# 구간별 사용자 요구 패턴을 이용한 NOD 에서의 캐싱 방법

○최태욱, 박용운, 김영주, 정기동
부산대학교 전자계산학과

# NOD Caching Strategy using User-Preference Pattern for Time-Window

Tae-uk Choi, Young-woon Park, Young-ju kim, Ki-dong Chung
Dept. of Computer Science, Pusan National University

## 요 약

NOD 데이타는 VOD 데이타에 비해서 life cycle 이 짧다. 그리고 사용자의 접근성이 높으며, 접근패턴도 시간에 따라 달라질 수 있다. VOD 데이타와 같이 NOD 뉴스기사의 경우 특정 기사들에 집중적으로 접근되다. 그리고 이러한 인기있는 기사들은 시간대에 따라 변할 수 있다. 본 논문에서는 이러한 인기도의 변화를 예측하기 위해서 시계열분석방법중의 하나인 지수평활법(exponenital smoothing method)을 사용한다. 시간대별 타임윈도우로 나누고 이전의 윈도우들의 접근패턴을 분석하여 다음 접근을 예측한다. 그리고 이 예측값을 이용해서 캐시정책을 새운다. 즉 예측값이 높은 기사순으로 캐시에 배치하는 것이다. 실시간 멀티미디어데이타의 경우 데이터의 방대함으로 연산의 오버헤드가 크다. 따라서 정적인 캐싱전략을 사용하데, 하나의 윈도우동안 재배치는 한번으로 한다는 것이다.

전통적인 block 단위 캐싱은 멀티미디어데이타에 적합하지않다. 따라서 기사단위의 캐시구조를 제안한다. 사용자는 기사단위로 요청을 하기 때문에 재사용을 위해서는 기사단위로 캐시되야 한다.

### 1. 서론

멀티미디어 처리기술과 통신기술의 발전으로 인터넷상에서 실시간 멀티미디어 데이터를 제공하는 것이 점점 대중화되어가고 있다. [97 Tew]에 따르면 향후 5 년안에 인터넷을 통하여 접근되는 데이터의 50%가 실시간멀티미디어 데이터가 될 것이라 한다. 이러한 멀티미디어 정보를 제공하는 대표적인 서비스로는 VOD 와 NOD 를 들 수 있는데, 이 두 가지 시스템에서의 데이터는 상당한 차이가 있다.

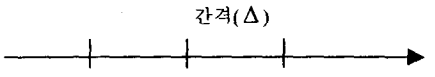
첫째로 데이터의 수명(life span)이다. 비디오프로그램의 경우 매일 새로운 프로그램이 출시되기 보다는 일주일 내지는 보름등의 비교적 긴 주기로 출시된다. 그러므로 사용자들의 비디오 프로그램 선호도도 그 주기와 함께 서서히 변하는 속성을 가진다. 반면 뉴스데이타의 경우 매일 많은 기사가 새로 생성되기 때문에 그 선호도의 주기가 짧다. 이들이 경과한 뉴스의 경우 거의 요구될 가능성이 없다.[97 이]

둘째로 사용자의 접근성이다. VOD 의 사용자가 데이터를 한번 요구하게 되면 VCR 기능 외에는 특별한 사용자의 추가적인 요구가 필요 없다. 그러나 NOD 데이타의 경우는 다르다. 사용자는 언제든지 다른 기사를 원할 수가 있다. 즉 사용자의 접근성이 VOD 에 비해서 높다.

셋째, 데이터의 재사용률이다. 비디오데이타의 경우 인기가 높다고 하더라도 데이터의 크기가 방대한 만큼

전체 데이터를 전부 캐싱하지는 못한다. 하지만, 뉴스데이타의 경우 접근빈도가 높은 기사들을 캐시에 올리는 것이 가능하다.

NOD 데이타의 경우는 사용자들의 접근 패턴을 예측할 수 있다면 효율적인 캐싱전략을 세울 수가 있다. 이를 예측하기위해서는 과거의 접근패턴을 알아야 된다. 이를 위해서 사용자의 접근요구를 시간대별로 나누고 각 시간대단위로 접근패턴을 분석해야 한다.




*W(i) : T-1 에서 T(현재)까지의 시간 간격Δ*

[그림 1]

이러한 방법중의 하나가 시계열 분석 인데, 이를 이용하면 과거의 시간대별 기사들의 접근 패턴을 분석하여 미래의 시간대에 기사의 접근 빈도를 예측할 수 있다. [그림 1]에서 보면, 이전의 n 개의 윈도우 W(T) ~ W(T-n-1)에서의 기사들의 접근빈도를 가지고 다음윈도 W(T+1)의 접근빈도를 예측할 수 있다.

