기존 시스템 환경에서의 병렬 미디어 서버의 설계 및 구현

김경훈(金敬勳), 류재상(柳再相), 김서균(金瑞均), 남지승(南址昇)
전남대학교 컴퓨터공학과, RRC
전화 : (062) 530-0422 / 팩스 : (062) 530-1809

Design and Implementation of parallel Media server in current system environment

KyungHun Kim, JaiSang Ryu, SeoGyun Kim,JieSeng Nam
Dept. Computer Engineering of Chonnam National University, RRC
pluit@mdclab.chonnam.ac.kr

Abstract

As network resources have become faster and demands for multimedia service through network have increased , the demand for Media server system has increased . These kinds of media server solve their bottle neck problem of internal storage device by using parallel system which takes advantage of fast network resource .

Many vendors have suggested each of their media server system to solve these problem radically , but most of them require major modification of infra component and additional drawback has added . For example , storage mechanism for specific media requires new file system which is totally different from traditional one , and algorithm for enhancing performance may not suit for traditional operating system environment . In this paper , we designed a parallel media server based on web interface of traditional system and implemented a program for media server . Implemented server system performs parallel processing through web interface without any modification of traditional system , and controls which is related to merging load by distributed data is charged only to client and control server and consequently load of storage server can be minimized .

And also , data transfer protocol for streaming media includes Retransfer algorithm and client Admission control policy relevant to performance of whole system .

1. 시스템의 구조

기존의 병렬 미디어 서버들과 같이 설계된 서버는 사용자에게 서비스할 데이터 테이블과 서비스 상태를 저장하고, 인터페이스를 위한 웹 서버가 설치된 Front-End와 실제 데이터가 저장된 저장 서버들로 구성되고 이들은 기본적으로 기존의 운영체제(LINUX)환경 위에 구현 되었으며 제어 서버인 Front-End에는 미디어 서비스를 위한 각종 클라이언트 서비스 도구들이 컴포넌트로 저장되어 있으며 웹 환경에 통합된 프로그램의 형태로 관리자와 서비스 이용자들에게 전송되어 진다.

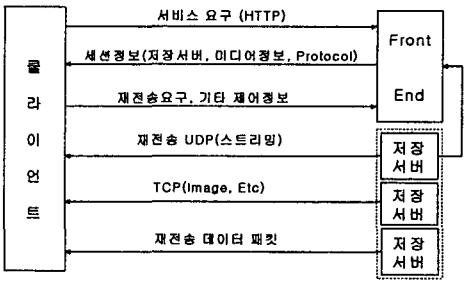

그림 1-1. 미디어 서버 통신 구조

클라이언트에는 분산 환경의 데이터를 재 조합하거나 스트리밍 하기 위한 부하가 추가로 부가되며 클라이언트의 재생 프로그램과 미디어 프로그램들은 네트워크 상태와 저장 서버들의 상태를 Front-End 서버로부터 전송 받아 감시 할 수 있다. 새로운 미디어의 저장은 서버 관리자의 클라이언트가 행하게 되는데 새로운 미디어 데이터의 분산저장, 미디어 정보 저장 서버의 상태 조정 등의 모든 정보는 Front-End에 저장이 되며 Front-End는 분산 저장 정보를 가지고 있는 유일한 시스템이 되고 이는 저장 서버들에 병렬처리를 위한 특별한 메커니즘이 적용될 필요가 없음을 뜻한다.

이 같은 구조는 몇 가지 면에서 장점을 갖는데 첫째로 실제 미디어 서버의 가용성에 많은 영향을 미치는 저장 서버 노드들은 단지 데이터의 검색과 전송만을 담당하여 기존 시스템의 변경 없이 사용자 증가에 따라 새로운 노드를 쉽게 추가 할 수 있으며 서비스 대상 미

디어의 특성에 따라 관리자는 성능에 관계되는 미디어 데이터를 그 특성에 따라 분류하여 저장 할 수 있다.

