PMSS 시스템에서 서버/클라이언트 간 서비스 모델의 성능분석

이 민 홍† · 김 경 훈†† · 남 지 승†††

요 약

본 논문은 단일 주문형 비디오 환경에서의 서버/클라이언트 간 미디어 서비스 모델을 분석하고 이를 병렬 주문형 비디오 환경에 적용함으로써 보다 나은 사용자 QoS(Quality of Service)를 제공하고자 한다. 미디어 서비스 모델로는 클라이언트가 데이터를 요청하고 서버가 전달해주는 Client Pull 모델과 서버 측에서 일방적으로 데이터를 전달해 주는 Server Push 모델 그리고 두 모델을 통합한 IPP(Interleaving Pull & Push) 서비스 모델로 크게 나뉜다. 병렬 주문형 비디오 환경을 위해 병렬형 미디어 스트리밍 서비스, 즉 단일 클라이언트를 위해 다수의 서버들이 동시에 서비스에 참여하는 PMSS(Parallel Media Streaming Service) 시스템을 구축하였으며, 단일/병렬 주문형 비디오 환경에서 네트워크상 전달지연과 클라이언트 버퍼 내 데이터 잔여량 인자를 통하여 각 서비스 모델간 성능을 비교 분석하였다. 실험을 통해 병렬 주문형 비디오 환경에서 IPP 서비스 모델이 가장 적은 전달지연과 안정적인 클라이언트 버퍼를 유지함을 알 수 있었다. 이를 통해 사용자에게 보다 나은 서비스를 제공할 수 있음을 확인하였다.

Performance Analysis of Service Model between server and client on PMSS System

MinHong Lee† · KyungHoon Kim†† · JiSeung Nam†††

ABSTRACT

This paper provides the higher user QoS(Quality of Service) by analyzing media service model between server and client in single VoD(Video on Demand) environment and applying it to parallel VoD environment. Media service model is divided into the Client Pull, Server Push, and IPP(Interleaving Pull & Push) model. A server sends data based on client's request in the Client Pull Model. A server one-sidedly sends data without client's request in the Server Pull model. And the IPP model unites above two models. For a parallel VoD environment, We built the PMSS system which provides the parallel media streaming services that one client is simultaneously served by several servers. In the single and parallel VoD environment, We compare and analyze the performance of service models with respect to network delay and data size in buffer. In this experiment, we found that IPP service model keeps the least network delay and stable client buffer in the parallel VoD environment. This result shows that PMSS can provide the more quality of service.

키워드 : VOD(Video On Demand), QoS(Quality of Service), PMSS(Parallel Media Streaming Service), IPP(Interleaving Pull & Push)

1. 서 론

현재의 인터넷 환경은 다양한 멀티미디어 데이터의 활용이 증가하면서 새로운 도약의 시기를 맞이하고 있다. 이러한 환경의 변화는 서비스 매체의 변화를 이끄는 계기가 되었으며, 텍스트 기반의 단일 매체 서비스에서부터 인터넷을 통한 비디오, 오디오, 텍스트 등과 같은 다양한 매체들을 하나로 종합한 멀티미디어 응용 서비스 형태로 발전하였고 이를 위한 연구가 활발히 진행되고 있다.

이러한 연구를 기반으로 주문형 비디오, 원격 화상 회의, 홈쇼핑 등의 상업용 서비스들이 현재 보편화 되어지고 있으며, 특히 인터넷상에서 스트리밍 기술을 기반으로 한 주문형 비디오 서비스가 증가하고 있다[1-4].

이러한 서비스 증가 추세는 보다 많은 클라이언트 지원을 보장하기 위한 서버성능의 향상과 효율적인 서비스 모델을 도출하게 되었으며, 이를 위해 다수의 서버를 이용한 병렬형 주문형 비디오 시스템과 이에 알맞은 서버/클라이언트간 미디어 서비스 모델의 개발이 필요하게 되었다[5-11].

