이중 캐쉬 서버를 사용한 실시간 데이터의 광대역 네트워크 대역폭 감소 정책

박 용 운† · 백 건 효†† · 정 기 동†††

요 약

연속형 미디어 데이터는 대용량이고 실시간으로 전송되어야 하므로 데이터 전송 시에 네트워크에 많은 부하를 주게 된다. 이러한 네트워크의 부하 문제를 해결하기 위하여 프락시 서버가 사용되며 프락시 서버에는 자주 접근되는 데이터가 저장되어 원래의 데이터가 존재하는 서버로의 네트워크 교통량을 줄이게 된다. 그러나 현재의 프락시 서버는 텍스트나 이미지 데이터등의 비 연속형 데이터만을 고려하여 설치되었으므로 연속형 미디어 데이터의 캐싱에는 적합하지 않다.

그러므로 본 연구에서는 연속형 미디어 데이터의 특징을 고려하여 프락시 서버를 두 계층으로 나누어 배치하여 데이터를 캐싱하고 데이터의 접근 패턴과 크기를 동시에 고려한 제할당 정책을 사용하여 캐쉬공간을 관리하는 프락시 서버 관리 정책을 제안한다. 제안한 정책에서는 각각의 LAN 마다 하나의 프락시 서버가 존재하며 각 LAN은 여러 개의 서브 LAN으로 나뉘어 지서 이러한 각각의 서브 LAN에는 또한 하나의 서브 LAN 프락시가 존재한다. 이에 병행하여 각각의 데이터들도 각각 전방 분할(front-end partition)과 후방 분할(rear-end partition)로 나뉘어져서 해당 데이터의 참조 유형에 따라 하나의 프락시에 동시에 저장되기도 하고 LAN 캐쉬 서버와 서브 LAN 캐쉬 서버에 각각 따로 저장되기도 한다. 이러한 정책을 사용함으로써 전체 데이터를 단위로 캐싱할 경우보다 데이터 공간의 할당과 재할당에 따른 오버헤드가 감소함으로써 궁극적으로는 원래의 저장 서버로의 네트워크 교통량을 보다 더 감소시킬 수 있다.

A Strategy To Reduce Network Traffic Using Two-layered Cache Servers for Continuous Media Data on the Wide Area Network

Yong-Woon Park† · Kun-Hyo Beak†† · Ki-Dong Chung†††

ABSTRACT

Continuous media objects, due to large volume and real-time constraints in their delivery, are likely to consume much network bandwidth. Generally, proxy servers are used to hold the frequently requested objects so as to reduce the network traffic to the central server but most of them are designed for text and image data that they do not go well with continuous media data. So, in this paper, we propose a two-layered network cache management policy for continuous media object delivery on the wide area networks. With the proposed cache management scheme, in each LAN, there exists one LAN cache and each LAN is further divided into a group of sub-LANs, each of which also has its own sub-LAN cache. Further, each object is also partitioned into two parts - the *front-end* and *rear-end* partition. They can be loaded in the same cache or separately in different network caches according to their access frequencies. By doing so, cache replacement overhead could be reduced as compared to the case of the full size data allocation and replacement, this eventually reduces the backbone network traffic to the origin server.

† 준 회 원 : 동의공업대학 전자계산과 교수
†† 정 회 원 : 동의공업대학 사무자동화과 교수
††† 종신회원 : 부산대학교 전자계산학과 교수
논문접수 : 2000년 4월 4일, 심사완료 : 2000년 9월 29일

1. 서 론

인터넷의 보급이 대중화됨에 따라 네트워크를 통한 원격지 서버의 데이터를 접근하는 응용 시스템 사이트 및 사용자들이 증가하고 있다. 이러한 사용자와 사이트의 급속한 증가에 따른 네트워크의 교통량의 문제를 해결하기 위하여서는 네트워크 자원의 증설 특히 백본 네트워크 자원의 증설이 필요하다. 그러나 단지 네트워크 자원을 증가 시키는 것만으로는 증가하는 네트워크 교통량을 효과적으로 처리할 수 없다. 왜냐하면 현재의 추세에 따르면 인터넷 사용자와 사이트의 증가가 급속하게 이루어지기 때문에 네트워크 자원을 추가한다고 하더라도 사용자와 사이트들의 증가 속도를 따라잡을 수 가 없다[1].

또한 사용자의 연속형 미디어 데이터에 대한 선호도가 증가함에 따라 2005년경에는 인터넷상에서 접근되는 데이터의 50%가 오디오 또는 비디오 등의 실시간형 데이터가 될 것으로 추정된다[2]. 인터넷 환경에서는 서로 상이한 네트워크들이 연결되어 있으며 이러한 상이한 네트워크들은 백본 네트워크에 의하여 서로 연결되어 다수의 사용자들을 지원하고 있다. 최근의 기가 비트 이더넷(Gigabit Ethernet)이나 광 채널(Fibre Channel) 등의 네트워크 기술의 발달로 인하여 지역통신망(LAN : Local Area Network)의 설치와 운영에 필요한 비용은 점차로 감소해가는 추세이다. 그러므로 실시간으로 연속형 미디어를 서비스하는 응용 서비스 시스템에서는 백본 네트워크 교통량을 줄이는 것이 효율적인 시스템 운영에 가장 중요한 부분이 된다[3].

따라서 본 논문에서는 연속형 미디어 데이터의 특징을 고려하여 프락시 서버를 두 계층으로 나누어 배치하고 이와 동시에 데이터도 분할 저장하여 캐싱하는 프락시 서버 관리 정책을 제안한다. 이러한 정책을 사용함으로써 오브젝트를 단위로 캐싱할 때보다 데이터 공간의 할당과 재할당에 따른 오버헤드가 감소함으로써 궁극적으로는 원격지에 존재하는 서버에서의 데이터 접근에 필요한 네트워크 교통량을 보다 더 감소시킬 수 있다.