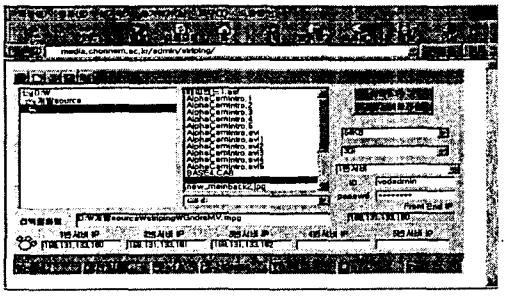

그림 1-2. 관리자 분산 저장 유틸리티

그림 2는 서버 관리자가 데이터를 특성에 따라 특정 블록 사이즈로 저장하거나 또한 대용량 동영상 데이터를 특정 저장 노드부터 저장이 가능하게 하는 저장 유틸리티를 나타낸 것이다. 미디어 서버를 운영하게 되면 사용자 측면에서 자주 접근되는 미디어에 대한 배려를 위해 그 배치정책과 또한 다른 성능 이를테면 저장 서버들이 확장되어 다른 용량의 메모리와 다른 네트워크 스피드의 시스템 자원을 갖는 시스템 상황을 고려하여 최적의 서비스를 위한 데이터 저장은 시스템 성능의 최적화를 위해 필요할 수 있다. 이는 응용 수준에서 기존의 RAID와 같은 시스템 하부의 메커니즘을 통해서는 조절하기 어려운 부분 중의 하나였다. 데이터 배치 정책은 디스크와 IO 대역폭 같은 복잡한 메커니즘을 이해하지 않고서도 응용 수준에서 어는 정도의 효과를 이를 수 있을 것임을 우리는 확신하고 있으며 시스템 구현이 완전히 끝난 상태에서 그 같은 결과를 실험해볼 것이다.

2. 미디어 서비스 프로그램 구조

설계된 분산 서버 시스템은 클라이언트에 재조합 메커니즘을 필요로 한다. 분산된 정책에 의해 재 조합 방법 역시 결정되지만 클라이언트에는 실제로 사용자들에 미디어 서비스를 이용하게 할 프로그램과 인터페이스가 필요하고 또한 우리가 주목하고 있는 실시간 대용량 영상 서비스는 스트리밍 메커니즘의 따라 성능에 중요한 변수가 되는 수신 버퍼관리와 재전송 요구 메커니즘이 포함된다. 이와 같은 클라이언트 프로그램 구성 요소들은 관리자 서비스와 마찬가지로 컴포넌트화 된 웹 인터페이스 삽입 요소들이며 사용자들은 단순히 Front-End의 웹 서버에 접속하는 것으로 서비스를 제공 받을 수 있다. 분산 서버들로부터 데이터를 전송 받기 위한 분산 프로토콜 수신 모듈은 미디어를 재생하거나 Viewing하는 응용 프로그램이 데이터를 읽기 위한 버퍼로 데이터를 재 조합 혹은 스트리밍 하게 하는 역할을 한다. 응용 프로그램은 이 같은 메커니즘에 의해 수정되거나 우리의 서버시스템에 맞게 새로 만들어 질 필요가 없다. 동영상 재생을 위한 플레이어는 Microsoft의 Media Player를 그대로 사용할 수 있으며 플레이어가 지원하는 모든 형식의 동영상 데이터를 재생할 수 있다. 그를 위해 우리는 이 응용의 입력 필터 부분을 우리의 프로토콜을 처리하기 위한 모듈로 제작 하였으며 동영상이 아닌 대용량 이미지와 같은 데이터들은 재 조합된 클라이언트에 의하여 기존의 뷰어에 그대로 적용될 수 있다.

분산 도착하는 동영상 데이터의 스트리밍을 위해 새로운 플레이어가 구현될 필요가 있고 이와 같은 것은 기존의 벤더들과 같은 수준에서의 거대한 스트리밍 메커니즘이 필요하다. 우리의 시스템은 기존의 재생기를 그대로 사용하도록 하였으며 그 같은 것을 가능하게 하기 위하여 MS 윈도우 Media의 기본 클라이언트인 Media Player를 사용하도록 하였다. Media Player는 입력 데이터 공급원에 따라 그 입력 Filter를 지정할 수 있는데 기본적으로 로컬 파일 시스템과 Http, 그리고 Ms media의 streaming 프로토콜들이다[1,2]. 이와 같은 Player의 구조가 우리의 분산된 데이터들을 그대로 받아서 처리하게 하기 위해 DirectShow를 이용한 입력 필터의 구현이 필수적이다[2]. 기존 환경은 대부분은 이러한 분산 데이터를 클라이언트 뷰어가 단순이 단일 서버 전송인 것처럼 보이게 하는 것은 전통적으로 서버 시스템의 몫이었지만[6], 이와 같은 서버 구성시의 방법은 기존의 시스템 환경을 고려한 우리의 목적에 부합하지 않는다.