주문형 비디오 서비스에 있어서 데이터 전송을 위한 서버/클라이언트간의 서비스 모델은 접속한 클라이언트에게 서버가 일방적으로 데이터를 전달해주는 방식 즉 주도권을 서버가 가지는 Server Push 서비스 모델과 클라이언트가

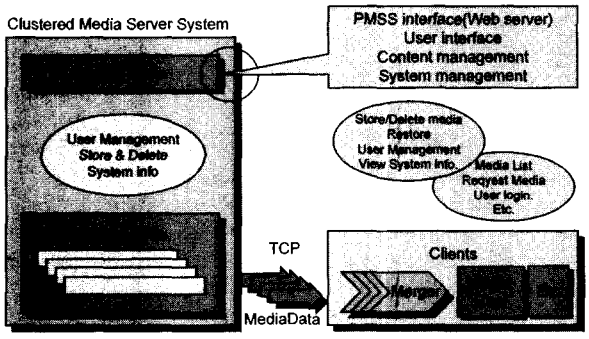
† 준 회 원 : (주)포스트림
†† 준 회 원 : (주)포스트림 대표이사
††† 종신회원 : 전남대학교 컴퓨터공학과 교수
논문접수 : 2003년 5월 6일, 심사완료 : 2004년 6월 11일

일정 크기의 데이터 블록을 요청하고 서버는 요청 받은 데이터 블록만을 전달하여 주는 방식 즉 데이터 흐름 제어의 주도권을 클라이언트가 가지는 Client Pull 서비스 모델로 양분되어 발전되고 있다[12-16].

본 논문에서는 단일 서버 주문형 비디오 환경에서의 서버/클라이언트 간 데이터 서비스 모델을 분석하고 이를 병렬 서버 주문형 비디오 시스템인 PMSS에 적용하여 보다 나은 서비스 모델의 제시와 사용자 QoS의 향상을 제공하고자 한다.

2. PMSS 시스템 구조

주문형 비디오솔루션의 가용성을 높이기 위해서는 시스템 근간을 이루는 하드웨어와 미디어를 전송하고 관리하는 소프트웨어의 두 가지의 구성 요소가 필요하다. 하드웨어는 다수의 미디어 저장을 위한 충분한 용량과 큰 전송 속도를 보장하는 저장 장치, 미디어에 대한 네트워크의 충분한 데이터 전송 능력을 보장하는 입출력 장치가 가장 중요한 요소를 차지하며, 소프트웨어는 미디어 데이터의 특성에 구애 받지 않는 효율적인 데이터 배치 및 미디어 관리를 위한 보다 효율적인 사용자 인터페이스가 주문형 비디오 솔루션의 주요 한 요소로 포함된다[1, 2].



(그림 1) PMSS 시스템 구조

(그림 1)은 PMSS시스템의 전체적인 구조를 나타낸 것이다[8].

구현된 시스템은 기본적으로 Linux 운영체제를 기반으로 한 여러 대의 서버가 동시에 하나의 클라이언트를 위해 다중의 접속 경로를 갖는 형태로 구성된다. 서버 시스템은 하위 저장 서버의 제어흐름을 조정하고 관리하기 위한 컨트롤 노드와 미디어를 저장하고 사용자에게 서비스하는 스트리밍 노드군으로 구성된다.

컨트롤 노드는 미디어를 스트리밍 노드군으로 스트라이핑하고 미디어 컨텐츠를 관리 재배치하는 미디어 관리 모듈, 사용자 접속요청과 서비스 상태 정보를 관리하는 클라이언트 관리 모듈, 컨트롤 서버와 저장 서버의 부하측정을 위한 시스템 정보 모듈로 구성되고 이를 위해 웹 서버와 데이터베이스 서버 기능을 갖춘다.

스트리밍 노드에는 컨트롤 노드의 제어 하에 클라이언트가 요구하는 데이터를 전송하는 프로그램이 설치되며 실제적인 데이터 전송을 담당한다. 이러한 기능적 분배는 서로 다른 타입의 데이터 전송 예를 들어 인터페이스용 웹 데이터, 데이터 베이스 트랜잭션 데이터 등을 전송 서버로부터 분리시키기 위한 목적으로 존재한다[5].

클라이언트에는 다중 접속 경로를 갖는 데이터 수신 모듈이 탑재되고 수신된 데이터 블록을 합병하여 플레이어에 공급하는 역할을 수행한다[17].

3. 서버 스트라이핑

스트라이핑은 병렬 주문형 비디오 서버 구조의 근간을 이루는 원리로 다수의 장치로 데이터를 분산하는 기법을 말하며 작업 처리량의 증가와 잠재적인 신뢰성 향상을 위해 사용되며 RAID(Redundant Array of Inexpensive Disks)가 대표적인 적용 사례이다[9].

