주문형 비디오 시스템에서 클라이언트 버퍼를 활용한 요구 스케쥴링 기법

한 금 희[†] · 김 종 훈[††] · 원 유 헌[†††]

요 약

주문형 비디오 시스템에서 요구 스케쥴링이란 시스템의 효율을 높이고, 한번의 디스크 검색으로 여러 사용자들의 요구를 만족시키기 위한 방법이다. 본 논문에서는 클라이언트 버퍼를 활용함으로 서버의 디스크 I/O와 버퍼 소비를 줄이는 Stream Relay Scheme (SRS)를 제안하였다. 본 논문에서 제안한 SRS 기법은 모든 서비스를 서버가 처리하는 것이 아니라 해당 비디오를 저장하고 있는 클라이언트로 하여금 전송하도록 함으로써 시청자들의 초기지연시간을 단축시키고 서버의 처리량을 증가시키는 기법이다. 또한 SRS에 배칭 기법을 통합한 SRS-BAT 기법도 소개하였다. 이 논문에서 제안한 기법과 기존의 기법의 성능을 시뮬레이션으로 비교·실험한 결과 제안한 기법이 요구당 초기 지연시간과 서버의 처리량에 있어서 효율적인 성능을 나타내었다.

A Request Scheduling Strategy using Client's Buffer in VOD Systems

Kum-Hee Han[†] · Jong-Hoon Kim[††] · Yoo-Hun Won[†††]

ABSTRACT

In a VOD(video-on-demand) system, a scheduling strategy is designed to increase the system efficiency and to satisfy the isochronous requirements of showing a video to multiple viewers with one disk access. In this paper, we have proposed the Stream Relay Scheme (SRS) which utilizes the client's buffer space to reduce the server's disk I/O, buffer consumption, and stream capacity. Under the SRS which we have proposed in this paper, the server does not service all the requests directly but forwards the new request selectively to the client who is being served currently in order to reduce the initial latency and increase the system throughput. The SRS-BAT which integrated the SRS with the Batching technique is introduced. The results of the simulated experiments which compared the SRS with the conventional technique have shown the noticeable performance improvements in terms of initial latency and the server throughput.

1. 서 론

최근 오디오와 비디오 데이터를 처리할 수 있는 멀티미디어 워크스테이션이 출현되고 ATM[1]과 Myrinet[2]와 같은 초고속 네트워크들이 개발됨에 따라 주문형 비디오 시스템이 고안되고 있다[3]. 주문형 비디오 시스템은 영화, 비디오게임, 방송 프로그램 등 각종 영상물을 컴퓨터에 데이터 베이스화한 다음 일반 가입자들에게 네트워크를 통하여 원하는 멀티미디어 정보를 제공하는 시스템이다. 그러나 이와 같은 기술을 현실화하기 위

※ 본 연구는 1999년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음
† 정 회 원 : 가톨릭대학교 컴퓨터공학부 교수
†† 종신회원 : 제주교육대학교 컴퓨터교육과 교수
††† 정 회 원 : 홍익대학교 컴퓨터공학과 교수
논문접수 : 1999년 4월 15일, 심사완료 : 2000년 1월 9일

해서는 대용량의 데이터를 디스크 기억장치에 효율적으로 저장하고 있다가 보다 많은 사용자들에게 그들이 원하는 서비스를 제공하는 주문형 비디오 서버 기술의 지원이 있어야 한다[4]. 이와 같은 지원을 위한 노력은 여러 방면에서 시도되고 있는데 브릿징(bridging), 피기백킹(piggybacking), 배칭(batching) 등과 같은 요구 스케쥴링 정책[5]이 이에 해당된다.

주문형 비디오 시스템에서는 일반적인 텍스트 데이터와는 달리 동일한 비디오에 대한 요구가 많이 발생할 것이므로 특정 비디오에 대한 요구가 발생했을 때 해당 비디오의 내용이 다른 클라이언트의 버퍼에 유지되고 있을 확률이 높을 것이다. 그러므로 이를 효율적으로 활용하는 연구가 필요하나 주문형 비디오 시스템에서는 이를 고려한 연구는 진행된 바가 없다. 또한 최근 클라이언트의 성능이 놀라울 정도로 발전하여 일반 분산 시스템 환경에서는 클라이언트 자원을 효율적으로 활용하는 연구[6, 7]가 진행되고 있는데 주문형 비디오 시스템 환경에서는 이에 대한 연구 결과나 진행이 미진한 상태에 있다. 앞서 언급한 스케쥴링 정책들도 이를 간과하고 예전의 중앙집중식 컴퓨팅과 유사하게 서버의 능력에만 업무의 대부분을 의존하고 있다.

그러므로 본 논문에서는 이러한 점을 고려한 요구 스케쥴링 정책을 제안한다. 제안한 정책에서는 요구가 발생했을 때 무조건 서버가 서비스를 제공하는 것이 아니라 해당 비디오를 저장하고 있는 클라이언트가 있는지를 검색해서 있을 경우에 작업을 클라이언트에게 넘겨 동작하게 된다. 제안한 정책의 효과를 검증하기 위해서 시뮬레이션을 통해 배칭 기법과의 성능을 비교한다. 성능 비교를 통해 본 논문에서 제안한 정책이 요구당 초기 지연시간과 서버의 처리량에 있어서 효율적인 성능을 나타내는 것을 확인한다. 특히 본 논문에서 제안한 정책에 배칭 기법을 통합한 경우에 더욱 효율적인 성능을 나타내는 것을 확인한다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 기존의 요구 스케쥴링 정책들과 버퍼링에 대해 살펴보고 문제점을 지적한다. 3장에서 본 논문에서 제안한 스케쥴링 정책에 대해 설명하고 4장에서는 성능 평가를 위한 실험 환경을 기술하며 실험 결과와 분석 내용을 설명한다. 그리고 5장에서 결론과 향후 연구를 논의한다.