NOD 데이타와 같은 멀티미디어 실시간 데이터의 경우는 그 크기가 방대하여 연산을 하는데, CPU 오보헤드가 크다. 따라서 간단한 캐시알고리즘이 효율적이다. 따라서 정적인 재배치방법을 수행하는데, 이는 하나의 윈도우동안에 재배치는 한번만 일어나는 것을 말한다. 즉 이전 윈도우의 인기도를 반영하여 접근빈도가 높은 기사들을 다음 윈도우동안에 캐시에 올리는 것이다. 캐시에 올라간 기사집합은 다음 시간간격(Δ)동안 변하지 않는다.

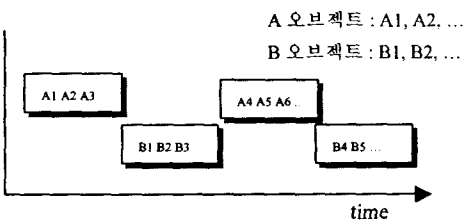
## 2. 관련 연구

실시간 멀티미디어 데이터의 캐싱이나 버퍼링에 관한 연구는 데이터의 재사용률을 높이는 목적보다는 디스크 스케쥴링 측면에서 디스크 사용률을 높이기 위한 목적이었다. [95Tew]에서는 일정시간구간을 두어 버퍼링하는 구간캐싱방식으로 임의의 비디오 오브젝트가 요청되었을 경우 해당 비디오 오브젝트를 디스크로부터 읽어서 캐쉉한다. 그리고 일정 기간동안 그 비디오를 요구하는 사용자가 있을 때에는 후속 사용자는 캐쉉된 데이터를 읽으면 선행 사용자는 캐싱을 계속해 나가고 그렇지 않을 때에는 캐싱을 멈추는 방식이다. 이 경우 요청된 비디오가 다양하고 캐쉉된 비디오를 요청하는 사용자 수가 많지 않을 때에는 방대한 메모리만을 요구할 뿐 효율은 높지 않다.

[95Kun]의 경우는 뉴스데이타를 대상으로 하고 있으나 데이터의 재사용률보다는 디스크의 효율적인 사용을 위한 부가적인 메커니즘 쪽에 초점을 마추고 있다. [97Wei]에서는 비디오 데이터의 버퍼링을 재사용보다는 입출력시에 발생하는 병목현상 해결을 위하여 여러 비디오 요청들 사이에 버퍼를 적절히 분배하여 전체 성능향상을 얻고자 한다. [97 박]에서는 사전에 사용자의 접근패턴을 조사하여 접근 가능성이 높은 비디오 오브젝트 전체를 프리패칭한다. 이 방법은 디스크 효율을 높이고 데이터 재사용률을 높인다는 측면과 오브젝트단위의 버퍼링전략을 사용한다는 데에 있어서는 상당한 의미를 가지나, 실제로 프리패칭할 대상을 결정하는 것은 그렇게 쉬운 문제가 아니고, 요구도 동적으로 변하는 상황에는 적합하지 않다.

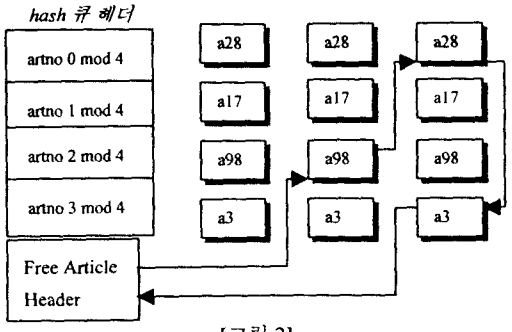
## 3. 캐시 구조 (Article Cache)

전통적인 캐시정책에서는 block 크기단위로 하였지만, 멀티미디어 데이터의 경우 그 크기가 방대하여 block 단위로 캐싱할 경우 효율이 좋지않다. 동영상 뉴스 데이터의 경우 최적의 블록사이즈로 알려진 256KB를 블록단위로 계산할 경우 2 분짜리 MPEG II 뉴스데이타는 120Mbytes/256Kbytes=480 개의 블록이 된다. 따라서 블록단위로 접근을 할 경우 그림 3 에서와 같이 디스크에서 버퍼로 읽어오는 순서에 의해 한 오브젝트의 중간 블록이 버퍼 재할당에 의해 다른 오브젝트로 넘어가는 경우가 생긴다. [박 98]




[그림 1]

하나의 기사는 실시간 데이터와 비실시간 데이터를 포함한다. 실시간 데이타들은 비디오나 오디오를 말하며, 비실시간 데이터들은 텍스트나 정적인 이미지를 말한다. 이들 모두는 하나의 기사를 이루는 요소들이다. 따라서 사용자가 그 기사에 대한 요청을 했을 때 이 모든 요소가 클라이언트로 전송되어야 하며, 또한 재사용을 위해서 캐시되야한다.




