공학석사학위논문

대용량 미디어 스트리밍 서비스를 위한 선호도 기반의 버퍼 관리 기법

방 철 석

강원대학교대학원

컴퓨터정보통신공학과

2004년 8월

정 인 범 교 수 지 도
공 학 석 사 학 위 논 문

대용량 미디어 스트리밍 서비스를 위한 선호도 기반의 버퍼 관리 기법

A Preference-Based Buffer Management for Large-Volume Media Streaming Service

강 원 대 학 교 대 학 원

컴퓨터정보통신공학과

방 철 석

방철석의 석사 학위논문을
합격으로 판정함

2004년 6월

심사위원장 정 인 범 인

위 원 최 창 열 인

위 원 이 구 연 인

대용량 미디어 스트리밍 서비스를 위한 선호도 기반의 버퍼 관리 기법

방 철 석

강원대학교 대학원 컴퓨터 정보통신공학과

미디어 스트리밍 서비스를 이용하면 사용자들은 아무 때나 자신이 원하는 영화를 볼 수 있다. 사용자에게 제공되는 편리성과 네트워크 환경의 발달에 힘입어 고화질의 미디어 데이터를 실시간으로 전송하는 미디어 스트리밍 서비스에 대한 요구가 증가하고 있다. 이런 서비스는 특정시간에 그 이용이 집중되기 때문에 모든 사용자에게 좋은 품질의 서비스를 제공하기 위해서 서버의 많은 자원을 필요로 한다. 서버의 자원은 한정되어 있기 때문에 이 한정된 자원을 효과적으로 이용하기 위한 방법이 필요하다.

서버의 여러 가지 자원 중에서 디스크에 비해 비싸고 용량이 적은 메인 메모리는 대용량의 미디어 데이터를 다루는 미디어 스트리밍 서버 성능에 중요한 역할을 한다. 그러므로 제한된 메인 메모리를 효과적으로 이용할 수 있는 방법이 필요하다. 그 대표적인 방법이 미디어 서버 내 버퍼 히트 율을 높이는 것이고 이를 위해 가장 중요한 것이 효과적인 버퍼 관리 정책을 선택하는 것이다.

본 논문에서는 미디어 스트리밍 환경에서 나타나는 특성을 알아보고 특성을 잘 반영한 미디어 스트리밍 서버를 구현하여 그 성능을 보여준다. 버퍼의 대체가 정적으로 이루어지는 메모리 관리 기법과 능동적으로 버퍼 대체를 수행하는 동적 버퍼 관리 기법에 대해 소개한다. 그리고 효과적인 미디어 스트리밍 서비스를 위해 버퍼의 히트 율을 높일 수 있는 버퍼 대체 정책을 제안한다. 또한 성능 평가를 통하여 사용자 선호도와 데이터 접근 특성을 반영한 대체 정책이 서버 시스템의 성능 향상에 기여함을 보인다.

목 차

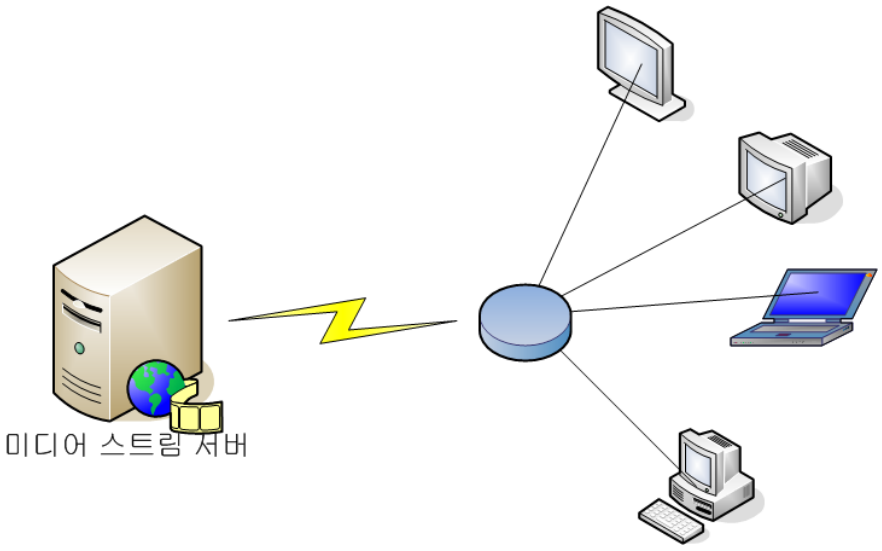
표 목 차

그림목차

Ⅰ. 서　론

컴퓨터 기술의 발달에 힘입어 사용자의 컴퓨팅 환경이 화려한 그래픽 환경으로 변하였다. 이제 검은 바탕에 하얀 텍스트 화면은 일부 개발자들의 전유물이 되어 버렸다. 많은 사용자들은 좀더 멋진 그래픽 환경과 함께 고화질의 미디어 데이터를 요구하게 되었다. 이런 요구를 충족시키기 위해 많은 고화질의 멀티미디어 데이터들이 만들어지고 있고 이를 위한 애플리케이션도 많이 개발되고 있다.

일반적으로 미디어 데이터를 이용하기 위해서는 디스크에 저장해야 한다. 비교적 용량이 큰 미디어 데이터를 디스크에 저장하는 일은 사용자에게 많은 불편함을 주게 된다. 고화질 미디어 데이터에 대한 사용자의 요구와 일일이 디스크에 저장하는 번거로움을 동시에 충족하는 것이 미디어 스트리밍 서비스이다. 사용자의 요구 증가와 네트워크 환경의 발달에 힘입어 미디어 스트리밍 서비스에 대한 관심이 증가하고 있다.



[그림 1] 미디어 스트리밍 서비스

미디어 스트리밍 서비스를 이용하면 원하는 시간에 원하는 영화를 볼 수 있다. 또한 현재 재생에 필요한 데이터만을 그때그때 네트워크를 통해 받음으로써 대용량의 데이터를 일일이 디스크에 저장하는 번거로움을 피할 수 있다. 사용자가 늘어남에 따라 모든 사용자에게 좋은 품질의 서비스를 하기 위해서는 서버의 많은 자원이 필요하다. 그러나 서버의 자원은 한정되어 있기 때문에 한정된 자원을 효과적으로 이용하기 위한 방법이 필요하다.

서버의 자원을 크게 메모리, 네트워크, CPU, 디스크로 나누어 볼 때 각각의 자원을 효과적으로 사용하는 것은 전체적인 성능에 큰 영향을 미친다. 그 중에서도 디스크에 비해 그 가격이 비싸고 용량이 적은 메인 메모리는 대용량의 미디어 데이터를 다루는 미디어 스트리밍 서버 성능에 중요한 역할을 하고 있다. 그러므로 제한된 메인 메모리를 효과적으로 이용할 수 있는 방법이 필요하다. 그 대표적인 방법이 미디어 서버 내 버퍼 히트율을 높여 재사용하는 것이다. 버퍼 내 히트율을 높이는 위해서는 효과적인 버퍼 관리 정책이 필요하고 이는 서버의 성능향상에도 중요한 요인이 된다.

효과적인 버퍼 관리를 위해 미디어 스트리밍 서버에서 먼저 고려되어야 하는 것이 사용자의 선호도이다. 스트리밍 서비스에서는 사람들의 요구가 대부분 비슷한 시간대에 이루어지고 그 대상 또한 한정되는 특성이 있다. 다수의 미디어 데이터를 저장하고 있더라도 대부분의 서비스는 현재 인기가 있는 몇 가지로 한정된다. 특정 시간대에 몇몇 미디어들에게 서비스가 집중되기 때문에 미디어 서버의 성능 향상을 위해서는 선호도를 고려해야 한다. 다음으로 고려되어야 하는 것이 미디어 데이터에 대한 접근 특성이다. 대용량의 데이터에 대해 순차적이고 반복적인 접근이 이루어진다. 이런 특성을 반영하여 버퍼 관리를 수행하면 효과적인 미디어 스트리밍 서비스를 할 수 있다.

본 논문에서는 사용자의 요구 선호도와 미이어 데이터에 대한 접근 특성을 연구하고 이런 특성을 잘 반영한 미디어 스트리밍 서버를 구현하여 그 성능을

측정하였다. 먼저 버퍼의 대체가 수동적으로 이루어지는 정적인 환경에서의 메모리 관리 기법에 대해 연구한다. 미디어 서버로 클러스터 병렬 시스템을 구현하여 성능 평가를 해 보고 이런 환경에서 발생하는 문제점에 대해 알아본다. 본 논문에서는 이런 정적인 환경에서의 문제점을 개선 할 수 있는 동적 버퍼 관리 기법을 기반으로 한 효과적인 미디어 스트리밍 서비스를 위한 버퍼 대체 정책을 제안한다. 성능 평가를 통하여 사용자 선호도와 데이터 접근 특성을 반영하기 위해 본 논문에서 제안한 MSS(Minimum Service Segment) 기법이 클러스터 미디어 서버의 성능향상에 기여함을 보인다.

논문의 구성은 다음과 같다. 2장에서는 미디어 스트리밍 서버의 버퍼관리와 관련된 기존의 연구에 대해서 알아보고 3장에서는 정적 버퍼 관리 기법으로 본 논문에서 제안한 미디어 스트림 가속기에 대한 성능 측정의 결과를 제시한다. 4장에서는 동적 버퍼 관리 기법으로 본 논문에서 제안하는 MSS 대체 정책과 이에 대한 성능을 평가 한다. 마지막으로 5장에서는 결론 및 향후 연구 과제에 대해 기술한다.

Ⅱ. 관련연구

효과적인 버퍼 관리는 스트리밍 서비스에만 국한 되는 문제가 아니다. 운영체제, 데이터베이스, 웹 서비스 등 다른 많은 분야에서도 이는 중요한 문제이다. 이번 장에서는 사용자의 선호도가 어떤 분포를 가지며 선호도를 반영하는 것이 얼마나 효과적인지에 대하여 구현한다. 또한 서비스 구현 시 사용하는 버퍼 모델과 기존의 버퍼 대체 정책, 미디어 데이터에 대한 접근 특성을 고려한 버퍼 관리 기법들에 대한 연구에 대해 알아본다.

2.1. 선호도

다수의 사용자가 많은 종류의 미디어 서비스를 이용하고 있다. 어떤 서비스를 이용하느냐에 따라 차이는 있지만 각각에는 선호도가 존재한다. 선호도가 짧은 시간 집중되는 대표적인 예가 뉴스이다. 뉴스는 대부분의 이용자가 가장 최근의 소식에 관심을 가지기 때문에 최근 하루 혹은 최근 몇 시간 동안의 내용에 대한 선호도가 매우 높다. 웹 서버의 경우에도 많은 웹 페이지 중에서 사용자들이 자주 이용하는 페이지는 몇몇으로 한정된다. 본 논문에서 다루는 미디어 스트리밍 서비스에도 선호도에 대한 특성이 나타난다. 영화 데이터는 뉴스보다는 그 주기가 길지만 최근에 개봉한 영화 혹은 인기영화에 대한 선호도가 다른 영화에 비해 매우 높다.

미디어 데이터를 다루는데 있어서 선호도는 매우 중요하다. 미디어 데이터는 용량이 크기 때문에 디스크에 비해 용량이 적은 메모리의 효과적인 사용이 필요하다. 이때 선호도를 반영하는 버퍼 대체 정책을 이용하면 버퍼 재사용율이 증가되므로 적은 메모리로 많은 클라이언트를 수용 할 수 있다.

선호도가 중요하기 때문에 어떤 형태로 선호도가 분포하는지에 대한 연구가 있었다. 기존연구에서 선호도에 대한 조사는 오프라인에서 비디오 대여점의 대여현황을 이용하였다[10]. 실제 대여 현황을 이용하여 선호도를 조사해본 결과는 zipf 분포로 나타나고 본 논문에서도 실험에 zipf 분포를 이용하였다.

2.2. 메모리 버퍼 모델

일반적으로 디스크 접근인한 속도 지연을 줄이기 위해 메모리 공간에 버퍼를 둔다. 미디어 스트리밍 서버에서도 디스크 접근 시간의 불규칙성과 운영 체제 내부에서 일어나는 프로세스 간의 문맥 교환, 네트워크의 파동 현상으로 인한 전송 지연을 방지하기 위해 메모리 공간에 버퍼를 이용한다[12,13]. 미디어 스트리밍 서비스에서 사용하는 버퍼 모델은 크게 두 가지 이다. 하나는 서비스를 담당하는 프로세스마다 독립적으로 버퍼를 두는 독립 버퍼 모델이고 다른 하나는 서비스 프로그램 전체에서 버퍼를 공유하는 공유 버퍼 모델이다.

독립 버퍼 모델은 각 서비스마다 버퍼를 따로 가지고 관리하게 된다[12,14]. 버퍼에 대한 참조가 서로 독립적으로 이루어지기 때문에 관리나 구현이 용이하고 서로 다른 데이터를 서비스하는 경우에는 큰 문제가 없다. 그러나 같은 데이터에 대해 두 개 이상의 서비스를 하는 경우 각각의 프로세스들은 자신의 버퍼 영역에 따로따로 데이터를 가지게 되어 같은 데이터가 메모리에 중복되는 현상이 발생한다. 미디어 스트리밍 서비스는 대부분의 서비스가 특정 시간에 한 두 개의 데이터로 집중되는 특성이 있기 때문에 독립 버퍼 모델은 메모리를 낭비를 초래하여 매우 비효과적이다.

이런 문제점을 해결하기 위한 방법이 공유 버퍼 모델이다[14,15]. 공유 버퍼 모델은 서비스를 하는 여러 프로세스들이 하나의 버퍼 공간을 공유하는 것이다. 일단 하나의 프로세스에 의해 데이터가 버퍼에 저장되면 여러 프로세스

들이 이 데이터를 사용할 수 있다. 여러 프로세스에 의해 버퍼 공간이 공유되므로 독립 버퍼 모델에 비해 관리나 구현이 쉽지는 않지만 미디어 스트리밍 서비스와 같은 서비스에 매우 효과적이다.

버퍼 공간을 공유한다고 할지라도 메모리공간은 디스크에 비해 턱없이 부족하다. 서비스가 늘어나면 더 많은 버퍼 공간을 계속 요구하지만 시스템의 메모리공간은 물리적으로 한계가 있다. 이런 한계를 극복하기 위해서는 효율을 높일 수 있는 버퍼 대체 정책이 필요하다.

