연속 미디어 서버를 위한 이중 모드 버퍼 캐쉬 관리 기법

서 원 일† · 박 용 운†† · 정 기 동†††

요 약

본 논문에서는 연속미디어 데이터의 접근 유형을 캐싱 정책에 반영하기 위하여 사용자 접근 패턴을 관찰한 다음, 관찰된 접근 유형을 기준으로 데이터의 캐싱 모드를 구간 단위 또는 오브젝트 단위로 운영하는 이중적 버퍼 캐쉬 운영 정책을 제안한다. 시뮬레이션을 통하여 제안한 알고리즘을 평가한 결과 제안한 방법이 기존의 구간 캐싱 방법보다 효율적이고 가변적인 시스템 환경에 안정적인 성능을 보인다는 것을 알 수 있었다.

A Dual Mode Buffer Cache Management Policy for a Continuous Media Server

Won-Il Seo† · Yong-Woon Park†† · Ki-Dong Chung†††

ABSTRACT

In this paper, we propose a new caching scheme for continuous media data where the buffer allocation unit is divided into two modes : interval and object. All of objects' access patterns are monitored and based on the results of monitoring, a request for an object is decided to cache its data with either interval mode or object mode. The results of our simulation show that our proposed caching scheme is better than the existing caching algorithms such as interval caching where the access patterns of the objects are changed with time.

1. 서 론

연속미디어 서버에서 디스크 장치는 메모리에 비하여 저 비용이면서 실시간으로 스트림을 지원할 수 있는 전송 대역폭을 제공하기 때문에 데이터의 저장과 데이터의 전송 대역폭 확보에 필요한 저장 장치로써 널리 사용된다[2, 4, 9, 11, 16]. 그러나 디스크 장치는 데이터 입출력시의 처리율이 데이터의 저장된 위치에 따라 가변적이다. 따라서 이러한 단점을 보완하기 위해 버퍼링 또는 캐싱 기법을 사용한다[6, 13, 15]. 재래식 데이터의 입출력에서는 완료시한(deadline)이라는 개념이 존재하지 않으므로 버퍼 할당 정책에 있어서도 주로 디스크 입출력을 최소화하여 디스크 장치의 수행 능력을 향상시키는 측면만 강조되었다. 그러나 연속미디어 데이터의 경우 데이터 접근시의 주기성(periodicity) 및 데이터 도착의 완료시한이 존재할 뿐만 아니라 저장 장치로부터 사용자에게 연속적으로 전달되어야 한다[11, 14]. 이러한 연속미디어 데이터의 특징을 고려할 때, 연속 미디어 데이터의 버퍼 할당 정책에 있어서 재래식 버퍼 캐싱 정책에서 사용되는 블록 단위의 관리 정책을 사용할 경우 위에서 언급한 연속미디어의

† 정 회 원 : 부산대학교 대학원 전자계산과
†† 준 회 원 : 부산대학교 대학원 전자계산과
††† 종신회원 : 부산대학교 전자계산과 교수
논문접수 : 1999년 7월 12일, 심사완료 : 1999년 11월 12일

특성을 반영하지 못하여 충분한 성능을 기대할 수 없다. 따라서 이 논문에서는 대규모의 연속 미디어 서버에서 데이터의 접근 유형과 전체 저장 서버의 효율성을 충분히 반영하여 각 데이터에 대한 요청 빈도를 기준으로 캐싱의 단위를 구간과 오브젝트 두 가지로 달리하는 정책을 제안한다. 즉, 데이터의 접근 유형을 반영하기 위하여 사용자 접근 패턴을 관찰하여, 각 데이터 별로 데이터 전체를 캐싱할 경우에 한 스트림을 서비스하기 위해 투자되는 평균 버퍼 량(오브젝트 캐싱 효율)과 연속된 스트림 간의 거리(distance)만을 캐싱하는 구간 캐싱 정책을 사용할 경우의 투자되는 평균 버퍼량(구간 캐싱 효율)을 계산하여 구간 캐싱 효율을 초과하는 데이터들은 데이터 전체를 버퍼풀에 캐싱하고, 나머지 버퍼풀은 구간 단위 캐싱 정책으로 운영한다.

본 논문의 구성은 다음과 같다. 제2장에서는 연속미디어 데이터의 캐싱 정책에 관한 관련 연구를 소개하고, 연속미디어 데이터의 버퍼 캐쉬 관리 정책에 관한 기존의 방법들의 문제점을 제시한다. 제3장에서는 본 논문에서 제안하는 버퍼 캐쉬 운영 정책 및 알고리즘을 소개한다. 제4장에서는 시뮬레이션을 통하여 제안한 방법을 분석하고 마지막으로 제5장에서 결론을 내리고 향후 연구 방향을 제시한다.

2. 관련 연구 및 문제점

캐싱 정책의 경우 캐싱되는 데이터의 단위에 따라 다음과 같이 3가지로 분류할 수 있다.

첫째는 블록 단위로 캐싱(block-level caching)하는 정책이다[5]. 재래식 데이터의 경우 대부분의 데이터가 소수의 데이터 블록으로 구성되고 데이터의 입출력 시 완료시한이 존재하지 않고 시간적, 공간적 지역성(locality)이 높으므로 블록단위 캐싱이 적합하나 연속미디어 데이터인 경우 대용량이고 순차적 접근을 하므로 블록단위 캐싱으로는 수행 능력의 향상을 기대하기는 어렵다.