2. 관련연구

본 장에서는 본 논문과 관련된 연구로 기존의 요구 스케쥴링 기법과 멀티미디어 시스템에서의 버퍼링에 관한 연구에 대해 살펴본다.

2.1 요구 스케쥴링에 관한 연구

브릿징[8, 9]은 동일한 비디오에 대한 요구가 있을 때 가장 먼저 온 요구에 대해서는 디스크를 검색하지만 이 검색한 데이터를 메모리에 버퍼링 함으로 뒤에 발생한 요구들에 대해서는 디스크가 아닌 메모리에서 서비스해 주는 정책이다. 그러나 이 기법은 매우 많은 양의 메모리 공간을 필요로 하는 단점이 있다.

피기백킹[10]은 비디오의 상영속도가 5%정도 빨라지거나 느려지는 것은 사용자가 감지하기 힘들다는 점을 이용한 방법으로 서비스 중에 동일한 비디오에 대한 요구가 들어오면 앞서고 있는 비디오의 속도를 늦추고 새로 시작하는 비디오의 속도를 빠르게 조정하여 두 사용자에게 동일한 스트림으로 서비스하는 기법이다. 그러나 서버가 검색해야 하는 데이터의 양의 변화가 매우 심하게 되어 디스크 입출력 양이 큰 폭으로 변하고, 각기 다른 속도로 검색하는 데에 서버의 능력이 요구되는 문제점이 있다. 또한 널리 이용되는 압축 기법인 MPEG 형식의 파일에는 적용하기가 힘들다는 문제점도 있다.

배칭[11, 12]은 배칭 간격 동안 동일한 사용자 요구들을 모아 하나의 입출력 스트림으로 서비스하는 방법이다. 이러한 배칭은 브릿징이나 피기백킹처럼 큰 메모리의 요구도 없고 서버가 데이터의 검색 속도를 빠르거나 느리게 하는데 능력을 소모하지 않아도 된다는 장점이 있다. 하지만 모든 요구에 동일하게 배칭 간격이 적용되므로 클라이언트가 불필요하게 기다리는 일이 생길 수 있다. 즉 인기 없는 비디오를 요구한 사용자 또는 평일 새벽시간대처럼 사용자가 거의 없는 시간에 요구한 사용자가 이에 해당된다.

체이닝[13]은 클라이언트들의 디스크를 활용함으로써 네트워크 I/O 요구량을 감소시킬 뿐만 아니라 더불어 서버의 자원을 절감하는 데에 목적이 있다. 체이닝에서는 첫 번째로 도착한 시청자의 요구는 서버로부터 서비스 받게 되며 동일한 비디오를 신청한 그 다음 번에 두 번째로 들어온 요구는 첫 번째 시청자인 클라이언트의 디스크로부터 제공받는 것이다. 세 번째 도착

한 요구는 물론 두 번째 클라이언트로부터 제공받는 식으로 진행된다. 그러므로 일찍 도착한 시청자의 요구가 배칭 간격 내에 늦게 도착하는 요구들을 기다릴 필요가 없게 되는데 이는 클라이언트의 워크스테이션으로부터 직접 멀티캐스트로 서비스 받게 되기 때문이다. 이런 식으로 시청자의 요구는 서버에 더 이상 부담을 준다기보다 공헌하는 역설적인 일면을 지니게 된다. 시청자의 요구가 드물게 들어오면 사실상 체이닝 기법을 사용하기가 어렵게 된다. 체인이 길어지면 더 많은 클라이언트가 같은 비디오 스트림을 공유하게 되므로 바람직하다. 그러나 클라이언트의 디스크 자원을 활용한다는 것은 비현실적인 정책으로 사료된다.

2.2 버퍼링에 관한 연구

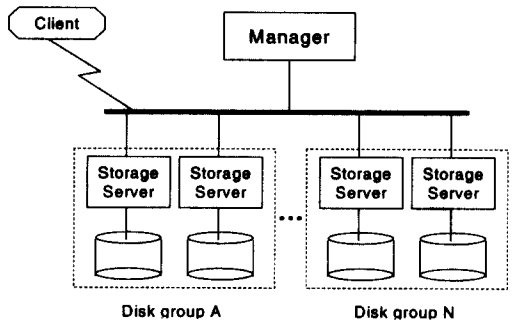
전통적인 시스템에서 버퍼링에 관한 연구[14, 15, 16]가 많이 이루어 졌는데, 이들 정책들을 멀티미디어 시스템에 그대로 도입하는데는 다음과 같은 이유 때문에 적합하지 않다. 첫째는 기존의 정책들은 블록 단위로 동작하나 멀티미디어 응용에서는 영화와 같은 완전한 멀티미디어 객체 단위로 동작한다는 점이고 두 번째 이유는 기존의 정책들은 과거의 참조 기록을 근거로 동작하나 멀티미디어 응용에서는 데이터의 접근을 예측할 수 있다는 점이다. 그러므로 멀티미디어 시스템에서의 버퍼링은 전통적인 시스템과는 다른 각도에서 접근이 이루어져야 하는데 이러한 점을 고려한 버퍼링에 관한 연구는 다음과 같다. Buffer sharing[8] 기법은 선행 요구에 의해 접근(access)되는 데이터를 필요로 하는 요구인 후행 요구가 발생하게 되면 선행 요구에 의해 읽혀진 데이터를 버퍼에 보관함으로 후행 요구에 대한 서비스는 버퍼에서 수행하게 되므로 디스크 접근을 최소화하게 된다. Interval caching[9] 기법은 buffer sharing을 하되 모든 요구들에게 동일하게 공유를 제공하는 것이 아니라 선행 요구와 후행 요구간의 간격(interval)이 작은 순으로 버퍼 공유를 제공한다. 그러므로 버퍼에서 서비스하는 요구의 수를 최대화할 수 있게 된다. Controlled buffer sharing[17] 기법 역시 버퍼 공유를 제공하되 모든 요구들에 대해 버퍼 공유를 제공하는 것이 아니라 선행 요구와 후행 요구간의 간격이 distance threshold라는 크기 보다 작은 경우에 대해서만 버퍼 공유를 제공하는 것이다. 이는 선행 요구와 후행 요구간의 시간 간격이 큰 경우에 버퍼 공유를 허용하면 이로 인한 손실이 크기 때문이다.