2.3. 버퍼 대체 정책

2.3.1 정적 버퍼 대체

정적 버퍼 대체는 버퍼의 내용이 대체되는 주기가 길고 수동적으로 이루어지게 된다. 이때 적재되는 데이터는 대부분 쪼개어지지 않고 메모리의 연속적 공간에 할당된다. 이를 위하여 메모리의 일부 혹은 전체를 특정 데이터를 위해 할당해야 한다. 이미 데이터가 적재될 공간을 마련해 놓았기 때문에 필요한 만큼만 그때그때 할당 받지 않고 연속적인 데이터 할당이 가능하다. 메모리에 적재되는 데이터에 대한 내용을 관리자가 직접 선택 하거나 프로그램 상에서 이루어지더라도 일정기간동안의 로그를 이용하기 때문에 버퍼 대체의 주기가 길다.

정적인 방법은 시스템의 현재 상태를 실시간으로 반영할 수 없고 버퍼대체에 대한 부하가 따르기 때문에 많이 애용되고 있지는 않다. 그러나 동작이 일정하거나 특정 목적을 위해 사용되는 일부 시스템에서는 유용하게 이용되고 있다. 본 논문에서도 선호도를 반영하는 정적인 버퍼 관리 기법을 통해 미디어 스트림 가속기를 구현하여 성능 측정을 해보았다.

2.3.2 동적 버퍼 대체

효과적인 버퍼 관리를 위한 대체 정책 중 가장 많이 이용되는 것이 LRU(Least Recently Used)이다. LRU는 과거 참조거리를 기준으로 우선순위를 부여하여 대체 대상를 선정한다. 과거 참조거리가 짧을수록 높은 우선순위가 부여되므로 대체 선정에서 제외되어 버퍼 공간에 오래 잔류하고 재사용될 가능성도 높아지게 된다. LRU는 거의 모든 응용 프로그램에서 공통적으로 나타나는 시간적 국부성을 이용하는 정책이므로 대부분의 환경에서 어느 정도 만족스러운 성능을 보이며 많은 시스템의 버퍼 대체 정책으로 채택되어 사용되고 있다. 그러나 LRU 정책에는 가장 최근에 접근된 것이 항상 최우선순위를 가질 뿐 다른 사항은 고려하지 않고 있다. 이런 단점을 보완하기 위하여 LRU에 대한 많은 연구가 진행되어 졌고 많은 변종된 LRU 정책이 있다.

LRU 정책에서 가장 큰 문제로 지적되는 것은 접근빈도에 따른 배려가 없다는 것이다. 자주 사용되는 데이터와 그렇지 않은 데이터에 대한 구분이 없기 때문에 자주 사용되지 않는 데이터도 버퍼 공간에 오래 잔류하는 문제가 발생한다. 얼마 전까지 빈번한 접근이 이루어진 것보다 최근 한번의 접근이 우선순위가 더 높다. 이런 단점을 극복하고자 나온 정책들로는 LRU-K[1], LRFU[2] 등이 있고 기본 LRU의 단점을 어느 정도 해결하고 있다. 이 방법들의 주요 해결책은 버퍼에 카운터를 두거나 과거 참조 거리를 기반으로 대체될 버퍼를 정하는 것이다. LRU 대체 정책에 접근 빈도라는 하나의 변수를 더 추가하여 데이터에 대한 접근 빈도를 고려하는 것이다. LRU 정책에서 과거 참조 거리가 버퍼 보다 클 경우 연속적인 버퍼 미스가 발생하게 된다. 버퍼의 크기가 주기적으로 접근하는 데이터보다 크기 때문에 반복적인 접근임에도 불구하고 연속적인 미스가 발생하게 된다. 이런 문제를 해결하고자 연구된 것이 EELRU 정책이다[3].

위에서 열거한 LRU의 문제점은 시스템 환경이나 서비스 종류에 따라 발생하는 참조 특성은 고려하지 않고 단지 시간적 국부성만을 이용하여 과거 참조 거리가 대체 대상을 선정하는 유일한 조건이라는 것에서 기인한다. 따라서 각 시스템 환경에 적합한 버퍼 대체 정책을 개발하기 위해서는 먼저 대체 정책이 적용되는 시스템 환경이나 서비스 특성을 파악하여 알맞은 정책을 수립하는 것이 필요하다.

2.3. 그룹 버퍼 관리

미디어 데이터는 대용량이고 대부분의 접근이 순차적이고 반복적으로 이루어진다. 미디어 데이터의 대용량 특성을 잘 반영하는 것이 그룹 단위의 버퍼 관리이다. 그룹 단위 버퍼 관리란 크기가 작은 여러 개의 버퍼를 하나로 묶고 이를 하나의 기본 단위로 하여 버퍼를 관리하는 것이다[11,16].

하나의 그룹이 관리의 기본 단위가 되므로 어떤 조건을 가지고 그룹을 형성하는 가도 중요하다. 보통은 시간적, 공간적으로 비슷한 서비스를 하는 것들을 하나의 그룹으로 형성한다. 그중에서 가장 단순하게 그룹을 나누는 방법이 고정적인 크기를 주기적으로 검사하는 방법이다. 특정 시간에 전체 버퍼에 대한 접근 현황을 통해 그룹을 만든다. 대체가 이루어져야 하는 경우 해당 그룹을 대상으로 대체작업을 수행한다. 이렇게 형성된 그룹은 끝까지 유지 될 수도 있고 중간에 다른 조건들에 의해 합쳐지거나 둘로 나눠지게 하여 효율을 더 높이는 연구도 있다[16].

위에서 본 것처럼 미디어 스트리밍 서비스의 효과적인 메모리 관리를 위해서는 선호도, 메모리 모델, 서비스의 특성, 데이터의 접근 특성 등 많은 점들을 고려하여야 한다. 그중 한두 가지만을 고려하는 것보다는 여러 가지 특성을

고루 반영하는 것이 더욱 효율적인 시스템을 만들 수 있다. 그러나 대부분의 기존 연구들은 그룹화에 대한 연구와 선호도를 반영한 연구들이 각 분야별로 이루어져왔다. 본 논문에서는 위에서 살펴본 여러 가지 방법들을 함께 고려한 시스템을 설계하고 이를 직접 구현하여 성능을 평가해 본다.

Ⅲ. 정적 버퍼 관리

대부분의 정적 버퍼 관리 시스템에서는 버퍼의 대체주기가 길고 수동적으로 대체 작업이 이루어지게 된다. 한번 메모리에 적재된 데이터는 서비스가 완료될 때 까지 대체 되지 않고 메모리에 상주한다. 필요로하는 모든 데이터가 메모리에 이미 적재되어 있기 때문에 메모리 관점에서만 보면 가장 좋은 성능을 내는 가장 이상적인 방법이다. 이런 메모리 기반의 서비스는 디스크 입출력으로 인한 지연시간을 줄여 사용자 입장에서 느끼는 초기 지연을 줄여준다. 또한 버퍼 대체로 인한 부하가 감소하기 때문에 전체적으로 성능이 향상되고 보다 많은 서비스를 안정적으로 제공할 수 있게 된다.

메모리 기반의 시스템을 구성하는 방법은 크게 두 가지가 있다. 하나는 시스템에서 사용하는 버퍼 메모리 공간의 일정영역을 할애하는 것이고 다른 하나는 메모리 기반의 시스템을 따로 구성하는 것이다. 본 논문에서는 후자의 방법을 이용하여 미디어 스트림 가속기를 설계 및 구현하였다. 이번 장에서는 그 구성 방법과 성능에 대해 보여준다. 또한 메모리가 충분한 경우 서비스의 저해요인이 무엇인지 살펴보고 정적 버퍼 관리가 가지는 문제점에 대해 알아본다.

3.1. 미디어 스트림 가속기 (Media Stream Accelerator)

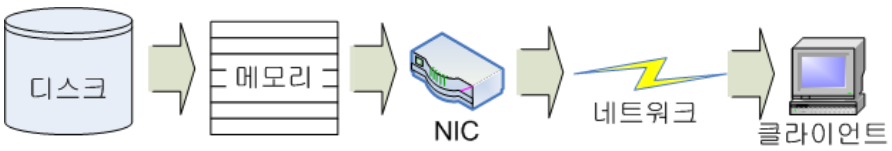
일반적으로 미디어 스트리밍 서버의 미디어 데이터는 디스크에 저장 된다. 이런 미디어 데이터는 사용자의 요구에 따라 메모리에 적재되고 서비스가 이루어진다. 디스크는 메모리에 비해 저렴하고 용량이 크기 때문에 많은 양의 데이터를 저장할 수 있다. 그러나 근본적으로 메모리에 비해 접근 시간이 길기 때문에 디스크에 있는 데이터를 사용하기 위해서는 기다림이 필요하다. 이런 시

간 차이를 줄이고자 메모리에 버퍼 공간을 사용한다. 버퍼는 디스크의 내용을 임시적으로 저장하여 디스크에 접근할 때 느끼는 시간 지연을 줄여준다.

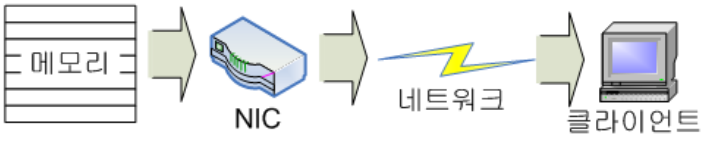
메모리 공간은 디스크에 비해 적기 때문에 버퍼를 이용한다고 할지라도 많은 양의 데이터를 적재할 수는 없다. 대용량의 데이터를 다루는 미디어 스트리밍 서비스에서는 많은 양의 버퍼가 성능에 큰 도움이 된다. 미디어 스트림 가속기는 시스템에 많은 양의 메모리를 장착하고 선호도가 높은 일부 미디어 데이터를 메모리에 직접 저장하여 서비스 한다. 본 논문에서 구현하는 미디어 스트림 가속기는 별도의 시스템으로 동작하며 리눅스 클러스터 기반의 VOD 서버인 VODCA[4]를 기반으로 개발되었다.

3.2. 미디어 스트림 가속기의 구성 및 동작

그림 2는 일반적인 미디어 스트리밍 서버에서 이루어지는 미디어 데이터의 서비스 과정이다. 모든 미디어 데이터는 디스크에 저장되고 요구에 의해 메모리를 거쳐 실제 서비스가 이루어지게 된다. 그림 3은 미디어 스트림 가속기에서 이루어지는 미디어 데이터의 서비스 과정이다.

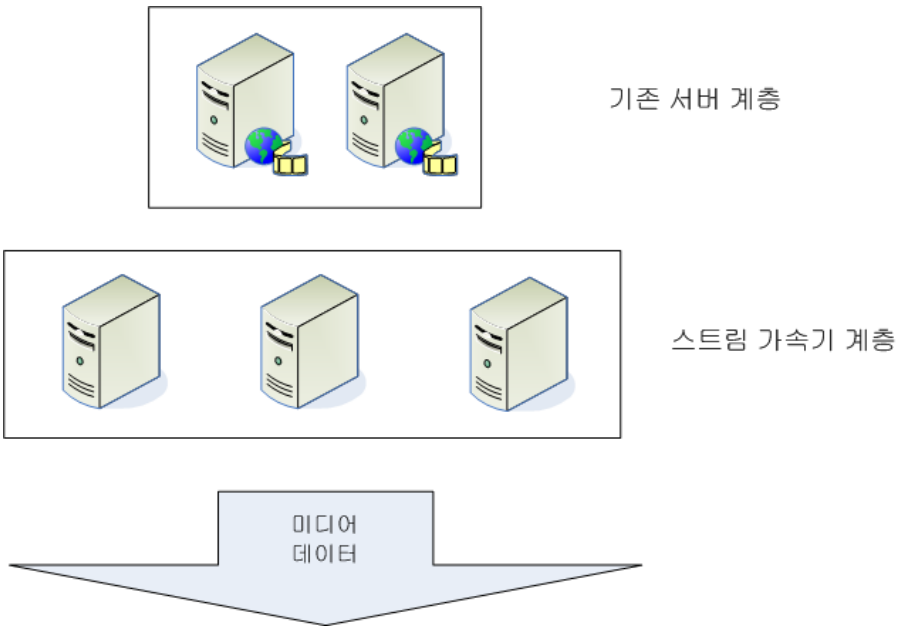


[그림 2] 일반적인 미디어 데이터 서비스 과정



[그림 3] 미디어 스트림 가속기의 미디어 테이터 서비스 과정

그림에서 보여주는 것처럼 미디어 스트림 가속기에서는 디스크의 입출력 과정이 없다. 미디어 스트림 가속기는 프로그램의 시작과 함께 네트워크를 통해 미디어 관리 서버로부터 지금까지 선호도가 가장 높은 미이어 데이터를 전송 받게 된다. 이렇게 전송받은 데이터는 디스크에 저장되는 것이 아니라 프로그램의 메모리 공간에 직접 저장되고 사용자 요청에 대해 빠른 서비스가 이루어진다.

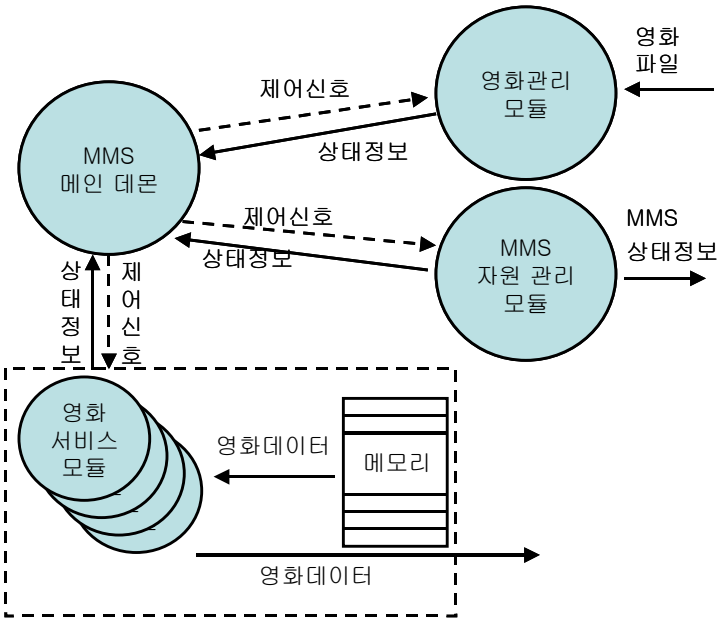


[그림 4] 미디어 스트림 가속기의 서비스 계층도

그림 4는 스트림 가속기가 위치하는 곳을 논리적으로 그린 것이다. 스트림 가속기는 기존의 서버 시스템과는 별도의 노드로 시스템 전면에 위치하게 된다. 스트림 가속기에는 과거 접속 로그를 통해 선호도가 높은 일부 데이터만을 메모리에 상주시켜 서비스한다. 과거의 접속 로그대로 서비스가 이루어질 경우 기존 서버에서 행해지는 많은 동작이 스트림 가속기로 분산되어 더 많은 서비스를 할 수 있게 된다. 그러나 과거의 로그가 실제 서비스 요청과 맞지 않으면 메모리의 내용을 다른 것으로 대체하는 작업을 해야 하고 별도의 노드를

추가한 효과를 볼 수 없게 된다.