둘째는 구간 단위로 캐싱(interval-level caching)하는 정책이다. 이 방법은 동일한 데이터에 대한 스트림들에 대하여 선발 스트림이 디스크 읽기를 통하여 버퍼에 읽어들인 디스크 블록의 그룹(구간)을 후발 스트림은 디스크 접근 없이 선발 스트림이 버퍼링한 데이터를 읽음으로써 디스크 오버헤드를 줄임과 동시에 스트림의 수를 증가시키고자 하는 방법이다[1, 3, 7]. 구간 캐싱의 경우 동일한 데이터에 대한 스트림이 연속적으로 생성되어야만 구간이 발생하므로 데이터의 시작 블록과 후행 스트림이 도착하여 선행 스트림이 캐싱을 시작하는 시점까지의 간격만큼은 캐싱에서 제외되므로 그 부분에 대하여는 디스크 읽기를 하여야 하는 문제점을 가진다. 또한 구간 캐싱 방법은 동일 데이터의 스트림간의 구간의 길이의 함수로만 캐싱 정책을 수립하였으며, 접근 형태를 고려하지 않기 때문에 빈번한 버퍼의 재할당이 발생할 가능성이 존재하며 디스크의 부하 균형에도 문제가 발생할 수 있다.

셋째는 오브젝트 단위로 캐싱(object-level caching)하는 정책이다. 이 방법은 데이터 전체를 캐싱함으로써 데이터의 연속성 보장성을 가능하고자 하는 방법이다[10]. 그러나 이러한 방법은 오브젝트 단위로 캐싱된 스트림에 대하여서는 디스크 입출력이 제외되므로 디스크 대역폭을 안정적으로 가져갈 수 있고 구간 캐싱에서 구간 정렬에 위한 계산 오버헤드가 존재하지 않지만 데이터 요청의 빈도가 떨어질 때에는 데이터 접근의 순간적인 변화를 반영하지 못하여 버퍼 공간의 효율성이 떨어지는 단점이 존재한다.

3. 연속미디어 뉴스 데이터를 위한 효율적 버퍼 캐싱 정책

3.1 연속미디어 데이터의 접근 빈도와 최근성

재래식 버퍼 캐쉬 운영 방법에서는 버퍼 캐쉬의 적중률을 최대화함으로써 디스크 입출력을 최소화하여 시스템의 수행 능력을 증가시키고자 하였다. 이러한 재래식 데이터 캐싱 방법은 일반적으로 데이터의 참조된 시점(reference time)을 기준으로 데이터의 접근 유형을 반영하여 운영된다. 재래식 데이터의 경우 트랜잭션당 발생하는 데이터의 크기가 작으므로 일정한 양 이상의 버퍼 메모리를 확보할 경우, 참조 시점만을 기준으로 정책을 수립하여도 적절한 버퍼 캐쉬의 운영이 가능하다. 그러나 연속미디어 데이터인 경우 요청된 데이터의 데이터 블록들이 주어진 완료시한 내에 연속적으로 읽혀져야 함으로 재래식 데이터의 버퍼 할당 정책에서처럼 각각의 블록의 참조 시점만을 기준으로 한 블록 단위의 버퍼 캐쉬 관리 정책은 연속적으로 접근되는 데이터의 특징을 고려할 때 적절하지 못하다[3, 12].

연속 미디어인 경우 주기성(periodicity)과 연속성

(sequentiality)이 존재하므로 블록 단위의 적중률보다는 전체 데이터 또는 동일 데이터에 대한 두 개 이상의 연속적으로 하는 스트림 간의 블록들에 대한 적중률이 더욱 중요하며 이를 위해서는 각각의 블록의 참조 유형뿐만 아니라 각각의 데이터에 대한 접근 유형 또한 고려되어야 한다. 이를 위하여 이 논문에서는 연속 미디어 데이터의 접근 최근성과 접근 빈도를 다음과 같은 개념으로 사용한다.

(1) 접근 최근성 : 연속미디어 데이터는 주기적인 접근 형태를 가지므로 같은 주기에 접근하는 모든 데이터 블록은 같은 최근성을 가진다. 구간의 최근성은 구간에 속하는 블록들이 접근된 라운드(round)중 가장 오래된 라운드로 한다.

(2) 접근 빈도 : 연속미디어 데이터는 실시간성이 보장되어야 하고, 연속적인 접근 형태를 가지므로 데이터별 평균 동시사용자 수를 접근 빈도로 한다.

연속미디어 데이터는 실시간성이 보장되는 조건에서 연속적으로 서비스되어야함으로 캐싱 단위는 최소한 동일 데이터를 요구하는 두 스트림 간의 구간이어야 하고, 최대로는 요청된 데이터 전체 크기이어야 한다.

3.2 오브젝트 단위 캐싱 정책과 구간 단위 캐싱 정책의 비교

이번 절에서는 오브젝트 단위 캐싱 정책과 구간 단위 캐싱 정책의 문제점을 제기하고 그 대안을 제안하기 위하여 두 캐싱 방법을 해석적으로 전개하여 비교 설명하겠다. 시스템의 단위 시간당 요청 도착률을 λ라하고, 시스템에서 제공되는 데이터의 수는 N이라고 하고 각 데이터의 크기는 s_i (bytes)이고, 데이터의 재생률은 c_i (byte/sec), 총 재생 시간은 $t_i=\frac{s_i}{c_i}$ 이고, 각 데이터에 대한 접근 확률은 p_i($\sum_{i=1}^{N} p_i=1$)이고 데이터들은 접근 확률이 감소하는 순으로 정렬($p_i \geq p_{i+1}$)되어 있다라고 가정하자. 이때, 각 데이터 i ($1 \leq i \leq N$)에 대한 단위 시간당 요청 도착률을 λ_i 라고 하면, $\lambda_i = p_i \lambda$ 이고, $\sum_{i=1}^{N} \lambda_i = \lambda$ 이다. 따라서 시스템이 지원하여야 할 데이터 i의 스트림의 수는 다음과 같다.

$$u_i = \lambda_i \cdot t_i \tag{1}$$

그러므로, 시스템이 지원하여야할 전체 스트림의 수는 다음과 같다.