본 논문의 구성은 다음과 같다. 제2장에서는 네트워크 상에서의 멀티미디어 데이터의 입출력 향상을 위한 연구들을 소개한다. 제3장에서는 본 논문에서 제안하고 있는 이중 캐쉬 서버의 운영 정책 및 알고리즘을 소개한다. 제4장에서는 본 논문에서 제안하는 정책을 시뮬레이션을 통하여 분석하고 마지막 장인 5장에서는 결론을 내리고 향후 연구 방향에 대해서 논한다.

2. 관련 연구

지금까지 네트워크 상에서 데이터 전송에 필요한 대역폭을 효율적으로 사용하기 위한 많은 연구가 있었다. 배칭(batching)[4, 5]은 일정한 주기를 정하여 사용자의 요청을 바로 처리하지 않고 그 일정한 주기동안 생성된 요청을 한꺼번에 묶어서 처리하는 방식이다. 이 방식은 간단하고 자원을 효율적으로 사용할 수 있다는 장점이 있으나 서비스 대기 시간이 서비스 주기에 비례하여 길어진다는 단점이 있다. 멀티캐스팅(multicasting)[6]은 배칭의 변형된 형태로써 사용자의 QoS를 미리 알 수 있는 화상회의 등의 응용에는 적합하나 비디오나 뉴스 서비스 같이 사용자의 수와 접근 유형이 알려지지 않은 경우에는 적합하지 않다. 패칭(patching)[7]은 멀티 캐스팅과 유사하나 디스크나 메모리 등의 사용자의 자원을 사용하여 네트워크 자원의 효율적으로 사용하고자 한 방법이다 사용자가 데이터의 전송을 요청하였을 때, 사용자 측에서는 주 채널(primary channel)과 부 채널(secondary channel)의 두 개의 채널을 열어서 주 채널로는 서버로부터 직접 데이터를 읽음과 동시에 부 채널로는 동일한 데이터를 서비스 받고 있는 다른 사용자의 채널을 공유하여 데이터를 자신의 메모리나 디스크에 저장한다 사용자의 데이터 읽기가 부 채널로 저장하기 시작한 지점에 이르러서는 주 채널로의 데이터 읽기를 중단하고 부 채널로 저장한 데이터를 사용한다. 그러나 이 방법은 네트워크 자원의 사용에는 효율적이지만 사용자의 자원에 의존한다는 한계가 있다. 일반적으로 네트워크 대역폭의 효과적인 관리를 위하여 프락시 캐싱[8]이 널리 사용되고 있으나 주로 이미지나 텍스트 등의 웹 데이터를 다루어 왔다 최근에 들어서야 연속형 미디어 데이터의 프락시 캐싱이 이루어지고 있으나 주로 초기 지연을 줄이는 측면[9]과 VBR 스트림의 대역폭 평활화(bandwidth smoothing)[10] 측면으로 초점이 맞추어져 있다.

3. 이중 캐쉬를 사용한 네트워크 캐쉬 구조

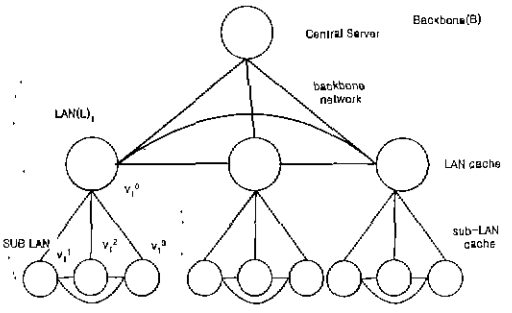
3.1 이중 네트워크 캐쉬 서버 구조

전체 캐쉬 구조는 (그림 1)과 같이 그래프 $G = <s, G', E>$ 표현될 수 있다(s 는 중앙서버, G'는 LAN 그

름, E는 s와 G'사이의 네트워크 경로이다). G'가 n개의 LAN으로 구성된다고 가정하자. 그러면 G'는 $G' = \sum_{k=1}^{n} CT_k$(CT는 LAN)로 표현된다.

정의) LAN k는 트리 $CT_k = (V_k, E_k^h, E_k^s)$(V_k는 m개의 네트워크 캐쉬로 이루어진 집합을, E_k는 각각의 LAN k에 존재하는 네트워크 캐쉬 간의 네트워크 경로의 집합을 의미)로 표현되며 아래와 같은 조건을 만족하여야 한다.

1) $V_k = \{ v_k^i \mid 0 \le i \le m, \forall sub-LAN\ i (i \ge 1) \exists v_k^i, \exists\ v_k^0 \forall\ v_k^i (i \ge 1)\ such\ that\ v_k^0 = parent(v_k^i) \}$
2) $E_k^h = \{ (v_k^i, v_k^0) \mid i \ge 1 \}$
3) $E_k^s = \{ (v_k^i, v_k^j) \mid i, j \le m \}$



(그림 1) 이중 캐쉬 서버 구조

3.2 이중 캐쉬 서버를 사용한 데이터 저장

3.2.1 데이터 분할

이미지나 텍스트 데이터와는 달리 오디오나 비디오 등의 연속형 미디어 데이터는 대용량의 저장 공간이 필요하므로 캐쉬에 저장되는 단위가 객체 단위라면 연속형 미디어 데이터의 크기를 고려할 때 빈번한 재할당으로 인한 오버헤드가 발생할 수 있다. 그러므로 본 논문에서는 각각의 미디어 데이터를 전방분할(front-end partition)과 후방분할(rear-end partition)로 분할하여 캐쉬 공간의 할당 및 재배치 단위를 객체가 아닌 분할 단위로 한다.

