비디오 서버를 위한 적응적 예약기반 피기백킹 알고리즘의 설계 및 평가

배 인 한† · 이 경 숙††

요 약

주문형 비디오 시스템의 성능에서 핵심적인 문제는 클라이언트 요청들을 만족시키기 위해 요구되는 입출력 대역폭이다. 공유를 통하여 비디오 서버의 입출력 요청을 감소시키는 다수의 방법들. 일괄처리, 브리징, 피기백킹이 사용되고 있다 피기백킹은 객체의 대응하는 입·출력 스트림들을 그룹으로 서비스할 수 있는 하나의 스트림으로 병합하기 위하여 진행중인 요청들의 디스플레이 율을 변경하는 정책이다. 본 논문에서는 인기 있는 비디오에 대한 요청들이 즉시 스케줄될 수 있도록 인기 있는 비디오를 위한 비디오 서버의 입출력 스트림 용량을 비디오 요청 도착율에 따라 동적으로 예약해 두는 적응적 예약기반 피기백킹이라는 새로운 기법을 제안한다 시뮬레이션을 통해 그것의 성능을 평가하고, 단순 피기백킹과 그 성능을 비교한다. 그 결과, 적응적 예약기반 피기백킹이 단순 피기백킹에 비해 더 나은 서비스 확률, 평균 대기 시간, 프레임 절약 백분율을 제공함을 알 수 있다.

Design and Evaluation of an Adaptive Reservation-Based Piggybacking Algorithm for Video Servers

Ihn-Han Bae† · Kyung-Sook Lee††

ABSTRACT

A critical issue in the performance of a video-on-demand system is the I/O bandwidth required in order to satisfy client requests. Several approaches: batching, bridging, piggybacking are used to reduce the I/O demand on the video server through sharing. Piggybacking is the policy for altering display rates of requests in progress for the same object, for the purpose of merging their corresponding I/O streams into a single stream, which can serve the entire group of merged requests. In this paper, we propose a new policy called an adaptive reservation-based piggybacking that dynamically reserves the I/O stream capacity of video server for popular videos according to video server loads to immediately schedule the requests for popular videos. The performance of the proposed policy is evaluated through simulations, and is compared with that of simple piggybacking. As the result, we know that the adaptive reservation-based piggybacking provides better service probability, average waiting time and percentage saving in frames than simple piggybacking.

※ 이 논문은 (1998년) 한국학술진흥재단의 학술연구비에 의하여 지원되었음

† 정 회 원 대구효성가톨릭대학교 컴퓨터정보통신공학부 교수
†† 준 회 원 · 대구효성가톨릭대학교 대학원 전산통계학과

논문접수 1999년 12월 28일, 심사완료 : 2000년 1월 28일

1. 서 론

최근 정보통신 분야의 기술적인 발전으로 주문형 비디오, 홈쇼핑 등과 같은 여러 가지 주문형 멀티미디어 시스템들이 실용화되고 있다. 오늘날의 정보 시스템은 단순히 커다란 멀티미디어 객체를 저장하고 검색하는 것뿐만 아니라 그 객체를 일정한 대역폭에서 계속적으로 제공하는 엄격한 실시간 요구사항을 만족시킬 수 있어야 한다. 멀티미디어 시스템은 교육용 응용, 오락 기술, 도서관 정보 시스템 등에서 중요한 역할을 하고 있으며, 이러한 시스템에서 가장 중요한 것은 다수의 클라이언트들에게 주문형 서비스를 동시에 제공하는 것이다. 즉, 사용자들은 비디오 등과 같은 객체를 요청하고, 적절한 지연 시간 내에 요청한 객체를 관람하기를 기대한다. 여기서 지연 시간은 요청이 도착한 시점부터 시스템이 디스크로부터 객체 읽기를 초기화하는데 까지 시간으로 정의되고, 데이터가 실제적으로 디스플레이 장치에 전달될 때까지의 추가적인 지연은 디스크 지연에 비해 상대적으로 작기 때문에 무시할 수 있다. 이러한 지연은 서비스 요청을 위한 불충분한 대역폭, 디스크로부터 판독 내용을 스케줄링 하기 위한 불충분한 버퍼 공간, 불충분한 디스크 기억장치 등의 요인으로 발생한다. 이러한 지연 요소 중에서 입출력 대역폭(I/O bandwidth)은 매우 중요한 자원이므로 공유를 통하여 기억장치 서버의 입출력 요청을 감소시켜 동시에 서비스할 수 있는 사용자 요청의 수를 증가시키는 다수의 접근 방법들이 다음과 같이 제안되고 있다[1, 2, 3, 4].

- 일괄처리(batching) : 일정한 일괄처리 윈도우 동안 도착하는 동일한 객체에 대한 요청들을 묶어서 단일 입출력 스트림으로 전체 그룹을 서비스하는 방법이다.
- 브리징(bridging) : 중앙 처리기의 메모리를 버퍼로 이용하는 방법이다. 특정 비디오에 대해 고정된 개수의 프레임이 버퍼링 된다면, 대응하는 시간 간격 내에 발생하는 비디오에 대한 어떤 요청은 디스크가 아닌 버퍼로부터 읽혀질 것이다. 그러나 브리징은 많은 양의 버퍼 공간을 요청하는 단점이 있다.
- 피기백킹(piggybacking) : 동일한 객체에 대한 입출력 스트림이 하나로 병합될 때까지 진행중인 스트림의 디스플레이 율을 조정하는 기법이다