스트라이핑 방식으로는 고정된 크기의 블록 단위로 데이터를 분할하는 스트라이핑 기법을 Space Striping이라고 하며 절차는 간단하지만 프레임 별 서비스가 클라이언트 측에서 행해져야 한다는 제약이 있다. 반면 고정 크기가 아닌 비디오 프레임의 단위에 맞추어 데이터를 분할하는 것을 Time Striping 이라고 하는데 이 방식은 프레임 별 서비스가 서버 측에서도 가능하지만 프레임을 구별하기 위한 부가적인 작업 부하를 가진다. MPEG(Motion Picture Expert Group)과 같은 동영상 데이터는 프레임 타입에 따라 프레임 사이즈가 매우 크게 변화하므로 스트라이핑 크기가 일정한 Space Striping 방식을 적용하는 것이 각각의 프레임을 구별하는데 소모되는 부하를 줄일 수 있으며 이 방식을 통하여 각각의 서버에 있어 동일한 작업량을 할당하는 자동 부하 분산의 효과 또한 얻을 수 있다[1, 10, 11].

PMSS 시스템에서는 Space Striping 방식을 이용하였으며 하나의 비디오 매체는 일정 크기의 블록들 즉 *V0*, *V1*, *V2*, …으로 분할된 후 라운드 로빈 방식을 통해 각각의 서버 *S0*, *S1*, *S2*, … *SN*에 저장된다. 여기에서 *V0*, *V1*, *V2*, …은 스트라이핑 블록을 나타낸 것이다.

4. 서비스 모델의 종류

병렬 주문형 비디오 시스템에서 서버와 클라이언트 간에 데이터를 전달하는 서비스 모델로는 Server Push 모델과 Client Pull 모델 그리고 두 모델의 장점만을 취한 IPP 모델로 크게 나눠 진다[12].

4.1 Server Push 모델

이 서비스 방식은 단일서버 주문형 비디오 시스템에서 일반적으로 연구되었으며 데이터 흐름의 주도권을 서버가 가지게 되며 클라이언트로부터의 특정 요청이 필요 없이 주기적으로 적절한 데이터를 클라이언트로 전송하는 방식으로, 브로드캐스트 서비스에 적합하나 클라이언트로부터의

역 채널이 없는 관계로 데이터를 공급하는 서버에서 데이터 전달 속도를 적절하게 조절하여야만 클라이언트에서의 버퍼 오버플로우 및 언더플로우를 방지할 수 있다는 단점이 있다[13-16].

4.2 Client Push 모델

Client Pull 서비스 방식은 클라이언트가 데이터 흐름의 주도권을 가지고 서버에 원하는 데이터를 요청하면 서버는 요청된 데이터를 클라이언트로 전달하는 요청/응답 형식을 가지는 일종의 Polling 방식으로서, 일반적인 운영체제의 파일 시스템 I/O가 여기에 해당하며 클라이언트에서의 버퍼 언더플로우 및 오버플로우를 클라이언트가 제어 할 수 있다. 이처럼 Server Push 방식에서의 클라이언트에 비해 보다 능동적인 역할을 수행한다 라는 장점을 가지는 반면 유니캐스트 서비스에 보다 적합하고, 클라이언트는 서버로 요청 메시지를 전송할 추가 채널을 가지고 있어야 하며, 서버는 요청을 처리하기 위해 연속적으로 인터럽트 되어야 하고 다수의 클라이언트로 인한 확장성 병목 현상이 쉽게 만들어진다 라는 단점을 가진다. WMT(Window Media Technology) 분야에서 주로 사용되는 방식이다[13, 14, 17].

4.3 IPP 모델

IPP서비스 방식은 Server Push방식과 Client Pull 방식이 결합된 서비스 방식으로 Client Pull 방식과 동일하게 데이터 흐름의 주도권을 클라이언트가 가지며, 단일 요청에 대한 다중 응답의 서비스 형태를 가진다. 클라이언트는 역 채널을 통하여 원하는 데이터 블록을 요청하게 되고 이 요청 메시지에는 요청하는 블록의 번호와 블록의 수가 포함되며 이를 통하여 서버는 요청 받은 번호의 블록부터 지정된 블록의 수만큼 해당 블록 데이터들을 클라이언트에게 전송하는 서비스 방식이다. 이를 통하여 데이터 전달 소고 조절을 통한 클라이언트 버퍼의 능동적 제어가 가능하고, 과다한 메시지 전송을 줄임으로써 미디어 데이터의 네트워크 전달 지연 문제 등을 해결할 수 있다[13, 14].

5. 단일 서버 환경에서의 서비스 모델의 성능 분석

서비스 모델의 전체적인 서비스 흐름은 다음과 같다. 먼저 버퍼 상태를 파악하고 비디오 상영에 필요한 데이터 요청 메시지를 네트워크를 통해 서버에 전달하는 단계인 데이터 요청, 서비스 스케줄러를 통해 요청 받은 데이터를 디스크로부터 전달 받아 해당 클라이언트에게 전송하는 단계인 서버 스케줄링, 서버로부터 전송되기 시작한 데이터가 네트워크를 통해 클라이언트로 전달되는 단계인 데이터 응답, 그리고 클라이언트 버퍼로 들어와 쌓인 데이터를 소비하는 단계인 데이터 소비 순으로 전개된다.