다음에 보여 지는 그림 5는 실제 서비스를 위해 시스템에 구현된 미디어 스트림 가속기의 구성도 이다.



[그림 5] 실제 구현된 미디어 스트림 가속기의 구성도

그림 5에서 가장 큰 특징은 네모난 점선으로 그려지는 부분이다. 일반적으로 디스크에서 미디어 데이터를 읽어오는 반면에 가속기는 메모리에서 영화 데이터를 바로 읽어온다. 또한 하나의 클라이언트 당 하나의 쓰레드를 생성하여 처리하는 쓰레드 모델을 이용하여 구현하였다. 이런 시스템에는 물리적인 메모리의 한계로 인해 많은 수의 영화를 서비스 하지는 못한다. 영화 저장 면에서 메모리는 디스크에 비해 그 양이 매우 적고 비싸기 때문이다. 하지만 이런 가속기 시스템을 사용하면 디스크 접근으로부터 오는 시간 지연을 줄여 사용자가 느끼는 서비스 초기 지연을 줄일 수 있다. 또한 자주 요청 되는 몇몇 영화만을 서비스하기 때문에 해당 영화에 대한 요청이 많을 경우 전체 시스템 성능 향상에 큰 효과를 거둘 수 있다.

미디어 스트림 가속기는 기존 시스템과 별도로 독립된 하나의 시스템으로 동작한다. 기존 시스템에 추가하는 형태로 구성되기 때문에 전체 시스템 확장도 쉽게 할 수 있다. 확장이 용이하기 때문에 향후 네트워크 대역폭이 늘어날 경우 발생할 수 있는 일련의 문제들에 대해 유연하게 대처 할 수 있다.

3.3. 미디어 스트림 가속기에서 성김도 정책

미디어 스트리밍 시스템의 병렬성을 높이기 위하여 미디어 데이터는 여러 개의 가속기에 분산 저장되어 진다[4,5,6]. 일반 재생을 위한 영화 데이터는 GOP 단위로 나뉘어 분산 저장되고 고속 재생을 위한 영화 데이터는 I-Frame 만을 이용하여 저장된다. 저장 순서는 라운드 로빈 방식과 스핀 방식을 지원하며 저장된 데이터는 저장 시 만들어진 헤더 파일을 통해 참조되어 진다. 이렇게 작은 단위로 영화를 분산 저장하면 부하를 여러 개로 분산하는 효과가 있어 보다 효과적인 서비스가 가능하다.

3.4. 성능 평가 및 결과

이번 절에서는 구현된 미디어 스트림 가속기의 성능 평가를 위한 실험 환경을 소개하고 실제 성능평가를 수행한 결과와 분석을 보여준다.

3.4.1. 실험 환경

미디어 스트림 가속기는 네트워크를 통해서 미디어 관리 서버로부터 최근에 가장 인기가 많은 5개의 영화를 전송받아 메모리에 저장한다. 하나의 영화는 병렬처리를 위한 성김도 정책에 기준하여 여러 개의 미디어 가속기들에 병

렬 저장된다. 본 논문에서는 병렬처리를 위한 미디어 데이터의 최소 분할 단위를 GOP 단위로 하였고 각 가속기들은 하나의 100Mbps 스위칭 허브에 연결되고 각각의 가속기를 이루는 하드웨어 사양은 표1과 같다.

CPU	Intel(R) Celeron(R) CPU 2.00GHz
메모리	1GB SDRAM
운영체제	RedHat 7.3 (Kernel 2.4.18)
네트워크	100Mbps fast-ethernet (Hub : 3Com 3C16465C Super Stack 3, 24port)

[표 1] 미디어 스트림 가속기의 하드웨어 사양

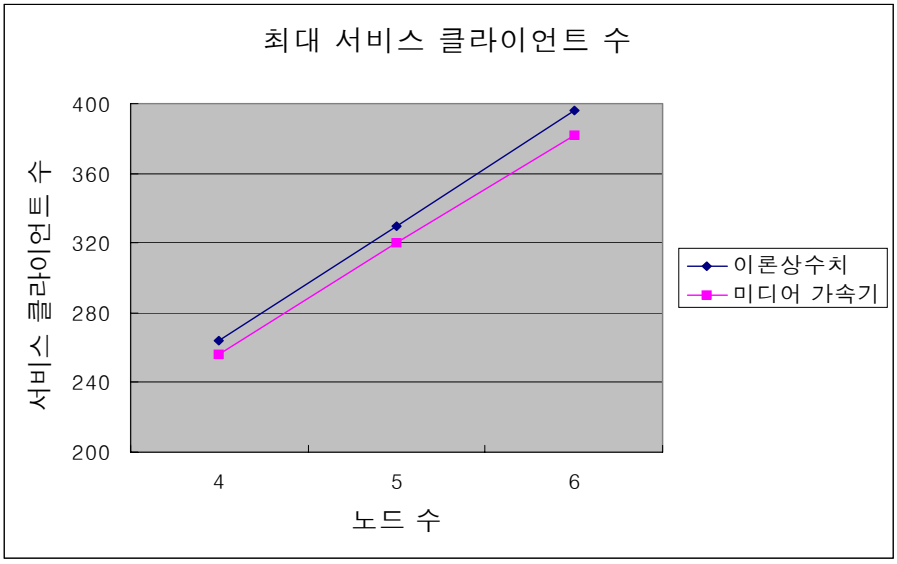
3.4.2. 성능 측정 방법

성능 측정은 VODCA 시스템 성능 측정 방법[4]을 사용하였다. 실제로 동작하는 하나의 클라이언트를 이용하여 서비스를 받게 하고 여러 개의 가상 부하 클라이언트를 이용하여 실제 서비스와 같은 부하를 서버에 걸어준다. 이때 가상 클라이언트 수를 점점 증가시켜 부하를 증가 시키고 실제 서비스에 영향을 미치는 시점을 측정한다.

3.4.3. 성능 측정 결과

실험에 사용한 MPEG1 영화 데이터는 1.5Mbps를 보장해주어야 원활한 상영이 가능하다[7]. 100Mbps 네트워크 환경에서 서비스를 한다면 하나의 서버에서 약 66개의 클라이언트를 처리할 수 있다. 이를 근거로 4,5,6개의 가속기들에서 최대 서비스 가능한 이론적 수치를 계산하였고 실험을 통하여 QoS가 보장되는 실제 서비스 가능한 스트리밍 수를 측정하였다.

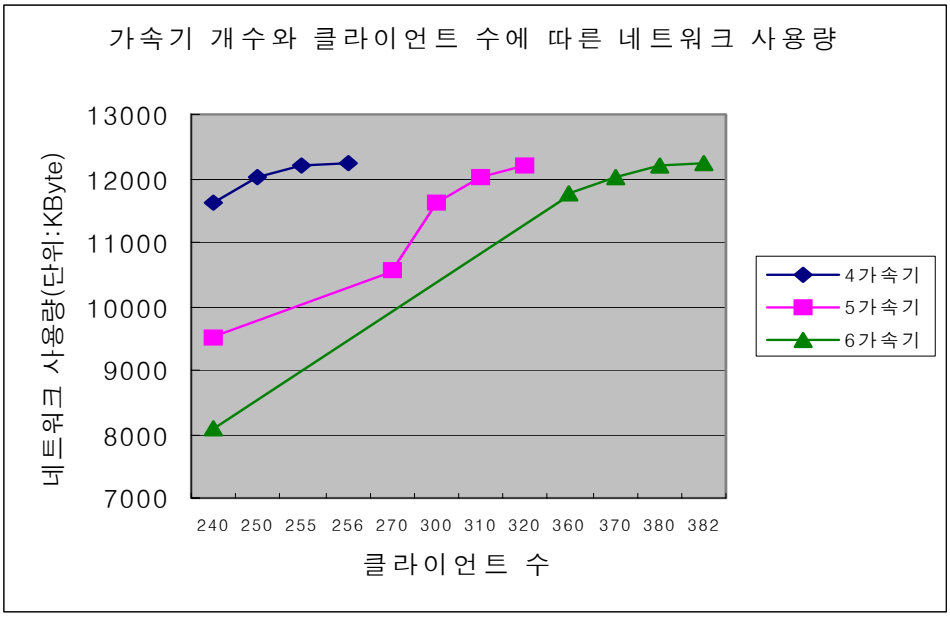
그림 6은 영화를 분산 저장하는 가속기의 수에 따라 서비스 가능한 최대 클라이언트 수를 보여주고 있다. 미디어 스트림 가속기는 그 수가 증가함에 따라 서비스 가능한 클라이언트 수가 이론상의 최대치에 근접하여 선형으로 증가하고 있다. 그림에는 나타나지 않았지만 과거 시스템과 비교해보면 많은 성능 향상을 보여주고 있는 것이다. 실험 시 미디어 스트림 가속기의 수를 최대 6개까지 밖에 하지 못하였다. 이는 100Mbps 24포트 스위칭 허브 환경에서는 더 이상의 클라이언트 측정이 불가능하였기 때문이다. 그러나 그림 6의 결과가 보여주는 것처럼 가속기가 몇 개 더 연결되어도 이론상의 수치에 근접하는 선형 증가를 예측할 수 있다.



[그림 6] 미디어 스트림 가속기의 최대 서비스 클라이언트 수

그림 7은 여러 개의 미디어 스트림 가속기를 이용하여 서비스를 할 때 클라이언트 연결 수에 따라 가속기 각각의 네트워크 사용량 변화를 보여주고 있다. 네트워크의 사용량은 KByte 단위로 표시하였고 클라이언트 수는 가속기의 수에 따라 한계에 이르는 시점을 위주로 나타내었다.

먼저 4,5,6개의 가속기 사용 시 240개의 클라이언트 연결을 처리할 때 네트워크 사용량을 보자. 4노드는 11601KByte, 5노드는 9502KBye, 6노드는 8082KByte 씩을 각각 사용하고 있다. 이는 분산되는 가속기의 수가 많을수록 하나의 가속기에서 담당하는 서비스의 양이 적어지는 것을 나타낸다. 즉, 영화 데이터를 많은 수의 가속기에 분산 저장할 경우 서비스 가능한 스트리밍의 수가 증가함을 보여주는 것이다.



[그림 7] 가속기의 수와 클라이언트 수에 따른 네트워크 사용량

전체적인 그림을 보면 12000KByte가 좀 넘는 시점에서 한계를 나타내고 있다. 실험을 한 네트워크 환경은 100Mbps 이고 이 환경에서 이론상으로 사용 가능한 네트워크 대역폭을 KByte 단위로 나타내면 12800KByte 이다. 본 실험에서 한계를 나타내는 수치는 이론상 가능한 최대 서비스 대역폭의 약 95%에 해당하는 수치이다. 즉, 미디어 스트림 가속기를 사용하는 경우에는 네트워크 대역폭이 서비스의 장애 요인이 되는 것이다.

그림 6,7을 보면 이론적으로 서비스 가능한 수치의 약 95%의 달하는 성능

을 보여주고 있다. 약 5%의 성능 저하가 나타나는데 이는 네트워크 장비에서 발생할 수도 있고 운영체제나 시스템에서 데이터를 처리하는 오버헤드로 볼 수도 있다.

본 실험의 결과로 볼 때 향후 기가비트 대역폭을 확보하는 네트워크 망이 상용화될 경우 미디어 가속기를 사용하여 선호도가 높은 영화들을 서비스 한다면 VOD 시스템의 성능 확장성에 크게 기여할 것으로 기대되어 진다.

3.5. 정적 버퍼 관리의 문제점

본 연구에서는 미디어 서비스를 담당하는 미디어 스트림 가속기를 설계하고 구현하여 본 시스템이 확장성 있는 미디어 스트리밍 서비스 구축에 기여 할 수 있음을 보였다. 또한 실험을 통하여 향후 네트워크 망이 기가비트로 상용화될 경우 구현한 미디어 스트림 가속기가 시스템의 전체 성능 향상에 중요한 역할을 할 것임을 볼 수 있었다.

반면에 미디어 스트림 가속기는 메모리 기반의 서비스이기 때문에 이를 위해서는 시스템에 많은 양의 메모리가 필요하다. 메모리는 휘발성이기 때문에 시스템의 부팅 때마다 미디어 관리 서버로부터 미디어 데이터를 전송받아야만 서비스가 가능하다. 이런 일련의 과정에서 또 다른 문제가 야기된다.

먼저 문제가 되는 것이 선호도 반영에 대한 것이다. 지난 일정 기간동안의 선호도를 기반으로 서비스 미디어가 선택되고 서비스가 이루어진다. 그러므로 가속기에서 서비스되는 데이터는 과거의 선호도를 반영할 뿐 현재의 선호도는 반영하지 못한다. 또한 데이터를 전송받는 작업은 시스템에 많은 부하를 주는 작업이므로 한가한 시간이나 오프라인 상태에서 작업이 이루어져야 하는 문제가 있다. 이러한 작업들을 위해 주기적인 관리가 필요하지만 미디어 데이터에 대한 선호도가 수시로 바뀌기 때문에 정확한 주기를 정하기도 어렵다. 수동적

으로 관리를 직접 하거나 프로그램 상에서 이루어진다고 하더라도 갱신주기난 방법 등에 대한 연구가 필요하게 된다.

Ⅳ. 동적 버퍼 관리

동적 버퍼 관리는 버퍼의 크기가 고정된 형태로 구현된다. 이런 동적 버퍼 관리에서는 정적 버퍼관리에 비하여 같은 메모리공간에 서로 다른 종류의 내용을 저장하는 버퍼가 많이 존재하게 된다. 때문에 이를 관리하기 위한 비용이 소모되는 단점이 있지만 사용자들의 메모리 요구를 능동적으로 할당, 환원 시킬 수 있는 장점이 있다.

한정된 메모리 공간을 효율적으로 사용하기 위해서는 버퍼의 히트 율을 높일 수 있는 효과적인 버퍼 대체 정책이 필요하다. 버퍼 대체 정책에서는 데이터의 접근 특성이나 패턴을 고려하여 앞으로 많이 사용될 만한 데이터를 예측하고 이에 따른 정책을 수립한다. 미디어 스트리밍 서비스 뿐 만 아니라 데이터베이스, 웹 서비스, 운영체제 등 많은 분야에서 자신의 서비스 특성에 적합한 효과적인 대체 정책에 대한 연구가 수행되고 있다. 이번 장에서는 미디어 스트리밍 서비스의 특성을 고려한 버퍼 대체 정책에 대해 소개한다. 또한 소개된 대체 정책을 기반으로 동적 버퍼 관리 시스템을 구현하고 성능평가를 수행하여 그 결과에 대해 분석한다.