그러나 이들 연구에는 서버의 버퍼만을 활용하는 정책들로 클라이언트 버퍼에 대해서는 고려하지 않고 연구가 진행되었다. 그러므로 본 논문에서는 클라이언트 버퍼를 활용한 정책을 제안하고자 한다.

3. 클라이언트 버퍼를 활용한 요구 스케줄링 기법

본 장에서는 시스템 구조에 대하여 살펴보고 본 논문에서 제안한 스케줄링 기법을 배칭을 고려하지 않은 SRS와 배칭을 통합시킨 SRS-BAT로 구분하여 설명한다.

3.1 시스템 구조



(그림 1) 시스템의 구조

(그림 1)은 본 논문에서 가정하는 주문형 비디오 시스템의 구조를 나타낸 것으로 각 사용자를 클라이언트로 하고 서비스 제공자를 서버로 하는 클라이언트-서버 구조이다. 특히 서버는 저장 서버(storage server)와 관리자(manager)로 구성되어 있다. 저장 서버는 비디오 데이터를 검색하여 요청한 클라이언트에게 데이터를 보내주는 일을 담당하는 부분으로 디스크 효율을 높이기 위해 그룹화 하여 그룹내 다수의 디스크들에 스트라이핑[18, 19, 20]하여 데이터를 저장한다. 그룹내 요구들에 대한 서비스는 라운드-로빈 방식으로 이루어진다. 관리자는 클라이언트의 요구들에 대한 관리와 이를 처리하는 부분으로 요구에 대한 서비스가 가능한지를 검사하여 서비스를 지시하거나 큐잉하여 저장·관리하는데 구체적인 동작은 뒤에서 언급한다.

3.2 Stream Relay Scheme (SRS)

주문형 비디오 시스템에서는 일반적인 텍스트 데이터와는 달리 동일한 비디오에 대한 요구가 많이 발생

하므로 특정 비디오에 대한 요구가 발생했을 때 해당 비디오 스트림이 다른 클라이언트의 버퍼에 유지되고 있을 확률이 높다. 이러한 점을 고려하여 SRS 기법에서는 요구가 발생했을 때 무조건 서버가 서비스를 제공하는 것이 아니라 해당 비디오 스트림을 버퍼에 저장하고 있는 클라이언트가 있는지를 검색하여 있을 경우에 작업을 클라이언트에게 넘겨 동작하게 된다. 이렇게 하여 서버의 부하가 감소하게 되고 더욱 많은 사용자들에 대한 서비스를 제공하게 된다. 최근 클라이언트의 성능은 놀라울 정도로 발전하여 자신의 버퍼에 저장되어 있는 비디오 스트림을 다른 클라이언트에게 보내는 단순한 작업을 처리하는 것은 전혀 문제가 되지 않는다..

서비스하는 비디오의 시작 부분을 버퍼에 저장하고 있는 클라이언트가 있는지에 대한 정보를 관리자가 테이블에 유지하고 있다. 관리 테이블은 매우 간단한 구조를 지니고 있다. 만약 200가지 종류의 비디오를 서비스하는 시스템이라면 크기가 200인 두 개의 정수형 배열로 이루어진 테이블 하나만 있으면 된다. 만약 1번 비디오를 10번 클라이언트가 보기 시작하면 TABLE[1][1]에 클라이언트 번호 10을 저장하고 TABLE[1][2]에는 클라이언트가 비디오를 보기 시작한 시간을 저장하면 된다. 그리고 주기적으로 TABLE[i][2]의 내용을 검색하여 비디오 시작 부분이 TABLE[i][1]번 클라이언트의 버퍼에서 쫓겨날 시간이 되면 TABLE[i][1]에 NULL을 저장한다.