단일 서버 환경에서 세 가지의 서비스 모델 즉 Server Push 모델, Client Pull 모델, IPP 모델을 비교하여 보고 각 방식의 장단점을 분석한다. 단, 여기에서의 비교 분석 대상은 네트워크 지연과 서버 처리시간만을 고려한 작업시간의 총량으로 한정한다.

단일서버 환경에서 클라이언트에서 메시지가 출발하는 시간을 0으로 정하고, 클라이언트가 수신할 블록의 수를 B, 한 블록의 크기는 Q(byte)이다. 그리고 Client Pull 모델에서 클라이언트가 데이터를 수신한 이후 새로운 메시지를 생성하고 출발시키는 데까지 걸리는 시간은 무시한다.

5.1 단일 서버 환경에서의 Server Push 모델

Server Push 모델은 클라이언트에서 서버로의 역 채널은 존재하지 않으며, 최초 서비스를 위한 한 번의 데이터 요청, 서버 처리, 데이터 수신의 과정을 거치며 서비스가 이루어진다.

i번째 블록을 요청한 후 해당 블록을 수신하는 데 소요되는 시간을 Ti, 클라이언트에서 서버까지 블록 요청 메시지를 전달하는 데이터 요청 단계에서 소비되는 시간을 Treq(i), 서버에서의 메시지를 처리하는 서버 스케줄링 단계의 소비 시간을 Tsch(i), 서버에서 클라이언트까지의 블록 데이터가 전달되는 데이터 응답 단계의 소비 시간을 Tres(i) 이라고 한다. 여기에서 Ti는 데이터 소비단계의 처리 시간은 제외된다.

Server Push 모델에서 B개의 블록 데이터를 전달되는 데 소비되는 총 소비시간 Tsp은 식 (1)과 같다.

$$Tsp = Treq + B \times Tsch + Tres \quad (1)$$

5.2 단일 서버 환경에서의 Client Pull 모델

Client Pull 모델은 클라이언트에서 서버에게 데이터 요청 시 하나의 블록씩만을 요구하며, 데이터 요청, 서버 처리, 데이터 수신의 과정을 반복하면서 서비스가 이루어진다. Client Pull 모델의 총 소비시간 Tcp은 식 (2)와 같다.

$$Tcp = B \times (Treq + Tsch + Tres) \quad (2)$$

Server Push 모델에서의 서버 스케줄링 시간 Tsch은 서버가 데이터 요청을 수신한 이후 디스크로부터 데이터를 읽어 들여 네트워크로 전송하는 데 걸리는 시간으로 서버에서의 메시지 처리를 위해 데이터를 디스크로부터 전달받아 디스크 버퍼에 저장하는 디스크 라운드DR, 그리고 네트워크 전송을 위해 디스크 버퍼로부터 네트워크 버퍼로 이동시킨 후 전송하는 네트워크 라운드NR 이렇게 두 개의 라운드로 구성된다. 이렇게 두 개의 라운드로 구성되는 이유는 끊김 없이 지속적으로 데이터를 전송하기 위해서인데 이는 클라이언트로부터의 데이터 요청이 없고 오직 서버의 스케줄링에 의해서만 데이터가 전달되기 때문이다. 이 두 개의 라운드 시간은 서로 동일하다라고 가정한다. 즉 DR = NR = R, Tsch = 2R(R = Round time)이다.

이러한 서버 스케줄링을 통하여 서버에서 데이터를 처리하는 데 걸리는 시간은 처음 요청 메시지를 받은 경우를 제외하고 실제적으로 Tsch = R×(1+1/B)이다. 반면 Client

Pull 방식은 단지 요청된 블록만을 요청 받은 시기에 전달하면 된다. 즉 지속적인 데이터 블록 전달이 불필요하기 때문에 디스크 라운드와 네트워크 라운드로 나눌 필요가 없다. 즉 Client Pull 방식에서의 서버 처리 시간은 Tsch = 2R 이다.

