 이런 결과는 사용자 명령 전달 지연과 버퍼링 문제로 분석되었다.

사용자 명령 전달 지연 문제는 GUI 모듈에서 사용자 입력을 검사하는 속도가 0.5초 간격이기 때문에 최대 0.5초의 지연이 발생되며, 클라이언트에서 HS를 거쳐 각 MMS 노드에 전달되는 네트워크 지연이 원인이었다. 버퍼링 문제는 각 MMS 노드의 버퍼를 비우고 다시 채우는 시간에 따른 지연과 클라이언트의 버퍼에 남은 비디오 데이타가 재생되면서 그 시간만큼 지연이 발생되는 것으로 분석된다. 사용자 명령 전달 지연 문제는 사용자 입력 검사 속도를 빠르게 함으로서 개선할 수 있을 것이다. 버퍼링의 문제는 각 MMS 노드에서는 버퍼를 채우지 않고 바로 전송하는 기능을 추가하며, 또한 하나의 스트림을 지원하는 버퍼의 개수를 증가시켜 버퍼 선 인출 및 사용 버퍼 내용의 재사용을 통하여 문제의 지연 시간을 단축시킬 수 있을 것이다. 클라이언트 버퍼의 경우 버퍼 내용을 비우는 기능을 개선하여야 할 것으로 보인다.

6. 동적 진입 제어

6.1 가용 자원 기반의 동적 진입 제어

성능 측정 결과 실제로 구현된 클러스터 VOD 서버에서 성능 제한의 요인이 메모리에 있음을 알 수 있었다. 따라서 클러스터 비디오 서버의 진입 제어의 기준을 각 노드의 메모리 크기에 따라 결정해야 한다. 시험에 사용된 영화 데이타 하나의 경우 일반 재생 클라이언트 수가 200을 넘지 않도록 진입을 제어하면 보다 안정적인 QoS를 제공 할 수 있다. 고속 재생이 포함되는 경우에는 고속 재생의 비율이 20% 이하로 유지하면서 각 노드 당 일반 재생 클라이언트 수를 메모리 크기와 GOP 크기를 주요 매개 변수로 하여 진입 제어의 기준을 정하여 QoS를 만족시킬 수 있는 서비스를 제공 할 수 있음을 알 수 있었다.

VODCA에서의 진입 제어는 각 노드의 메모리에서 현재 사용하고 있는 메모리의 사용량을 뺀 값에 새로 요청한 영화에서 사용할 메모리의 량을 비교하여 결정할 수 있다. 메모리의 사용량과 네트워크의 사용량을 이용하여 진입 제어를 위한 기준을 세웠다. 표 5는 진입 제어 기준을 위해 사용된 기호 및 의미를 나타내고 있다.

HS에서 영화를 등록할 때 각 MMS 노드에 영화를 분산 저장하기 위해 생성하는 헤더 정보를 모아 HS의 로컬 하드디스크에 저장한다. 저장된 헤더 정보들은 영화 및 노드 별로 구분되며, 시퀀스 번호와 GOP 크기, I 프레임 크기 정보를 가지고 있다. 본 연구의 진입 제어 기준은 HS에 저장된 헤더 정보를 이용하여 영화 등록 단계에서 계산하여 저장한다. 계산된 정보는 메모리의 $m1_{ij}$(일반 재생 시 메모리 사용량), $m2_{ij}$(고속 재생 시 메모리 사용량), $m3_{ij}$(역방향 재생 시 메모리 사용량), $n1_{ij}$(일반 재생 시 네트워크 사용량), $n2_{ij}$(고속 재생 시 네트워크 사용량), $n3_{ij}$(역방향 재생 시 네트워크 사용량)의 6가지 배열로 저장된다. 1시간 길이의 영화의 경우 각각 3600개의 정보를 저장하는 것이다. 임의의 영화 A를 사용자가 요청하면, HS에서는 영화 A에 대해 미리 계산된 정보들을 읽어 메모리에 저장한다. 예를 들어, MMS 노드 1의 영화 A의 일반 재생 시의 메모리 사용량은 $m1_{1A}$가 되고, 현재 일반 재생 중이라면 $m_{1A} = m1_{1A}$가 된다. 영화 A가 요청된 시점에서 각 MMS 노드의 가용 메모리 량과 가용 네트워크 대역폭 량을 계산하여 이 값이 모두 음수가 되지 않는다면 진입을 허용한다.