$$u = \sum_{i=1}^{N} u_i = \sum_{i=1}^{N} \lambda_i \cdot t_i \tag{2}$$

또, 데이터 i의 요청들의 평균 도착 시간 간격은 다음과 같다.

$$d_i = \frac{1}{\lambda_i} = \frac{t_i}{u_i} \tag{3}$$

따라서, 평균 도착 시간 간격이 d_i 인 경우의 데이터 i 의 스트림의 수는 다음과 같다.

$$u_i = \frac{t_i}{d_i} \tag{4}$$

오브젝트 캐싱 정책은 캐싱 대상 데이터의 선정 기준으로 평균 도착 시간 간격을 사용한다. 즉 버퍼의 크기와 사용자 접근 패턴에 따라 캐싱할 데이터와 디스크 대역폭만으로 서비스할 데이터를 선정한다. 따라서 이러한 선정 기준이 되는 데이터 i의 평균 도착 간격의 기대값을 ad_i라고 하면, 각 데이터가 캐싱될 확률은 식 (5)와 같다.

$$P[ad_i \leq d'] = p[u_i \geq \frac{ti}{d'}] \tag{5}$$

식 (5)는 결국 식 (6)과 같다.

$$P[d_i \leq d'] = 1 - \sum_{j=0}^{\frac{t_i}{d'}} \frac{e^{-\lambda_i \cdot t_i} \cdot (\lambda_i \cdot t_i)^j}{j!} \tag{6}$$

캐쉬가 서비스할 데이터 i의 평균 스트림의 수는 다음과 같다.

$$u_i' = u_i \cdot P[ad_i \leq d'] \tag{7}$$

따라서 데이터 i의 요청 중 버퍼 캐쉬로 서비스 될 평균 스트림의 수는 식 (8)과 같다.

$$u' = \sum_{i=1}^{N} u_i \cdot P[ad_i \leq d'] \tag{8}$$

식 (8)에 식 (6)와 (1)을 적용하면 식(9)와 같다.

$$u' = \sum_{i=1}^{N} \frac{t_i}{d_i} \cdot \left(1 - \sum_{j=0}^{\frac{t_i}{d'}} \frac{e^{-\lambda_i \cdot t_i} \cdot (\lambda_i \cdot t_i)^j}{j!}\right) \tag{9}$$

구간 캐싱 정책은 캐싱 대상 구간의 선정 기준을

두 개의 연속하는 스트림 간의 간격을 기준으로 한다. 구간 캐쉉 정책을 사용할 경우 데이터 i의 두 개의 연속하는 스트림 사이의 거리 id_i가 d'' 이내일 확률은 식 (10)과 같다.

$$P[id_i \leq d''] = 1 - e^{-\lambda_i \cdot d''} \quad (10)$$

식 (1)에 의해 데이터 i를 접근하는 스트림의 수는 u_i개이고, 따라서 이러한 스트림이 형성한 $u_i - 1$개의 구간이 존재하고, 이 중에서 버퍼 캐쉬가 할당된 구간이 $k(0 \leq k \leq u_i - 1)$개 존재한다면, 디스크로 서비스되는 스트림의 수는 $u_i - k$개 존재한다. 즉 구간 캐쉉 기법을 사용할 경우 적어도 하나의 디스크 읽기는 반드시 존재한다. 데이터 i의 j번째 스트림을 $S_{ij}(1 \leq j < = u_j)$라 하고, 데이터 i의 구간 중 캐쉉된 구간의 수를 I_i라 하면, $I_i = k(0 \leq k \leq u_i - 1)$일 확률은 다음과 같다.

$$P[I_i = k] = \binom{u_i}{k} (P[id_i \leq d''])^k (1 - P[id_i \leq d''])^{u_i - k} \quad (11)$$

따라서 캐쉬가 서비스할 데이터 i의 스트림의 수는 식 (12)와 같다.

$$u_i'' = \sum_{k=0}^{u_i - 1} k \cdot p[I_i = k] \quad (12)$$

따라서, 구간 캐쉉 정책을 사용할 경우 버퍼 캐쉬로 서비스될 스트림의 수는 다음과 같다.

$$u'' = \sum_{i=1}^{N} \sum_{k=0}^{u_i - 1} k \cdot p[I_i = k] \quad (13)$$

식(13)에 식(11)과 (10)을 적용하면 다음과 같다.

$$u'' = \sum_{i=1}^{N} \sum_{k=0}^{u_i - 1} k \binom{u_i}{k} (1 - e^{-\lambda_i \cdot d''})^k e^{-\lambda_i d'' (u_i - k)} \quad (14)$$

식 (9)과 (14)를 비교하면, 사용자 접근 패턴에 따라 버퍼가 지원하는 동시 사용자 수가 다름을 알 수 있다. 즉 사용자 접근 패턴에 따라 $u' > u''$인 경우에는 오브젝트 단위 캐쉉 정책이 구간 단위 캐쉉 정책보다 지원 가능한 동시 사용자 수가 많고, 반대인 경우에는 구간 캐쉉 정책이 더 낮은 성능을 보인다는 것을 알 수 있다. 연속미디어 데이터에 대한 사용자의 접근 패턴은 일반적으로 지프(Zipf) 분포를 가지며, 이러한 분포는 요일, 시간대, 사회적 여건에 따라 매우 가변적이다. 따라서 이러한 두 가지 캐쉉 정책을 혼용할 필요가 있다. 즉 다음과 같은 경우를 생각할 수 있다.

$$\hat{u} = \sum_{i=1}^{l} u_i' + \sum_{i=l}^{n} u_i'' \quad (15)$$

본 논문에서는 캐쉬가 지원 가능한 사용자 수를 증가시키기 위해서 기본적으로 식 (15)와 같은 방법을 사용한다. 즉 오브젝트 단위의 평균 구간 값을 기준으로 오브젝트 단위로 캐쉉할 데이터를 선정하고 나머지 버퍼 공간은 구간 단위로 할당하는 이중적 방법을 사용한다. 이것은 결국은 l을 선정하는 문제이다. 단, 제한 조건은 버퍼의 크기이며, 목적은 $\hat{u}$의 극대화이다.