$$Tsp = Treq + B\times(R\times(1 + 1/B)) + Tres \quad (3)$$

$$Tcp = B\times(Treq + 2R + Tres) \quad (4)$$

5.3 단일 서버 환경에서의 IPP 모델

IPP 모델은 클라이언트가 역 채널을 통하여 원하는 데이터 블록을 요청하고 이 요청 메시지에는 요청하는 블록의 번호와 블록의 수가 포함되며 이를 통하여 서버는 요청 받은 번호의 블록부터 지정된 블록의 수만큼 해당 블록 데이터들을 클라이언트에게 전송하는 서비스한다. 블록을 요청하는 전체 횟수를 N이라 하면, IPP 모델의 총 소비시간 Tipp은 식 (5)과 같다.

$$Tipp = N\times Treq+(B + N)\times R + N\times Tres \quad (5)$$

6. 병렬 서버 환경에서의 서비스 모델의 성능 분석

다음은 병렬 주문형 비디오 서버 환경에서 네트워크 지연과 서버 처리시간만을 고려한 작업시간의 총량에 있어서의 각 서비스 모델간의 비교 분석이다. 단일 서버의 경우 세 가지 모델 모두를 비교하였으나 Server Push과 IPP방식이 거의 비슷한 지연시간을 소모하고 Server Push 모델이 대부분 브로드캐스트 서비스에만 국한적이라는 사실을 통해 병렬 환경에서는 Client Pull 모델과 IPP 모델만을 비교 분석한다.

클라이언트가 다수의 서버에게 요청 메시지를 전달하는 시간차는 실제적으로 네트워크 전달 지연 변이에 대부분 흡수된다. 또한 서버의 수를 S라 할 때, 각각의 서버는 동일한 작업 처리 능력을 가진다. 즉, 클라이언트가 요청하는 S개의 메시지는 동시에 출발하며, 각각의 서버는 동일한 작업 처리 시간 Tsch 값을 가진다.

병렬 서버 환경에서의 Client Pull 서비스 모델과 IPP방식을 적용할 경우의 데이터 전달 지연시간 TP-CP, TP-IPP은 단일 서버 환경의 소요시간을 서비스에 참여하는 스트리밍 서버의 수로 나눈 값이 된다.

$$TP\text{-}CP = (B\times(Treq + 2R + Tres))/S \quad (6)$$

$$TP\text{-}IPP = (N\times Treq + (B + N)\times R + N\times Tres)/S \quad (7)$$

이 된다.

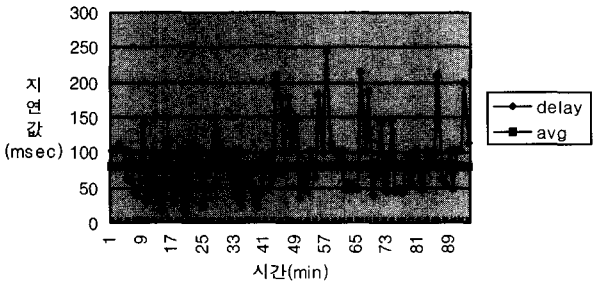
즉 병렬 서버 환경에서의 Client Pull 서비스 모델과 IPP 방식을 적용할 경우의 데이터 전달 지연시간 TP-CP, TP-IPP은 서버의 수(S)만큼 지연시간이 감소하게 되는 것이다.

단 위의 식에서는 클라이언트 측에서 데이터를 소비하는데 걸리는 시간은 포함되지 않았다. 즉 각각의 서버로부터 전송되어온 데이터 블록을 클라이언트 버퍼로 수신될 때까지의 시간만을 고려하였으며 클라이언트가 데이터를 정렬하고 다시 디코딩 모듈로 넘겨 비디오 데이터가 PLAY 되도록 하는데 걸리는 시간은 제외하고 계산된 결과이다.

7. 성능평가

인터넷의 경우 여러 시스템들이 연결되어 있고, 서비스의 다양화에 따라 각 네트워크 세그먼트들의 전송 데이터 유형이나 전송용량 및 패킷 손실 등의 트래픽 특성이 매우 가변적으로 나타난다. 여기에서 사용된 네트워크 전송간 평균지연은 다음과 같은 측정 시나리오를 통해 계산되었다.

본 시험의 측정 시나리오는 임의로 선정한 전남대학교와 서울 삼성소프트웨어멤버쉽 간의 인터넷을 대상으로 시험을 수행하였고, 두 노드 간에는 7 HOP의 서버 네트워크를 경유한다. 가장 네트워크 부하가 심한 것으로 조사된 오후 2시부터 4시까지의 시간에 측정되었으며, (그림 2)와 같은 지연 특성을 확인 할 수 있었다.