4.1. 선호도 기반 버퍼 관리

미디어 스트리밍 서비스에서는 다수의 사용자들이 같은 미디어에 대한 요구를 동시에 그리고 산발적으로 발생한다. 그 결과 대용량의 데이터에 대한 순차적이고 반복적인 접근 형태를 나타내게 된다. 기존의 버퍼 관리 정책들은 이러한 미디어 데이터 접근 특성을 충분히 반영하지 못하므로 좋은 성능을 보장하지 못한다. 이렇게 대용량의 데이터에 순차적인 접근이 많은 시스템에서는

성능 개선을 위하여 지역 국부성을 고려하기보다는 데이터에 대한 선호도나 프리패치와 같은 기법이 더 효과적일 수 있다[8,9].

본 절에서는 LRU(Least Recently Used), LFU(Least Frequently Used) 정책에 사용자 선호도를 추가하여 버퍼에서 히트 율을 높이는 선호도 기반 버퍼관리 기법을 알아본다. 선호도를 추가한 버퍼 대체 정책은 기존의 LFU와 유사하지만 미디어 데이터의 선호도 수치에 대한 가중치를 두어서 선호도가 높은 데이터가 대체 되는 것을 한 번 더 보호할 수 있게 하였다. 사용 빈도에 선호도 수치를 더해주어 최근 사용빈도가 낮더라도 선호도가 높은 것은 대체 선정에서 제외시키는 것이다. 성능 평가에서 사용한 선호도 수치는 다음 식에 의해 결정하였다.

$$favor(i) = zipf(i) \times N \qquad (1)$$

zipf(i) : i 번째 영화에 대한 요청 확률
N : 전체 서비스 요청 수

식(1) 에서도 알 수 있듯이 선호도 수치는 zipf[10] 분포와 전체 서비스 요청 수인 N을 곱하여 이용하였다. zipf 분포는 해당 인덱스에 대한 요청 확률을 나타내기 때문에 1 미만의 실수로 나타난다. 이런 실수 값을 그대로 사용할 수 없기 때문에 정수화 과정이 필요하다.

한 번의 참조가 이루어지면 참조에 대한 가중치는 1씩 증가하는데 선호도에 따른 차이가 1미만이라면 선호도 수치는 의미가 없어진다. 또 실수 수치를 그대로 이용한다면 계속되는 실수 연산으로 인해 성능 저하를 초래 할 수도 있다. 이런 이유로 사용자의 요청에 유연하게 대응하면서 선호도 수치를 정수화하는 방법으로 확률 값에 N을 곱하여 가중치 수치로 이용하였다. 이 수치는 해

당 미이어 데이터에 대해서 서비스를 요청하는 수이기도 하다. 이렇게 수치를 부여하면 전체 사용자의 요청이 증가함에 따라 자연스럽게 선호도 수치도 증가하고 반대의 경우에는 수치가 감소하여 전체 요청에 맞게 선호도 수치를 반영할 수 있다.

4.2. 선호도 기반 버퍼 관리의 성능 평가

4.2.1. 성능 평가 방법

성능 평가는 AMD 2000+ MP Dual CPU와 2GByte의 메모리가 탑재된 리눅스 시스템에서 시뮬레이션을 수행하였다. 총 3가지 대체 정책(LRU, LFU, LRU+가중치)들에 대한 성능평가를 수행하였다. 기존의 버퍼관리 정책으로는 LRU와 LFU를 이용하였고 여기에 미디어 데이터의 선호도를 더하여 우선순위를 결정하는 LRU+가중치 방법을 추가한 것이다.

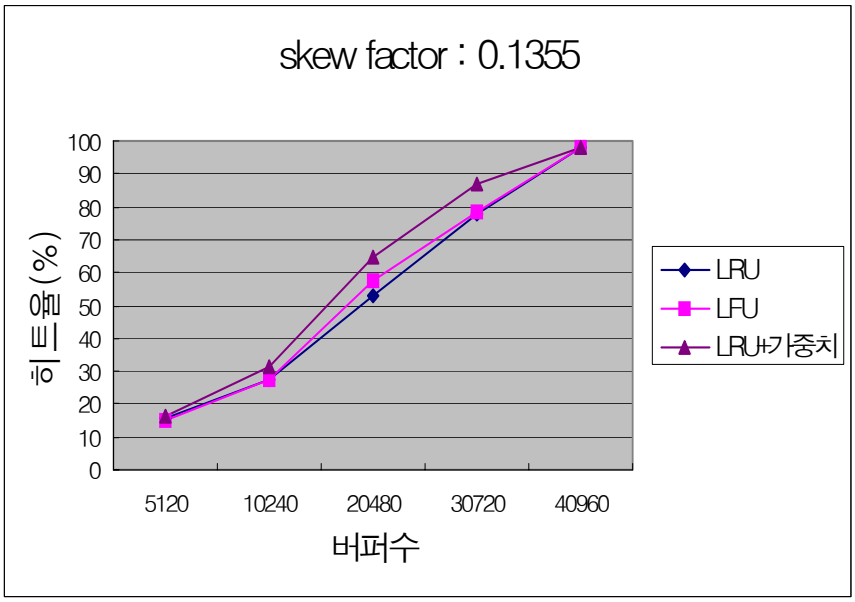
시뮬레이션을 수행할 때 식 (1)에서 전체 서비스 요청을 나타내는 N은 500으로 하였고 버퍼의 사이즈는 10KByte로 하였다. 버퍼 사이즈를 10KByte로 한 이유는 네트워크 상에 데이터를 전송할 때 10KByte 씩 이루어지기 때문에 이를 반영한 것이다[4].

미디어 데이터의 분포는 zipf 분포를 이용하였다. zipf 분포의 선호도를 결정짓는 skew factor는 0.271을 기준으로 0.1355, 0.542 3가지 경우로 나누었다[10]. 각각의 경우에 따라 버퍼 수를 달리 하여 버퍼 히트율을 측정하였으며 사용자의 요청에 대한 빈도는 λ가 2인 포아송 분포를 이용하여 초당 2개의 요청을 발생시켰다.

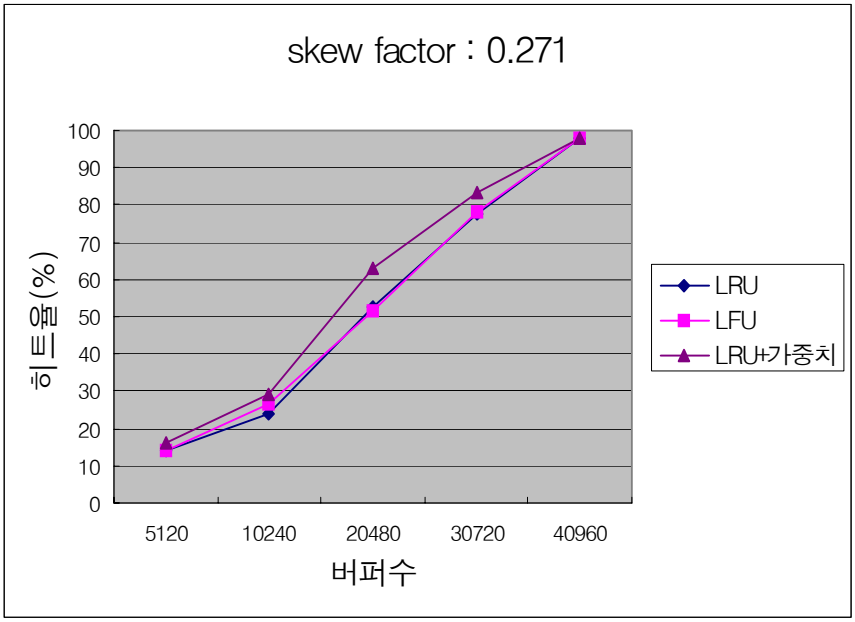
4.2.2. 성능 평가 결과

성능측정에서 우선 수행한 것은 zipf 분포의 skew factor를 달리 하였을 경우 버퍼 수 변화에 따른 히트율 이다. skew factor 값은 0에 가까울수록 선호도의 변화가 많이 나타나고 1에 가까울수록 고르게 나타난다. 본 실험에서 skew factor 0.271을 기준으로 선호도의 변화가 보다 심한 0.1355와 정도가 덜한 0.542를 이용하여 성능을 측정하였다[10]. 선호도 변화가 심한 경우인 skew factor 0.1355를 스트리밍 서비스에 비추어 보면 500 번의 사용자 요청 중 150 번이 한 개의 미디어 데이터에 집중되는 경우를 의미한다.

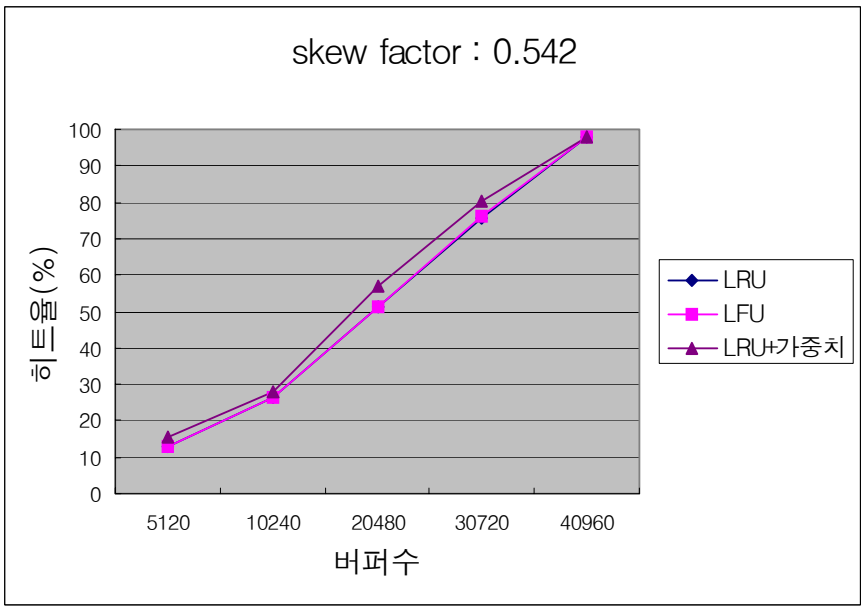
그림 8,9,10 은 3가지의 skew factor에 대하여 각각 성능을 측정한 결과이다. 대체적으로 3가지의 경우가 큰 차이를 보이지는 않는다. LRU와 LFU는 거의 비슷한 성능을 보이고 있으며 선호도에 따라 가중치를 부여한 정책이 약 4~5% 정도의 성능 향상을 보여주고 있다. 기존의 정책에 선호도만을 추가하여 이와 같은 성능 향상을 볼 수 있었다.



[그림 8] skew factor가 0.1355 인 경우 히트율 변화



[그림 9] skew factor가 0.271 인 경우 히트율 변화

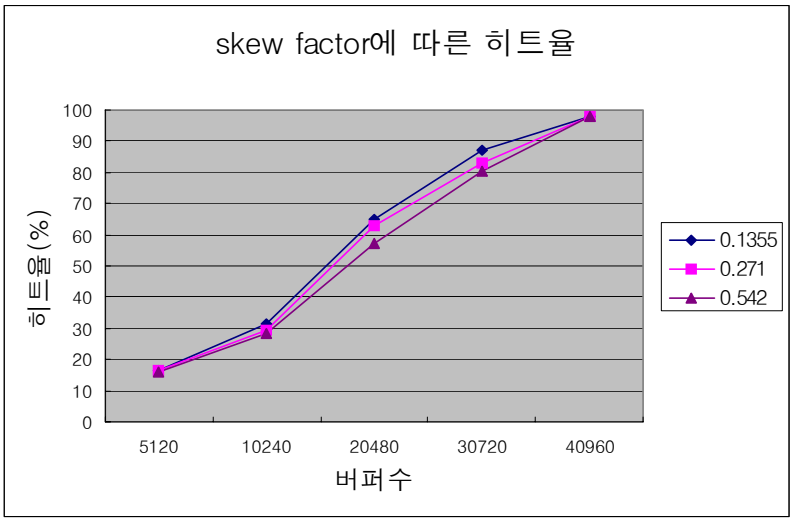


[그림 10] skew factor가 0.542 인 경우 히트율 변화

그림에서 보면 버퍼 수에 따라 차이정도가 다른 것을 볼 수 있다. 버퍼 수가 적거나 클 경우는 알고리즘에 따른 버퍼 히트율의 차이가 거의 없다. 버

퍼의 수가 적으면 빈번한 대체가 이루어지기 때문에 알고리즘에 대한 차이가 거의 없는 것이고 반대로 버퍼의 수가 많으면 대부분의 데이터가 메모리에 적재될 수 있기 때문에 그 차이가 없는 것이다. 그림에서 보면 버퍼수가 10240, 20480 일 때 차이정도가 크게 나타나는 데 메모리 크기로 보면 100M, 200M에 해당하는 크기이며 나중에 실제 시스템을 구현할 때는 약 140M정도의 메모리를 이용한다.

그림 11의 결과에 나타나듯이 선호도를 반영한 버퍼관리 정책들 간의 비교에서는 선호도 변화가 심한 skew factor 0.1355에서 가장 좋은 성능을 보였다. 이런 결과는 접근하는 데이터가 1,2개의 미디어 데이터에 집중하면서 버퍼의 재사용이 늘어나기 때문이다. 그러나 skew factor에 따라 많은 변화를 기대하였으나 성능 평가에서는 약 2%의 성능 차이만을 보여 주였다. 이는 skew factor가 증가하는 만큼 사용자의 요구가 증가하지는 않기 때문이다. 예를 들어 skew factor가 0.271에서 0.1355 로 두 배 변하여도 전체 사용자 요구 500 회 중 특정 미디어에 대한 요구는130회에서 150회로 그 변화의 폭이 크지 않다.



[그림 11] skew factor 변화에 따른 버퍼 히트 율

실험 결과로부터 사용자 선호도를 미디어 서버의 버퍼관리 기법에 적용하여 성능 향상을 관찰할 수 있었다. 선택된 선호도 수치들 사이에서 성능 차이가 크게 나타나지는 않았지만, 실제 미디어 서버 환경은 이번 실험에서 사용한 사용자 선호도 분포보다 특정한 미디어에 대한 사용자 요구 분포가 더욱 극단적일 것으로 예상되므로 선호도 기반 버퍼 관리 기법이 더욱 효과적일 것으로 예상된다.

4.3. 그룹 버퍼 관리

미디어 스트리밍 서비스에서 효과적인 메모리 관리를 위해 선호도와 더불어 고려되어 져야 하는 것이 데이터에 대한 접근 특성이다. 미디어 데이터는 용량이 크고 순차적, 반복적인 접근이 이루어지기 때문에 작은 단위로 관리하는 것보다는 그 관리 기준을 크게 하는 것이 더욱 효과적이다.