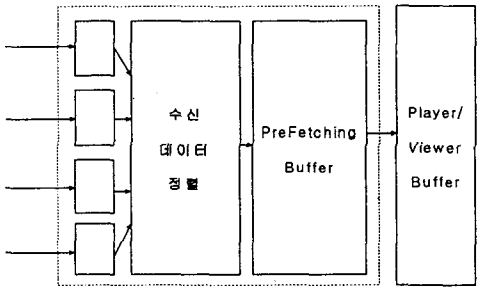

그림 2-1. 수신 클라이언트 프로그램의 재생 구조

클라이언트 뷰어는 그래서 실제로 분산 데이터 수신 프로토콜 머신이 별도로 존재하고 player의 입력 필터는 로컬 파일을 읽는 형식의 필터로서 스트리밍을 가능케 한다. 이와 같은 구조는 시스템이 동영상 되의 Download Viewing이 가능한 데이터에도 적용 될 수 있게 하기 위한 것이다.

동영상 데이터의 수신에는 플레이어의 버퍼링을 위한 공간 이외에 Pre-fetching 영역 메모리 공간이 필요한데 이는 서버에서 UDP로 전송하는 데이터의 재전송을 위한 시간적인 기준이 된다[3]. 이 메모리 영역의 사이즈는 실시간 데이터 전송을 위해 사용되는 손실 확률이 높은 UDP 데이터의 재전송을 위한 손실 데이터를 검사하고 플레이어 버퍼의 재생을 위한 충분한 데이터 수신을 확인하면 손실 데이터를 특정 저장 노드에 재 전송 요구를 송신한다. 시간적인 여유가 없을 때 이 공간의 데이터는 Player 버퍼에 의해 폐기되고 새로운 데이터를 다시 Pre-fetching한다. 이 영역의 크기는 스트리밍의 성능에 많은 영향을 미칠 것으로 예상되며 이는 전송 프로토콜과 망의 상태에 따라 다른 상태로 결정 되어야 한다. 스트리밍은 대부분 실시간 전송 특성을 강조하며 대다수의 사용자들은 망 상태에 의한 지연에서 기존의 영상을 기다리기를 원하지 않는다[3]. 하지만 극심한 네트워크 지연에 의해 많은 장면들이 단지 음성만이 전송된다든지 하는 등의 Rate-Control 방법들을 사용한 기존의 서비스들은 가용 대역폭을 충분히 활용하지 못하는 단점들이 부각되고 있다[2]. 우리는 그 같은 점을 해결하기 위한 방법으로 클라이언트의 데이터 버퍼링과 또 스트리밍의 방법상의 여러 가능성을 고려하고 있다. 이것은 데이터 타입에 관한 정보를 더욱 신중하고 자세히 저장하고 그 같은 데이터 타입에 각기 다른 알고리즘을 각각 적용하는 점을 생각 하고 있는데 구체적인 예로 심지여 데이터 스트리밍에 사용할 프로토콜의 일부는 TCP로 전송될 것이다. 캠퍼스 망과 같은 중간 정도의 지연 특성을 갖는 네트워크 망에서 UDP데이터는 높은 대역폭을 점유하지 못하고 그 손실 특성에 의해서만 많은 데이터가 클라이언트에 의해 버려질 확률이 증가한다. 재전송 UDP 메커니즘이 어느 정도 이 같은 특성을 보완 한다고 해도 궁극적으로 확장 구현될 우리의 서버는 TCP를 선택하여 데이터를 전송하게 될 것이다. 이와 같이 전송 프로토콜들 다중 선택할 뿐 아니라 사용자 QoS확보를 위한 노력도 병행하는데 그 같은 것은 Front-End Server를 통하여 수행된다. VBR 로 인코딩된 데이터는 그 특성에 따라 수용 가능한 스트림의 수도 달라질 것을 고려하여[4], 그 특성에 맞는 사용자 수용에 관한 알고리즘도 추가되었다. 이와 같은 것들의 제어는 Front-End서버의 Load를 가중 시키므로 전체 시스템 향상에 아무런 영향을 주지 않는다 그 예로 Front-End 서버와 저장 서버들은 전체적으로 다른 Load를 취급한다. 또한 저장 서버는 어떠한 알고리즘을 위한 프로세스나 설정이 필요치 않고 모든 것을 Front-End와 클라이언트에 맡김으로써 쉽게 확장될 수 있다. 이것은 서버 설계의 중요한 요소중의 하나로서 단일 저장 서버의 성능을 최대로 하는데 사용되는 RAID 제어기와 같은 옵션들처럼 고정 시킬 필요가 없게 구현한 것은 그와 같은 목적이다. 사용자수가 늘어난 만큼 추가되는 저장 서버에 단순히 전송 프로그램을 설치하는 것으로 시스템은 쉽게 확장된다.