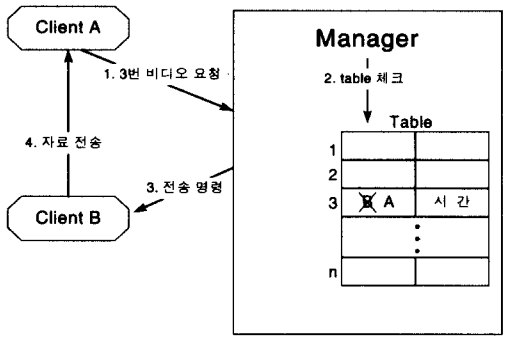
주문형 비디오 시스템은 전통적인 시스템과는 달리 접근 형태를 예측할 수 있고 블록 단위로 동작하지 않고 완전한 멀티미디어 객체(예, 영화) 단위로 동작하므로 본 정책에서 클라이언트 버퍼 교체 정책은 FIFO(First-In First-Out)를 따른다.

다음은 SRS 기법의 동작을 요구의 형태에 따라 분류하여 나타낸 것이다.

3.2.1 클라이언트로부터 발생한 요구

클라이언트로부터 요구가 발생했을 때 요구된 비디오의 시작 부분이 다른 클라이언트의 버퍼에 있는지의 여부에 따라 동작이 달라진다.

- 요구된 비디오의 시작 부분이 다른 클라이언트 버퍼에 있을 경우 (그림 2)의 예를 통해 동작을 살펴본다. 클라이언트 A로부터 3번 비디오에 대한 요구가 들어오면 관리자는 관리 테이블을 검색하여 3번 비디오의 시작 부분을 버퍼에 저장하고 있는 클라이언트가 있는지를 점검한다. 만약 해당 비디오를 저장하고 있는 클라이언트(B)가 있을 경우에 관리자가 자료를 전송하라는 명령을 클라이언트 B에게 보낸다. 그러면 클라이언트 B에서 해당 비디오 스트림을 클라이언트 A에게 전송하게 된다. 그리고 관리자는 관리 테이블의 내용을 B에서 A로 갱신하고 시간 정보도 갱신한다.



(그림 2) 클라이언트 A가 요구한 3번 비디오가 클라이언트 B의 버퍼에 있을 경우의 동작

- 요구된 비디오의 시작 부분이 다른 클라이언트 버퍼에 없을 경우 해당 요구를 관리자의 대기 큐에 큐잉한다.

(그림 3)은 클라이언트로부터 요구가 발생했을 때의 동작을 알고리즘으로 나타낸 것이다.

```
while (클라이언트로부터 요구 발생) {
  table check;
  if (요구된 비디오의 시작 부분이 다른 클라이언트(source_client)에 존재) {
      source_client에게 자료전송 요구;
      table 갱신;
  }
  else {
      enqueue;
  }
}
```

(그림 3) 클라이언트로부터 요구가 발생했을 때의 동작 알고리즘

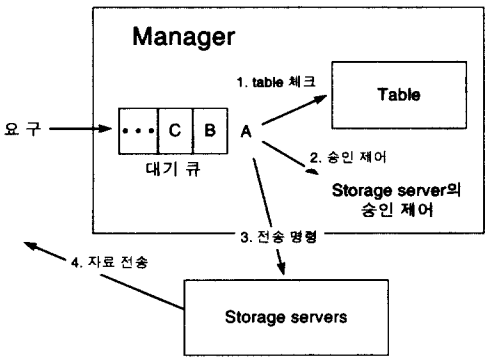
3.2.2 큐잉되었던 요구

주기적으로 대기큐에서 요구를 디큐하여 처리하는데 다음 과정을 따른다.

- 요구된 비디오의 시작 부분이 다른 클라이언트 버퍼에 있을 경우 관리 테이블을 검색하여 해당 비디오의 시작 부분을 저장하고 있는 클라이언트가 있으면 (그림 2)와 동일하게 처리한다.

- 저장 서버가 요구를 처리할 수 있을 경우 만약 요구된 비디오의 시작 부분을 저장하고 있는 클라이언트가 없을 경우에는 저장 서버가 요구를 승인할 수 있는지를 판단한다. 승인 제어는 묵시적으로 FCFS (First-Come First-Served) 방식으로 처리되지만, 저장 서버는 디스크 대역폭, 스트림 용량, 버퍼 및 네트워크 대역폭 등이 충분할 때 새로운 비디오 스트림의 연속적인 전송을 시작할 수 있는 것이다. 승인이 가능하면 요구를 저장 서버로 전송하여 저장 서버에서 비디오 스트림을 클라이언트로 전송하도록 한다. 그리고 관리자는 관리 테이블의 내용을 갱신한다.
- 처리가 불가능한 경우 만약 저장 서버가 요구를 처리할 수 없을 때 즉 자원이 충분하지 않을 경우에는 큐에서 대기한다.

(그림 4)는 이를 그림으로 나타낸 것이고, (그림 5)는 큐잉된 요구들을 처리하는 알고리즘이다.



(그림 4) 디큐된 요구를 처리하는 동작

```
while (큐에 요구가 존재) {
   dequeue;
   table check;
   if (요구된 비디오의 시작 부분이 다른 클라이언트에 존재) {
        source_client에게 자료전송 요구;
        table 갱신;
    }
     else if (저장 서버가 서비스 가능) {
        저장 서버가 자료 전송;
        table 갱신;
    }
    else {
        큐에서 대기;
    }
}
```

(그림 5) 큐잉된 요구들을 처리하는 알고리즘

3.3 Stream Relay Scheme with Batching (SRS-BAT)

앞에서 언급한 것처럼 배칭 기법은 미리 정해진 시간간격(batching interval)동안에 동일한 비디오를 보겠다고 신청한 다수의 요구들을 모아서 하나의 스트림으로 일괄처리 하는 방법으로 클라이언트들의 동일한 비디오에 대한 요구를 일정한 시간동안 모아서 한꺼번에 서비스함으로써 필요한 디스크 I/O와 버퍼 요구량을 감소시키는 방법이다. SRS 기법에서는 클라이언트로부터 직접 들어온 요구가 다른 클라이언트를 통해 서비스되지 않을 경우에 대기큐에 저장되어 스케줄링을 기다리게 되는데, 이 때 대기큐에 저장된 요구들은 FCFS 방식으로 한번에 하나씩 처리하도록 되어 있다.