연속형 미디어 데이터의 경우에는 라운드 별로 데이터를 주기적으로 전송하며 데이터 재생 시에 시간적인 제한이 존재한다. 따라서 중앙 서버 또는 다른 캐쉬 서버에 존재하는 후방 분할의 첫 라운드 분량에 해당하는 데이터가 서브-LAN 캐쉬에 존재하는 전방 분할이 클라이언트에게 전달되어 재생되는 동안 서브-LAN 캐쉬에 도착하여야 한다. 그러므로 데이터 재생의 지연을 발생 시키지 않는 전방 분할의 최소 크기는 이러한 후방 분할의 첫 라운드 분량의 데이터가 서브-LAN 캐쉬에 완전히 도착할 수 있는 시간 동안을 재생할 수 있는 크기여야 한다.

전체 네트워크가 1개의 backbone 과 m개의 LAN으로 구성되며 각각의 LAN 구역은 하나의 LAN 캐쉬와 k-1개의 서브 LAN 캐쉬가 완전 연결(fully connected)되어 있다고 하자. 또한 오브젝트는 r초를 라운드로 주기적으로 서비스 되고 있다고 하면 오브젝트 A의 후방 분할의 전송에 따른 전달 지연을 상쇄하기 위한 전방 분할의 최소 크기($A_{\min}^F$)는 다음과 같다.

$$A_{\min}^F = \{ N_{packet} * D_{trans} + D_p + \{ (m-1)w(l) + (k-1)w(k) \} * (D_{initial} + \sum_{l=1}^{N_{packet}-1} D_{inter-arr}) \} * pl_r^A \quad (1)$$

where pl_r^A : 오브젝트 A의 라운드 r 당 재생률 (부록 참조)

따라서 후방 분할의 최대 크기($A_{\max}^R$)는 다음과 같이 정해진다.

$$A_{\max}^R = A_{size}^{TOT} - A_{\min}^F \quad (2)$$

where A_{size}^{TOT} : 오브젝트 A의 크기

3.2.2 서브 LAN 캐쉬 서버

서브 LAN 캐쉬 서버는 사용자가 원하는 데이터를 읽기 위하여 가장 먼저 접근하는 캐쉬 서버이다. 사용자가 데이터를 접근하기 위하여 요청을 하면 먼저 해당 사용자가 속해있는 서브 LAN 캐쉬 서버를 접촉하여 요청된 데이터가 저장되어있는지를 체크한다. 서브 LAN 캐쉬는 데이터의 전방 분할 또는 전체를 저장하는데 사용된다. 만약에 요청한 데이터 전체 또는 전방 분할이 해당 서브 LAN 캐쉬 서버에서 발견되지 않는 경우 요청한 데이터를 캐쉬 할당 알고리즘에 따라 데이터 전체 또는 전방 분할을 캐싱한다.

3.2.3 LAN 캐쉬 서버

LAN 캐쉬 서버는 각각 여러 개의 서브 LAN그룹으

로 구성된 LAN 상에 존재하며(하나의 LAN에 존재하는 서브 LAN들은 (그림 1)에 나타난 것처럼 완전 연결(fully connected)되어있다) 해당 LAN내의 서브 LAN 캐쉬 서버의 부모 역할을 한다. LAN 캐쉬 서버는 요청된 데이터의 후방 분할을 저장하고 전송하기 위하여 존재한다.

3.3 데이터 공간 재할당

기존의 파일 캐싱 정책은 공간 할당 및 재 할당 단위가 블록인 반면 프락시 캐싱에서는 블록이 아닌 오브젝트 또는 데이터 전체가 캐싱의 단위가 되며 이 때 데이터의 크기는 서로 다르다. 따라서 두 개의 데이터가 동일한 접근 빈도를 보인다면 크기가 작은 데이터를 캐싱하는 편이 히트율을 최대화할 수 있다. 그러므로 데이터가 요청되고 해당 데이터의 캐싱 여부를 결정할 때 해당 데이터의 접근 빈도 수 뿐만 아니라 데이터의 크기 및 전송에 필요한 네트워크 대역폭 등의 요소도 고려하여야 한다[5]

LRU 등의 최근성에 기반한 알고리즘이 주로 데이터의 참조 국부성(locality of reference)을 이용하는 반면 LFU 등의 접근 빈도수에 기반한 알고리즘은 많은 데이터 중에서 소수의 데이터에 집중적으로 참조가 발생하는 편기된 접근 유형(skewed access pattern)을 보이는 응용에 적합하다. 따라서 본 논문에서는 위에서 언급한 사항을 고려하여 최근성과 접근 유형을 동시에 접목하는 캐쉬 공간 재할당 정책을 사용한다. 우선 각각의 네트워크 캐쉬 d에서의 데이터 i에 대한 접근 유형을 파악하기 위하여 해당 캐쉬에서의 데이터 i의 평균 접근 간격을 구한 다음 데이터 i의 크기를 반영하기 위하여 데이터 i의 크기를 가중치로 준 값을 *가중 평균 간격* $w_dist(i)^d_{avg}$으로 두고 재할 당 기준으로 선정한다.

$$w\_dist(i)^d_{avg} = \left\{ \sum_{j=2}^{l} \left\{ T(i)^d_j - T(i)^d_{j-1} \right\} \Big/ N^d_i - 1 \right\} * s(i) \quad (3)$$