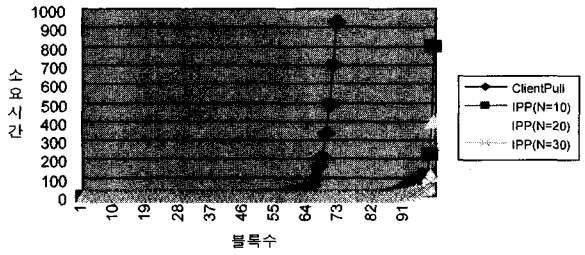


(그림 2) 네트워크 지연(전남대-서울삼성소프트웨어멤버쉽)

실험에 사용될 데이터는 MPEG-2 데이터 파일로서 2.5 Mbitrate 즉 약 300Kbyte/sec의 전송률을 가지는 미디어 파일이다. 위 미디어 데이터는 32Kbyte크기를 스트라이핑 단위로 사용하여 약 20000여 개의 블록으로 나누어지며 병렬 서버 환경의 경우 서버의 수(S)만큼 분산되어 저장된다.

7.1 단일 서버 환경에서의 서비스 모델간 소요시간 비교

(그림 3)은 단일 서버 환경에서 Client Pull 모델과 IPP 모델간 데이터 전달 소요시간을 비교한 그래프이다. X축은 전송된 블록의 수를 나타내며, Y축은 전송에 걸리는 시간을 지수함수를 통하여 확장한 값이다.



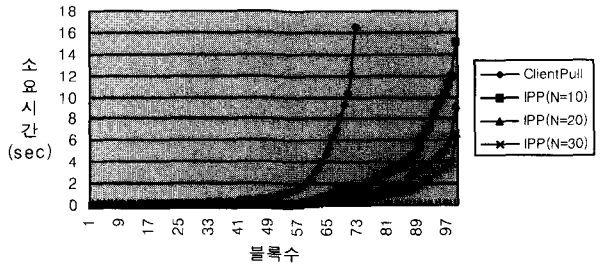
(그림 3) 단일 서버 환경에서의 서비스 모델간 소요시간 비교

전송 시간의 측면에서 Server Push 방식은 네트워크 지연에 큰 영향을 받지 않으므로 송수신되는 블록의 수가 증가함에 따라 수식과 거의 흡사하게 증가하여 그래프에 포함시키지 않는다. 또한 클라이언트 측에서 각각의 서버를 통해 들어오는 데이터의 조합에 걸리는 시간은 모두 제거된다. 즉 네트워크로부터 데이터를 수신하고 시간을 체크한 후 데이터는 곧바로 버리는 방식을 택함으로써 전달소요시간 측정을 위한 장애 요소 값들을 제거하였다.

IPP 방식은 한번에 전송되는 블록의 수(N)가 10, 20, 30으로 증가함에 따라 소요시간의 증가도 즉 그래프의 기울기가 줄어들어 점차적으로 Server Push 방식의 소요시간과 비슷해짐을 확인할 수 있다. 반면 Client Pull 방식은 IPP 방식과 Server Push방식에 비하여 훨씬 많은 시간을 소모하게 되며 빠른 시간 내에 서비스 한계에 다다르게 된다. 이로 인해 클라이언트 측에 많은 양의 메모리 요구와 빈번한 서비스 끊김 현상을 초래한다.

7.2 병렬 서버 환경에서의 서비스 모델간 소요시간 비교

(그림 4)는 병렬 서버 환경에서 Client Pull 모델과 IPP 모델간 데이터 전달 소요 시간을 비교한 그래프이다. X,Y 축은 (그림 3)과 동일하지만 소요시간의 감소로 인해 y축의 값의 범위가 축소되었다. 서버의 수(S)는 5로 고정하고, IPP 모델은 한번에 전송되는 블록의 수N(N = 10, 20, 30)에 따라 분리하였다.



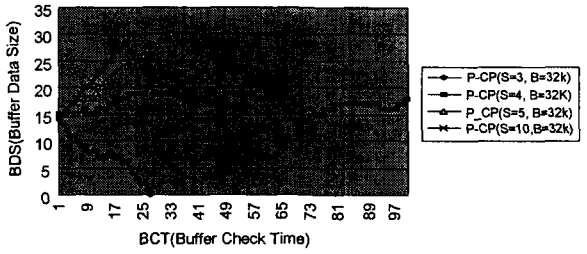
(그림 4) 병렬 서버 환경에서의 서비스 모델간 소요시간 비교

P-CP(병렬 Client Pull) 방식과 P-IPP(병렬 IPP) 방식에서의 소요시간이 단일 환경에 비해 현저하게 감소하였음을 알 수 있는데 이는 단일 서버에게 주어지던 부하가 다수의 서버들로 분산되면서 나타나는 장점이다. 이는 (그림 3)과의 비교를 통하여 확인할 수 있으며 식 (6), 식 (7)에 따라 서버의 수만큼 즉 서버의 수가 증가할수록 서비스 지연이 감소한 것으로 볼 수 있다.