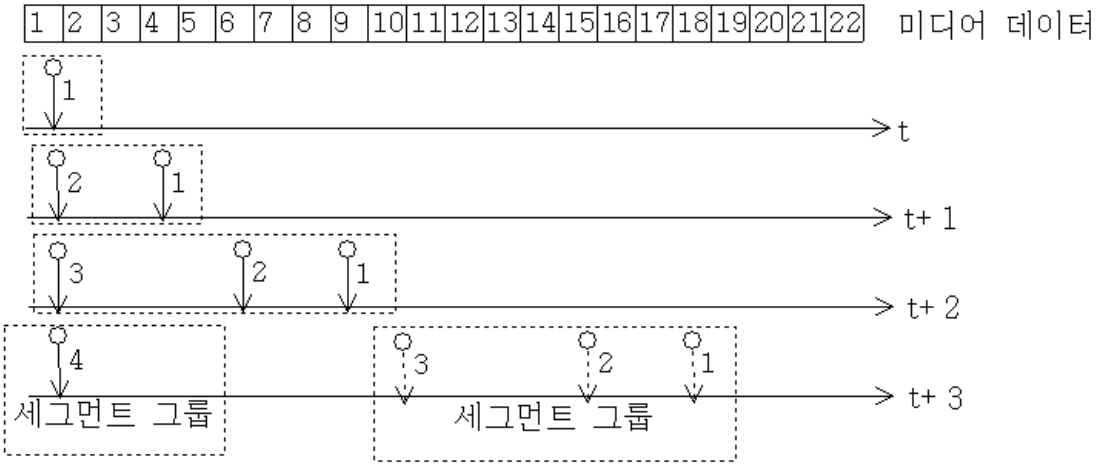
많이 사용되는 MPEG 미디어 데이터에서 자체적으로 재생이 가능한 기본 단위는 GOP(Group of Picture)이다. 미디어 데이터의 접근이 대용량으로 이루어진다는 점에서 볼 때 독립 재생이 가능한 기본 데이터 단위인 GOP를 여러 개 묶어서 관리하는 것이 더 효과적이다. 본 장에서는 여래 개의 GOP를 묶어서 하나의 세그먼트 그룹을 만들고 이를 기준으로 대체 정책을 수립하는 방법에 대해 소개한다.

4.3.1. 세그먼트 정의

세그먼트 그룹은 동일한 미디어 데이터에 대해 비슷한 부분을 서비스 받는 여러 클라이언트를 기준으로 하나 이상의 클라이언트 서비스 요청에 의해 동적

으로 생성된다. 하나의 세그먼트는 1개 이상의 GOP 정보와 1개 이상의 클라이언트 서비스 요청을 가지고 있다. 세그먼트 단위로 현재 서비스가 이루어지고 있는 클라이언트 수와 버퍼의 히트 수 그리고 가용한 버퍼 수 등의 정보를 저장하고 이를 기반으로 버퍼의 대체를 수행한다.

그림 12는 세그먼트와 그 생성 과정을 보여주고 있다. 임의의 시간 t 에 1 사용자의 요청이 도착한다. 이전에 만들어진 세그먼트가 있으면 자신이 포함될 수 있는 지 검사 한다. 자신이 처음 서비스거나 이전 세그먼트와 차이가 기준 이상이 되면 독립된 하나의 세그먼트를 생성한다. 이 후 시간이 지남에 따라 2,3,4 사용자의 요청이 도착한다. 이렇게 하나의 미디어 데이터에 대해 여러 사용자의 요청이 있을 때 시간적으로 비슷한 시간대에 도착한 1,2,3은 하나의 세그먼트를 형성하게 되고 요구 4와 같이 기존에 형성된 세그먼트에 대하여 시간적으로 거리가 먼 서비스 요청은 독립된 다른 세그먼트를 만들게 된다. 세그먼트는 버퍼 대체 시 기준이 되어 하나의 세그먼트가 대체 될 때 해당 세그먼트 안에 있는 모든 데이터가 같이 대체된다.

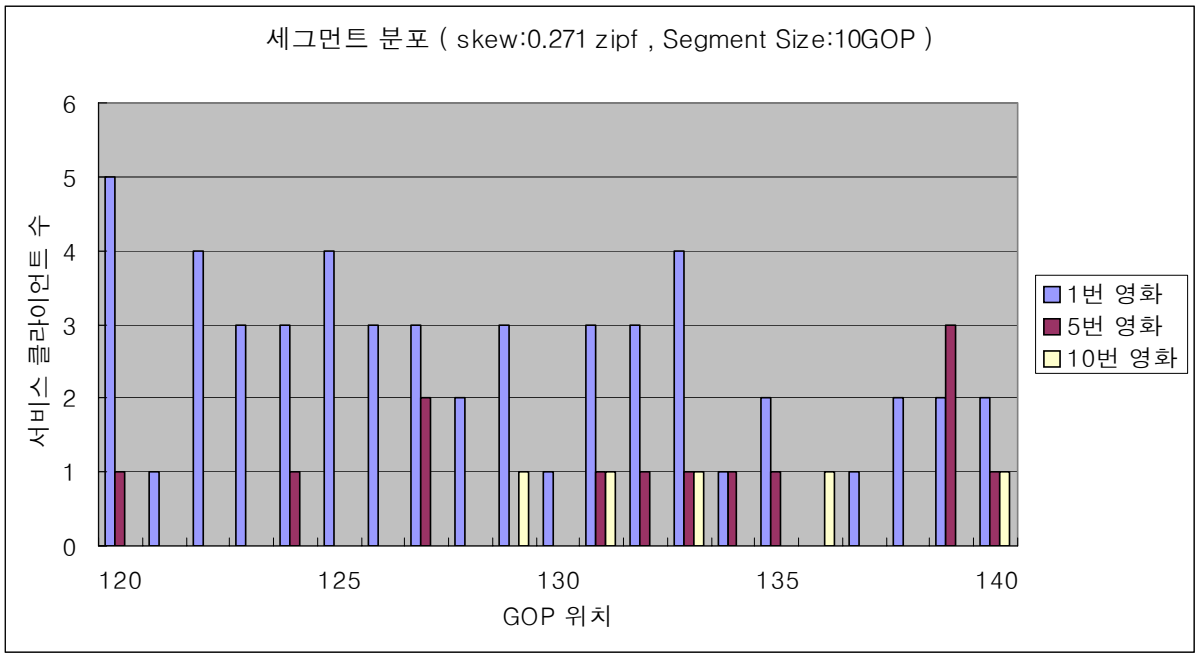


[그림 12] 사용자 요구에 따른 세그먼트 생성 과정

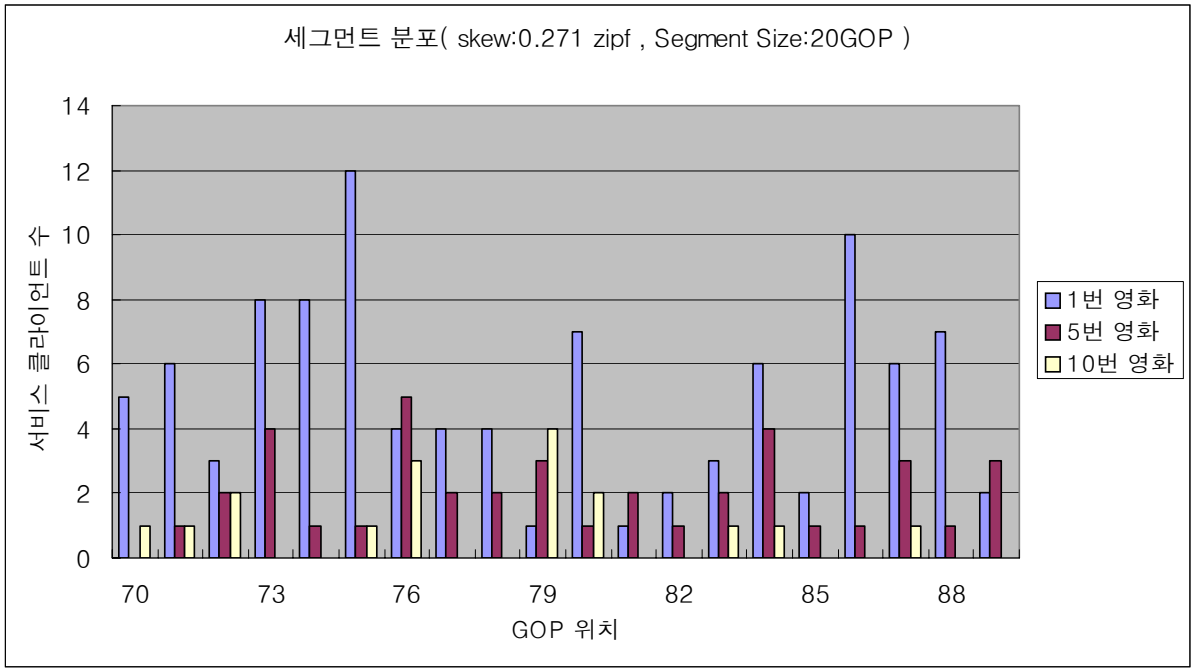
4.3.2. 세그먼트 분포

본 절에서는 세그먼트 단위의 효과적인 대체 정책을 수립하기위해 사용자들의 요구를 세그먼트 단위로 처리 시 사용자들이 서비스 받고 있는 세그먼트들의 분포를 모의시험을 통하여 관찰한다. 세그먼트 크기와 클라이언트 요청수를 달리하여 그 분포를 살펴보고 적용할 수 있는 대체 정책에 대해 알아본다.

시뮬레이션은 클라이언트 요청을 1000개, 2000개 5000개 발생 시키고 각각의 경우에 대하여 하나의 세그먼트 크기를 GOP 5개, 10개, 20개, 50개 로 하여 측정 하였다. 영화에 대한 분포는 skew factor가 0.271인 zipf 분포를 사용하였고 클라이언트 요구 빈도는 λ가 2인 포아송 분포를 사용하였다.



[그림 13] 하나의 세그먼트 크기가 GOP 10개 인 경우 세그먼트 분포

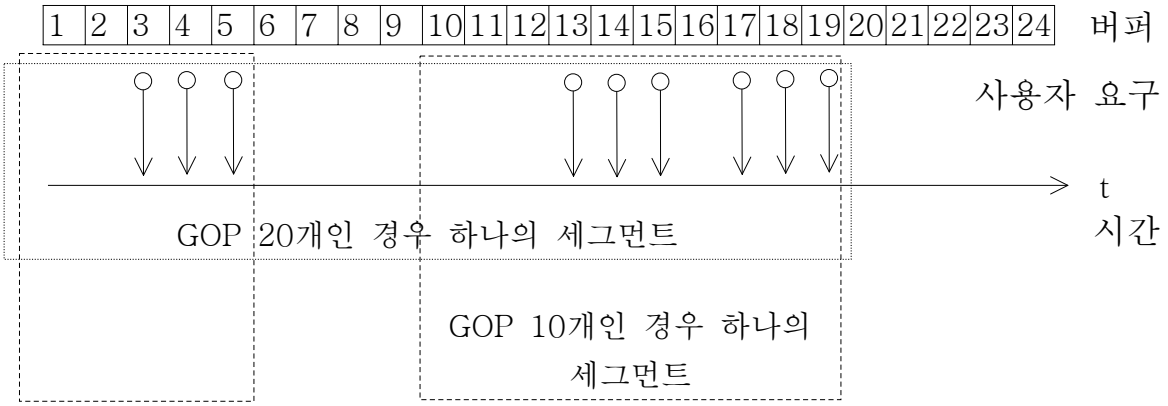


[그림 14] 하나의 세그먼트 크기가 GOP 20개 인 경우 세그먼트 분포

그림 13,14는 클라이언트 수가 1000개 인 경우 세그먼트의 크기를 다르게 하였을 때의 결과이다. 그림에서도 알 수 있듯이 세그먼트의 크기를 크게 하면 하나의 세그먼트에 속하는 서비스 클라이언트 수가 증가 하게 된다. 하나의 세그먼트로 더 많은 수의 클라이언트를 대표할 수 있어 더 효과적인 것처럼 보일 수도 있다. 하지만 세그먼트의 크기가 커지면 중간에 사용하지 않는 버퍼공간도 하나의 세그먼트 단위로 처리되기 때문에 버퍼를 세그먼트로 묶어 관리하는 효과가 반감 될 수 있다.

다음 그림 15는 세그먼트 크기가 클 경우 발생할 수 있는 문제에 대해 보여주는 것이다. 20개의 GOP를 하나의 세그먼트로 관리 하면 그림 15에 보여지는 사용자의 요구는 모두 하나의 세그먼트로 관리된다. 반면 같은 경우에 10개의 GOP를 하나의 세그먼트로 관리하면 두 개의 세그먼트로 나뉘어 관리된다. 하나의 세그먼트로 관리하게 될 경우 그림에서처럼 중간에 사용하지 않는 부분이 같은 세그먼트로 관리되기 때문에 비슷한 시간대의 접근에 대한 특성을

제대로 반영하지 못한다. 즉, 시간적 국부성에 대한 특성을 살리지 못하는 것이다. 그림에서 보여주고 있지는 않지만 세그먼트의 크기를 너무 작게 하면 세그먼트로 묶는 의미가 없어지기 때문에 효율적인 크기를 정하는 것도 성능 향상에 하나의 요인이 된다.



[그림 15] GOP 크기에 따른 하나의 세그먼트 형태

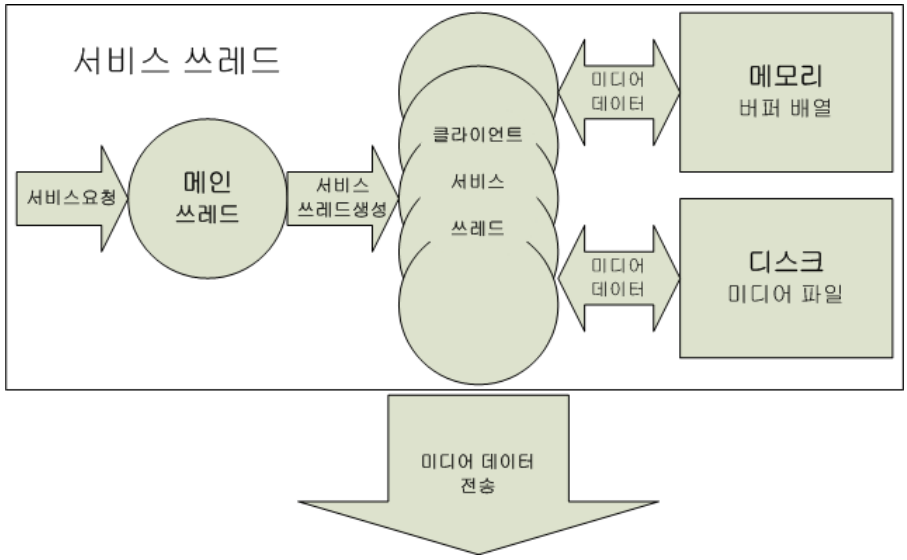
4.4. MSS (Minimum Service Segment) 대체 정책

MSS 대체 정책은 세그먼트를 기본 단위로 하고 선호도를 기반으로 하는 대체 정책이다. 대체 선정 시 우선적으로 선호도가 가장 낮은 미디어 데이터를 선택한다. 다음으로 해당 미디어 데이터 내에서 가장 선호도 낮은 세그먼트를 대체 대상으로 선정한다. 이 방식은 지금까지 위에서 살펴본 미디어 스트리밍 서비스의 두 가지 특성을 모두 고르게 반영하여 효과적인 버퍼 관리를 가능하게 한다.