피기백킹에서는 인기 있는 비디오들이 즉시 스케줄되어야 디스플레이 스트림들을 많이 병합할 수 있으므로 현재 비디오 요청들을 서비스하는데 사용되는 입출력 스트림들을 많이 절약할 수 있다. 따라서 같은 비디오 서버의 입출력 용량으로 가능한 한 많은 사용자들에게 주문형 비디오 서비스를 동시에 제공할 수 있다. 본 논문에서는 비디오 요청 도착율에 따라 인기 있는 비디오들을 위한 비디오 서버의 입출력 스트림 용량을 동적으로 예약해 두는 적응적 예약기반 피기백킹을 제안하고, 그것의 성능을 시뮬레이션을 통하여 평가하고, 단순 피기백킹과 그 성능을 비교한다. 본 논문의 구성은 다음과 같다 2장에서는 공유를 통한 저장 서버의 입출력 대역폭 감소에 대한 관련 연구를 살펴보고, 3장에서는 본 논문에서 제안하는 적응적 예약기반 피기백킹을 설명하고, 4장에서는 시뮬레이션을 통하여 제안하는 알고리즘의 성능을 평가하고, 그것의 성능을 단순 피기백킹과 비교한다. 그리고 마지막으로 5장에서 결론을 맺는다

2. 관련 연구

디스크 입출력 대역폭은 서비스의 지연 시간을 결정하는 중요한 자원이다 지연 시간을 줄이는 한가지 방법은 더 많은 디스크를 구매하는 것이다. 그러나 더 흥미롭고 경제적인 방법은 데이터 배치 기법(data layout techniques)과 스케줄링 기법을 향상시키거나 같은 객체에 대한 요청들간의 데이터 공유를 통하여 서비스 중인 각 요청의 입출력 요청을 감소시키는 것이다. 여기서는 공유를 통하여 저장 서버의 입출력 요청을 감소시켜 동시에 서비스되는 사용자 요청의 수를 증가시키는 방법을 설명한다.

2.1 일괄처리

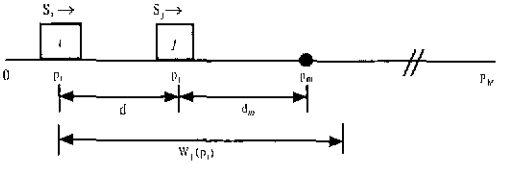
일괄처리는 같은 객체에 대한 요청들을 묶어서, 기억장치 서버에 대한 하나의 입출력 요청을 만드는 것이다. 일괄처리 방법에서 스트림 요청의 지연 시간과 기억장치 서버의 입출력 요청의 감소는 서로 상반관계에 있다 일괄처리 정책은 크기에 의한 일괄처리와 시간에 의한 일괄처리로 분류할 수 있다. 각 객체 i에 대해 미리 정해진 요청 개수 B_i를 일괄처리 윈도우라 할 때, 크기에 의한 일괄처리는 객체 i에 대해 B_i 만큼의 요청 개수가 누적되면 객체 i를 위해 입출력 스트림을 초기화한다. 따라서 객체 i에 대해 감소되는

입출력 스트림의 크기는 B_i-1이다. 시간에 의한 일괄처리는 시간 단위로 일괄처리 윈도우를 설정하는 것이다. 하나의 요청이 기억장치 서버에 도착하고 같은 객체 i에 대한 요청이 존재하지 않으면 타이머가 설정된다. 일괄처리 윈도우 B_i 동안 모아진 같은 객체 i에 대한 요청은 타이머가 만기되면 하나의 입출력 스트림으로 서비스된다[2].

좋은 비디오 스케줄링 정책은 일괄처리 윈도우뿐만 아니라 관람자의 이탈 확률과 대기 시간을 고려해야 한다. 일괄처리를 위한 두 가지 일반적인 스케줄링 정책은 가장 긴 대기 시간을 갖는 비디오 요청을 먼저 스케줄 하는 FCFS(First Come First Served) 정책과 최대 개수의 대기 요청을 갖는 비디오를 먼저 선택하는 MQL(Maximum Queue Length) 정책이 있다. MQL은 일괄처리 되는 요청의 개수를 최대로 하기 위하여 큐 길이만 고려함으로써 인기 있는 비디오의 스케줄링에 너무 적극적인 반면, FCFS는 이탈을 줄이기 위하여 도착 시간에 초점을 맞추고 큐 길이를 완전히 무시함으로써 MQL의 반대 효과를 가진다[5]. [6]에서는 factored queue length의 개념을 도입하여 maximum factored queue length를 갖는 비디오를 스케줄 하는 일괄처리 정책인 MFQ를 제안하였다. Factored queue length는 차별적인 가중 요소를 다른 비디오의 큐 길이에 적용함으로써 얻어진다. 여기서 factored queue length는 비디오의 큐 길이를 그것의 상대적 요청 회수의 제곱근으로 나눈 것으로 정의하였다. 그리고 MFQ, FCFS, MQL을 시뮬레이션한 결과, MFQ가 평균 지연 시간, 이탈 확률, 공평성에서 우수한 실험 결과를 보였다. [7]에서는 FCFS 정책을 확장한 FCFS-n 정책을 제안하였다. FCFS-n 정책에서는 서버 용량의 일부분이 n개의 인기 있는 비디오에 대한 요청들을 일괄처리하기 위하여 예약되고 선할당되어 진다. 인기 있는 비디오를 위해 서버 용량을 선할당 함으로써 인기 있는 비디오에 대해 초과될 수 없는 최대 대기 시간을 제공할 수 있고, 그리고 인기 없는 비디오에 대한 요청이 인기 있는 비디오에 대한 서비스에 간섭하지 않으므로 상대적으로 적은 서버 용량으로 높은 수용 확률을 보장하였다.