[그림 2]

위의 그림은 Article Cache 의 구조이다. 여기서 hash 큐 헤더는 해당 기사들을 쉽게 찾기위해서 사용되고 있다. 그리고 Free Article Header 는 자유화된 기사들의 리스트를 가르킨다. 실제로 디스크와의 입출력기능을 위해서 기사들은 여러 개의 연속된 블록으로 이루어진다. 따라서 하나의 기사를 할당하고 자유화할 때 여러 개의 블록들이 할당되고 자유화되는 것이다.

## 4. 구간별 분석 방법

뉴스 기사의 경우 그 시간대별 접근분포는 변화할 것이다. 이러한 가정은 시계열적 분석방법을 적용시킬 수 있는데, 과거의 수집된 자료를 바탕으로 미래를 예측하는 것이다. 가장 대표적인 시계열적 예측방법은 이동평균법 (moving average method)와 지수평활법(exponential smoothing method)를 들 수 있다.

### 4.1 이동평균법(moving average method)

이 방법은 과거의 관측값중 최근에 관측된 N 개를 가지고 추정을 한다. 즉 최근 N 개의 관찰값들의 평균하는 것이다. 그리고 이 평균값을 미래의 예측값으로 사용한다.

$y_t$ : 시간 t 에서의 관측치

$$M_T = \frac{1}{N}\sum_{t=T-N-1}^{T} y_t \qquad [식\ 1]$$

이동평균법은 N의 크기에 따라 그 결과가 달라진다. N 이 클 때는 그 변화의 반응이 천천히 보이게 된다. 이는 N 시간이 경과한 후에야 예측이 가능하기 때문이다.

### 4.2 지수평활법(exponential smoothing method)

이동평균법은 구가지 주된 제한점이 있다. 첫째는 이동평균을 계산하기 위해서는 N 개의 관측치 전부가 필요하기 때문에 그 예측값을 구하기 위해 많은 자료를 저장해야 한다. 둘째는 최근의 자료가 더 많은 정보를 간직한다고 볼 수 있기 때문에 최근의 자료에 더 큰 가중값을 주는 것이 타당하다. 그러나 이동평균법에서는 N 개의 자료에 동일한 가중값을 주고있다.

지수평활법은 최근의 관측치에 가중치를 더 많이 두는 방법이다. 그리고 과거로 갈수로 가중치는 기하급수적으로 줄어들게 된다.

$a$ : 평활 계수(smoothing coefficient) , $0<a<1$

$S_T$ : 지수 평활값(exponential smoothed valud)

$$S_T = ay_T + (1-a)S_{T-1} \quad [식\ 2]$$

여기서 주목할 것은 초기추정값 $S_0$ 를 정하느냐 하는 것이다. 이전의 자료가 있다면 이들의 평균또는 최근 N 개의 이동평균값으로 사용할 수 있다. 그러한 자료가 없다면 주관적으로 추정한 값을 사용한다. 지수평활법에서는 평활계수 $a$ 의 값에 따라 시계열 예측값이 달라지는데, 이 값이 작으면 예측값은 시계열 변화에 대하여 느리게 반응을 보인다.

## 5. 캐시 정책

캐시정책에서 가장 중요한 것은 미래에 대한 예측이다고 할 수 있다. 즉 가장 많이 접근될 것 같은 기사를 캐시에 올리는 것이다. 앞서 살펴본 시계열 분석방법중 지수평활법이 최근의 데이터에 가중치를 둠으로서 더 신뢰성있는 예측을 한다. 따라서 이 방법을 이용해서 캐시에 올릴 기사를 결정한다. NOD 데이타를 지수평활법에 적용시킬경우 각각의 기사마다 지수평활값과 현재 윈도우의 관측값을 가지고 있어야 한다. 이 값을 바탕으로 현재의 지수평활값을 구하고 이 값이 가장 큰 기사가 다음에 가장 많이 접근될 가능성이 높다고 예측할 수 있다. 따라서 이 지수평활값이 큰 기사순으로 캐시에 배치해야 한다.

초기추정량은 이전날의 마지막 윈도의 평활값으로 선택한다. 이경우 재배치 알고리즘은 [그림 3]과 같다.