4.4.1 버퍼 관리를 위한 프로그램 모델

버퍼 관리를 위한 프로그램은 쓰레드 모델을 이용하여 설계 되었다. 쓰레

드 모델은 시스템내의 병렬성을 높여주고 같은 기능을 하는 프로세스 모델에 비해 생성 및 관리에 대한 부하가 감소한다. 이번 장에서 제안 하는 버퍼 관리는 애플리케이션 단계에서 동작하기 때문에 각각의 클라이언트에게 미디어 데이터를 서비스하는 작업들 끼리 버퍼 공간을 공유해야 한다. 해당 기능을 프로세스 모델로 설계할 경우 일반적으로 IPC(Inter Process Communication) 설비를 이용하여 자원을 공유하므로 운영체제의 부담이 증가한다. 반면에 쓰레드는 프로세스와는 다르게 스택 영역만 독립적으로 가지기 때문에 같은 기능을 쓰레드 모델로 설계하면 버퍼 공간에 대한 공유나 새로운 서비스 클라이언트의 생성 면에서 많은 부하가 감소한다.



[그림 16] 프로그램 모델의 쓰레드 동작 과정

그림 16은 설계된 프로그램 모델의 동작 과정을 보여주는 것이다. 그 과정을 살펴보면 데몬 형태로 서비스 요청을 기다리는 메인 쓰레드가 있고 요청이 발생하면 클라이언트에게 서비스를 수행하는 서비스 쓰레드가 생성 된다. 각각의 쓰레드들은 GOP 정보 관리 노드를 통해 해당 미디어 데이터에 대해 접근하게 된다. 이때 읽고자하는 내용이 버퍼 공간에 있으면 버퍼 히트가 읽어나고

해당 내용에 대한 서비스가 바로 이루어지게 되고 버퍼 공간에 내용이 없을 경우 디스크로부터 내용을 읽고 버퍼 공간에 채워 넣은 후 서비스가 이루어지게 된다.

4.4.2 MSS 알고리즘 및 자료구조

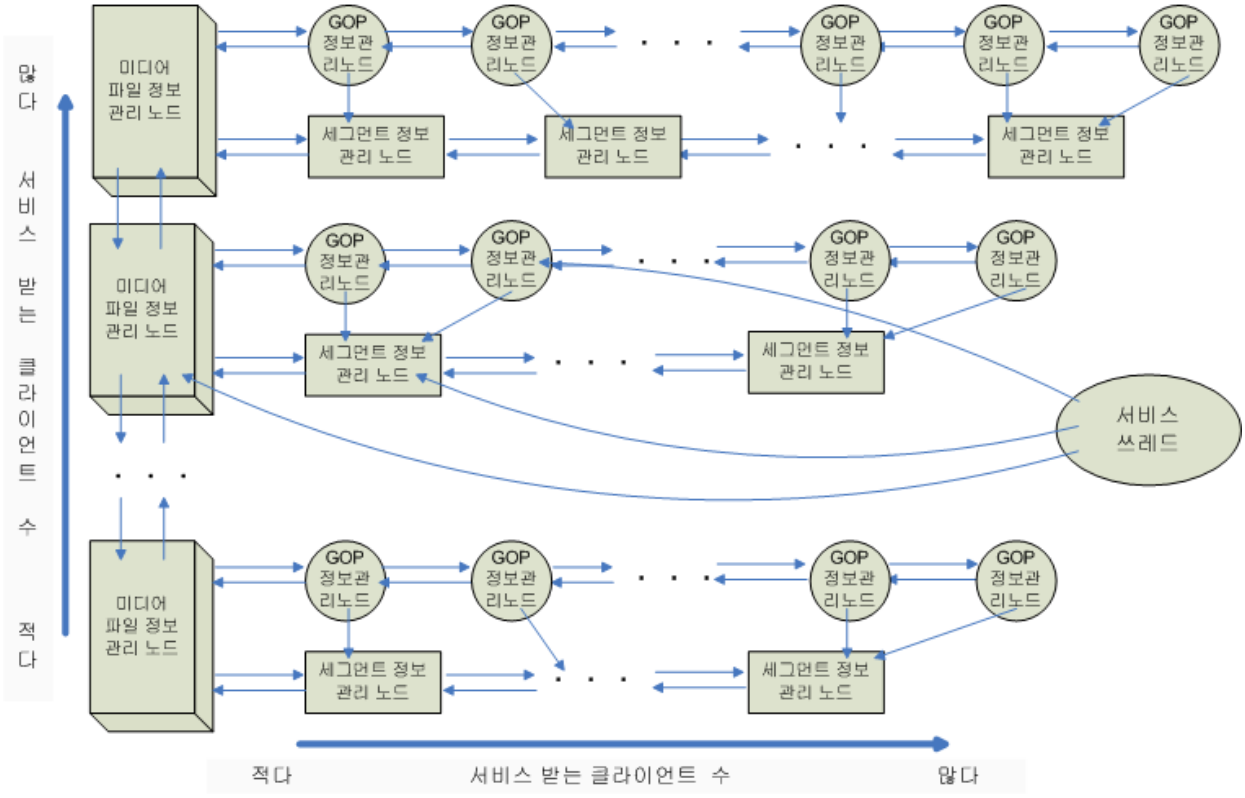
서비스 구현을 위해 사용하는 자료구조는 그림 17과 같다. 해당 자료구조는 쓰레드 안에서 전역 변수 형태로 존재하며 모든 서비스 쓰레드에 의해 참조될 수 있다.

자료 관리를 위해서 크게 세 가지 종류의 구조체가 이용된다.

첫째는 미디어 데이터에 전반적인 내용을 관리하는 미디어 파일 정보 관리 노드이다. 하나의 미디어 데이터에 대해서 이를 관리하는 하나의 미디어 파일 정보 관리 노드가 생성된다. 이 노드는 해당 파일을 서비스 받고 있는 전체 클라이언트 수, 해당 파일의 히트 율, GOP 정보 노드 관리를 위한 포인터, 세그먼트 리스트를 관리하기위한 포인터 등을 관리한다.

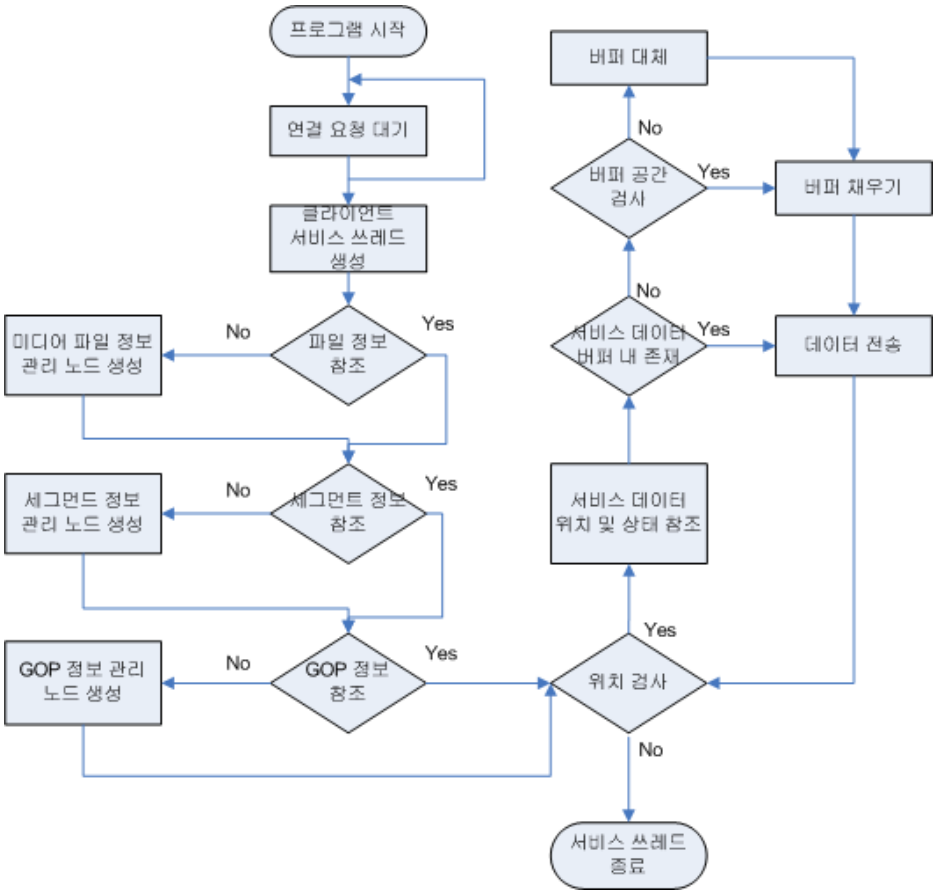
두 번째는 GOP 정보 관리 노드이다. GOP 정보 관리 노드에는 해당 파일의 위치, 미디어에 대한 고유번호, 할당 받은 버퍼의 시작 인덱스, 사용하고 있는 버퍼 수, GOP 번호, 크기, 위치 등의 정보를 담고 있고 실제 데이터를 디스크로부터 읽어 올 때 이용한다.

세 번째는 세그먼트 정보 관리 노드이다. 세그먼트 정보 관리 노드에는 해당 세그먼트를 참조하는 클라이언트 수, 세그먼트 내 히트 율, 사용하고 있는 버퍼 수, 리스트 유지를 위한 포인터 등이 있다.



[그림 17] 정보 관리를 위한 자료구조

그림 17에서 보는 것처럼 모든 노드들은 이중 링크드 리스트로 연결되어 진다. 이 링크를 이용하여 클라이언트 서비스 쓰레드가 생성되거나 삭제될 때 현재 서비스하고 있는 클라이언트 수를 기반으로 오름차순 정렬을 하게 된다. 먼저 미디어 파일 정보 관리 노드에 대한 정렬이 이루어지고 다음으로 파일 내 세그먼트 정보 관리 노드에 대한 정렬을 수행한다. 서비스의 연결이 이루어져 새로운 클라이언트가 추가 될 때나 연결 해제로 인해 클라이언트가 감소할 때 마다 새로 정렬을 수행한다. 이런 동작은 실시간으로 정보를 수정하기 때문에 현재의 선호도를 그대로 정책에 반영할 수 있다. 또한 항상 정렬된 상태를 유지하기 때문에 버퍼 대체 시 대체 대상 노드를 찾고 선정하는 걸리는 부하를 감소시킬 수 있다.



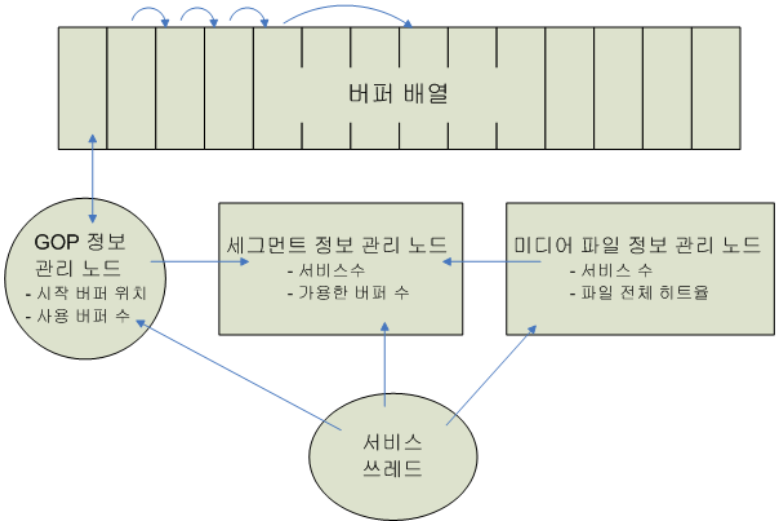
[그림 18] 프로그램 순서도

하나의 서비스 쓰레드가 생성될 때 미디어 데이터에 대한 정보와 서비스를 위한 클라이언트 정보를 실행인자로 받게 된다. 생성된 쓰레드는 해당 미디어 데이터에 대한 미디어 파일 정보 관리 노드가 있는지 먼저 검사한다. 만약 이 쓰레드가 해당 미디어 데이터에 대한 첫 번째 서비스이면 미디어 파일 정보 관리 노드를 새로 생성하게 된다. 다음으로 자신이 속하게 될 세그먼트를 찾는다. 기존에 만들어진 세그먼트에 자신이 속할 수 있는지를 검사하고 조건이 맞지 않거나 자신이 첫 서비스이면 새로운 세그먼트 정보 관리 노드를 생성하게 된다. 그림 18은 서비스 쓰레드의 동작과정을 보여주는 프로그램 순서도 이다. 서비스 쓰레드는 자신이 서비스하는 미디어 데이터에 대한 정보, 자신이 속한 세그먼트 정보, 현재 서비스하고 있는 GOP 정보 등에 대한 포인터를 가지고

해당 서비스를 하게 된다.

4.4.3 버퍼 할당 및 대체

버퍼 메모리에 대한 관리는 세그먼트를 기준으로 이루어지만만 실제 메모리 공간에 존재하는 버퍼는 10KByte 단위로 나뉘어져 있다. 미디어 데이터의 GOP 사이즈가 수십 킬로바이트에서 수백 킬로바이트까지 가변적이고 프로그램 상에서 네트워크로 전송하는 기본 단위가 10KByte기 때문에 버퍼를 10KByte 단위로 나누었다. 버퍼의 기본 단위를 크게 하면 작은 GOP 정보를 저장할 경우 메모리 낭비를 초래한다. 반면에 버퍼의 기본 단위가 너무 작은 경우 빈번한 메모리 복사 동작으로 인해 전체적인 성능을 저하 시킬 수 있다.