3.3 캐쉉 효율의 계산

본 논문에서 제안한 이중 모드 캐쉉 전략은 기본적으로 각 데이터에 대한 사용자의 과거 접근 유형을 기반으로 한다. 과거의 관찰값을 바탕으로 미래를 예측하는 가장 일반적인 방법은 이동평균법(moving average method)과 지수평활법(exponetial smothing method)이 있다. 이동평균법은 과거의 관찰 값 중 최근의 N개의 관찰 값을 평균하여 미래의 값을 예측하는 방법이다. 이 방법의 문제점은 N값이 크면 사용자 접근 패턴의 변화에 대한 반응이 느리고, N값이 작으면, 지나치게 민감하게 반응한다. 또 N개의 값을 유지하고, 계산하는데 필요한 오버헤드가 크다. 지수평활법의 이동 평균법과는 달리 최근의 자료에 더 큰 가중치를 주는 방법으로써 다음과 같은 식에 의해 구해지며 과거로 갈수록 그 가중치는 기하급수적으로 줄어든다.

$$M(t) = \alpha \cdot C(t) + (1 - \alpha) \cdot M(t-1) \quad (16)$$

α : 평활 계수(smoothing coefficient), $0 \leq \alpha \leq 1$
$C(t)$: 현재의 관찰 값
$M(t-1)$: 직전의 예측 값
$M(t)$: 현재의 예측 값

이 방법은 α의 값에 따라 예측 값이 달라지며 α가 크면 빠르게 작으면 느리게 반응한다. 또, 바로 직전의 값 하나만 유지하면 되므로 계산 오버헤드가 적다. 과거 접근 유형은 오브젝트의 캐쉉 모드를 결정하는데 이용되며, 동시사용자 수와 요청간의 도착 구간 값의 예측은 접근 확률에 의존적이다. 이 논문에서는

데이터 O_i의 평균 동시 사용자 수 u'_i를 예측하기 위해서 식 (17)과 같은 지수 평활법을 사용한다.

$$u'_i(t) = \alpha \cdot u_i + (1-\alpha) \cdot u'_i(t-1) \quad (17)$$

이중 모드 캐싱 정책에서는 선반입할 데이터를 선정하기 위해서 기본적으로 캐싱 효율(caching efficiency)에 기반한다. 캐싱 효율은 투자된 버퍼에 비례한 절약될 수 있는 디스크 대역폭의 양으로 한다. 이를 위해서 먼저 데이터 별로 오브젝트 캐싱 효율을 계산한다. 데이터 O_i의 평균 동시사용자 수가 u'_i이고 데이터의 크기가 s_i이고, 재생률이 c_i면, 이 데이터 전체가 캐싱되어 있을 경우, 평균적으로 하나의 스트림을 서비스하기 위해서 $\frac{s_i}{u'_i}$ 크기의 버퍼가 투자된다. 그러므로 하나의 버퍼 블록을 투자하여 절약하는 디스크 대역폭의 양은 $c_i / \frac{s_i}{u'_i}$가 된다. 따라서, 데이터 O_i의 캐싱 효율 CE_i는 식 (18) 과 같이 구할 수 있다.

$$CE_i = \frac{u'_i \cdot c_i}{s_i} \quad (18)$$

오브젝트 캐싱 대상으로 선정된 오브젝트에 대한 연속하는 요청들이 형성하는 구간을 제외한 각 구간 j의 크기를 d_j 라고 하면, 구간 캐싱으로 인하여 절약되는 디스크 대역폭의 합은 $\sum_{d_j \le d''} c_j$이고 투자된 버퍼의 량은 버퍼풀의 크기는 $\sum_{d_j \le d''} d_j$ 가 된다. 따라서 구간 단위 캐싱 대상의 캐싱 효율의 평균은 식 (19)와 같고, 이는 오브젝트 캐싱 대상을 결정하기 위한 기준이 되는 임계값(Threshold ; CT)으로 사용할 수 있다.

$$CT = \frac{\sum_{d_j \le d''} c_j}{\sum_{d_j \le d''} d_j} \quad (19)$$

제안한 캐싱 정책은 새로운 사용자 요청이 있을 때마다 요청된 데이터의 캐싱 효율 CE_i와 캐싱 임계값 CT를 계산하여, $CE_i \ge CT$ 라면 해당 데이터는 오브젝트 단위 캐싱 대상이 되고, 아니라면 구간 단위 캐싱 대상으로 선정한다. 그리고 오브젝트 단위 캐싱 대상으로 선정된 데이터에 대하여 우선하여 버퍼를 할당하고, 나머지 데이터에 대한 스트림들은 구간 단위 캐싱 정책을 사용하여 운영한다.

3.4 알고리즘

본 논문에서 제안하는 이중 모드 버퍼 캐싱 알고리즘은 데이터에 대한 새로운 요청이 발생할 때마다 데이터에 적용되는 캐싱 모드를 결정하는 알고리즘과 스트림에 버퍼를 할당하는 알고리즘으로 구성된다.

3.4.1 오브젝트 캐싱 모드의 결정

버퍼풀의 크기를 B라고 하고, 오브젝트 캐싱에 할당된 버퍼의 크기를 B_o라 하면, 오브젝트의 캐싱 모드를 결정하는 알고리즘은 개략적으로 다음과 같다

(1) 오브젝트 O_i에 대한 새로운 요청이 도착하면 새로운 스트림 S_{ij}에 관한 정보를 생성한다.