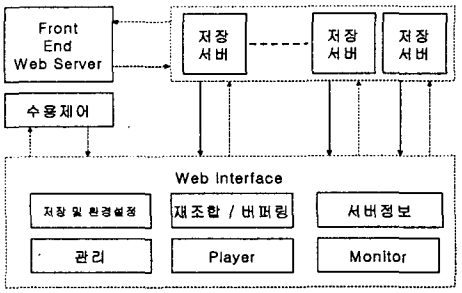

그림 2-2. 서버와 클라이언트의 기능 구조

3. 성능관련요소와 사용자 제어 방법

향후 확장 구현될 서버의 최종 목표는 많은 동시 사용자에게 서비스 할 수 있는 서버 시스템의 구축이다. 많은 시스템 구성요소를 특정 목적에 부합하게 하는 것이 아닌 기존의 서버 자원과 운영체제 상에서도 이와 같은 고가의 시스템의 역할을 대신하게 하는 것이 우리 시스템의 목적이다. 그렇지만 시스템은 높은 가용성을 확보하기 위한 응용 수준의 많은 도구들을 제공하고 알고리즘이 적용될 것이다. 그 같은 것들에는 서비스 미디어의 특성이 반영되고 서버 가용용량에 따른 동적인 사용자 수용 알고리즘도 포함된다. 버퍼와 시스템 자원의 구성은 이 같은 목적에 맞게 최적화 되고 이는 주로 제어 서버인 Front-End와 사용자 클라이언트에 구현되어 저장서버의 확장과 추가적인 부하 부담을 줄여 우리의 기본 목적에 가장 적합한 형태가 될 것이다. 처음 사용자 클라이언트가 서비스 웹에서 얻는 정보는 서비스 받을 데이터의 종류나 서비스 되고 있는 상황 뿐만이 아니라, 데이터의 종류 분산 저장되어 있는 형태 그리고 연결을 요청할 저장 서버의 정보 등이 모두 포함된다. 그것은 기존의 서비스들이 고려하지 않았던 사항들로 단순히 웹 서버가 플레이어를 미디어 서버로 연결 설정을 변경하는 기능 뿐이었던 것을 보완한 것이다. 이 같은 세션 설정 정보를 받은 클라이언트는 그 같은 정보와 본인의 네트워크 상황과 QoS 요구를 모두 반영하여 저장 서버로 요구 Request를 전송하고 저장 서버들은 이와 같은 요구에 따라 데이터를 전송한다. 데이터 전송이전에 저장 서버는 전송 서비스 상황을 Front-End에 보고 하는데 이 정보는 사용자를 수용할 것인지를 결정하는 Front서버의 기준 값이 된다. 초기 클라이언트가 특정 미디어를 요구할 때 서버는 이 값을 통하

여 선택된 미디어가 미디어 자체의 QoS를 보장하여 전송할 수 있는 지를 판단하고 서비스 가부를 클라이언트에 전송한다. 같은 비트율 크기의 재생률을 갖는 데이터는 이와 같은 것의 예측판단이 비교적 정확히 이루어질 수 있지만 VBR미디어 데이터는 클라이언트의 요구당시 저장 서버의 VBR데이터 서비스 상황에 따라 유동적인 기준이 적용된다. 시간당 재생률이 다른 데이터를 고정 비트율 데이터와 같은 기준으로 사용자 수용제어를 하는 것은 시스템의 가용성에 도움이 되지 않기 때문에[4], 사용자 QoS를 준수하는 조건에서 이러한 데이터를 고려한 최적의 수용제어가 필요하고 우리 시스템은 빨리 감기나, 되감기 등의 사용자 요구의 저효율을 감수하고 아직 서비스 하지 않은 데이터 중 최대 값을 이용하는 방법으로 시스템의 효율을 증가 시켰다.