2.2 피기백킹

피기백킹(Piggybacking)이란 다수의 디스플레이 스트림을 하나의 입출력 스트림으로 서비스해 주는 방법으로 일괄처리는 정적 피기백킹이다. 동적 피기백킹 방법은 다음과 같이 수행된다. 요구에 따라 각 디스플레이 스트림에 대한 하나의 입출력 스트림이 초기화되고, 하나의 디스플레이 스트림을 같은 객체에 대한 다른 디스플레이 스트림의 입출력 스트림을 향하여 적응적으로 피기백되는 것을 허용한다. 이것은 두 개의 입출력 스트림을 하나로 병합하는 것으로 볼 수 있다. 병합 전에는 각각 한 개 이상의 디스플레이 스트림을 서비스하는 두 개의 입출력 스트림이 있다면, 병합 후에는 두 개의 디스플레이 스트림을 동시에 서비스하는 한개의 입출력 스트림이 있다. 이러한 병합은 요청의 디스플레이 속도를 조정함으로써 이루어질 수 있다. 즉, 각 요청에 대해 정상적인 속도로 디스플레이 하는 것이 아니라, 디스플레이들간의 간격을 좁히기 위하여 더 느린 율로 또는 더 빠른 율로 각 요청의 디스플레이 속도를 조절할 수 있다. 그렇게 하여 간격이 없어지면 두 개의 디스플레이 스트림들은 하나의 입출력 스트림으로 서비스될 수 있다. 즉, 두개의 입출력 스트림이 하나의 입출력 스트림으로 병합되는 것이다 이러한 디스플레이 속도 조정이 어떻게 이루어지는지를 간단하게 살펴보자. 우선 기억장치 서버에 의해 작동되는 디스플레이 단위는 NTSC(National Television System Committee) 표준인 30 프레임/초의 율(fps)로 디스플레이 한다고 가정한다 디스플레이 율의 가속과 감속은 비디오에 부가적인 프레임을 추가하고 삭제함으로써 가능하다. 그리고 정상율의 ±5% 정도의 디스플레이 율 변경은 관람자에 의해 인지되지 않는다. 따라서 실제 디스플레이 율을 5% 낮추거나 높이는 것은 비디오 품질을 손상시키지 않고 가능하다는 가정에 기초한다[1].



(그림 1) 시스템의 상태

(그림 1)은 전체 디스플레이 기간이 P_M인 객체의 디스플레이 상태를 나타낸다. 각 디스플레이 스트림은 그 객체의 디스플레이내의 그것의 위치로 식별된다. 예를 들어, 스트림 i는 p_i로 식별되고, 특별한 디스플

레이 속도 S_i로 이동한다. 입출력 스트림 i와 j를 병합하기 위하여, 먼저 $S_i > S_j$가 보장되어야 하고, 그 다음 병합 기회를 확인하기 위하여 피기백킹이 사용될 수 있는 거리 제한을 정의할 수 있다. 여기서 p_m은 입출력 스트림 i와 j가 병합되는 객체의 디스플레이 위치를, d는 입출력 스트림 i와 j간의 프레임 거리를, d_m은 병합 위치와 j의 현재 위치간의 프레임 거리를 나타낸다. 그리고 $W_g(p_i)$는 정책 g에 대한 앞선 스트림 i와 스트림 j간의 병합 가능한 최대 가능 거리인 격차 해소 윈도우(catch-up window)로 정의된다. $W_g(p_i)$는 객체의 디스플레이내의 위치 p_i에 비례하여 계산되어진다.

동적 피기백은 다음 네 가지 사건들 : 도착(arrival), 병합(merge), 종료(dropoff), 그리고 윈도우 횡단(window crossing) 중의 하나가 발생할 때 속도를 조정하는 정책들을 고려한다. 도착 사건은 새로운 입출력 스트림의 초기화에 해당하고, 병합 사건은 두 개의 입출력 스트림의 병합에 해당하고, 그리고 종료 사건은 어떤 객체의 디스플레이의 끝에 해당한다. 윈도우 교차 사건은 (그림 1)의 격차 해소 윈도우의 경계를 지나는 것을 나타낸다. 병합이 빠를수록 더 많은 자원이 다른 요청들을 서비스하기 위해 사용될 수 있다. 그러므로 일반 속도에서 최대 가능 편차를 가정한 다음 세가지 디스플레이을 : 가장 느린율(S_{min}), 정상율(S_n), 가장 빠른율(S_{max})만을 고려한다. 위와 같은 기본 개념을 바탕으로 제안된 여러 가지 동적 병합 정책들은 다음과 같다[1]

(1) 홀짝 감소 정책

홀짝 감소 정책은 최대 50% 정도의 입출력 요구를 감소시킨다. 이 정책은 연속적인 도착 스트림들을 병합하기 위하여 짝 지운다. 새로운 입출력 스트림이 시스템에 도착하고, 최대 격차 해소 윈도우 $W_{oe}(0)$내에 S_{min}의 속도로 앞서 가는 동일한 객체를 위한 입출력 스트림이 있을 때 그 새로운 요청의 속도를 S_{max}로 설정한다. 그러면 이 두 개의 스트림은 일정 시간 후에 병합되어진다

(2) 단순 병합 정책

단순 병합 정책은 객체 디스플레이의 시작에서 상대적으로 측정된 격차 해소 윈도우 $W_{sm}(0)$와 함께 어떤 객체의 디스플레이 시각에서 상대적으로 측정된 최대 병합 윈도우 $W_{sm}^{m}(0)$를 사용한다. $W_{sm}^{m}(0)$는 두 개의 스트림을 병합할 수 있는 가장 늦은 가능 지점을 나타낸다. 즉, 스트림 i가 시스템에 도착했을 때, $W_{sm}(0)$내에 스트림 i에 앞서 있는 스트림 j를 찾았다면, 스트림 i와 j는 $W_{sm}^{m}(0)$의 오른쪽 경계에서 병합될 것이다 단순 병합 정책의 기본 개념은 스트림들을 "병합 그룹"에 할당하는 것이다. 여기서 하나의 스트림 즉, 스트림 i가 그룹을 초기화하고, 스트림 i가 $W_{sm}(0)$에 있을 동안 시스템에 도착하는 모든 스트림들은 스트림 i에 병합된다 단순 병합 정책은 (그림 2)와 같다.

```
Algorithm Simple merging policy
begin
   Case arrival of stream i :
      If no stream within W_sm(0) is moving at S_max
         S_i = S_min ;
      else
         S_i = S_max ;
   Case merge of i and j
      drop stream i;
      S_j = S_min ;
   Case window crossing, W_sm^m(0)
         S_j = S_n;
end
```