$T(i)^d_j$: 캐쉬 d에서 데이터 i의 j번째 스트림의 도착 시간

N^d_i : 캐쉬 d에서 서비스되는 데이터 i의 스트림수

$s(i)$: 데이터 i의 크기

만약 해당 캐쉬에 새로이 요청된 데이터 i의 캐싱에 필요한 공간이 존재하지 않는다면 $w_dist(i)^d_{avg}$를 해당 캐쉬에 존재하는 다른 분할 또는 데이터 j의 $w_dist(j)^d_{avg}$와의 비교를 통하여 i의 가중 평균 간격 보다 큰 값을 가지는 데이터 또는 분할을 재할당 후보로 둔다. 이렇게 선정된 재할당 대상들에 대하여 현재 접근 중이 스트림이 있는지의 여부를 조사하여 재 할당 대상인 스트림 j에 대한 현재 접근 스트림 존재 여부를 $act(j)_d$로 표시한다.

$$act(j)_d = \begin{cases} 0, & if\ active\ stream\ exists \\ 1, & otherwise \end{cases} \quad (4)$$

최종적으로 재할당 후보로 선택된 데이터 또는 분할에 대하여 접근 중인 스트림이 없는 데이터 또는 분할을 최종 재 할당 대상으로 삼는다.

$$w\_dist(i)^d_{avg} < w\_dist(j)^d_{avg}\ and\ act(j)_d = 1 \quad (5)$$

위의 과정을 캐싱하고자 하는 데이터 j의 크기에 해당하는 공간을 확보할 때 까지 반복한다. 자세한 공간 할당 알고리즘은 (그림 2)에 나타나 있다.

```
Caching_policy(network_cache v, object partition p)
{
   네트워크 캐쉬 i의 가용 공간 체크;
   if가용공간 > 요청된 분할의 크기 {
      요청된 분할의 크기만큼 공간을 할당한다,
      return,
   }
  else {
      노드 v에 존재하는 모든 분할 또는 데이터 체크
      가용 공간을 0로 셋팅;
      총 가용공간 += 가용 공간;
      요청된 데이터 o의 분할 p에 대한 가중 평균 간격
      w_dist(p)^v_avg계산 ;
      노드 v에 존재하는 모든 분할의 가중 평균 간격
      계산하여 값에 따라 정렬
      k를 1로 셋팅;
     loop until
       w_dist(p)^v_avg <= w_dist(k)^v_avg {
       k번째 크기의 분할을 재할당 후보로 선정,
       if k번째 분할을 접근하는 스트림이 없음
          총 가용공간 += k번째 분할의 크기,

       if 총 가용공간 > 요청된 분할 p {
          재할당 후보로 선정된 분할(들)이 점유한
                        캐쉬 공간을 deallocate;
          요청된 분할 p를 캐싱함,
          return;
       }
       k의 값을 1증기;
```

```
        }
    }
    요청된 분할 p의 캐싱 실패
    return;
}
```

(그림 2) 캐쉬 공간 할당 알고리즘

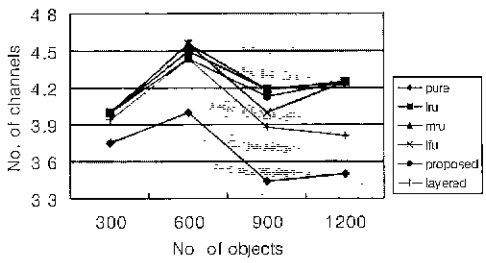
4. 실험 결과

본 장에서는 시뮬레이션을 통하여 제안한 네트워크 캐쉬 정책의 성능을 평가하였다. (그림 1)에 나타난 것처럼 전체 네트워크 구조는 트리 형태를 가정하였다. 중앙 서버가 트리의 루트로써 존재하고 있으며 4개의 LAN을 자식으로 하여 중앙 서버와 4개의 LAN이 완전 연결된 형태로 존재하고 이 네트워크 구역을 레벨 2 네트워크라고 두었다. 또한 각각의 LAN에는 두 개의 서브 LAN이 존재하며 이 때 각각의 서브 LAN에는 또한 서브 LAN 캐쉬 서버가 존재한다. 이와 같이 LAN과 서브 LAN이 구성하는 네트워크 구역을 레벨 1 네트워크로 두었다. 사용자의 데이터 접근 패턴은 *Zipf* 분포를 따르며 모든 미디어 데이터는 MPEG II 형식으로 압축되어 초당 4Mbps의 전송 속도를 필요로 한다고 가정하였다. 또한 미디어 데이터의 크기는 서로 상이하며 그 크기는 30초에서 180초까지의 범위 내에서 평균 크기는 120초인 정규 분포를 하는 것으로 가정하였다. 네트워크 캐쉬 서버의 크기는 3기가 바이트, 초당 전송율은 10Mbps/초로 두었다. 데이터 분할은 식 (1)에 따라 개략적으로 10초 분량으로 두었다(최장 길로 6초 + 기타 전송 오버헤드 4초). 본 논문에서 제안한 캐쉬 정책의 성능을 평가하기 위하여 기존의 세 종류의 캐쉬 알고리즘(LRU, MRU, LFU)과 본 논문에서 제안하는 두 개의 정책(*proposed*, *layered*)의 실행한 결과를 비교하였다. 그림에서의 *pure*는 캐쉬 지원 없이 중앙 서버만으로 사용자의 요청을 서비스했을 경우의 결과를 의미하며 기존의 알고리즘은 각각 lru(LRU 알고리즘), mru(MRU 알고리즘), lfu(LFU 알고리즘) 등으로 표현하였다. 본 논문에서 제안한 알고리즘은 각각 *proposed*와 *layered*로 표현되며 그 차이는 다음과 같다. *proposed* 의 경우에는 데이터 분할을 하지 않고 (그림 2)의 데이터 재 할당 정책만을 적용한 경우이며 *layered*의 경우는 식 (1)에 의해 데이터 분할 까지도 적용한 본 논문에서 주장하는 알고리즘 전체를 의미한다. 이렇게 한 이유는 본 논문에서 제안한 알고리즘의 성능 개선이 재 할당 정책에서 오는 것인지 아니면 데이터의 분할에서 오는 것인지를 파악하기 위한 것이다 또한 전체 미디어 데이터의 수가 캐싱 정책에 미치는 영향을 조사하기 위하여 전체 미디어 데이터의 수를 각각 300, 600, 900, 1200 등으로 변화시키면서 성능을 측정하였다