SRS 기법을 배칭 기법과 결합시키는 것은 대기큐에 저장된 요구들을 스케줄링할 때 맨 앞에서 디큐(dequeue)된 요구와 동일한 비디오를 신청한 모든 요구들을 한꺼번에 처리하도록하기 위한 것이다. 이렇게 배칭 기법을 SRS에 적용하게 되면 클라이언트로부터 도착한 요구들은 SRS 기법의 장점을 취하게 되고, 대기큐에서 기다리고 있던 요구들은 배칭 기법의 장점을 얻게 되므로 더욱 효율적인 성능을 나타낼 것이다. 다시 말해서 SRS-BAT의 동작은 대기큐에 큐잉되었던 요구들을 처리하는 과정에서 약간의 차이가 있을 뿐 SRS 기법과 거의 유사하다. 차이점은 큐에서 디큐된 요구들을 처리할 때 SRS 기법에서는 디큐된 요구 하나만을 서비스하고 있고, SRS-BAT에서는 디큐된 요구와 동일한 비디오를 신청하고 큐에서 대기하고 있던 모든 요구들을 한꺼번에 서비스하는 것이다.

4. 실 험

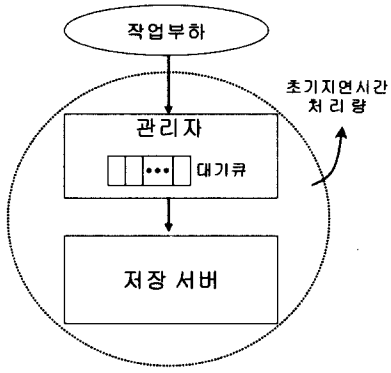
본 장에서는 성능 평가를 위한 실험 환경을 살펴보고 실험 결과와 그에 따른 분석 내용을 설명한다.

4.1 실험 환경

우선적으로 SRS 기법으로 얻어지는 성능 향상을 정확하게 파악하기 위하여 FCFS 방식을 적용한 배칭 기법의 성능과 비교하였는데, 배칭 기법은 배칭 간격을 다양하게 하여 실험한 결과중에서 가장 효율적인 성능을 나타낸 것으로 선택하였다. 이차적으로 SRS 기법에 배칭 기법을 적용한 SRS-BAT의 성능을 다양하게 살펴보았다.

실험은 시뮬레이션을 통해 평가하였다. 사용된 요구 작업부하는 실험의 편리함을 위하여 가상적으로 만들어 사용하였다. 가상 요구 작업부하는 주어진 수만큼의 요구를 생성할 수 있도록 되어있으며 작업부하는 요구하는 클라이언트의 번호와 원하는 비디오 번호로 이루어져 있다. 요구하는 비디오 번호는 인기 비디오와 비인기 비디오를 형성하기 위해 지수분포를 따르도록 하였다. 즉 인기있는 비디오의 요구는 자주 발생하고 비인기 비디오에 대한 요구는 드물게 발생한다. 본 실험에서는 20,000개의 요구로 이루어진 작업부하를 만들어 이용하였으며 요구 발생 시간은 다양하게 하며 실험하였다.

시뮬레이터는 C++를 사용하여 개발하였으며 구성은 (그림 6)과 같이 작업부하를 입력하는 부분과 관리자와 저장 서버로 구성된다. 작업부하에서 요구들은 주기적으로 서버 시뮬레이터로 입력되며 시뮬레이터에서 관리자는 대기큐를 두어 들어오는 요구들을 저장하며 큐에 들어온 순간부터 시간을 측정한다. 시뮬레이터의 동작은 알고리즘과 동일하게 작동하며 시뮬레이터에서는 각 요구에 대한 초기지연시간과 서버의 처리량을 측정한다.



(그림 6) 시뮬레이터의 구성

시뮬레이션 파라미터는 <표 1>과 같이 서버가 제공하는 비디오의 개수는 100개, 각 비디오의 길이는 100분, 서버 스트림 용량은 200개로 하였다. 그리고 한 라운드는 1초로 가정하였으며 한 라운드당 한 클라이언트에게 보내는 데이터의 양은 MPEG-1[21]을 기준으로 192KB로 하였다. 192KB 데이터가 네트워크를 통해 전송되는 시간은 ATM/155Mbps[1]를 고려하여 네트워크 오버헤드를 포함하여 9.8 miliseconds인데 10 miliseconds로 고정시키고 실험하였다. 각 정책에 대한 성능 비교 척도로는 요구당 초기 지연시간과 서버의 처리량을 사용하였으며 시뮬레이션에서 클라이언트 요구 발생 간격을 다양하게 하여 실험하였다. 단, 신청했던 비디오를 보던 중에 이탈하거나 신청했다가 취소하는 경우는 없는 것으로 가정하였다. 또한 VCR 동작은 고려하지 않았으며 이 동작을 고려하려면 정책에 있어서 추가적인 보완이 있어야 한다.