(그림 2) 단순 병합 정책

(3) 탐욕 정책

탐욕 정책은 객체의 전체 디스플레이 기간 동안 가능한 많이 입출력 요청을 병합하는 정책이다. 탐욕정책은 초기 격차 해소 윈도우 $W_g(0)$와 함께 객체 디스플레이내의 위치 p_i에 대해 상대적으로 측정되는 격차 해소 윈도우 $W_g(p_i)$를 사용한다. 객체에 대한 요청이 도착했을 때 속도 조절은 홀짝 감소 정책과 같이 동작한다. 만약 격차 해소 윈도우를 지나쳤고 스트림이 병합을 위해 아직 짝 지어지지 않았을 때, 앞선 스트림과의 병합 가능성을 고려하기 위해 $W_g(W_g(0))$을 검사한다. 만약 p_i 위치에서 병합이 일어나면, 새로운 격차 해소 윈도우 $W_g(p_i)$가 계산된다 만일 그 윈도우내에 입출력 요청이 없으면, 그 요청의 속도는 S_n으로

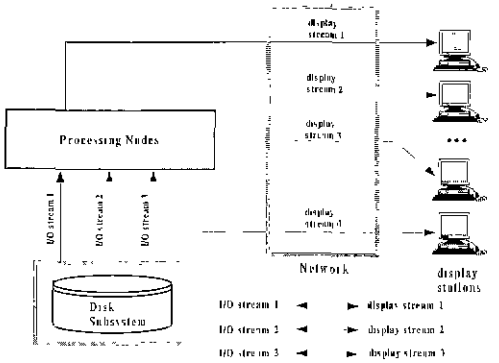
설정된다. 그러나, $W_p(p_i)$내에 약간의 요청이 있고, 바로 앞선 요청이 S_n의 속도로 디스플레이 되고 있다면, 그 요청의 속도는 S_{min}으로 설정되고, 위치 p_i의 요청의 속도는 S_{max}로 설정된다

3. 적응적 예약기반 피기백 병합 정책

본 논문에서 제안하는 적응적 예약기반 피기백 병합 정책은 기존의 동적 피기백 병합 정책에 인기 있는 비디오를 위한 비디오 서버의 입출력 스트림 예약이라는 개념을 도입한다. 기존의 피기백 병합 정책들은 도착하는 비디오 요청들에게 차례대로 비디어 서버의 입출력 스트림을 할당해 줌으로써 도착율이 어느 정도 이상 높아지면 모든 가용 입출력 스트림이 스케쥴 되어 도착한 모든 요청들이 무조건 기다려야 한다. 만일 이때 비디오 서버에 가용 입출력 스트림이 있어 인기 있는 비디오 요청을 스케쥴 하면, 격차 해소 윈도우 내에는 인기 있는 비디오에 대한 디스플레이 스트림이 항상 존재하므로 많은 디스플레이 스트림이 병합되어 비디오 서버의 입출력 스트림이 많이 감소하게 된다. 그렇지 않으면, 비디어 서버의 가용 입출력 스트림이 생길 때까지 기다려야 하므로 스트림을 감소시킬 수 있는 기회를 놓치게 된다. 따라서 본 논문에서는 이러한 단점을 해소하여 인기 있는 비디오의 경우에 앞선 디스플레이 스트림과의 계속적인 병합이 가능하도록 비디오 서버의 입출력 스트림 용량을 예약해 두는 적응적 예약기반 피기백 병합 정책을 제안한다.

3.1 시스템 모델

본 논문에서는 어떤 객체의 각 요청에 대해 하나의 디스플레이 스트림과 이에 대응하는 입출력 스트림이 존재하는 비디오 서버 시스템을 고려한다. 처리 노드들은 디스크들로부터 필요한 데이터를 검색하고, 다소의 방법으로 그 데이터를 수정 가능하고, 그리고 그 디스플레이 스트림들을 네트워크를 통하여 적절한 디스플레이 스테이션들에 전송하기 위하여 입출력 스트림들을 사용한다(예, (그림 3)에서 디스플레이 스트림 1과 2는 대응하는 입출력 스트림 1과 2를 사용하여 서비스된다). 저장 서버의 입출력 요구는 길은 객체에 대한 요청들에 대응하는 다수의 디스플레이 스트림들을 서비스하기 위하여 하나의 입출력 스트림을 사용함으로써 감소될 수 있다(예, (그림 3)에서 같은 객체에 대한 요청에 대응하는 디스플레이 스트림 3과 4는 하나의 입출력 스트림을 사용하여 서비스된다). 그리고 저장 서버는 요청의 디스플레이 율을 동적으로 변경할 수 있는 기능, 더 정확하게 말하면 어떤 객체의 디스플레이의 다소의 부분을 동적으로 시간 압축 또는 시간 확장하는 기능을 갖는다고 가정한다