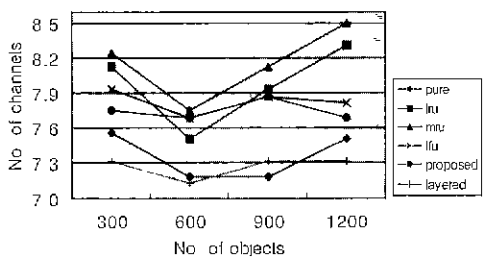
4.1 도착률과 네트워크 대역폭 사용률의 변화

성능 평가의 첫 번째 기준으로써 네트워크 채널의 사용도를 측정하였다. 네트워크 채널이란 사용자가 데이터를 서비스 받기 위하여 서버 또는 캐쉬를 선택하여 데이터를 전송 받을 때 설정된 경로의 집합을 의미한다. 실험의 결과가 (그림 3), (그림 4)에 나타나 있으며 각각의 값은 각 네트워크 레벨에서의 노드 간의 피크 타임에서의 최대 네트워크 채널 수를 구한 다음 그 값들의 평균을 네트워크 채널 사용도로 표시하였다.

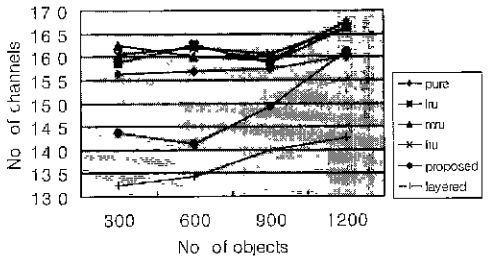
레벨 1 네트워크는 사용자 도착률이 낮은 경우(분당 10명의 경우)에는 캐싱을 하지 않는 경우(pure)가 캐싱을 한 경우보다 요구되는 네트워크 채널의 수가 적다 이는 사용자의 도착률이 낮은 경우 데이터의 캐싱을 통한 데이터의 재 사용 가능성이 떨어지기 때문이며 결과적으로 캐싱을 통한 성능 향상은 이루어지지 않았



(a) Arrival rate of 10/min

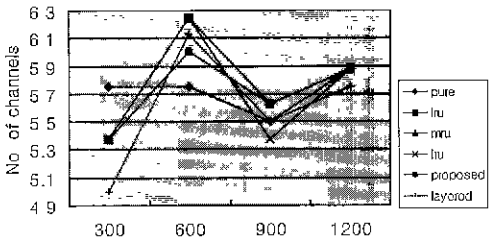


(b) Arrival rate of 30/min

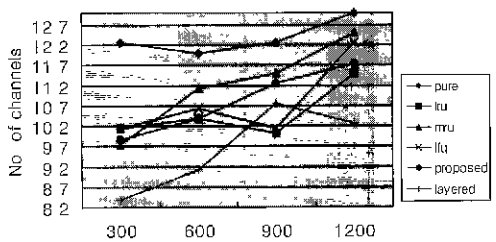


(c) Arrival rate of 60/min

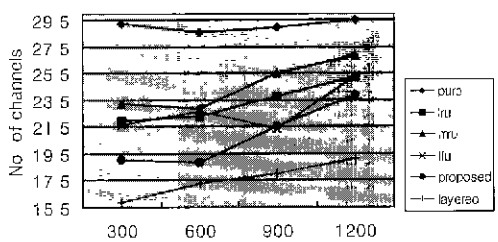
(그림 3) 사용자 도착률의 변화에 따른 레벨 1 네트워크 채널의 사용도의 변화



(a) Arrival rate of 10/min



(b) Arrival rate of 30/min



(c) Arrival rate of 60/min

(그림 4) 사용자 도착률의 변화에 따른 레벨 2 네트워크 채널의 사용도의 변화

다. 그러나 사용자 도착률이 높은 경우에는 요구되는 네트워크 채널의 수가 캐쉉 정책에 따라 변화하였다 (그림 3)에서 처럼 본 논문에서 제안한 2정책(*proposed* 와 *layered*)의 결과가 다른 캐쉉 정책보다 우수한 결과를 보임을 알 수 있다. 다른 캐쉉 알고리즘은 서로 간의 성능의 우열을 평가할 정도의 차이를 보이지는 않았다