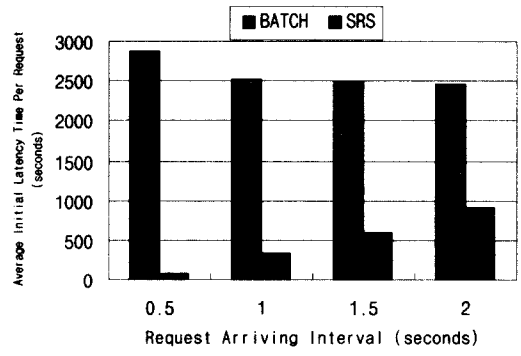
<표 1> 시뮬레이션 파라미터

비디오의 개수	100개
각 비디오의 상영시간	100분
서버 스트림 용량	200개
라운드 시간	1초
요구 발생 간격	0.5초, 1초, 1.5초, 2초
데이터의 양/ 각 라운드, 각 클라이언트	192KB
192KB 데이터의 네트워크 전송시간	10 miliseconds

4.2 실험 결과

4.2.1 SRS 기법과 배칭 기법의 비교

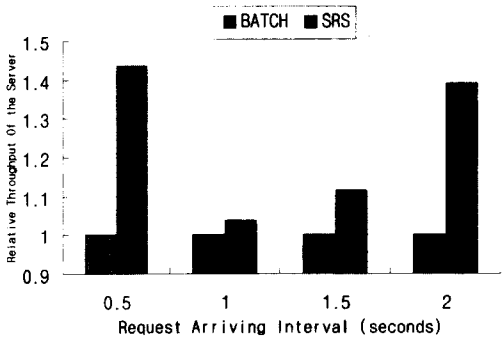
SRS 기법과 FCFS 방식을 적용한 배칭 기법의 성능을 비교하였다. 본 절에서 사용된 SRS 기법에서는 본 논문에서 제안하는 정책에 의한 성능향상을 파악하기 위하여 전혀 배칭을 하지 않았다.



(그림 7) 정책에 따른 초기지연시간

(그림 7)은 스케줄링 정책에 따른 초기지연시간을 나타낸 것이다. 그림에서 BATCH는 배칭 기법을 나타

낸 것이고 SRS는 stream relay scheme을 나타낸 것으로 각 클라이언트 버퍼의 용량은 압축된 데이터 5분 동안의 분량을 저장할 수 있는 크기이다. 그리고 X축은 클라이언트로부터 발생하는 요구 발생 간격을 의미한다. 그림의 결과에서 몇 가지 사항을 관찰할 수 있다. 우선 모든 경우에 있어서 SRS 기법이 배칭 기법에 비해 매우 효율적인 초기 지연시간을 나타내는 것을 볼 수 있다. 이는 자주 요구되는 비디오에 대한 서비스를 다른 클라이언트로 하여금 처리하게 함으로 전체 시스템 입장에서 더욱 많은 클라이언트들에 대한 서비스를 제공할 수 있기 때문이다. 다음으로 SRS 기법에서는 요구 발생 간격이 커짐에 따라 요구당 초기 지연 시간이 지연되는 것을 확인할 수 있는데 이는 요구 발생 간격이 커지면 커질수록 비디오의 시작 부분을 다른 클라이언트의 버퍼에 저장하고 있는 동안 발생하는 요구의 발생 횟수가 감소하기 때문이다.

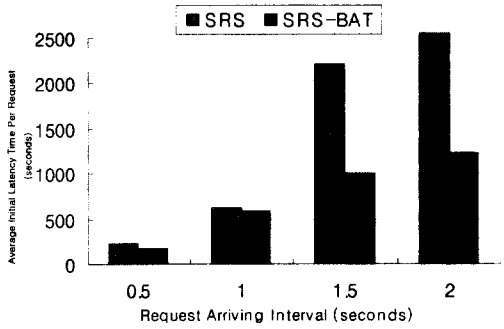


(그림 8) 정책에 따른 서버의 상대적인 처리량

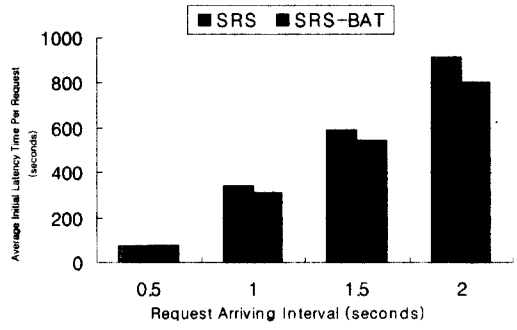
(그림 8)은 정책에 따른 서버의 처리량을 상대적으로 나타낸 것이다. 정책간의 성능에 있어서는 SRS 기법이 배칭 기법에 비해 4~45% 가량 서버의 처리량이 증가하는 것을 볼 수 있다. 그러므로 본 논문에서 제안한 SRS 기법은 클라이언트 입장에서 초기지연시간이 짧아짐은 물론이고 서버의 처리량 측면에 있어서도 좋은 성능을 나타내는 효율적인 정책임을 알 수 있다.