(그림 3) 시스템의 개략적인 구조

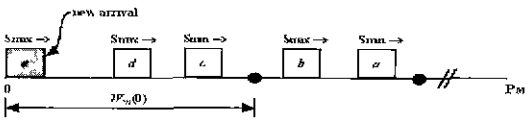
3.2 예약기반 단순 병합 정책의 시나리오

인기 있는 k개의 비디오를 서비스하기 위해 비디오 서버 입출력 스트림을 예약하는 적응적 예약기반 피기백 병합 정책을 *피기백킹-k*라 한다. 이러한 예약기반 피기백 병합 정책은 여러 동적 피기백 병합 정책들에 적용 가능하지만, [1]의 성능 평가 결과, 동적 피기백 병합 정책들의 성능은 거의 비슷하였고, 실험을 간단히 하기 위해 일반화된 단순 피기백 병합 정책[8]에 예약기반 피기백 병합 정책을 적용하였다. 일반화된 단순 피기백 병합 정책은 디스플레이 속도 S_{min}과 S_{max}만을 고려하고, 최대 격차 해소 윈도우 크기는 식 (1)과 같이 계산될 수 있다 여기서 L은 비디오의 프레임 길이이다.

$$W_n = \frac{S_{max} - S_{min}}{S_{max}} \cdot L \qquad (1)$$

우리는 윈도우 크기 W에 대한 단순화된 병합정책을 고려한다. W는 프레임으로 측정되는 매개변수이다 여기서 $0 \le W \le W_n$일 것을 요구한다 새로 도착하는 스트림은 그것의 W내에 늦은 스트림이 존재하면 빠른 스트림으로 설정되고, 그렇지 않으면 그 스트림은 늦

은 스트림으로 설정된다. 만일 이떤 빠른 스트림이 늦은 스트림과 병합되면, 빠른 스트림은 없어지고 늦은 스트림은 계속 진행한다



(그림 4) 예약기반 단순 병합 정책의 시나리오

*피기백킹-k*에서 한가지 가능한 시나리오는 (그림 4)와 같다. 입출력 스트림 *c*가 시스템에 도착했을 때, 입출력 스트림 *a*는 격차 해소 윈도우 *W* 밖으로 이미 이동하였으므로 입출력 스트림 *c*의 디스플레이 속도는 S_{min}으로 설정된다. 스트림 *b* (스트림 *d*)가 시스템에 도착했을 때, 스트림 *a* (스트림 *c*)는 격차 해소 윈도우 *W* 내에 있으므로 그것의 디스플레이 속도는 S_{max}로 설정된다. 스트림 *e*가 시스템에 도착했을 때, 그 스트림이 스케줄되면, 스트림 *e*, *d*, *c*는 하나의 입출력 스트림으로 병합될 수 있지만, 비디오 서버의 가용 입출력 스트림 용량이 없어 스트림 *e*를 스케줄할 수 없으면, 스트림 *c*와 *d*만 하나의 입출력 스트림으로 병합되므로 한 개의 입출력 스트림을 감소시킬 수 있는 기회를 놓치게 되는 것이다. 이 시나리오에서 스트림 *b*는 스트림 *a*와 병합되고, 저장 서버의 가용 입출력 스트림이 있으면 스트림 *d*와 *e*는 스트림 *c*와 병합되고, 아니면 스트림 *d*만 스트림 *c*와 병합된다 따라서, 인기 있는 비디오의 디스플레이 스트림은 항상 *W* 내에 존재하므로 인기 있는 비디오를 위해 비디오 서버의 입출력 스트림 용량을 예약하여 인기 있는 비디오 요청들이 즉시 스케줄 되도록 함으로써 가능한 한 많은 디스플레이 스트림을 병합시켜 비디오 서버 입출력 스트림이 많이 절약되도록 한다.

3.3 적응적 예약기반 피기백킹 알고리즘

k-피기백킹에서 중요한 매개변수는 인기 있는 비디오를 위해 예약되는 비디오 서버의 입출력 스트림 용량이다 본 논문에서는 비디오 서버의 입출력 스트림 예약 용량이 시스템 부하에 따라 동적으로 변하는 적응적 예약기반 피기백킹 알고리즘을 설계한다. 적응적 예약기반 피기백킹에서, 비디오 서버가 경부하인 경우에는 인기 있는 비디오 요청들이 항상 스케줄 될 수 있을 만큼의 비디오 서버의 입출력 스트림 용량을 예약한다. 그러나 비디오 서버가 과부하인 경우, 인기 있는 비디오를 항상 스케줄할 수 있을 만큼의 비디오 서버의 입출력 스트림 용량을 예약하면, 인기 없는 비디오를 위해 할당되는 입출력 스트림 용량이 거의 없어, 인기 없는 비디오 요청의 대기 시간이 아주 길어져 서비스를 포기하는 서비스 이탈 확률이 높아진다 따라서 인기 있는 비디오를 위해 비디오 서버의 적절한 입출력 스트림 용량이 예약되어야 한다. *k*개의 인기 있는 비디오를 위한 비디오 서버의 입출력 스트림 예약 용량은 다음과 같다.

$$SC_{r,min} = SC \times \sum_{i=1}^{k} Prob_i$$

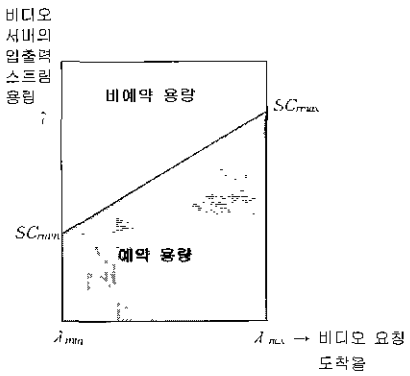
$$SC_{r,max} = \omega \times SC_{r,min}$$

$$SC_r = \begin{cases} SC_{r,min} & \lambda < \lambda_{min} \\ \dfrac{SC_{r,max} - SC_{r,min}}{\lambda_{max} - \lambda_{min}}(\lambda - \lambda_{min}) + SC_{r,min} & \lambda_{min} < \lambda < \lambda_{max} \\ SC_{r,max} & \lambda > \lambda_{max} \end{cases} \quad (2)$$