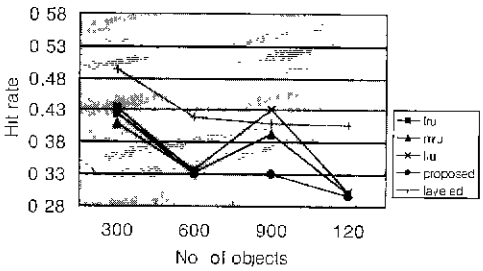
레벨 1과는 달리 레벨 2 네트워크는 백본 네트워크로써 많은 사용자에 의해 공유되므로 교통량의 증가가 발생하면 병목 현상이 일어나기 쉽다. 실험 결과를 보면 사용자의 도착률이 분당 10명을 초과하면서 캐쉉 정책을 사용하는 것이 캐쉉 정책을 사용하지 않는 *pure*의 경우보다 적은 네트워크 채널이 필요한 것을 알 수 있다. 캐쉉 정책을 사용하지 않는 경우와 캐쉉 정책을 사용하는 경우의 성능의 차이는 사용자의 도착률과 네트워크의 레벨이 높을수록 많이 나는 것을 알 수 있다. 특히 본 논문에서 제안하는 *layered* 방식과 *pure* 방식의 차이는 거의 두 배에 이르며 캐쉉 정책을 사용하는 경우 사용자의 도착률이 높아질수록(30 또는 60) 미디어 데이터의 수의 변화가 캐쉉 정책에 영향을 미치고 있음을 알 수 있다. 이는 미디어 데이터의 수가 증가하는 반면 네트워크 캐쉬의 크기는 고정되어 있으므로 불 필요한 제 할당이 일어나기 때문이다. 따라서 미디어 데이터의 수가 증가 할수록 캐쉉되는 미디어 데이터를 더욱 선별적으로 두어야 할 것으로 판단된다. 그러나 캐쉉을 적용하지 않은 *pure*의 경우 요구되는 네트워크 채널의 수가 미디어 데이터의 수의 변화에 영향을 받지 않는다. 이는 캐쉉을 적용하지 않는 경우 요청되는 데이터에 상관 없이 사용자와 서버 간의 네트워크 채널이 독립적으로 형성되기 때문이다. 본 논문에서 제안한 알고리즘은 레벨 1의 경우와 마찬가지로 사용자의 도착률이 낮은 경우에는 성능 향상을 보이지 않았지만 사용자의 도착률이 높아질수록 다른 캐쉉 정책에 비해 성능이 증가함을 알 수 있다.

4.2 도착률의 변화에 따른 캐쉬 히트율의 변화

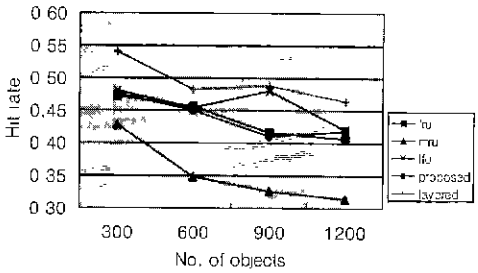
두 번째 성능 평가의 척도로는 사용자의 도착률과 미디어 데이터의 수의 변화에 따른 캐쉬 히트율의 변화를 측정하였다. 여기서의 힛트율은 각 레벨의 네트워크 캐쉬의 힛트율의 평균 값을 의미한다.

서브 LAN 캐쉬 서버의 힛트율의 변화가 (그림 5)에 나타나 있다. (그림 5)에 나타난 것 처럼 일부 경우를 제외하고는 본 논문에서 제안한 알고리즘의 힛트율이

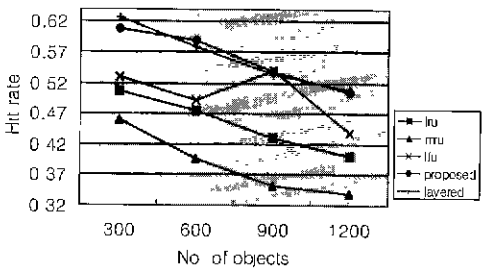
다른 알고리즘의 힛트율보다 높음을 알 수 있다(일부의 경우에서 LFU가 본 논문에서 제안한 알고리즘 보다 우수한 성능을 보였으나 성능의 차이를 관가름할 정도의 결과는 아니었다). 네트워크 체널의 사용도와 마찬가지로 힛트율 또한 미디어 데이터의 수에 영향을 받음을 알 수 있다((그림 5)에서 보듯이 미디어 데이터의 수가 증가할수록 힛트율이 떨어진다). LFU의 경우에는 사용자의 도착률에 따라 힛트율이 불규칙적으로 변하였다. (그림 6)에 사용자 도착률 및 미디어 데이터 수의 변화에 따른 LAN 캐쉬 서버의 히트율의 변화를 측정한 결과를 나타내었다. (그림 6)에서 알 수 있듯이 LAN 캐쉬 서버의 힛트율의 변화는 서브 LAN 캐쉬 서버의 힛트율의 변화와는 상이하다. 대부분의 경우에 있어서 본 논문에서 제안하는 알고리즘의 경우(*layered*)가 다른 알고리즘의 경우보다 힛트율이 떨어짐을 알 수 있다. 그러나 이는 본 논문에서 제안하는 알고리즘의 재 할당 정책이 잘못되었다기보다는 본 논문에서 제안하는 캐쉬 정책이 데이터 전체를 할당 기준으로 하는 다른 캐쉬 정책과는 달리 데이터의 분할을 통한 점진적 캐쉬 정책을 사용함으로써 발생한 결과라는 것을 본 논문에서 제안하는 또 다른 정책인 *proposed* (데이터 분할 없이 재 할당 정책만을 적용하였을 경우)의 결과를 보면 알 수 있다(본 논문에서의 LAN 캐쉬 서버는 서브 LAN 캐쉬와는 달리 후방 분할의 저장



(a) Arrival rate of 10/min

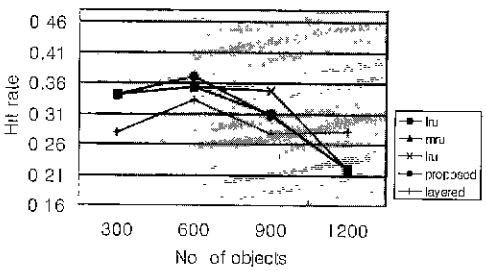


(b) Arrival rate of 30/min

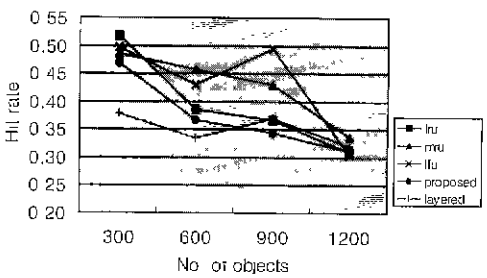


(c) Arrival rate of 60/min

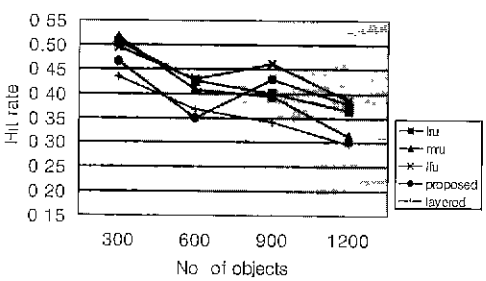
(그림 5) 사용자 도착률의 변화에 따른 서브 LAN 캐쉬 힛트율의 변화