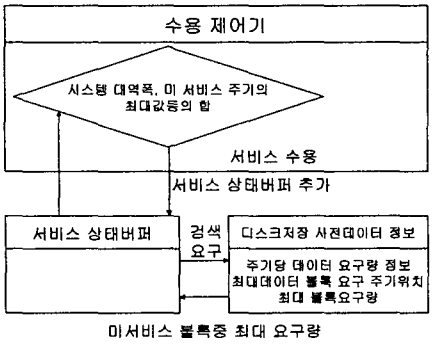

그림 3-1. 사용자 승인 제어기 구조

시스템의 성능은 기본적으로 저장 노드들의 성능의 영향을 받는다[5]. 우리가 사용한 저장 노드들은 모두 같은 성능 예로 같은 디스크와 같은 RAID 컨트롤러 같은 프로세서의 속도와 같은 용량의 메모리를 갖는다. 다만 Front-End는 더 빠른 프로세서를 사용하고 있다.

우리의 최종 목표는 실제 분산환경에서의 분산 데이터의 크기와 성능의 관계 운영체제의 파일 시스템과의 성능 상관관계 그리고 네트워크를 통한 1차적 데이터 분산과 저장 서버에서의 하드웨어에 의한 2차적 분산에 의한 가장 효율적인 분산 정책을 실제 시스템을 구현해 봄으로써 찾는데 있다. 여기에 버퍼의 관리와 좀더 효율적인 전송 방법상의 문제도 고려 대상이다. 시스템의 설계와 일부의 구현은 앞에서도 언급했듯이 이 같은 고려사항을 모두 제어할 수 있는 형태로 구현 하였으며 최종적인 구현 모습은 결국 이와 같은 요소를 반영한 미디어 서버 데이터의 통합한 네트워크 파일 시스템의 구성이 될 것이다. 하지만 이 논문에서는 기존 시스템 환경에서 간단한 프로그램 설치로 이와 같은 형태의 서비스를 구현 하는 것만을 언급하고 성능 분석과 거기에 따른 파일 시스템의 구현의 향후의 과제이다.

4. 결론 및 향후 과제

우리가 구현한 시스템은 기존의 환경에서 여러대의 저장 서버를 고속의 네트워크에 연결하여 동시에 서비스 할 수 있는 사용자 수를 극대화 하기 멀티미디어 데이터 서버이다. 분산 정책은 다양한 옵션들로 선택되어질 수 있고 실제 서비스를 위한 클라이언트 도구들도 구현 하였다. 미디어 서버의 모든 성능 향상 가능성을 고려 하였으며 웹 환경에 통합 적용 될 수 있는 형태로 구현 하였다. 시스템은 각 미디어 데이터에 따라 다른 옵션으로 저장되고 전송되며 사용자는 서비스 요구 시 서버의 능력을 고려한 수용제어도 포함되어 있다. 하지만 향후 이 같은 성능은 실험되어 검증될 필요가 있으며 이러한 환경에서 가장 일반적인 성능변수를 찾아 낼 필요가 있다. 향후 우리의 연구는 구현된 시스템을 확장하고 실험을 통해 이 같은 문제를 찾아 낼 것이며 그 것을 통해 통합된 분산 파일 시스템을 구성하는데 있다.

참고문헌

[1] MicroSoft ,"MicroSoft Media Service SDK ",1998

[2] MicroSoft, "MicroSoft Media Technologies White paper",1998

[3] Leopold. H.,"Classification of Multimedia Types",Computer Networks and ISDN Systems, Vol. 17,pp.324-327,1989

[4] Huang-Jen Chen T.D.C Little, "Storage Allocation Policies for Time-Dependent Multimedia Data", IEEE Trans. On Knowledge and Data Engineering , Vol. 8, No. 5. ,pp. 855-864, 1996

[5] S. Moyer and V. Sunderam, "Parallel I/O as a Parallel Application," Journal of Supercomputer Applications, Vol.9, No.2, 1995.

[6] H. Schulzrinne, A. Rao, R. Lanphier," *Real Time Streaming Protocol (RTSP) ",IETF RFC 2326,* April 1998