$$SC_{nr} = SC - SC_r$$

여기서 $Prob_i$는 *i*-번째 인기 있는 비디오의 인기도, *SC*는 비디오 서버의 전체 입출력 스트림 용량, $SC_{r,min}$은 입출력 스트림의 최소 예약 용량, $SC_{r,max}$은 입출력 스트림의 최대 예약 용량 (단, $SC_{r,max} \le SC$이다), SC_r은 *k*개의 인기 있는 비디오를 위해 예약되는 비디오 서버의 입출력 스트림 용량, SC_{nr}은 비예약 용량, 그리고 ω는 가중치를 나타낸다. 적응적 예약기반 피기백킹인 경우, 인기 있는 비디오를 위한 비디오 서버의 입출력 스트림 예약 용량을 도식화하면 (그림 5)와 같다

피기백킹-*k*의 구현 방법은 다음과 같다 시스템에 도착하는 비디오 요청들은 큐에서 FCFS 순서로 관리된다. 번지 주기적으로 수식 (2)를 사용하여 현재 주문형 비디오 시스템의 부하에 따라 인기 있는 비디오를 위한 비디오 서버의 입출력 스트림을 예약한다. 그리고 큐의 선두에 있는 비디오 요청이 인기 있는 비디오이면 비디오 서버의 예약 입출력 스트림을 할당하여 그 스트림을 스케줄하고, 인기 없는 비디오이면 비디오 서버의 비예약 입출력 스트림을 할당하여 그 스트림을 스케줄한다. 또한 인기 있는 비디오의 스트림이



(그림 5) 인기 있는 비디오를 위한 비디오서버의 입출력 스트림 예약 용량

병합, 혹은 완료되어 스트림이 해제되면, 그 스트림은 비디오 서버의 입출력 스트림의 예약 용량으로, 인기 없는 비디오의 경우 비디오 서버의 입출력 스트림의 비예약 용량에 반환된다. 본 논문에서 제안하는 *피기백킹-k* 알고리즘의 구조는 (그림 6)과 같다

```
Algorithm Adaptive Reservation-based Simple
Merging Policy(k)
  Begin
    Reserve the I/O stream capacity of video
    server for k-hottest
      videos according to arrival rate of video
      requests;
    Case arrival of stream i:
      If stream i is in the k-hottest streams
        If the reserved stream capacity of video
        server is
          not empty
          If no stream within W is moving at S_min
            S_i = S_min;
          Else
            S_i = S_max;
        Else
          Block the video request;
      Else
        If the unreserved stream capacity of
        video server is
          not empty
          If no stream within W is moving at S_min
            S_i = S_min;
          Else
            S_i = S_max;
        Else
          Block the video request;
    Case merge of i and j:
      Drop stream i;
      S_j = S_min;
  End
```

(그림 6) 적응적 예약기반 단순 피기백 병합 정책

4. 시뮬레이션 및 평가

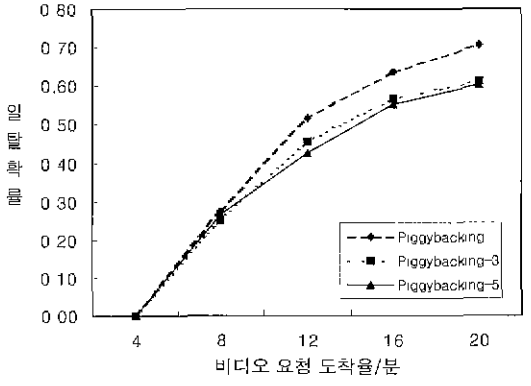
본 논문에서 제안하는 예약기반 단순 병합 정책의 성능을 시뮬레이션을 통하여 평가하고, 그 결과를 일반화된 단순 병합 정책의 성능과 비교 분석한다. 시뮬레이션에 사용된 매개변수는 표 1과 같다. 평가를 위해 단순 피기백킹인 *piggybacking*, 3개의 인기 있는 비디오에 대해 비디오 서버의 입출력 스트림 용량을 예약하는 예약기반 단순 피기백킹인 *piggybacking-3* 그리고 5개의 인기 있는 비디오에 대해 비디오 서버의 입출력 스트림 용량을 예약하는 예약기반 단순 피기백킹인 *piggybacking-5*를 고려하고, 평가 항목은 비디오 요청 도착율에 따른 관람자의 이탈 확률, 관람자의 평균 대기 시간 그리고 프레임 절약 백분율이다. 여기서 i-번째 인기 있는 비디오의 인기도는 Zipf의 법칙[9]을 따른다고 가정한다. x 단위 시간 떨어진 디스플레이 스트림들이 병합되는 시점은 $t_m = \frac{xS_{min}}{S_{max} - S_{min}}$으로 계산된다. 그리고 시뮬레이션의 공정성을 기하기 위해 서비스를 시작한 후 300분~400분 사이의 실험 결과로 평가하였다.