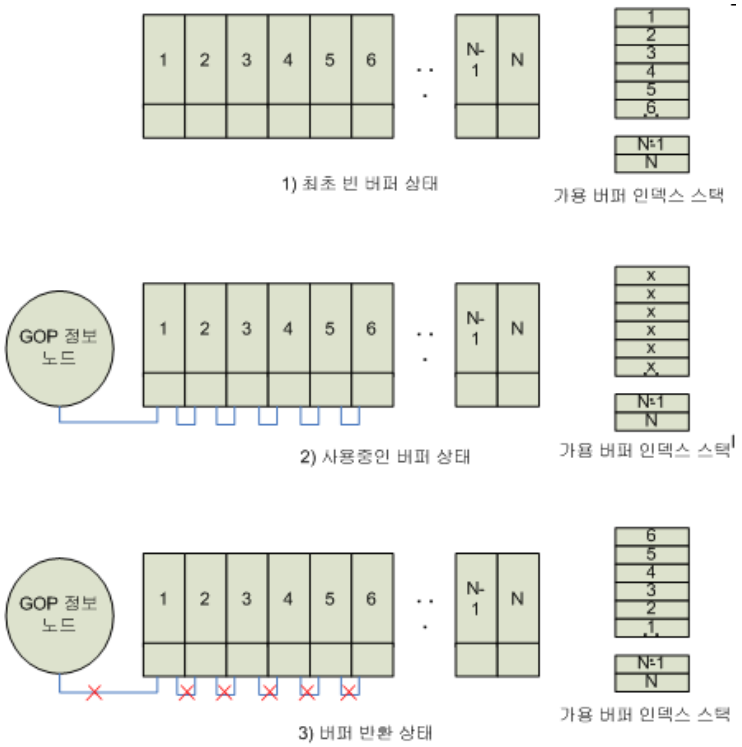


[그림 19] 서비스 쓰레드의 구성

크기가 가변적인 GOP를 관리하기 위해서 버퍼는 다음 부분을 저장하는 버퍼 위치를 알 수 있도록 인덱스를 가지고 있고 이 인덱스를 통하여 다음 버퍼를 참조한다. 그림 19는 하나의 서비스 쓰레드를 이루는 주요 구성 요소들을

보여주고 있다. 하나의 서비스 쓰레드는 자신이 속한 미디어 파일 정보 관리 노드, 세그먼트 정보 관리 노드, GOP 정보 관리 노드를 직접 참조 할 수 있는 포인터를 가지고 있어 필요할 때 바로 참조가 가능하다.

하나의 서비스 쓰레드는 현재 자신이 서비스하고 있는 GOP 정보 노드를 통해서 해당 데이터가 버퍼 내에 있는지 없는지를 알 수 있다. 이 정보를 이용하여 빈 버퍼에 데이터를 채우고 반환하는 작업을 수행 할 수 있다. 그림 20은 서비스 쓰레드에 의해 버퍼에 데이터가 할당 되고 대체 되는 과정을 보여주는 것이다.



[그림 20] 버퍼의 할당과 반환

전역으로 관리되는 버퍼 공간을 할당 받고 반납하는 과정에서 공간의 사용 유무를 알 수 있도록 가용 버퍼 인덱스 스택을 두었다. 버퍼 공간을 할당 받고자 하는 쓰레드들은 이 스택에서 버퍼 인덱스를 가져와서 해당 공간에 데이터

를 채워 넣는다. 이렇게 데이터를 채우게 되면 가용 버퍼 인덱스 스택의 크기는 줄어들게 되고 GOP 정보 노드에는 버퍼의 시작 인덱스를 저장하게 된다. 나중에 스택 내에 가용한 인덱스가 없으면 MSS 정책에 기반하여 선택된 세그먼트에 포함된 버퍼들의 대체가 이루어지게 된다. 어떤 공간이 대체 대상으로 선정되어 공간을 반납해야 할 경우 인덱스를 따라가면서 가용 버퍼 인덱스 스택에 반납하고자 하는 버퍼 공간의 인덱스를 저장하게 된다. 이렇게 반납된 버퍼 인덱스는 버퍼 요구 시 재사용 되는 것이다.

4.5 성능 측정 및 결과

이번 절에서는 MSS 대체 정책을 이용하여 구현된 시스템 성능을 측정하고 그 결과를 보여준다. 먼저 성능 측정에 사용된 환경에 대해 알아보고 결과에 대한 분석을 보여준다.

4.5.1 실험 환경

실험은 미디어 스트림 서버의 노드 수를 4~6개로 늘려가면서 수행되었다. 각 노드에는 MSS 정책을 기반으로 동작하는 미디어 스트림 서버 프로그램이 수행되고 야드 스틱 프로그램과 가상 부하 클라이언트 프로그램을 이용하여 노드에 부하를 걸어주었다. 표 2는 성능 측정 시 사용한 노드와 네트워크 장치의 하드웨어 사양이다.

미디어 데이터는 총 10개의 파일을 사용하였으며 목록은 표 3과 같다. 대부분의 영화가 초당 약 143KB 정도의 대역폭을 요구하는 데이터이고 마지막에 있는 2개의 데이터는 좀 더 많은 대역폭을 요구하는 데이터이다. 가상 부하 클라이언트 프로그램은 해당 영화가 요구하는 대역폭만큼을 인식하여 소비하도록

구성되었다.

CPU	Intel(R) Pentium(R) 4 CPU 1.60GHz
메모리	256MB DDR RAM
디스크	Seagate IDE 40G 7200 rpm 2개
운영체제	RedHat 7.3 (Kernel 2.4.18-4)
네트워크	100Mbps fast-ethernet (Hub : 3Com 3C16465C Super Stack 3, 24port)

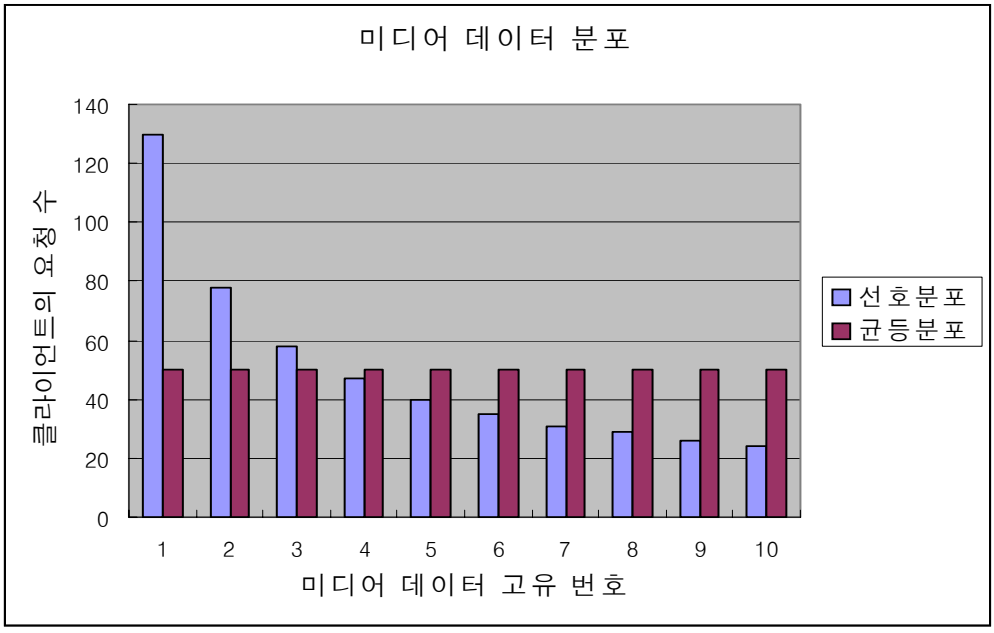
[표 2] 미디어 스트림 서버 노드 및 하드웨어 사양

미디어 데이터 고유 번호	제목	해상도	프레임(fps)	비트율(KB/s)
1	Matrix 3	352x240	29.97	143.8
2	PayCheck	352x240	29.97	143.8
3	Welcom to Jungle	352x240	29.97	143.8
4	Big Fish	352x240	29.97	143.8
5	Load of Ring	352x240	29.97	143.8
6	그녀를 믿지 마세요	352x240	29.97	143.8
7	홍 반 장	352x240	29.97	143.8
8	Spider Man	352x288	25.0	143.8
9	Resident Evil	352x288	25.0	155.8
10	Piter Pan	480x480	29.97	312.5

[표 3] 실험에 사용한 미디어 데이터

버퍼의 크기는 4.2절의 결과를 근거로 100M 바이트를 기준으로 하였다. 4.2절의 실험 결과를 보면 버퍼의 크기가 200M 바이트인 경우도 좋은 성능을 보여주고 있지만 시스템의 메모리가 256M 바이트 이고 버퍼 공간 외에 프로

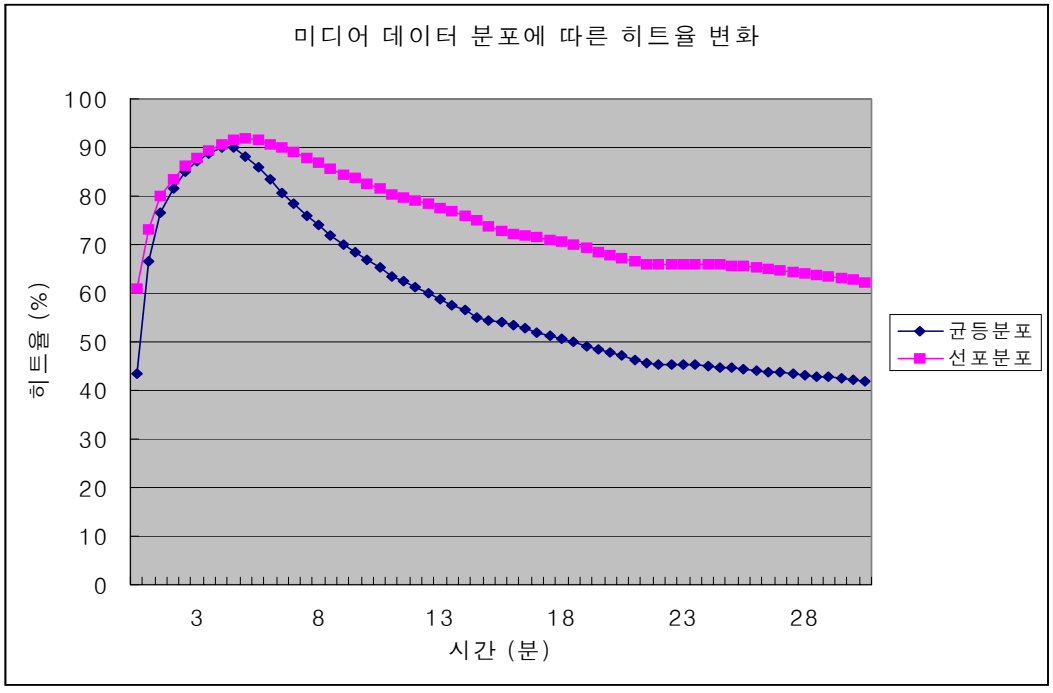
그램에서 추가로 메모리를 필요로 하기 때문에 100M 바이트를 기준으로 버퍼 크기를 정하였다. 또한 4.3절의 세그먼트 분포를 이용하여 하나의 세그먼트는 15개의 GOP로 구성하였다. 세그먼트의 크기는 시간적 국부성에 대한 반영을 결정짓는 중요한 요소인데 4.3절의 그래프에서는 하나의 세그먼트가 GOP 10개인 경우와 20개인 경우를 보여주고 있다. 어떤 것이 효과적인지는 미리 예측하기가 쉽지 않으므로 본 실험에서는 중간 크기인 15를 이용하였다. 사용자 요청에 대한 빈도는 λ가 1인 포아송 분포를 이용하여 초당 1개의 요구가 도착하도록 하였으며 그림 21은 미디어 데이터에 대한 분포를 보여주고 있다. 그림 21에서 보여주는 선호 분포는 skew factor가 0.271 zipf 분포를 말하고 균등 분포는 선호도의 배려 없이 모든 데이터에 대한 요청이 같은 수로 발생된 것을 말한다.



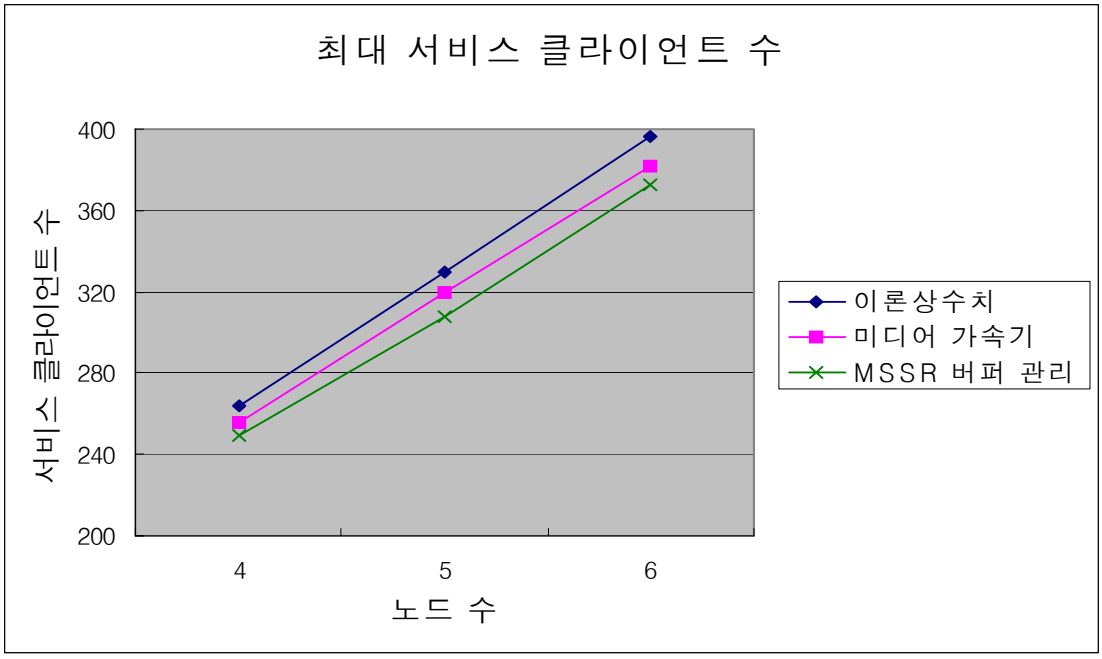
[그림 21] 실험에 사용한 미디어 데이터 선포도 분포

4.5.2 실험 결과 및 분석

그림 22에서 보여 지는 실험 결과는 선호도 반영에 따른 히트율의 변화이다. 영화에 대한 선호도가 버퍼 대체 정책에 어떠한 영향을 미치는지를 나타내고 있다. 버퍼의 크기는 100메가바이트를 할당 하였고 시간의 흐름에 따라 히트율의 변화를 측정하였다. 그림에서 보면 약 5분을 지나는 시점까지는 비슷한 히트 율을 보이며 같이 히트 율이 상승한다. 아직 버퍼 공간에 여유가 있기 때문에 대체가 이루어지지 않는 것이며 히트 율은 계속 상승하게 된다. 약 7분을 지나면서부터 버퍼 공간이 고갈되고 버퍼에 대한 대체가 이루어진다. 이 때부터 두 그래프는 히트율에 차이를 보인다. 사용자의 요청이 특정 영화에 대한 선호도를 따르면 그렇지 않은 경우에 비해 약 20%의 히트율 향상을 보여주고 있다. 실제 서비스에서는 본 실험에서 보다 더 많은 집중현상이 고려되므로 더 좋은 성능을 예측할 수 있다.