표 5 논문에서 사용된 기호 및 의미

M_i	MMS 노드 i의 메모리 량
m_{ij}	MMS 노드 i에서의 영화 j의 메모리 사용량
N_i	MMS 노드 i의 네트워크 대역폭
n_{ij}	MMS 노드 I에서의 영화 j의 네트워크 대역폭 사용량
S_i	MMS 노드 i의 시스템 메모리 사용량
AM_i	MMS 노드 i의 가용 메모리 량
AN_i	MMS 노드 i의 가용 네트워크 대역폭
CM_i	새로 요청된 영화의 MMS 노드 i에서의 메모리 사용량
CN_i	새로 요청된 영화의 MMS 노드 i에서의 네트워크 사용량
※모든 값은 1초 단위의 값을 가지는 배열이다.	

$$AM_i = M_i - S_i - \sum m_{ij} - CM_i \quad (1)$$

(단, j = MMS 노드 i에서 상영중인 각 영화)

$$AN_i = N_i - \sum n_{ij} - CN_i \quad (2)$$

(단, j = MMS 노드 i에서 상영중인 각 영화)

HS에서 식 (1)과 식 (2)를 통해서 모든 MMS 노드

의 AM_i과 AN_i를 1초 단위로 계산 된 값의 배열로 저장한다. 식 (1)과 식 (2)를 통하여 계산 된 값들의 배열의 길이는 현재 가장 늦게 종료되는 영화의 길이와 같다. 계산은 새로운 영화 요청이 진입 할 때 수행한다. M_i와 N_i는 실제 메모리 량과 네트워크 대역폭이므로 상수의 값을 가지며, S_i는 HS의 자원 감시 및 관리 모듈로부터 쉽게 얻을 수 있다.

VODCA HS의 경우 CPU와 메모리를 거의 사용하지 않으므로 이러한 진입 제어 계산을 위한 자원이 충분하다. 진입 제어를 위한 식 (1)과 식 (2)의 $M_i - \sum m_{ij}$와 $N_i - \sum n_{ij}$는 직전 영화 요청의 진입 때의 계산된 값이기 때문에 새로 계산 될 필요가 없기 때문에 실제 진입 제어 시의 계산은 CM_i와 CN_i의 값을 감산하는 것으로 충분하다. 따라서 진입 제어를 위한 계산의 오버헤드가 HS의 성능에 큰 영향을 주지 않는다. 영화의 상영 시간을 평균 2시간이라고 했을 때, 각 배열은 총 7200개의 정보를 가지며, 4바이트의 정수형 변수를 사용한다면, 하나의 영화에 대한 계산된 정보는 약 170Kbytes (7200 × 4 × 6bytes)의 크기를 필요로 한다. 현재 서비스 중인 영화가 100개일 경우 약 17Mbytes의 메모리를 요구하게 된다. 이는 현재 HS의 메모리 사용량을 볼 때 충분히 서비스 할 수 있는 크기이다. 사용자의 요청은 Zipf의 법칙을 따라 특정 영화에 집중되는 특성을 가지고 있다. 따라서 동일한 영화에 대한 정보들은 단 한번만 메모리에 저장하면 되기 때문에, HS의 메모리와 디스크 읽기 작업을 줄일 수 있다.

VODCA의 성능 측정에서 저장 장치의 대역폭이 네트워크의 대역폭보다 상당히 크다는 것을 보였다. 따라서 진입 제어 계산에는 이를 제외하였다.

6.2 동적 진입 제어의 구현 및 결과

앞에서 제안한 동적 진입 제어 방법을 구현하여 모듈의 형태로 VODCA 시스템에 장착하였다. 동적 진입 제어 모듈의 알고리즘은 pseudo 코드의 형태로 표 6에 나타난 바와 같다. 동적 진입 제어 모듈을 장착한 VODCA 시스템을 성능 측정에 사용한 성능 측정기를 사용하여 측정하였다. 측정 결과 목표하는 QoS를 벗어나면 새로운 스트리밍 요구 발생시 진입을 거부하는 것을 확인 할 수 있었다. 따라서 가상 부하 서버와 가상 부하 클라이언트는 종료되지 않고 일정한 스트리밍 수를 유지함을 확인하였다.