〈표 1〉 시뮬레이션 매개변수

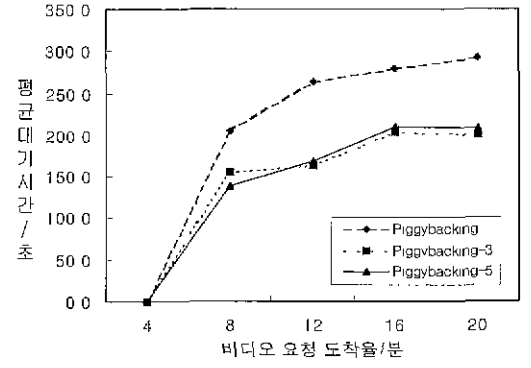
매개변수	값
비디오의 개수	60 개
비디오 서버의 입출력 스트림 용량 (SC)	400
비디오 요청 도착율 (λ)	4, 8, 12, 16, 20 요청/분 (포아송)
느린 디스플레이 율 (S_{min})	28.5 프레임/초
빠른 디스플레이 율 (S_{max})	31.5 프레임/초
영화의 길이 (L)	100 분
비디오 서버의 최대 입출력 스트림 예약을 위한 가중치 (ω)	2
최대 격차 해소 윈도우(W_m)	9.5 분
서비스 이탈 시간	3~6분 (랜덤)
시뮬레이션 시간	500 분

(그림 7)은 비디오 요청 도착율에 따른 서비스의 이탈 확률을 나타낸다. 여기서 이탈 확률은 서비스 이탈 회수를 전체 비디오 요청 회수로 나눈 것이다. 비디오 요청 도착율이 낮을 경우에 *piggbacking*과 *piggybacking-3*, *piggybacking-5*의 이탈 확률은 거의 비슷하지만 비디오 요청 도착율이 높아질수록 단순 피기백킹과 예약기반 피기백킹의 이탈 확률의 차이가 커짐을

알 수 있다. *piggybacking-3*과 *piggybacking-5*의 경우 인기 있는 비디오를 위해 비디오 서버의 입출력 스트림이 예약되어 있으므로 자주 발생하는 인기 있는 비디오에 대한 요청이 *piggybacking*에 비해 이탈 이전에 훨씬 많이 서비스 될 수 있다. 물론 상대적으로 인기 없는 비디오의 요청은 *piggybacking* 보다 높은 이탈 확률을 보이겠지만 전체적으로 이탈 확률이 낮아지므로 비디오 요청 서비스 확률은 높아진다는 것을 알 수 있다.

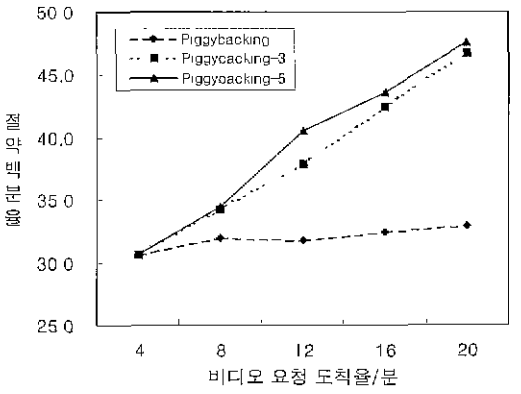


(그림 7) 비디오 요청 도착율에 따른 서비스 이탈 확률



(그림 8) 비디오 요청 도착율에 따른 평균 대기 시간

(그림 8)은 비디오 요청 도착율에 따른 서비스의 평균 대기 시간의 변화를 보여준다. 여기서 평균 대기 시간은 서비스된 비디오의 대기 시간의 합을 총 서비스된 비디오의 개수로 나누어 계산하였다. *Piggybacking*의 평균 대기 시간이 *piggybacking-3*과 *piggybacking-5*에 비해 최대 1.7분 (102초) 이상 높음을 알 수 있다. 이는 예약 기반 피기백킹이 인기 있는 비디오를 위해 비디오 서버의 입출력 스트림을 예약함으로써 단순 피기백킹에 비해 많은 디스플레이 스트림들의 병합이 발생하고 이로 인하여 완료 이전에 해제되는 입출력 스트림이 많기 때문이다. 따라서 비디오 서버의 가용 입출력 스트림의 개수가 단순 피기백킹에 비해 많아지므로 서비스의 평균 대기 시간은 줄어들게 된다.



(그림 9) 비디오 요청 도착율에 따른 프레임 절약 백분율

(그림 9)는 단순 피기백킹과 예약기반 피기백킹의 프레임 절약 백분율의 차이를 보여준다 프레임 절약 백분율은 절약된 프레임의 합을 전체 처리된 프레임의 합으로 나눈 백분율이다. 여기서 전체 처리된 프레임의 합은 (*서비스 된 비디오의 개수×한 비디오의 프레임 개수*)로서 한 비디오의 프레임 개수는 18,000 (=100×60×30) 프레임이다. *Piggybacking*의 경우 비디오 요청 도착율이 증가하더라도 프레임 절약 백분율에는 별 차이가 없음을 알 수 있다. 단순 피기백킹은 비디오 요청 도착율이 낮을 경우, 병합 가능한 스트림의 도착이 자주 발생하지 않으므로 병합 가능성이 낮고, 비디오 요청 도착율이 높을 경우, 많은 스트림 요청들이 도착하지만 전체 비디오가 하나의 큐를 통해 FCFS 정책으로 관리되므로 인기 있는 비디오 요청이 도착해도 비디오 서버의 가용 입출력 스트림이 거의 없는 상태이므로 서비스 이탈 이전에 스케줄을 받기가 힘들다 따라서 비디오 요청이 많이 발생할지라도 병합 확률이 높아지지 않는다 그러므로 비디오 요청 도착율에 상

관없이 *piggybacking*의 프레임 절약 백분율이 31~33% 정도의 일정한 수준을 유지한다. 그러나 *piggybacking-3*과 *piggybacking-5*의 경우 낮은 비디오 요청 도착율에서는 프레임 절약 백분율이 *piggybacking*과 비슷하지만, 높은 비디오 요청 도착율에서는 인기 비디오를 위해 비디오 서버의 입출력 스트림이 예약되어있는 상태이므로 디스플레이 스트림의 병합 가능성이 *piggybacking* 보다 훨씬 높아진다. 그러므로 *piggybacking-3*과 *piggybacking-5*는 비디오 요청 도착율이 높을수록 높은 프레임 절약 백분율을 얻을 수 있다. 결국, 같은 입출력 스트림 용량을 갖는 비디오 서버의 경우, 단순 피기백킹 보다 예약기반 피기백킹을 사용하는 것이 더 많은 사용자들에게 주문형 비디오 서비스를 동시에 제공할 수 있음을 알 수 있다.