[그림 22] 영화 분포에 따른 히트율 변화

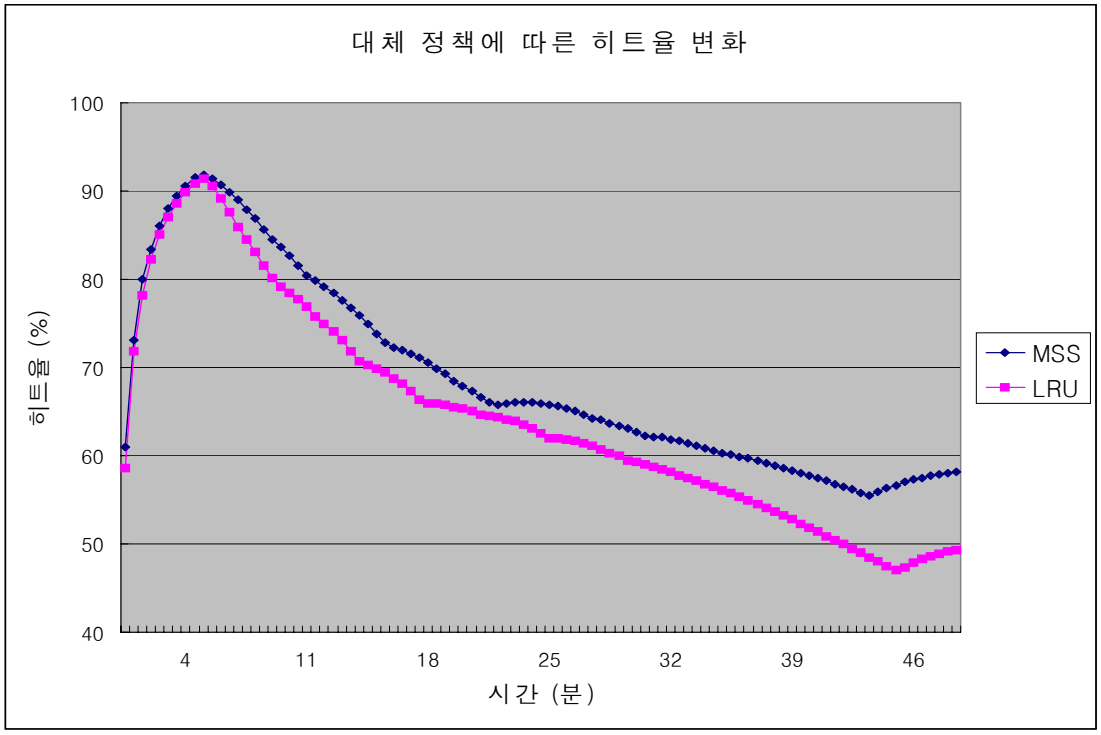


[그림 23] 서비스 가능한 최대 클라이언트 수

그림 23은 서비스가 가능한 총 클라이언트의 수를 나타내고 있다. 그림 23에서 보는 바와 같이 MSS 정책을 사용 시 서비스 수가 이론상 수치에 근접하여 선형적으로 증가하는 것을 볼 수 있다. 정적 버퍼 관리를 수행하는 미디어 가속기의 서비스 수와도 큰 차이를 보이지 않고 있다. 가속기 시스템이 1기가바이트의 메모리를 탑재하고 5편의 영화를 서비스 할 때의 성능이기 때문에 메모리 크기와 서비스 영화수를 고려한다면 MSS 정책이 좋은 성능을 나타내고 있음을 알 수 있다.

이론적으로 서비스 가능한 수치, 가속기, MSS 각각이 약 4-5%의 성능 저하가 나타나고 있다. 이론적인 수치와 차이가 나는 것은 실제 서비스를 하게 되면 운영체제나 네트워크 장비 등에서 데이터를 처리하는 오버헤드가 발생하기 때문이다. 가속기와의 차이 발생은 사용한 미디어 데이터가 다르기 때문이다. 가속기 시스템을 측정할 때는 메모리 제한 때문에 동일한 압축률을 가지는

5개의 영화만 이용하였다. MSS 측정 시에는 보다 능동적인 메모리 관리가 동적으로 이루어 질수 있으므로 압축률이 다른 10개의 영화를 이용하였고 이런 환경적 차이가 서비스를 받는 클라이언트 수에 차이를 발생시킨 것이다.



[그림 24] 대체 정책에 따른 히트율 변화

마지막으로 그림 24는 대체 정책에 따른 히트 율의 변화이다. 본 논문에서 제안하는 MSS 대체 정책과 LRU 대체 정책을 서로 비교 하였다. 기본적인 버퍼의 대체 단위가 세그먼트를 기준으로 하고 있기 때문에 LRU 대체 정책도 세그먼트 기반으로 작성되었다. LRU 정책은 과거 참조 거리를 기준으로 대체 대상을 선정한다. 하나의 세그먼트 안에 여러 개의 서비스 클라이언트가 있을 수 있는데 이런 경우 그 중 하나에 의해 참조가 이루어지면 최근 참조로 갱신된다. 그러므로 많은 수의 클라이언트가 있는 세그먼트는 대체 될 확률이 적어지게 된다. 이런 점만 보면 논문에서 제안하는 선호도 기반의 대체 정책과 차이

가 없다고 생각할 수 있다. 하지만 LRU의 경우는 단지 확률이 적어질 뿐 서비스 받는 클라이언트가 많은 세그먼트가 대체에서 배제되는 것을 완전히 보장해 주지는 못한다. 어느 시점에 많은 수의 클라이언트를 담당하는 세그먼트에 대한 접근이 없을 수 있고 이때 버퍼의 대체가 필요하다면 해당 세그먼트 내 버퍼는 모두 대체가 될 것이다. 하지만 금새 대체된 버퍼의 사용요구가 증가하게 될 것이며 버퍼 미스 또한 증가하게 된다. 이 경우 다시 세그먼트를 생성하고 버퍼를 채우는 비효율적인 작업을 수행해야 한다. 그림 24에서 보는 것처럼 제안된 대체 정책이 LRU에 비해 최고 10% 정도의 높은 히트율을 보여주고 있다.

Ⅴ. 결 론

컴퓨터 기술의 발달과 네트워크 환경의 발달로 인해 사용자의 컴퓨팅 환경에 많은 변화가 일고 있다. 비교적 데이터의 양이 크고 많은 자원을 필요로 하는 미디어 데이터를 다루는 일이 과거에는 힘든 일이었으나 지금은 많은 사용자들이 이용하고 있다. 나아가 더 고화질의 더 멋진 화면을 요구하고 있다.

사용자들의 요구를 충족시키면서 불편을 해소 해소하는 방법이 미디어 스트리밍 서비스이다. 미디어 스트리밍 서비스를 이용하면 사용자들은 원하는 시간에 원하는 서비스를 받을 수 있다. 이런 서비스를 보다 많은 사용자에게 보다 좋은 품질로 제공하기 위해서는 하기위해서는 서버에 많은 자원을 필요로 하지만 서버의 자원은 한정적이다. 아무리 좋은 성능의 서버라 할지라도 무한대의 자원을 가질 수는 없다. 항상 보다 나은 서비스를 위해서는 효과적인 자원 관리가 필요하다. 특히나 디스크에 비해 양이 적고 비싼 메인 메모리에 대한 효율적인 관리가 전체 시스템 성능에 많은 영향을 미치게 된다. 본 논문에서는 미디어 스트리밍 서비스의 특성을 알아보고 그런 특성을 잘 반영하는 대체 정책을 수립하여 효과적인 메모리 관리를 하고자한다.

본 논문에서 효과적인 메모리 관리를 위해 크게 두 가지 특성을 고려하였다. 하나는 데이터에 대한 선호도 이고 다른 하나는 데이터 대한 접근 특성이다. 서버는 다수의 미디어 데이터를 보유하고 사용자 요구가 있을 때마다 해당 서비스를 수행한다. 이 때 사용자의 요구를 살펴보면 많은 수의 요구가 한두 가지로 집중된다. 예를 들면 사람들이 영화나 비디오를 볼 때 지금 유행하는 혹은 새로 나온 몇몇 영화를 집중적으로 본다는 것이다. 또 다른 특성은 데이터의 크기가 상당히 크다는 것이다. 그리고 순차적이면서 반복적인 접근이 주를 이룬다. 이런 특성들은 메모리 관리에서 중요한 역할을 하게 된다.

본 논문에서는 버퍼 메모리에 대해 정직인 환경과 동적인 환경으로 나누어 접근했다. 먼저 정적인 버퍼 관리 환경에서는 미디어 스트림 가속기를 설계 및 구현하여 성능을 측정하여 보았다. 이런 환경은 성능 향상에는 도움이 되지만 또한 문제점을 가지고 있다. 먼저 많은 양의 메모리가 필요하고 현재의 선호도가 바로 반영되지 못하며 주기적인 관리를 필요로 하기 때문에 실제 서비스에 제약이 따르게 된다. 이런 문제점들을 해결하기 위해서 동적인 환경에서 효과적인 대체 정책을 수립하고자 했다. 먼저 선호도에 따른 효과를 알아보고 현재의 선호도를 그대로 반영할 수 있도록 하였고 대용량의 데이터에 대한 순차적, 반복적 접근 특성을 고려하여 비슷한 서비스를 하나의 세그먼트를 묶어서 관리하는 방법을 제안하고 시스템을 구현하여 실험을 해보았다.

실험 결과 제안한 대체 정책을 이용하여 좋은 성능을 보여 주었다. 향후에는 세그먼트를 관리하는 방법이나 대체를 위한 새로운 정책을 수립하여 비교 분석하는 것이 필요하다.

참 고 문 헌

[1] E. J. O'Neil, P.E. O'Neil, G. Weikum, "The LRU-K Page Replacement Algorithm For Database Disk Buffering", Proc. Of the 1993 ACM SIGMOD Conference, pp 297-306, 1993

[2] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H. Noh, Sang Lyul Min, Yook Cho, Chong Sang Kim, "On the Existence of a Spectrum of Policies that Subsumes the Least Recently Used(LRU) and Least Frequently Used(LFU) Policies", Proc. Of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp 134-143, 1999

[3] Yannis Smaragdakis, Scott Kaplan, and Paul Wilson, "EELRU: Simple and Effective Adaptive Page Replacement", Proc. OF the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp 122-133, 1999

[4] 서동만, 방철석, 이좌형, 김병길, 정인범, “QoS를 지원하기 위한 리눅스 클러스터 VOD 서버의 성능 분석,“ 정보과회 제 30회 춘계학술발표회 논문집, 2003

[5] Stergios V. Anastasiadis, Kenneth C. Sevcik, Michael Stumm, “Maximizing Throughput in Replicated Disk Striping of Variable Bit-Rate Streams“, In Proceedings of the 2002 USENIX Annaual Technical Conference , June 10-15, 2002

[6] 안유정, 원유헌, “주문형 비디오 저장 서버에서 디스크 성능을 고려한 저장 시스템의 구조와 비디오 데이터의 특성에 따른 배치 정책”, 정보과학

회논문지(A) 제26권 제11호, pp. 1296-1304, 1999

[7] LeGall, D., "MPEG: A Video Compression Standard for Multimedia Applications", Communicatios of the ACM, Vol. 24, No. 4, pp. 55-63, Apr. 1991

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

[9] 이상호, 문양세, 황규영, 조완섭, “주문형 비디오 시스템에서의 동적 버퍼 할당 기법”, 정보과학회논문지 : 시스템 및 이론 제28권 제9호, pp. 442-460, 2001

[10] C.C.Aggarwal, J.L.Wolf, and P.S.Yu, "On optimal batching policies for video-on-demand storage servers", Proc. of IEEE ICMCS'96, pp.253-258, Hiroshima, Japan, June 1996

[11] K. A. Hua, Y, Cai, S. Sheu, "Patching : A Multicast Techniques for True Video-on-Demand Services", ACM Multimedia '98, 1998

[12] T. Chiueh, M. Vernick, C. Venkatramani, "Performance Evaluation of Stony Brook Video Server”, ECSL-TR-24, February 1997

[13] J. Gafsi and E. W. Biersack, "Impact of Buffer Sharing in Multiple Disk Video Server Architecture”, In Proceedings in the 6th Open Workshop on High Speed Networks, October 1997

[14] W. Shi, S. Ghandeharizadeh, "Buffer Sharing in Video-On-Demand Server", SIGMETRICS Performance Evaluation Review 25, pp. 13-20, 1997

[15] C. Martin, P. S. Narayan, B. Ozden, R. Rastogi, and A. Silberschatz, "The Fellini Multimedia Storage System", Journal of Digital

Libraries, 1997

[16] S. Shen, K. A. Hua, and W. Tavanapong. "Dynamic grouping: An efficient buffer management scheme for video-on-demand servers", Technical Report CSTR-97-02, University of Central Florida, Orlando, Florida, February 1997. CS Technical Report.

A Preference-Based Buffer Management for Large-Volume Media Streaming Service

Cheol-Seok Bang

Department of Computer, Information and Telecommunications Graduate Engineering School of Kangwon National University

Abstract

The media streaming service makes users be able to watch the movie whenever they want. With the convenience for the user and the growth in network environment, the demand for the media streaming service that transmits the media data of high quality in real time is increasing rapidly. Because requests for this kind of service are clustered at specific time, the service requires many resources in the server to provide high quality service to every user. However the resource in the server is limited so that there is the necessity for the method to utilize the limited resource efficiently.

Among the various resources in the server, the main memory which is more expensive but has less capacity than the disk, takes important part in the performance of media streaming server that deals with large-volume media data. So the method to utilize the limited memory is necessary. Increasing the hit ratio for the buffer in the server is one of typical way and selecting the efficient buffer management policy is very important for this.

This paper inquires about the characteristic of media streaming service and shows the performance of the implemented media streaming server reflecting the inquired characteristics. The memory management technique that the buffer replacement is done statically and the dynamic buffer management technique that executes buffer replacement actively are introduced in the paper. And the paper proposes the buffer replacement policy that increases hit ratio for the efficient media streaming service. The performance evaluation shows that the replacement policy reflecting the preference of user and the characteristic of data access contributes for the performance of server.