(a) Arrival rate of 10/min



(b) Arrival rate of 30/min



(c) Arrival rate of 60/min

(그림 6) 사용자 도착률의 변화에 따른 LAN 캐쉬 힛트율의 변화

이나 서브 LAN캐쉬가 포화 상태에 이르렀을 때 서브 LAN 캐쉬의 대리역할을 하는 캐쉬이다). 또한 이 실험 결과로써 계층 구조의 캐쉬 구조에서의 전체 성능은 사용자 측에 가까운 캐쉬 서버의 성능에 영향을 받으며(본 논문의 경우를 예를 들면 서브 LAN캐쉬 서버) 힛트율과 전체 네트워크의 성능이 반드시 비례하지는 않는다는 것을 알 수 있다.

5. 결론 및 향후 연구 방향

본 연구에서는 연속형 미디어 데이터의 특징을 고려하여 프락시 서버를 두 계층으로 나누어 각각의 LAN 마다 하나의 프락시 서버가 존재하며 또한 각각의 LAN 은 여러 개의 서브 LAN으로 나뉘어지며 이러한 각각의 서브 LAN에는 또 하나의 서브 LAN 캐쉬 서버가 존재한다. 이에 병행하여 각각의 데이터들도 전방 분할과 후방 분할로 나뉘어서 점진적인 캐쉬정책을 적용하고 데이터 재할당을 접근패턴과 최근성을 동시에 반영하는 프락시 서버 관리 정책을 제안하였다. 실험 결과 본 논문에서 제안한 방식을 사용하여 네트워크 캐쉬를 운영하는 것이 다른 캐쉬 정책을 사용한 경우보다 우수한 성능을 보였다. 향 후 연구로는 실험 결과에서 언급한 것처럼 LAN캐쉬 서버의 성능 향상을 통하여 전체 캐쉬 서버의 성능을 더욱 높이는 연구가 필요하다. 또한 데이터의 분할에 있어서 분할 비율을 어떻게 두는 것이 최적인지를 파악하는 부분에 대한 연구도 병행되어야 할 것으로 생각된다.

부 록

전방 분할 크기 설정

전체 네트워크는 (그림 1)에서 처럼 하나의 backbone 과 m개의 LAN 그룹으로 구성되어있다. 하나의 LAN 그룹에는 하나의 LAN 캐쉬(k=0) 서버가 존재하고 또한 각각의 LAN 은 k-1(k>1이고 모든 LAN에서 k는 동일하다고 가정)개의 서브 LAN 캐쉬로 구성되며 중앙서버와 각 LAN 캐쉬 또는 각 LAN 캐쉬 간의 간의 경로의 가중치의 값은 $w(l)$, 각 LAN에서의 LAN 캐쉬와 각 서브 캐쉬 또는 각 서브 캐쉬 간의 경로의 가중치는 $w(c)$로 일정하다고 하자.

LAN j로 부터 임의의 사용자 U에 의한 오브젝트 A에 대한 요청이 발생했다고 하면 전송 노드에서 목적 노드까지의 이미 거친 노드는 다시 방문하지 않는 최장 경로 길이($MAX(m,k)^{server}_{LAN,(U)}$)는 사용자 U로부터 사용자가 있는 LAN의 캐쉬 서버까지의 최장 경로 $MAX(P(k))^{LAN,(C)}_{LAN,(U)}$와 해당 LAN 캐쉬에서 중앙서버까지의 최장 경로 $MAX(P(m))^{server}_{LAN,(C)}$의 합으로 표시될 수 있다.

$$MAX(P(m,k))^{server}_{LAN,(U)} = MAX(P(m))^{server}_{LAN,(C)} + MAX(P(k))^{LAN,(C)}_{LAN,(U)}$$

$MAX(P(k))^{LAN,(C)}_{LAN,(U)}$ 는 서브 LAN 캐쉬의 수에 따라 따라 달라진다.

1) 서브 LAN 캐쉬가 없을 때 (k=1)

LAN 캐쉬와 서브 LAN 캐쉬 사이의 경로가 존재하지 않는다. 따라서 총 경로의 길이는 다음과 같다.

$MAX(P(1))^{LAN,(C)}_{LAN,(U)} = (1-0) * w(c) = 0$ 이다.

2) 서브 LAN 캐쉬가 하나 일 때 (k =2)

LAN 캐쉬와 서브 LAN 캐쉬 사이의 경로가 하나(2-1)밖에 없다. 따라서 최장 경로의 길이는 다음과 같다

$MAX(P)^{LAN,(C)}_{LAN,(U)} = (2-1) * w(c)$

즉, $MAX(P(2))^{LAN,(C)}_{LAN,(U)} = MAX(P(1))^{LAN,(C)}_{LAN,(U)} + w(c)$ 이다.

3) k = 가 n 일때 성립한다고 가정하자. 즉,

$MAX(P(n))^{LAN,(C)}_{LAN,(U)} = (n-1) * w(c)$이 된다.