(2) 오브젝트 캐싱효율 CE_i와 구간 캐싱 임계값 CT을 계산한다.

(3) 오브젝트 O_i의 현재 캐싱 모드가 OBJECT이고 $CE_i \ge CT$ 이면 스트림 S_{ij}의 캐싱 상태값을 OBJECT로 설정한다.

(4) 오브젝트 O_i의 현재 캐싱 모드가 OBJECT이고 $CE_i < CT$이면 캐싱 모드의 값을 INTERVAL로 변경하고, $B_o = B_o - s_i$로 재설정하고, 스트림 S_{ij}의 캐싱 상태 값을 INTERVAL로 설정한다.

(5) 오브젝트 O_i의 현재 캐싱 모드가 INTERVAL이고 $CE_i < CT$이면 스트림 S_{ij}의 캐싱 상태 값은 INTERVAL로 설정한다.

(6) 오브젝트 O_i의 현재 캐싱 모드가 INTERVAL이고 $CE_i \ge CT$이면, 오브젝트의 크기가 $B - B_o$보다 작은지 검사한다. 만약 작다면, 캐싱 모드를 OBJECT로 변경하고, $B_o = B_o + s_i$로 재설정한다. 아니라면, 현재 캐싱 모드가 OBJECT인 오브젝트 중 캐싱 효율이 데이터 O_i 보다 낮은 데이터에 할당된 버퍼를 해제하면 충분한 공간이 존재하는지를 검사한다. 존재한다면, 교체 대상이 되는 오브젝트의 캐싱 모드를 INTERVAL로 변경하고, 오브젝트 O_i의 캐싱 모드를 OBJECT로 설정하고, B의 값을 재계산한다. 오브젝트 O_i의 캐싱 모드가 OBJECT이면 스트림 S_{ij}의 캐싱 상태 값을 OBJECT로 설정하고, 아니라면 INTERVAL로 설정한다

3.4.2 버퍼 공간의 할당

버퍼 공간 할당을 위해서 스트림별로 상태 정보를 의미하는 source와 status 변수를 유지한다. source는 데이터 블록을 읽어들일 매체를 의미하는데, DISK와 BUFFER 중 하나의 값을 가질 수 있다 status는 읽어들인 데이터 블록을 다음 스트림을 위해서 유지할 것인지 아닌지를 의미하는 변수인데, CACHE와 NOCAHE 중 하나의 값을 가질 수 있다.

S_{ij}를 오브젝트 i에 대한 j번째 스트림을 의미하고, 고 할 때, 버퍼 공간 할당 알고리즘은 다음과 같다.

(1) 스트림 $S_{i(j-1)}$의 캐쉉 상태가 OBJECT이고 스트림 S_{ij}의 캐쉉 상태 값이 OBJECT이면 S_{ij}의 source와 status를 BUFFER와 CACHE로 설정한다.

(2) 스트림 $S_{i(j-1)}$의 캐쉉 상태 가 INTERVAL이고 스트림 S_{ij}의 캐쉉 상태 값이 OBJECT이면 데이터 O_i의 모든 스트림의 source와 status를 BUFFER와 CACHE로 설정한다. 구간 캐쉬의 크기를 $B-B_o$로 설정하고 캐쉉 구간을 재 선정한다.

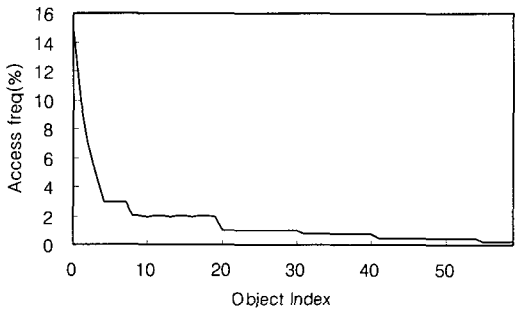
(3) 스트림 $S_{i(j-1)}$의 캐쉉 상태가 OBJECT이고 스트림 S_{ij}의 캐쉉 상태 값이 INTERVAL이면 S_{ij}의 source와 status를 BUFFER와 NOCACHE로 설정한다.

(4) 스트림 $S_{i(j-1)}$의 캐쉉 상태가 INTERVAL이고 스트림 S_{ij}의 캐쉉 상태 값이 INTERVAL이면 S_{ij}를 위한 버퍼 공간의 크기 만큼의 가용공간이 있는지를 조사한다. 존재한다면 $S_{i(j-1)}$의 status는 CACHE로 아니라면, 구간 캐쉉 상태에 있는 스트림 중, 이들을 위해 할당된 버퍼의 크기의 값이 S_{ij} 보다 큰 값을 가지는 스트림들에 할당된 버퍼 공간을 해제하여 캐쉉 공간을 확보할 수 있는지 검사하여 가능하면, 이러한 캐쉬 공간을 해제하고, $S_{i(j-1)}$의 status를 CACHE로 설정한다. $S_{i(j-1)}$의 status가 CACHE이면 S_{ij}의 source를 BUFFER로, 아니면 DISK로 설정한다. S_{ij}의 status는 NOCACHE로 설정한다.

4. 시뮬레이션

4.1 시뮬레이션 파라메터

본 시뮬레이션에서는 논문에서 제안한 버퍼 캐쉬 방법의 성능을 평가하기 위하여 멀티미디어 주문형 뉴스 시스템(Multimedia News On Demand) 저장 서버를 가정하였다. 인터넷 신문의 경우 현재 가장 널리 서비스되고 있는 인터넷 응용의 하나이며 지금은 주로 이미지나 텍스트 위주의 데이터가 주류를 이루지만 향후 인터넷의 발전 추이를 고려할 때 멀지 않은 장래에 연속형 미디어 데이터가 주류를 이루게 되기 때문이다 [8]. 또한 인터넷 신문의 경우[12]에 나타난 것처럼 사용자의 접근 패턴이 알려져 있어서 본 연구에서 제안한 알고리즘의 성능 평가를 위하여 적절하였다. 본 시뮬레이션을 위하여 위해서 유닉스 시스템 환경에서 C 언어를 사용하여 가상의 주문형 시스템 저장 서버를 시뮬레이터하기 위한 시뮬레이션 프로그램을 작성하였다. [17]에서 조사한 인터넷 신문의 접근 패턴은 (그림 1)에 나타나있다.