4.2.2 SRS 기법에 배칭 기법을 적용하였을 경우의 효과

본 절에서는 SRS 기법과 배칭 기법을 통합했을 경우 성능에 어떠한 영향을 미치는지를 살펴본다. (그림 9)는 SRS 기법을 배칭에 적용했을 경우와 적용하지 않았을 경우의 초기 지연 시간을 나타낸 것이다. 특히 (a)는 클라이언트의 버퍼 크기가 압축된 비디오 데이터를 4분간 저장할 수 있는 용량으로 했을 때이고 (b)는 5분간 저장할 수 있는 용량으로 하였을 경우의 성능을 나타낸 것이다. 그림의 결과에서 확인할 수 있듯이 SRS 기법에 비해 배칭 기법을 통합한 SRS-BAT 기법이 더욱 효율적인 성능을 나타내고 있음을 알 수 있다. 이는 앞서 언급하였듯이 SRS는 클라이언트로부터 발생하는 요구에 효율적인 기법이고 배칭 기법은 큐에서 대기하고 있는 요구들에 대해 효율적인 기법이므로 이들의 통합은 더욱 효율적인 성능을 나타내게 되는 것이다. 그러나 버퍼의 크기가 커짐에 따라 성능의 차이가 심하지 않은 것을 확인할 수 있는데 이는 버퍼의 크기가 큰 경우에는 클라이언트가 요구들에 대한 서비스를 하는 경우가 많기 때문이다.

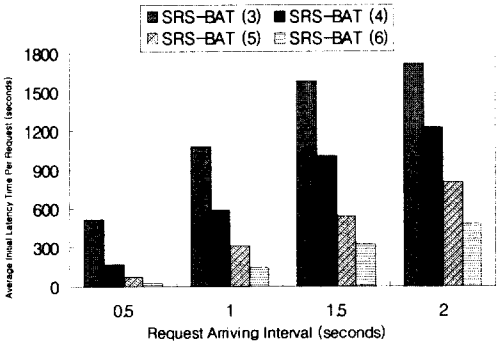


(a) 버퍼의 크기가 4분



(b) 버퍼의 크기가 5분

(그림 9) SRS와 SRS-BAT의 초기 지연 시간



(그림 10) SRS-BAT에서 클라이언트 버퍼 크기를 달리했을 경우의 초기 지연 시간

(그림 10)은 SRS-BAT에서 각 클라이언트 버퍼의 크기를 달리할 경우의 초기 지연시간을 나타낸 것이다. 그림에서 SRS-BAT (3)에서 3이란 클라이언트 버퍼 용량이 3분의 내용을 저장할 수 있다는 것을 의미한다. 클라이언트 버퍼의 크기가 커짐에 따라 성능이 좋아지는 것을 확인할 수 있다. 특히 (그림 9)에서 확인할 수 있듯이 SRS 기법에서는 클라이언트 버퍼 크기가 작은 경우에는 성능이 매우 저하되는 것을 확인할 수 있는데 SRS-BAT에서는 버퍼 크기가 작아지더라도 배칭 기법의 이점을 얻을 수 있기 때문에 큰 성능 저하를 나타내지 않는 것을 알 수 있다. 즉 SRS에 배칭 기법을 통합하면 클라이언트 버퍼 자원이 충분하지 않아도 SRS 기법에 비해 매우 효율적인 성능을 나타내는 것을 알 수 있다.

앞의 실험에서는 모든 클라이언트 버퍼의 크기가 동일하다는 전제하에 실험을 하였다. 만약 클라이언트 버퍼의 크기가 이질적인 경우에는 단순히 클라이언트로부터 비디오 요청시 버퍼 크기에 대한 정보만 추가적으로 받아 크기 정보를 관리테이블에 추가 저장하여 동작하면 된다.

5. 결 론

최근 시스템 환경의 발전으로 주문형 비디오 시스템 개념이 시선을 끌고 있다. 이러한 주문형 비디오 시스템에서 보다 많은 사용자들에게 서비스를 제공하기 위해 브릿징, 피기백킹, 배칭 등과 같은 요구 스케줄링 정책이 제안되었다. 본 논문에서도 한번의 디스크 검색으로 더욱 많은 사용자들의 요구를 서비스하는 스케줄링 기법을 제안하였다. 주문형 비디오 시스템에서는 일반적인 텍스트 데이터와는 달리 동일한 비디오에 대한 요구가 많이 발생하므로 특정 비디오에 대한 요구가 발생했을 때 해당 비디오의 내용이 다른 클라이언트의 버퍼에 유지되고 있을 확률이 높다. 본 논문에서 제안한 정책은 이러한 점을 활용하였다. 본 논문에서 제안한 정책에서는 요구가 발생했을 때 무조건 서버가 서비스를 제공하는 것이 아니라 해당 비디오를 저장하고 있는 클라이언트가 있는지를 검색해서 있을 경우에 작업을 클라이언트에게 넘겨 작동하게 하였다. 제안한 정책의 효과를 검증하기 위하여 시뮬레이션을 통해 SRS 기법과 배칭 기법의 성능 비교를 다각적으로 시행하였다. 성능 비교를 통해 본 논문에서 제안한 정책이 요구당 초기 지연시간과 서버의 처리량에 있어서 매우 효율적인 성능을 나타내는 것을 확인하였다. 특히 큐잉되었던 요구들을 처리하는데 배칭 기법을 SRS 기법에 적용하였을 경우에 더욱 효율적인 성능을 나타내는 것을 확인하였다.

본 논문에서 제안한 정책은 클라이언트의 버퍼만을 활용하는데 앞으로 서버의 버퍼도 함께 고려하여 동작하는 정책으로 확장할 것이다.

참 고 문 헌

[1] D. E. McDysan and D. L. Spohn, *ATM : Theory and Application*, McGraw-Hill, 1995.

[2] N. J. Boden et al., "Myrinet : A Gigabit-Per-Second Local Area Network," *IEEE Micro*, 15(1) : pp.29-36, February 1995.