NOD 의 경우 하나의 기사크기가 비디오데이타를 포함할 경우 수십에서 수백메가바이트가 된다. 이런 크기의 데이터를 캐시에 저장한다는 것은 그만큼 오버헤드가 크다. 그리고 이들을 재배치할 경우 그 비용이 너무 큼을 알 수 있다. [TRS97]에서는 caching algorithm 의 복잡도가 시스템성능에 상당한 영향을 줌을 알고 너무 복잡한 재배치 전략을 피했다. 일정주기 동안에 요구된 log file 을 분석하여 가장 좋은 value 를 가진 문서들을 캐시에 배치한다. 캐시된 문서들은 일정 주기동안 변하지 않고 남아있게 된다.

NOD 와 같은 멀티미디어데이타의 캐시정책은 복잡한 알고리즘보다는 간단한 재배치 알고리즘의 성능이

더 좋게 나올 수 있다. 따라서 하나의 윈도우동안에 일어나는 재배치는 한번으로 고정시킨다. 즉 그 주기동안 캐시에 올라가는 문서의 집합은 변하지 않는다. 새로운 윈도우가 시작하는 시점에서 재배치를 위하여 이전 윈도동안에 모든 기사들의 지수평활값을 구한다음 그 값이 큰값을 가진 기사들부터 캐시에 채워넣는다. 이렇게 채워진 기사들은 그 주기동안 변하지 않게 된다.

*ESR(Exponential Smooth Replacement) 알고리즘*
*입력: 모든 기사들의 이전 윈도의 추정량* $S_{T-1}$
*이전 윈도 동안 접근된 기사들과 그 빈도*
*평활 계수* $0<a<1$
*출력: 없음*
{
*현재 캐시의 내용을 모두 지운다.*
*While(모든 기사들에 대하여)*
{
*지수평활값* $S_T$ *을 구한다.*
}
*지수평활값이 큰기사들 순으로 sorting 함*
*캐시가 다 찰때까지 순서대로 배치한다.*
}

[그림 3]

## 6.실험

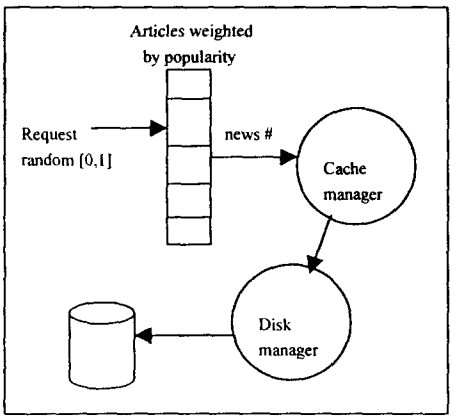
본 실험은 Sun Ultra Sparc 30 워크스테이션에서 수행되었으며, 프로그램은 C로 코딩하였다. 기사수는 100개로 제한했으며, 평균도착률은 24초로 하루평균 3600번의 사용자 요구가 발생한다고 가정하였다. 윈도우의 크기는 1간으로 총 24개의 윈도우가 생긴다. 그리고 사용자요구의 도착시간은 지수분포를 따른다고 가정하였다.

### 6.1 사용자 요구 발생

하나의 윈도안에서 기사들의 인기도는 [식 3]의 Zipf 분포를 따른다고 가정한다.

$$Zipf(i) = C/i, \quad C = 1/(\sum_{i=1}^{N} 1/i) \quad [식\ 3]$$

즉, 특정 윈도안에서 사용자들은 가장 인기있는 기사 몇개를 집중적으로 요구하게 된다. 그리고 이런 인기있는 기사들이 윈도우마다 일정한 패턴으로 달라지게 된다. [그림 4]에 보면 이러한 요구를 발생시키기 위한 시뮬레션 모델을 보여주고 있다. 먼저 각 윈도마다 기사들을 그 인기도에 따라 가중치를 매겨놓는다. 그리고 사용자 요구가 왔을 때 램덤하게 0과 1사이의 난수를 발생시키고 이 값을 인기도의 가중기사(Article weighted by popularity)에 매핑시켜 기사번호를 발생시킨다. 그리고 이 발생된 기사번호를 캐시메니저(cache manager)에게 보낸다.




[그림 4] 시뮬레이션 모델

### 6.2 ESR 성능분석

ESR 성능은 미래에 대해 얼마나 정확하게 예측할 수 있냐는 것이다. 따라서 현재의 평활계수값, 즉 다음 윈도의 예측값과 다음 윈도의 실제관측값의 차이가 적을수록 그 예측률이 좋다고 할 수 있다.