7. 관련 연구

다양한 사용자의 요구와 영화 데이타의 연속적인 특징을 고려하여 보다 많은 사용자에게 안정적인 서비스를 제공하기 위한 VOD 서버에 대한 많은 연구들이 있었다[3-5,7,12,14,15,26,27]. 또한 연구용 또는 상업용의 목적으로 설계 구현된 VOD 시스템에 관한 연구들이 있었다[3,12,17-19]. 이러한 VOD 서비스에는 영화 데이타들의 연속적인 특징과 압축 기술의 특징으로 인해 일반 VCR과는 달리 고속 재생과 역방향 재생에 몇 가지 문제점들을 가지고 있다.

최근 클러스터링 비디오 서버에서의 안정적인 QoS를 지원하기 위한 많은 연구가 있었다. 비디오 데이타의 특성을 고려하여 실시간 스케줄링 기법인 EDF를 이용하는 연구가 있다[20]. 비디오 데이타인 MPEG의 특성을 고려하여 각 프레임간에 우선 순위를 부여하고 서비스 상황에 따라 높은 우선 순위의 프레임을 우선 전송하는 방법에 관한 연구가 있었다[2]. 이러한 연구들은 비디오 데이타의 실시간 적 특징과 MPEG의 특성을 고려한 것으로서 오프라인 상에서 모든 영화에 대한 정보를 알아내야 하고, 서비스 도중에도 많은 오버헤드가 있다는 단점이 있다. 또한, 비디오 서버의 메모리를 효율적으로 관리하여 QoS를 제공하여 주는 연구가 있었다[21, 22]. 이러한 연구들은 오프라인에서의 계산이 필요하며, 서비스 도중에 계산되어야 하는 오버헤드로 인하여 서버의 성능이 오히려 떨어질 수 있는 단점이 있다. 따라서 QoS를 지원하기 위한 내부의 오버헤드를 최소화하면서 안정적인 QoS를 제공하기 위한 연구가 필요하다.

실질적으로 VOD 시스템을 구성하기 위해서는 서버, 네트워크, 단말, 컨텐츠 관련 전 분야에 대한 통합적 연구가 필요하다. 과거의 VOD 시스템에 대한 연구들은 상기의 분야들 중 일부를 구현하거나, 또는 모의 시험에 의존하여 성능 결과를 제시한 것이 대부분이다. 본 논문에서와 같이 클러스터형 VOD 서버와 셋탑박스에 내장형을 목적으로 하는 클라이언트 모델 모두를 구현하여 성능 측정 및 문제점, 대안을 제시한 연구가 미진하다. 일부 상업용 VOD 시스템이 나와 있기는 하지만 대부분 성능 및 상세 구성 내역을 공개하지 않고 있으며, 성능 측정 결과에 대한 자료를 얻을 수 없을 뿐만 아니라 시스템을 구성한 부품들의 구성 내역도 공개하지 않으므로 절대적 기준으로 본 논문의 결과물과 성능 비교를 할 수가 없다.

8. 결론 및 향후 연구 계획

본 논문에서는 리눅스 기반의 클러스터 VOD 서버 시스템과 내장형 VOD 클라이언트 시스템을 설계, 구현하였다. 구현된 시스템의 성능을 측정하여 결과를 분석하고 이를 바탕으로 진입 제어의 기준을 제안하였다.

클러스터형 VOD 서버 시스템인 VODCA는 사용자의 요청을 처리하고, MMS 노드들을 관리 제어하며, 영화 데이타를 분석, 분산 저장, 관리하는 HS 노드와 HS 노

표 6 동적 진입 제어 pseudo 코드

문제 : 새로운 스트리밍 요청을 수락할 것인가?
입력값(파라미터) : 새로 요청된 영화 번호
출력 : 1 또는 0 (수락하면 1, 거부하면 0)

int DAC (int movie_no)
{
for(i = 1; i <= MMS 노드의 수; i++) {
CM[i] = 새로 요청된 영화의 각 노드에 대한 메모리 요구량;
CN[i] = 새로 요청된 영화의 각 노드에 대한 네트워크 요구량;
if (현재 서비스 스트림 수 == 0) {
for(i = 1; i <= MMS 노드의 수; i++) {
AM[i] = M[i] - S[i] - CM[i];
AN[i] = N[i] - CN[i];
 	