(그림 1) 사용자 접근 패턴 분포

[12]의 조사에 의하면 하루에 생성되는 뉴스 데이터의 개수는 총 600개 정도이고 하루 평균 10,000건 정도 접근되며 이 중에서도 사용자가 10번 이상 언급한 경우는 60개 정도였다. 따라서 이를 근거로 총 데이터의 수를 60개로 정하였다. 데이터의 크기는 본 실험을 위하여 국내외의 뉴스 길이를 측정한 결과 대부분의 경우 2분 내외의 길이라는 사실에 근거하여 60개의 뉴스 데이터는 각각 30 MB의 크기가 5개, 60 MB의 크기가 10개, 90 MB의 크기가 20개, 120 MB의 크기가 15개, 150 MB의 크기가 10개로 정하였다. 각 스트림은 모두 동일한 재생률(8 Mbps/sec)로 재생된다고 가정하

였다. 또 데이터 별 동시 사용자수를 계산하기 위한 평활 계수 α값은 0.33으로 가정하였다.

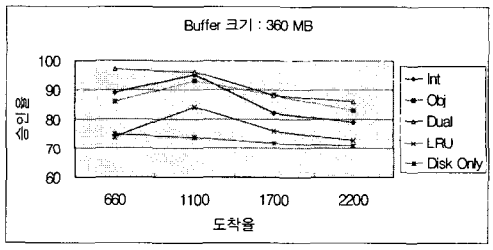
사용자가 뉴스를 보기 위해 서버에 머무르는 시간은 평균 10분으로 가정하였으며 시뮬레이션을 위한 사용자의 시간당 도착률은 660, 1100, 1700, 2200으로 Poisson 분포를 따른다고 가정하였다. 본 시스템에서는 디스크 1대당 가격과 메모리 1 MB의 가격 비를 1 : 18로 두고 계산하였다. 따라서 디스크로만 저장 서버를 구성하였을 경우를 기준으로하여 디스크 한 대를 제외함으로써 얻는 메모리의 양을 180MB, 2대를 제외하였을 경우 360MB 등으로 계산하였다. 구체적인 시뮬레이션 파라메터는 <표 1>과 같다.

〈표 1〉 시뮬레이션 파라메터

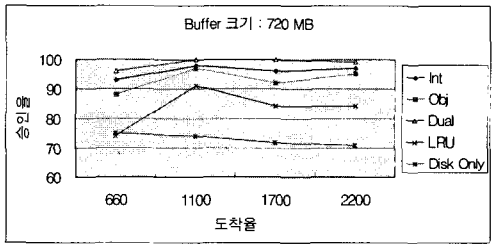
파 라 메 터	값
디스크 저장 공간	3.2 GB
디스크 평균대역폭	10 MBytes/sec
뉴스 데이터의 수	60 개
뉴스 데이터의 길이	30~150 MB

4.2 버퍼풀의 크기에 따른 사용자 승인율의 변화

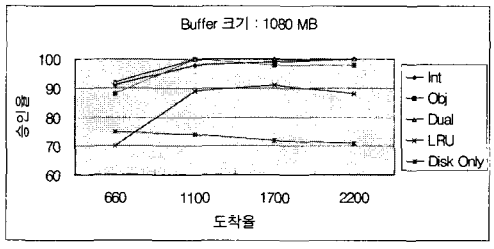
(그림 2)~(4)에서는 버퍼풀의 크기를 고정했을 때, 단위 시간당 사용자의 도착률의 변화가 사용자의 뉴스 데이터에 대한 요청 건수의 승인율의 변화를 나타내었다. 여기서 승인율이란 데이터를 읽기 위하여 디스크 접근을 한 전체 요청 건수 중 디스크 대역폭이 존재하지 않아서 기각되는 요청의 건수를 제외한 승인된 건수의 비율을 의미한다. 그림에서 Int는 구간 캐싱, Obj는 오브젝트 캐싱, Dual은 이중 모드 캐싱, LRU는 LRU 방식의 블록단위 캐싱을 의미하고, Disk Only는 총 시스템 비용을 디스크에만 투자했을 경우, 즉 디스크만으로 사용자 요청을 처리한 경우를 의미한다. 대부분의 경우 버퍼 캐쉬를 사용하였을 경우가 디스크만으로 지원한 경우 보다 성능이 우수하게 나왔다. 단 모집단이 작은 경우 즉, 660인 경우에는 디스크만으로 서버를 운영하였을 때 보다 버퍼 캐쉬를 사용 적용하였을 경우의 효과가 낮게 나왔다. 그 이유는, 모집단이 작음으로 인해서 예상되는 최고 동시 스트림의 수도 작아져서 이를 지원하기 위한 디스크 수가 상대적으로 작아진. 버퍼 캐쉬를 사용하는 경우 디스크 비용의 일부분을 버퍼캐쉬를 위하여 할당하였으므로, 사용자의 뉴스 데이터에 대한 요청 빈도수가 적어지면 재 사용될 확률이 떨어지기 때문이다. 반면에, 버퍼의 크기가 일정한 수준 이상인 경우 즉, 720MB 이상인 경우 버퍼를 사용하였을 경우가 디스크만으로 서비스한 경우에 비하여 거의 모든 경우에 있어서 성능이 우수하게 나왔다. 버퍼의 크기가 상대적으로 작고, 도착률이 높은 경우에는, 오브젝트 캐싱이 구간 캐싱보다도 성능이 우수하게 나왔는데, 이는 사용자 접근 패턴의 편기(skew)의 정도가 높을수록 오브젝트 캐싱 정책이 버퍼 교체 오버헤드가 없기 때문이다. 모든 경우에 있어서 성능은 거의 이중 모드 캐싱 정책을 사용하였을 경우의 승인율이 높았다. 이는 본 논문에서 제안하는 이중 모드 캐싱 알고리즘이 단순하게 오브젝트 캐싱이나 구간 캐싱 방법을 사용하는 것 보다, 버퍼를 더욱 효율적으로 이용하고 있다는 것을 보여주는 것이다. 즉 버퍼의 양과 데이터의 선호도에 따라 일정한 수준 이상의 데이터는 데이터 전체를 캐싱하고, 나머지 데이터에 대해서는 구간 캐싱을 하는 것이 효과적임을 알 수 있다.