[3] A. Heybey, M. Sullivan, and P. England, "Calliope: A Distributed, Scalable Multimedia Server," In *Proceedings of the USENIX 1996 Annual Technical Conference*, January 1996.

[4] D. Gemmell, H. vin, D. Kandlur, P. Rangan, "Multimedia Storage Servers : A Tutorial and Survey," *IEEE Computer*, 1995.

[5] L. Golubchik, J. Lui, and R. Munts, "Reducing I/O Demand in Video-On-Demand Storage Servers," *ACM Sigmetrics Conference*, pp.25-36, May 1995.

[6] A. Leff, J. Wolf, and P. Yu, "Efficient LRU-Based Buffering in a LAN Remote Caching Architecture,"

IEEE Transactions on Parallel and Distributed Systems, 7(2):191-206, February 1996.

[7] M. Dahlin, R. Wang, T. Anderson, and D. Patterson, "Cooperative Caching : Using remote client memory to improve file system performance," In *Proceedings of the First Symposium on Operating System Design and Implementation*, pp.267-280, November 1994.

[8] Mohan Kamath, Krithi Ramamritham, and Don Towsley, "Continuous Media Sharing in Multimedia Database Systems," In *Proceedings of the 4th International Conference on Database Systems for Advanced Applications*, pp.79-86, 1995.

[9] Asit Dan and Dinkar Sitaram, "A Generalized Interval Caching Policy for Mixed Interactive and Long Video Environments," In *IS&T SPIE Multimedia Computing and Networking*, San Jose, January 1996.

[10] C. Aggarwal, J. Wolf, and P. Yu, "On Optimal Piggyback Merging Policies for Video-on-Demand Systems," *Technical Report, IBM RC 20337*, February 1996.

[11] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," In *Proceedings of the 2nd ACM Multimedia Conference*, pp.25-32, 1994.

[12] H. Shachnai and P. Yu, "The Role of Wait Tolerance in Effective Batching : A paradigm for Multimedia Scheduling Schemes," *IBM Research Report, RC 20038*, 1995.

[13] S. Sheu, K. Hua, and W. Tavanapong, "Chaining : A Generalized Batching Technique for Video-On-Demand Systems," In Proceedings of the IEEE International Conference on Multimedia Computing and Systems, pp.110-117, June 3-6, 1997.

[14] Theodore Johnson and Dennis Shasha, "2Q : A Low Overhead High Performance Buffer Management Replacement Algorithm," In *Proceedings of the 20th VLDB Conference*, pp.439-450, 1994.

[15] Elizabeth J. O'Neil, Patrick E. O'Neil, and Gerhard Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp.279-306, May 1993.

[16] Ramakrishna Karedla, J. Spencer Love, and Bradley G. Wherry, "Caching Strategies to Improve Disk System Performance," IEEE Computer, 27(3) : 38-46, March 1994.

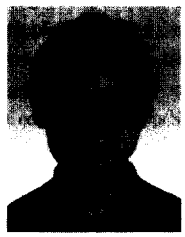
[17] Weifeng Shi and Shahram Ghandeharizadeh, "Trading Memory for Disk Bandwidth in Video-On-Demand Servers," In *Proceedings of 13th ACM Symposium on Applied Computing*, February 1998.

[18] T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R. Wang, "Serverless Network File System," *ACM Transactions on Computer Systems, 14(1) : 41-79*, February 1996.

[19] J. Hartman and J. Ousterhout, "The Zebra Striped Network File System," *ACM Transactions on Computer System*, 13(3) : 274-310, August 1995.

[20] P. Chen, E. Lee, G. Gibson, R. Katz, and D. Patterson, "RAID : High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, 26(2) : 145-185, June 1994.

[21] O. Rose, "Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modeling in ATM Systems," *University of Wurzburg Research Report Series No.101*, February 1995.



한 금 희

e-mail : hkumhee@www.cuk.ac.kr

1971년 성심여자대학교 화학과 (이학사)

1981년 미국 RPI 대학교 computer science(이학석사)

2000년 홍익대학교 대학원 전자계산학과 (이학박사)

1982년~현재 가톨릭대학교 컴퓨터공학부 교수

관심분야 : 멀티미디어 시스템, 프로그래밍 언어론, 컴파일러 이론

김 종 훈

e-mail : jkim@ns.cheju-e.ac.kr

1990년 목원대학교 수학교육과 (이학사)

1992년 동국대학교 대학원 통계학과 전산통계전공(이학석사)

1998년 홍익대학교 대학원 전자계산학과(이학박사)

1998년~1999년 홍익대학교 과학기술연구소 연구원

1998년~1999년 한국전자통신연구원(ETRI) 운영체제 연구팀 Post-Doc. 연구원

1999년~현재 제주교육대학교 컴퓨터교육과 전임강사

관심분야 : 웹 기반 교육, 컴퓨터 교육, 멀티미디어 시스템

원 유 헌

e-mail : won@cs.hongik.ac.kr

1972년 성균관대학교 수학과 (이학사)

1975년 한국과학원 전자계산학과 (이학석사)

1985년 고려대학교 수학과 전산전공 (이학박사)

1975년~1976년 한국과학기술연구소 연구원

1986년~1987년 R.P.I. 교환교수

1976년~현재 홍익대학교 컴퓨터공학과 교수

관심분야 : 프로그래밍 언어, 컴파일러 이론, 멀티미디어 시스템