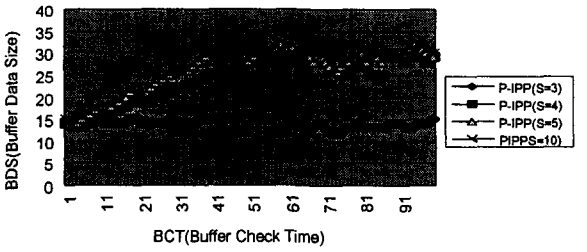
7.3 병렬 서버 환경에서의 클라이언트 버퍼 상태

(그림 5)는 병렬 서버 환경에서 Client Pull 모델을 이용하였을 경우 시간에 따른 클라이언트 버퍼의 데이터 적재량 변화도이다.

서버의 처리시간을 100msec, 서버의 스트라이핑 블록 크기(B)는 32kbyte일 때 서버의 수를 3, 4, 5, 10으로 증가시키면서 측정된 그래프로 버퍼 언더플로우나 오버플로우가 발생할 경우 버퍼 적재량의 측정을 중지하게 설정되었다. Client Pull 모델을 이용할 경우 서버의 수에 의해 버퍼 데이터 적재량의 변화가 매우 심한 것을 확인 할 수 있다. 이러한 현상은 앞에서 언급된 것처럼 데이터 전달 지연을 통해 발생하는 현상으로 서버의 수가 3일 경우 버퍼 데이터 적재량의 급격한 감소로 인해 버퍼 언더플로우가 발생하게 되고, 서버의 수가 4인 경우 버퍼 내 데이터 적재량이 큰 범위의 오실레이션을 일으킴으로써 버퍼의 안정성을 침해하게 되며 결과적으로 서비스의 질 하락을 유발할 수 있다. 서버의 수가 5 이상일 경우에만 버퍼 데이터 적재량 증가로 인해 버퍼의 안정화 상태가 유지된다. 이와 같이 서버의 수에 민감하게 반응하는 클라이언트 측 버퍼 상황은 추후 클라이언트 버퍼 설계와 서버 시스템의 확장성에 문제를 발생시키는 원인이 될 수 있다.



(그림 5) 병렬 서버 환경에서의 Client Pull 모델을 이용할 경우



(그림 6) 병렬 서버 환경에서의 IPP 모델을 이용할 경우

(그림 6)은 병렬 서버 환경에서 IPP 모델을 이용하는 경우의 그래프로 스트라이핑 블록의 크기가 16KByte이며 실험을 위해 실제 버퍼의 크기를 기존 버퍼보다 20% 크게 설정하였다. 서버의 수가 4 이상인 경우 충분한 버퍼 데이터 적재량을 유지하게 되며, 또한 서버의 수가 3인 경우에도 Client Pull 모델에 비해 보다 작은 오실레이션을 일으키며 이를 통하여 클라이언트 버퍼 안정화와 사용자 QoS를 보장할 수 있음을 확인 할 수 있다.

8. 결론 및 향후 연구 과제

본 논문은 단일 주문형 비디오 환경에서의 서버/클라이언트 간 데이터 서비스 모델을 분석하고 이를 병렬 주문형 비디오 환경에 적용함으로써 보다 나은 사용자 QoS를 제공하고자 한다. 이를 위해 단일 주문형 비디오 환경에서 데

이터 송수신의 주도권을 클라이언트가 가지는 Client Pull 모델과 이와 상반되는 Server Push 모델 그리고 두 모델을 통합한 IPP 모델을 네트워크 지연 시간의 측면에서 실험하였으며, 병렬 주문형 비디오 환경에서는 Client Pull 모델과 IPP 모델을 전달지연과 클라이언트 버퍼 내 데이터 잔여량 측면에서 비교 및 분석하였다.

실험을 통해 병렬 주문형 비디오 환경에서 IPP 서비스 모델이 가장 적은 전달지연과 보다 안정적인 클라이언트 버퍼를 유지함을 알 수 있으며 이를 통해 사용자에게 보다 나은 서비스를 제공할 수 있음을 검증하였다. 추후 클라이언트 버퍼로 들어오는 데이터의 병합 시간에 대한 연구가 진행되어야 하겠다.

참 고 문 헌

[1] D. J. Gemmell, H. M. Vin, D. D. Kandlur, P. V. Rangan, and L. A. Rowe, "Multimedia storage servers : A Tutorial," IEEE Multimedia Mag., Vol.27, pp.40-49, May, 1995.