(그림 2) 버퍼풀 크기에 따른 승인율(1)



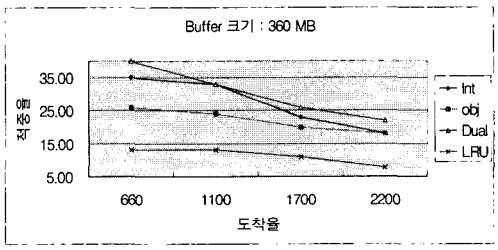
(그림 3) 버퍼풀 크기에 따른 승인율(2)



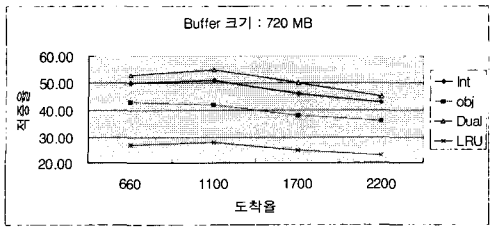
(그림 4) 버퍼풀 크기에 따른 승인율(3)

4.3 버퍼풀의 크기에 따른 버퍼 적중률의 변화

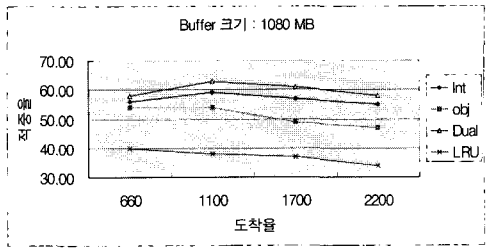
(그림 5)~(7)에서는 버퍼풀의 크기를 고정했을 때, 시간당 사용자의 도착률의 변화가 버퍼 적중율에 미치는 영향을 측정한 결과를 나타내었다. 버퍼 적중율은 전체 요청 중에서 버퍼 캐쉬에서 적중되어 디스크로 가지 않고 버퍼 캐쉬에서 서비스되는 요청 건수이다. 버퍼의 적중율은 승인율에서와 마찬가지로 이중 모드 캐싱 정책이 구간 캐싱, 오브젝트 캐싱, LRU 캐싱 방법 등의 다른 캐싱 정책보다 순서로 높다는 것을 알 수 있다. 그러나, 도착률이 증가함에 따라 각 정책간의 적중률의 차이가 상당히 있음에도 불구하고, 승인율은 점점 비슷해지는데, 이는 연속미디어 데이터인 경우 적중률이 높다고 해서 반드시 승인율이 높지 않다는 것을 의미한다. 즉, 구간 단위 캐싱과 같이 적중율만을 높이기 위한 정책은 버퍼 교체 오브헤드가 발생할 수 있고, 이는 대역폭 낭비로 이어진다는 것을 의미한다. 또, 사용자 도착률에 비하여 버퍼가 충분한 경우, 오브젝트 캐싱 방법만으로 버퍼를 운영하는 경우 구간 캐싱에 비하여 적중률이 낮은데, 이는 오브젝트 단위로 버퍼를 할당함으로써 버퍼로써 서비스되는 데이터의 수가 적게되어 전체 효율이 떨어지기 때문이다. 따라서 연속미디어 데이터의 캐싱은 적중률만을 목적으로 해서는 안되며 디스크 대역폭의 효율적 측면에서 고려되어야 한다는 것을 알 수 있다.



(그림 5) 버퍼풀 크기에 따른 적중율(1)



(그림 6) 버퍼풀 크기에 따른 적중율(2)



(그림 7) 버퍼풀 크기에 따른 적중율(3)

5. 결 론

다수의 사용자에게 연속 미디어 데이터를 제공하는 저장 서버 수용 가능한 스트림의 수는 디스크 대역폭의 제한을 받게 된다. 이러한 디스크 대역폭의 한계를 극복하기 위하여 재접근 가능성이 높은 데이터를 버퍼풀에 캐싱하는 방법을 사용함으로써 디스크의 입출력 한계를 극복하여 효과적으로 저장 서버를 운영할 수 있다. 캐싱 정책의 효율은 재접근 가능성이 높은 데이터를 선정하는 방법에 의존적이다. 연속미디어 데이터는 대용량이며 접근유형이 다양하며 시간에 따른 접근 빈도수의 편차가 심하다. 따라서 캐싱 정책은 이러한 특징을 잘 반영하여야 한다. 본 논문에서 제안한 방법은 오브젝트 단위 캐싱과 구간 단위 캐싱 방법을 절충하여 각 데이터별로 평균 스트림 수를 관찰하여, 각 데이터 캐싱 효율이 캐싱 임계값 보다 큰 경우 이러한 데이터는 데이터 전체를 버퍼링하고, 그렇지 않은 경우에는 데이터들에 대한 요청은 연속된 스트림 사이의 거리를 기준으로 스트림간의 간격을 캐싱하는 이중 모드의 버퍼 관리 정책을 제안한다. 시뮬레이션을 통하여 제안한 알고리즘을 평가한 결과 제안한 방법이 기존의 오브젝트 캐싱이나 구간 캐싱 방법보다 평균적으로 5%~10% 정도 이상의 추가적인 요청을 지원할 수 있다는 것을 알 수 있었다.