AM[i], AN[i]을 데이터베이스에 저장;
}
return 1;
}
 	
AM[i], AN[i]을 데이터베이스에서 읽어옴;
 	
for(i = 1; i <= MMS 노드의 수; i++) {
AM[i] = AM[i] - CM[i];
AN[i] = AN[i] - CN[i];
if (AM[i] < 0 || AN[i] <0) return 0;
}
AM[i], AN[i]을 데이터베이스에 저장;
return 1;
}

변수들 : CM[] = 새로 요청된 영화의 각 노드에 대한 메모리 요구량 CN[] = 새로 요청된 영화의 각 노드에 대한 네트워크 요구량 AM[] = 각 MMS 노드의 현재 가용 메모리 크기 AN[] = 각 MMS 노드의 현재 가용 네트워크 대역폭 M[] = 각 MMS 노드의 메모리 크기 N[] = 각 MMS 노드의 네트워크 대역폭 S[] = 각 노드의 시스템 메모리 사용량

드의 제어에 따라 사용자에게 영화 데이타를 전송하여 주는 MMS 노드로 구성하였다. MPEG-2 데이타의 병렬 처리를 위해 MPEG-2 미디어를 구성하는 GOP 계층을 이용하여, GOP 단위로 각 노드에 분산 저장하였고, 이를 통해 노드의 균등한 부하 분산을 구현하였다. 또한, MPEG-2의 픽처 계층의 I 프레임을 이용하여 VODCA에 VCR 기능인 고속 재생과 역방향 재생 기능을 구현하였다. VODCA의 성능 측정을 위하여 계측 프로그램(Yardstick Program)을 구현하였다. 구현된 프로그램을 이용하여 분산 저장 노드의 수와 VCR 기능 요청의 비율의 변화에 따른 VODCA의 성능을 측정하였다. 측정된 결과를 통해 VODCA는 최대 354개의 클라이언트를 지원할 수 있음을 보았다. 측정된 결과를 분석하여 성능 향상의 제한 요인이 메모리에 있다는 것을 밝혀 내었다. 이를 근거로 VODCA의 각 MMS 노드의 가용 메모리와 가용 네트워크 대역폭을 기준으로 한 VOD 서버의 진입 제어 방법을 제안하였다.

VOD 클라이언트 시스템은 사용자 친화도가 높고, 영

화를 시청하기 좋은 플랫폼으로 제공하기 위하여 TV에 연결하고 사용자의 입력을 리모트 콘트롤러로 받는 셋탑 박스의 형태로 구현하였다. 구현된 클라이언트 시스템을 사용자 응답 시간을 기준으로 성능을 측정, 분석하였다.

향후에는 MPEG-4의 병렬화 및 VCR 기능에 관한 연구와 VOD 성능 향상에 관한 연구와 VOD 시스템에서의 failure recovery에 대한 연구를 진행할 것이다.

참 고 문 헌

[1] Dinkar Sitaram, Asit Dan, "Multimedia Servers: Applications, Environments, and Design," Morgan Kaufmann Publishers, 2000.

[2] Joseph Kee-Tin Ng, Calvin Kin-Cheung Hui, Wai Wong, "A Multi-server Design for a Distributed MPEG Video System with Streaming Support and QoS Control", IEEE RTCSA, 2000.

[3] Calvin K. Hui, Joseph K. Ng, Wai Wong, Karl R.P.H. Leung, "The Implementation of a Multi-server Distributed MPEG Video System," IEEE RTAS, 2001.

[4] Jack Y.B. Lee, "Parallel Video Servers: A tutorial," IEEE Multimedia, pp. 20-28, 1998.

[5] 최숙영, 유관종, "병렬 VOD 서버의 확장을 위한 스트라이핑 기법", 정보과학회논문지 : 정보통신 제 28 권 제 3 호, 2001.

[6] 이호석, 김준기, "알기 쉬운 MPEG-2 소스코드 해설", 홍릉과학출판사, 2001.

[7] 배인한, 천성광, "분산 주문형 비디오 시스템을 위한 영화 할당 알고리즘의 설계 및 평가", 정보과학회논문지(A) 제 25 권 제 6 호, 1998.