K = n + 1 이면

$MAX(P(n+1))^{LAN,(C)}_{LAN,(U)} = (n+1-1) * w(c)$ 즉,

$MAX(P(n+1))^{LAN,(C)}_{LAN,(U)} = MAX(P(n))^{server}_{LAN,(U)} + w(c)$ 이 된다

따라서 $MAX(P(k))^{LAN,(C)}_{LAN,(U)} = (k-1) * w(c)$가 성립한다. 마찬가지로 $MAX(P(m))^{server}_{LAN,(C)}$에 대해서도 동일한 증명 방법을 적용하면

$MAX(P(m))^{server}_{LAN,(U)} = (m-1) * w(l)$ 을 증명할 수 있다

따라서 최종적으로

$$MAX(P(m,k))^{server}_{LAN(U)} = MAX(P(m))^{server}_{LAN(C)} + MAX(P(k))^{LAN(C)}_{LAN(U)} = (m-1)\,w(l)+(k-1)\,w(c)$$

가 성립한다.

후방 분할의 첫 라운드에 해당하는 데이터가 전방 분할에 존재하는 데이터 블록들의 서비스 완르되기 전에 도착하기 위하여 소요되는 총 지연 시간(D_{total})은 다음과 같다.

$$D_{total} = N_{packet} * D_{trans} + D_p + D_{trip} \quad (1)$$

where N_{packet} : 1 라운드에 해당하는 총 패킷의 수
D_{trans} : 전송 지연(transmission delay)
D_p : 전파 지연(propagation delay),
D_{trip} : 패킷이 목적지에 전달되는데 소요되는 시간

D_{trip}은 첫 번째 패킷이 전송될 때의 trip time($D_{initial}$)과 이후 $N_{packet}-1$개의 패킷 간의 도착 간격(inter-arrival time)의 합($D_{inter-arr}$)으로 표현될 수 있으므로 다음과 같이 표현된다

$$D_{trip} = \{(m-1)\,w(l)+(k-1)\,w(k)\} * (D_{initial} + \sum_{i=1}^{N_{packet}-1} D_{inter-arr}) \quad (2)$$

따라서 식 (1)은 다음과 같이 다시 표현된다.

$$D_{total} = N_{packet} * D_{trans} + D_p + \{(m-1)\,w(l)+(k-1)\,w(k)\} * (D_{initial} + \sum_{i=1}^{N_{packet}-1} D_{inter-arr}) \quad (3)$$

그러므로 오브젝트 A의 후방 분할의 전송에 따른 전달 지연을 상쇄하기 위한 전방 분할의 최소 크기 ($A^F_{\min}$)는 다음과 같다.

$$A^F_{\min} = \{N_{packet} * D_{trans} + D_p + \{(m-1)w(l) + (k-1)\,w(k)\} * (D_{initial} + \sum_{i=1}^{N_{r+1}-1} D_{inter-arr})\} * pl^A_r$$

where pl^A_r : 오브젝트 A의 라운드 r 당 재생률

참 고 문 헌

[1] By Michael A. Goulde, "Network Caching Guide Optimizing Web Content Delivery," March 1999/6/23 White Paper . Inktomi Corp. Available at http://www.inktomi.com/products/network/traffic/technology.html

[2] Garth A. Gibson, Jeffrey Scott Vitter, John Wilkes, "Strategic directions in storage I/O issues in large-scale computing," ACM Computing Surveys Vol 28, No.4 (Dec 1996), pp.779-793.

[3] Zhi-Li Zhang ; Du, D.H.C. ; Dongli S, Yuewei Wang, "A network-conscious approach to end-to-end video delivery over wide area networks using proxy servers," INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Volume · 2. Vol 2, pp 660-667, 1998.

[4] C. C Aggarwal and J. L. Wolf and P. S Yu, "On Optimal Batching Policies for Video-On-Demand Storage Server," Proc. of the IEEE Int'l Conf. On Multimedia Systems June 1996.

[5] Aggarwal, C : Wolf, J. L. : Yu, P S., "Caching on the World Wide Web," Knowledge and Data Engineering, IEEE Transactions on Volume · 111, pp.94-107, Jan.-Feb. 1999.

[6] K A Hua,, Simon Sheu, "Skyscraper Broadcasting : A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems," In Proceedings of ACM SIGCOMM'07, Cannes, France, pp89-100

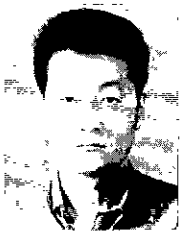
[7] Kien A. Hua Ying Cai Simon Sheu, "Patching : A Multicast Technique for True Video-on-Demand," ACM Multimedia Bristol, UK 191 200.

[8] Greg Barish and Katia Obraczka, "World Wide Web Caching Trends and Techniques," Univ. of Southern California USC tech reports, available at ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/99-713.ps.Z

[9] Sen, S. ; Rexford, J. ; Towsley, "Proxy prefix caching

for multimedia streams." D. INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Volume : 3, Vol.3, pp 1310-1319, 1999.

[10] Reza Rejaie, Mark Handley, Haobo Yu, Deborah Estrin, "Proxy Caching Mechanism for Multimedia Playback Streams in the Internet," 1999, USC tech reports, available at ftp://ftp.usc.edu/pub/csinfo/tech-reports/papers/99-693.ps.gzL.



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~1999년 부산대학교 전자계산학과 박사 수료

2000년~현재 동의 공업대학 전자계산과 전임강사

관심분야 : 병렬 파일 시스템, 멀티미디어, 캐싱

백 건 효

e-mail : ghbaek@dit.ac.kr

1988년 부산대학교 계산통계학과 졸업(학사)

1989년~1992년 공군 전산실 근무

1994년 부산대학교 전자계산학과 졸업(석사)

1996년~1998년 부산대학교 전자계산학과 박사 수료

1999년~현재 동의 공업대학 사무자동화과 조교수

관심분야 : 멀티미디어, 병렬파일시스템, 캐싱 등

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

관심분야 : 병렬처리, 멀티미디어