5. 결 론

본 논문에서는 인기 있는 비디오를 위해 비디오 서버의 입출력 스트림 용량을 별도로 예약해 두는 예약기반 피기백 병합 정책을 제안하였고, 시뮬레이션을 통하여 그것의 성능을 서비스 이탈 확률, 서비스 평균 대기 시간, 프레임 절약 백분율로 평가하였고, 그리고 그 결과를 기존의 일반화된 단순 피기백 병합 정책과 비교하였다. 비디오 서버의 부하가 낮을 때는 예약기반 피기백킹과 단순 피기백킹 간에 성능 차이가 거의 없으나 비디오 서버의 부하가 높아질수록 성능 차이가 많음을 알 수 있었다. 예약기반 피기백킹의 경우, 비디오 서버의 부하가 많아질수록 빈번하게 발생하는 인기 있는 비디오 요청에 대한 디스플레이 스트림의 병합 가능성이 단순 피기백킹에 비해 월등히 높아지므로 비디오 서버의 가용 입출력 스트림 개수가 많아지게 된다. 그러므로 예약기반 피기백 병합 정책은 비디오 요청 도착율이 증가할수록 단순 피기백 병합 정책에 비해 전체적으로 사용자의 서비스 이탈 확률이 작아지고, 서비스 평균 대기 시간이 짧아지며, 프레임 절약 백분율이 높아지게 된다. 따라서 예약기반 병합 정책은 주문형 비디오 시스템의 성능 뿐만 아니라 관람자의 QoS도 향상시키는 우수한 주문형 비디오 서버의 입출력 대역폭 감소 알고리즘임을 확인할 수 있었다. 아울러 *piggybacking-3*과 *piggybacking-5*가 비슷한 성능을 제공하고, 많은 개수의 인기 있는 비디오에 대해 비디오 서버의 입출력 스트림을 예약하면 인기 있는 비디오의 스케줄링에는 적극적인 반면 인기 없는 비디오에 대한 스케줄링에는 소극적이다. 따라서 작은 개수의 인기 있는 비디오에 대한 비디오 서버의 입출력 스트림을 예약함으로써 인기 있는 비디오 요청과 인기 없는 비디오 요청을 적절히 스케줄하여 주문형 비디오 시스템의 QoS와 성능을 모두 향상시킬 수 있음을 알 수 있었다. 향후 연구 내용은 다양한 시뮬레이션 환경하에서 다양한 입출력 대역폭 감소 정책들간의 성능 평가, 그 정책들간의 QoS에 관한 평가, 그리고 피기백킹과 일괄처리가 혼합된 예약기반 하이브리드 정책에 관한 연구 등이다.

참 고 문 헌

[1] L. Golubchik, J. Lui, and R. Munts "Reducing I/O Demand in Video-On-Demand Storage Servers," *In Proceedings of ACM Sigmetrics '95*, pp.25-36, May 1995.

[2] 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

[3] M. Kamath, K. Ramamritham, and D Towsley, "Continuous Media Sharing in Multimedia Database Systems," Department of Computer Science, University of Massachusetts, Technical Report 94-11, Feb 1994.

[4] A. Dan, D. M. Dias, R. Mukherjee, and D. Sitaram, "Buffering and Caching in Large-Scale Video Servers," IBM Research Division, Technical Report RC 19903, Jan. 1995.

[5] H. Shachnai and P. S Yu, "An Analytical Study of Multimedia Batching Schemes," IBM Research Division, Technical Report RC 20662, Dec. 1996.

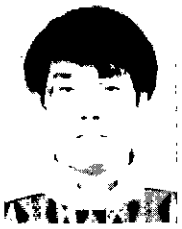
[6] C. C. Aggarwal, J L. Wolf, and P. S Yu, "The maximum Factor Queue Length Batching Scheme for Video-on-Demand Systems," IBM Research Division, Technical Report RC 20621, Nov. 1996.

[7] A. Dan, D Sitram, and P. Shahabuddin, "Dynamic Batching Polices for an On-Demand Video Server," *Multimedia Systems*, Vo.4, No.2, pp.112-121, June 1996.

[8] C Arrarwal, J. Wolf, and P. S. Yu, "On Optimal Piggyback Merging Polices for Video-On-Demand Systems," IBM Research Division, Technical Report RC 20337, Feb. 1996.

[9] G K. Zipf, *Human Behavior and the Principles of Least Effort*, Addison-Wesley, 1949.

[10] H. Shachnai, P S Yu, "Exploring wait tolerance in effective batching for video-on-demand scheduling," *Multimedia Systems*, Vol 6, No.6, Nov. 1998.



배 인 한

e-mail : ihbae@cuth.cataegu.ac.kr

1984년 경남대학교 전자계산학과 (학사)

1986년 중앙대학교 대학원 전자계산학과(석사)

1990년 중앙대학교 대학원 전자계산학과(박사)

1996년~1997년 Dept. of Computer and Information Science, The Ohio State University (Postdoc)

1989년~현재 대구효성가톨릭대학교 컴퓨터정보통신공학부 교수

관심분야 : 인터넷, 멀티미디어 시스템, 이동 컴퓨팅 시스템, 분산 시스템 등

이 경 숙

e-mail : g6721001@cuth.cataegu.ac.kr

1990년 대구효성가톨릭대학교 수학교육학과 (학사)

1993년 대구효성가톨릭대학교 대학원 전산통계학과 (석사)

1998년 대구효성가톨릭대학교 대학원 전산통계학과 (박사과정 수료)

관심분야 : 인터넷, 멀티미디어 시스템, 분산 시스템 등