[2] M. Buddhikot, G. Parulkar and J. Cox, "Design of a Large Scale Multimedia Storage Server," Proceedings of the INET '94/JENC5, Conference of the Internet Society and the Joint European Networking Conference, Prague, Ozech, June 1994.

[3] D. Jadav and A. Houdhary, "Design Issues in High Performance Media-on-Demand Servers," IEEE Parallel and Distributed Technology Systems and Applications, Summer, 1995.

[4] D. P. Wu, Y. W. T. Hou and W. W. Zhu, "Streaming Video over the Internet : Approaches and Directions," IEEE Transactions on Circuits & Systems for Video Technology, Vol.11, No.3, 2001.

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

[6] M. Wu and W. Shu, "Scheduling for Large-Scale Parallel Video Servers," Proc. Sixth Symp. On the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, Los Alamitos, Calif., pp.126-133, 1996.

[7] Y. B. Lee, "Parallel video servers-A Tutorial," IEEE Multimedia Mag., Vol.5, No.2, pp.20-28, 1998.

[8] 김서균, 김경훈, 류재상, 남지승, "리눅스 기반의 고성능 병렬 미디어 스트림 서버 설계 및 구현", 정보처리학회 논문지A, Vol.8-A, No.4, pp.287-292, 2001.

[9] P. Shenoy and H. Vin, "Efficient striping techniques for multimedia file servers," in NOSSDAV 97 (G. Parulkar, ed.), May, 1997.

[10] W. Liao and V. O. K. Li, "The Split and Merge Protocol for Interactive Video-on=-Demand," IEEE Multimedia, Vol. 4, No.4, pp.51-62, Oct., 1997.

[11] Kai Hwang, Hai Jin and Roy Ho, "Distributed Software RAID Architecture for Parallel I/O in Serverless Clusters," IEEE Transaction on Parallel and Distributed Systems, September, 2000.

[12] S. Rao, H. Vin and A. Tarafdar, "Comparative Evaluation of Server-push and Client-Push Architectures for Multimedia Servers," In Proc. of the 6th International Workshop on Network and Operation System Support for Digital Audio and Video, April, 1996.

[13] S. Acharya, M. Franklin and S. Zdonik, "Balancing Push and Pull for Data Broadcast," In Proc.of ACM SIGMOD Conference, Vol.26, No.2, pp.183-98, May, 1997.

[14] J. P. Martin-Flatin, "Push vs. Pull in Web-based Network Management," In Proc. of the Integrated Network Management VI, pp.3-18, May, 1999.

[15] T. Kiao, "Global Information Broadcast : an architecture for Internet push channels," IEEE Internet Computing, Vol.4, No.4, pp.16-25, July, 2000.

[16] J. Y. B. Lee, "Staggered push a linearly scalable architecture for push-based parallel video servers," IEEE Trans on Multimedia, Vol.4, No.4, pp.423~433, December, 2002.

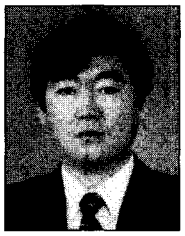
[17] Microsoft, "Microsoft Media Service SDK," 2002.

이 민 홍

e-mail : bluenomad@empal.com
2001년 전남대학교 컴퓨터공학과(공학사)
2003년 전남대학교 컴퓨터공학과(공학석사)
1999년~2001년 삼성소프트웨어멤버쉽
2003년~현재 (주)포스트림
관심분야 : 멀티미디어 네트워크, 병렬 주문형 비디오 시스템

김 경 훈

e-mail : pluit@mdclab.chonnam.ac.kr
1998년 대불대학교 컴퓨터과학과(이학사)
2000년 전남대학교 컴퓨터공학과(공학석사)
2000년~현재 전남대학교 컴퓨터공학과 박사과정
2000년~2003 (주)포스트림 기술이사
2003년~현재 (주)포스트림 대표이사
관심분야 : 병렬 주문형 비디오 시스템, 실시간 통신 시스템



남 지 승

e-mail : jsnam@chonnam.chonnam.ac.kr
1981년 인하대학교 전자공학과(공학사)
1985년 University of Alabama, Electrical Engineering(공학석사)
1992년 University of Arizona, Electrical & Computer Engineering(공학박사)
1992년~1995년 한국전자통신연구소 선임연구원
1995년~현재 전남대학교 컴퓨터공학과 부교수
1999년~현재 정보통신 특성화 센터 소장
관심분야 : 컴퓨터 네트워크, 실시간 멀티미디어 시스템, 병원정보시스템