참 고 문 헌

[1] A. Dan, D. Sitaram and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," In Proceedings of the ACM Multimedia, pp.391-398, 1994.

[2] A. Dan, M. Kienzle and D. Sitaram, "A Dynamic

Poilcy of Segment Replication for Load-Balancing in Video-On-Demand Servers," ACM Multimedia System, Vol.3, No.3, 1995, pp.93-103.

[3] A. Dan and D. Sitaram. "Multimedia Caching Strategy for Heterogeneous Application and Server Environments," IBM Research Report, RC 20670, Yorktown Heights, NY, 1996.

[4] Edward Chang and H. G. Molina, "Cost-Based Media server Design," Dept Of CS, U. Of Wisonsin, 1997.

[5] M. Bach, 'The Design of the Unix Operating System,' Englewood Cliffs, NJ, 07458 : Prentice Hall, Inc, 1986 pp.220 221.

[6] Raymod T. Ng, Jinhai Yang, "An Analysis of Buffer Sharing and Prefetching Techinques for Multimedia Systems," Tech. Report, Dept. of CS, Univ. of British Columbia, Ca, 1995.

[7] R. Tewari, A. Dan, etal, "Buffering and Caching in Large-Scale Video Servers," Tech-Reports, Texas Austin, 1995.

[8] R. Tewari, H. M. Vin, A. Dan and D. Sitaram, "Resource Based Caching for Web Servers," In Proceedings of ACM/SPIE Multimedia Computing and Networking, pp.191-204, January 1998.

[9] W. Shi and S. Ghandeharizade, "Trading Memory for Disk Bandwidth in Video On Demand Servers," In Proceeding of International Conference on Multimedia Computing and Systems, pp.172-180, Jun 1996.

[10] Y. Park, W. Seo and K. Chung, "Three-phase Interval Caching for a News On Demand Server," In Proceeding of the Communication Networks and Distributed Systems Modeling and Simulation Conference(CNDS), Jan, 1999, pp.25-29.

[11] Mohan Karmath and K. Ramamritham and D. Towsley, "Continuous Media Sharing in Multimedia Database Systems," Proceedings of the Internaltional Conference on Database Systems for AdvancedApplications(DASFAA'95), April 1995.

[12] B. Özden , R. Rastogi and A. Silberschatz, "Buffer Replacement Algorithm for Multimedia Storage Systems," In Proceeding of the IEEE International Conference on Multimedia Computing and Systems, Jun 1996, pp.172-180.

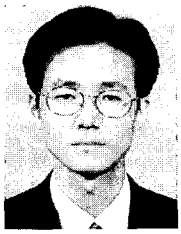
[13] Kun-Lung Wu and P.S. Yu, "Consumption-Based Buffer Management for Maximizing Systems Throughput of a Multimedia Systems," In Proceeding of the IEEE International Conference on Multimedia Computing and Systems, Jun 1996. pp.164-171.

[14] D.J. Gemmell, H.M. Vin, D.D. Kandlur, P.V. Rangan and L.A. Rowe, "Multimedia Storage Servers : A Tutorial," IEEE Computer Magazine, May 1995, pp.40-49.

[15] Huanxu Pan, L.H. Ngoh and A.A. Lazar, "A Buffer-inventory-based Dynamic Scheduling Algorithm for Multimedia-on-demand Servers," IEEE Multimedia Systems, 1998, Vol.6 pp.125-136.

[16] 류연승, 고건, "가변 데이터 율을 갖는 연속미디어 데이터를 위한 동적 버퍼 관리 기법", 한국 정보과학회논문지(A) 제25권 제3호, 1998. 3.

[17] 백건효, 박용운, 서원일, 정기동, "뉴스 비디오 데이터의 효율적인 운영 정책 수립을 위한 사용자 접근 패턴 분석", 한국 정보과학회 춘계학술 발표회, 제25권 제1호, pp.485-487, 1998.



서 원 일

e-mail : wiseo@jisan.ac.kr

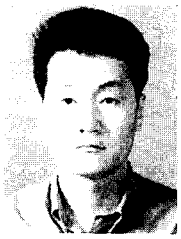
1988년 서강대학교 전자계산학과 졸업(학사)

1993년 서강대학교 전자계산학과 졸업(석사)

1995년~1997년 부산대학교 전자계산학과 박사 수료

1993년~현재 지산 대학 전산 정보처리과 조교수

관심분야 : 멀티미디어, 병렬 파일 시스템, 캐싱 등



박 용 운

e-mail : ywpark@melon.cs.pusan.ac.kr

1988년 부산대학교 계산통계학과 졸업(학사)

1988년~1995년 ㈜LG-EDS 근무

1997년 부산대학교 전자계산학과 졸업(석사)

1997년~현재 부산대학교 전자계산학과 박사과정

관심분야 : 병렬 파일 시스템, 멀티미디어, 캐싱

정 기 동

e-mail:kdchung@hyowon.cc.pusan.ac.kr

1973년 서울대학교 졸업(학사)

1975년 서울대학교 대학원 졸업(석사)

1986년 서울대학교 대학원 계산통계학과 졸업(박사)

1990년~1991년 MIT, South Carolina 대학 교환교수

1978년~현재 부산대학교 전자계산학과 교수

1993년~현재 부산대학교 부설 컴퓨터 및 정보통신 연구소 운영위원

관심분야 : 병렬처리, 멀티미디어