*예측오차 = 평활계수값 - 실제관측값*
*예측오차률 = 예측오차 / 윈도의 총요구수* [식 4]

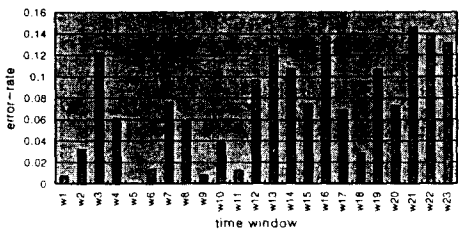
[식 4]에 따라 실험한 결과 예측오차는 평균 12.2 회로 나타났고, 예측오차률은 평균 0.0724 로 나타났다. [그림 5]는 각 윈도우별로 예측률을 나타낸 것이다.

평활계수값은 현재 관측치의 반영비율을 나타낸다. 이값이 크면 현재관측치에 많은 비중을 둠을 의미한다. [그림 6]를 보면 평활계수가 0.1 과 0.3 일 때 캐시히트률을 나타내고 있다. $a$가 0.1일때 평균 캐시히트율은 33% 이고 0.3 일 경우에 39%로 적으로 $a$가 클 때 더 성능이 좋아짐을 알 수 있다. [그림 6]에서는 평활계수값 $a$에 따른 평균 캐시히트율을 나타내고 있는데, 최적의 평활계수는 0.5 임을 알 수 있다. 이 때의 평균 캐시 히트율은 40%이다.

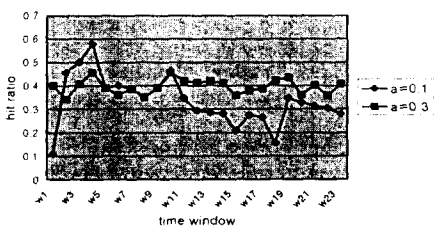
## 7. 결론

NOD 뉴스데이타의 경우 시간대별 기사들에 대한 선호도가 일정한 패턴으로 달라질 수 있다. 시계열 분석방법을 이용하면 이러한 변동을 분석하고 예측할 수 있다.

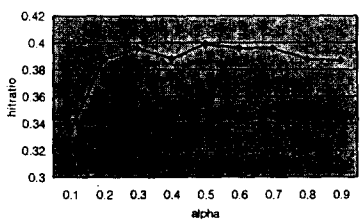
시계열 분석방법중 지수평활법(exponential smoothing method)을 이용하여 실험한 결과 그 예측오차율이 약 7%로 상당히 좋음을 알 수 있었고 캐시히트율이 약 40%까지 올라감을 알 수 있었다. 이 수치는 정적인 캐싱을 사용한 것으로 CPU 오버헤드를 줄이면서 얻은 값이다. 따라서 CPU 오버헤드를 고려하지않는다면 더 좋은 성능을 내는 재배치 알고리즘도 제시할 수 있다.




[그림 5]윈도별 예측률




[그림 6] $\alpha$=0,1 과 0.3 의 캐시히트율




[그림 7] 평활계수에 따른 히트율

참고논문

[97 Tew] Renu Tewari, Harrick M. Vin, Asit Dan & Dinkar Sitaram, "Resource Based Caching for Web Servers", Tech Report of CS of U. of Texas at Austin, 1997

[95 Tew] Renu Tewari, Asit Dan, etal Buffering and Caching in Large-Scale Video Servers, Tech-Reports, Texas Austin, 1995

[95 Kun] Kun Lung Wu, Philip S Yu, Consumption based Buffer management for Maximizing System Throughputs of News On Demand Multimedia System, 1995

[97 Wei] Weifeng Shi etal, Trading memory for disk bandwidth in Video on Demand, Tech report, Dept. of CS at USC, 1997

[97 TSR] Tatarinov, V.Soloviev, and A. Rousskov, Static Caching in Web Servers, submitted to the 6th international Conference on Computer Communications and Networks

[90 김] 김우철외, 현대통계학, 영지문화사 , 1990

[97 박] 박용운, 백건효, 서원일, 김영주, 정기동, NOD 데이터를 위한 새로운 버퍼링 기법, 한국방송공학회 학술대회, 1997

[97 이] 이주경, 박용운, 김영주, 정기동, NOD 데이타를 위한 데이터 배치방법, 한국정보과학회 추계 학술 발표대회, 1997

[98 서] 서원일 , 박용운, 백건효, 정기동, 짧은 비디오 오브젝트를 위한 오브젝트 단위 버퍼캐쉬 운영기법, 한국정보과학회 춘계학술발표대회, 1998