[8] Gerard Beekmans, "Linux From Scratch Version 3.3", (http://www.linuxfromscratch.org)

[9] "Qt/Embedded Whitepaper", trolltech, (http://trolltech.com/products/embedded/)

[10] mplayer 개발 사이트 (http://mplayerhq.hu/)

[11] Brian K. Schmidt, Monica S. Lam, J. Duane Northcutt, "The interactive performance of SLIM: a stateless, thin-client architecture," ACM SOSP '99, pp. 31-47, 1999.

[12] D, James Gemmell, Harrick M. Vin, Dilip D. Kandlur, P. Venkat Rangan, "Multimedia Storage Servers: A Tutorial and Survey," IEEE computer, 1995.

[13] Segate Barracuda ATA Ⅳ 데이타 시트. (http://www.seagate-asia.com/seagatefiles/korea/pdf/Barracuda_ATA4-KR(Datasheet).pdf)

[14] Florin Lahan, Irek Defee, Marius Vlad, Aurelian Pop, Prakash Sastry, "Integrated system for multimedia delivery over broadband ip networks," IEEE Transactios on Consumer Electronics, Vol. 48, No.3, pp.564-565, 2002.

[15] Sooyong Kang, Heon Y. Yeom, "Modeling the Caching Effect in Continuous Media Servers," Multimedia Tools and Applications, 2001.

[16] Prashant J. Shenoy, Pawan Goyal, Harrick M. Vin, "Issue in multimedia Server Design," ACM Computing Surveys, Vol.27, No 4, pp. 636-639, 1996.

[17] Craig S. Freedman, David J. DeWitt, "The SPIFFI Scalable Video-on-Demand System," ACM SIGMOD, 1995.

[18] SuperNAVA, http://archive.dstc.edu.au/SuperNOVA/

[19] VODKA, http://vodka.lfcia.org/

[20] Wanghong Yuan, Klara Nahrstedy, Kihun Jim, "R-EDF: A Reservation-Based EDF Scheduling Algorithm for Multiple Multimedia Task Classes," IEEE RTAS, 2001.

[21] 김순철, 조유근, "가변 비트율을 이용하는 주문형 비디오 서버에서의 효율적인 버퍼 관리 기법", 정보과학회논문지(A) 제25권 제2호, pp. 177-186, 1998.

[22] 원유집, "주문형 비디오 서버의 버퍼 최소화를 위한 가변적 서비스 모드 변환", 정보과학회논문지:시스템 및 이론 제28권 제5호, pp. 213-227, 2001.

[23] Tom Fawcett, "The Linux Bootdisk HOWTO," (http://www.tldp.org)

서 동 만

2002년 강원대학교 컴퓨터공학과 졸업(학사). 2004년 강원대학교 컴퓨터 정보통신공학과 졸업(석사). 2004년~현재 강원대학교 컴퓨터 정보통신공학과 박사과정 재학중. 관심분야는 병렬처리, 멀티미디어 시스템, 운영체제

방 철 석

2002년 8월 강원대 정보통신공학과 학사 2004년 8월 강원대 컴퓨터정보통신공학과 석사. 2004년 7월~현재 (주)씨인씨 연구원 재직. 관심분야는 내장형 시스템, 씬 클라이언트

이 좌 형

2003년 2월 강원대학교 정보통신공학과 학사. 2004년 3월~현재 현재 강원대학교 컴퓨터정보통신공학과 석사과정. 관심분야는 운영체제, 병렬처리, 파일시스템, 멀티미디어 시스템

김 병 길

2003년 2월 강원대 정보통신공학과 학사 2003년 3월~현재 강원대 컴퓨터정보통신공학과 석사과정. 관심분야는 유비쿼터스 컴퓨팅, 내장형 시스템

정 인 범

1985년 고려대학교 전자공학과 졸업(학사). 1985년~1995년 (주) 삼성전자 컴퓨터 시스템사업부 선임 연구원. 1992년~1994년 한국과학기술원 정보및통신공학과 졸업(컴퓨터공학 석사). 1995년~2000년 8월 한국과학기술원 전산학과 졸업(박사). 2001년~현재 강원대학교 전기전자정보통신공학부 컴퓨터전공 교수. 관심분야는 다중처리기 구조, 운영체제