論文
2-11-10

# Prefetch하는 웹 캐쉬 프록시의 성능에 대한 연구
# (A Study on the Performance of Prefetching Web Cache Proxy)

백 윤 철*
(Yun-Cheol Baek)

## 요 약

인터넷 사용자의 폭발적인 증가로 인해 웹 서비스는 심각한 성능상의 지연을 보이고 있다. 인기 있는 웹 사이트의 입장에서는 많은 요청으로 인하여 원활한 서비스를 제공하기 어렵고, 사용자 또한 만족스러운 수준의 응답시간을 제공받지 못한다. 이에 대한 해결책으로 제시된 웹 캐슁은 서버에 대한 요청을 흡수하여 전체적인 교통량을 줄이며 사용자에게는 보다 빠른 응답을 제공한다. 본 논문에서는 미국의 웹 캐쉬 프로젝트인 NLANR(National Laboratory for Applied Network Research)의 최상위 캐쉬들로부터 생성된 트레이스와 서울대에 위치한 교육망 캐쉬의 트레이스를 이용하여 웹 캐쉬 트래픽의 특성들에 관해 분석하고, 이들로부터 얻은 각종 특성 자료를 바탕으로, 미래에 필요하리라고 여겨지는 웹 오브젝트를 미리 가져오는 prefetch 방법을 제시하였으며 그로 인한 효과에 관해 분석하였다. 그 결과, 1~3% 정도 일일 평균 적중률의 향상과 최대 5% 정도의 평균 응답시간의 개선을 기대할 수 있음을 발견하였다.

## ABSTRACT

Explosive growth of Internet populations results performance degradations of web service. Popular web sites cannot provide proper level of services to many requests, and such poor services cannot give user a satisfaction. Web caching, the remedy of this problem, reduces the amount of network traffic and gives fast response to user. In this paper, we analyze the characteristics of web cache traffics using traces of NLANR(National Laboratory for Applied Network Research) root caches and Education network cache in Seoul National University. Based on this analysis, we suggest a method of prefetching and we discuss the gains of our prefetching. As a result, we find proposed prefetching enhances hit rate up to 3%, response time up to 5%.

## 1. 서론

월드와이드웹의 급성장으로 네트워크 교통량이 급속히 늘어나게 되었다. 이로 인하여 인기가 있는 웹 서버에는 많은 요청이 몰려 각 클라이언트에게 만족할 만한 서비스를 제공할 수 없는 현상이 일어나곤 한다.

인터넷의 병목 현상을 타개하기 위해 대두된 해결책의 하나로 웹 오브젝트의 캐슁(caching)을 들 수 있다. 이것은 원격 사이트에 있는 웹 오브젝트에 대한 반복적인 요구를 흡수하기 위해 인터넷의 각 요소마다 캐쉬를 배치하여 웹 오브젝트를 지리적으로 가까운 곳에 위치하도록 하는 방법이다[1].

* 정회원 : 상명대학교 소프트웨어학부 조교수

논문접수 : 2001. 11. 7.
심사완료 : 2001. 11. 13.

※ 본 연구는 2000학년도 상명대학교 자연과학연구소 연구비 지원으로 이루어졌음.

웹 캐쉬에 있어서 중요한 것은 얼마나 많은 요청을 국지화하여 사용자의 요청에 대한 응답 지연(latency)을 줄여 주는가 하는 것이다. 이러한 시도로 캐쉬 교체 정책을 개선하여 캐쉬 적중률을 높이는 연구들이 많이 진행되었다 [2,3,4]. 다른 한 측면에서 보면 웹 캐쉬 자체가 하나의 서버이므로 외부로부터의 요청이 많아질수록 성능 상의 저하를 가져오게 된다. 따라서 이에 잘 견딜 수 있는 서버 구조를 채택하는 등의 노력과 피크 시간의 요청을 줄이는 일 등이 서버의 성능 저하를 최소화하는데 기여하게 된다.

웹 캐쉬의 prefetching은 사용자가 오브젝트를 요청하기 전에 요청할 만한 오브젝트를 예측하여 웹 캐쉬에 미리 가져다 놓는 것이다. 그렇게 함으로써 더 많은 요청을 웹 캐쉬에서 흡수하여, 외부로 나가는 요청을 줄이고 응답시간을 개선하는 것을 목적으로 한다. Prefetching을 하기 위해서는 사용자가 미래에 요청할 오브젝트를 예측하는 것이 필요한데 웹 트래픽의 특징에 대한 기존 연구에 의하면 사용자가 요청하는 오브젝트는 매우 광범위하기 때문에 미래에 사용자가 요청할 오브젝트를 예측하는 것은 매우 어려운 일이다[5,6].

본 논문에서는 NLANR에서 제공하는 웹 캐쉬와 서울대에 위치한 교육망 웹 캐쉬의 access 로그를 분석하여 하루 중 요청이 많은 시간과 적은 시간대를 알아내고, 상대적으로 요청이 적은 시간대에 지난 로그를 바탕으로 다음 날에 요청될 오브젝트를 미리 가져다 놓음으로써 요청이 많은 시간대에 외부로 나가는 트래픽을 줄이는 prefetch를 연구하였다.

본 논문의 2장에서는 관련 연구로서 기존 prefetching에 대한 연구들에 대해 알아보고, 3 장에서는 로그 분석과 prefetching의 방법을 기술하고, 4 장에서 제시한 방법으로 시뮬레이션을 실행한다. 마지막으로 5장에서 실험에 대한 결론과 향후 연구과제에 대해 논한다.

## 2. Prefetching에 대한 기존 연구

Prefetching과 관련한 기존 연구는 대략 3가지로 구분할 수 있다. 첫 번째는 사용자가 지금 보고 있는 페이지의 hyperlink를 클릭 할 확률이 많다는 점을 고려하여 현재 페이지에 hyperlink 된 오브젝트들을 prefetching 하는 방법이다[7]. 두 번째는 사용자의 웹 오브젝트 요청에 대한 연관 관계 그래프를 유지하고 있다가 사용자가 하나의 오브젝트를 요청할 때 일정 조건을 만족하는 연관 관계가 있는 오브젝트를 같이 가져오는 방법이다[8,9]. 세 번째 방법은 클라이언트나 프록시가 이전 요청에 대한 history를 바탕으로 인기 있는 웹 서버의 리스트를 유지한다. 그리고 각 웹 서버에서는 자신이 가지고 있는 웹 오브젝트의 인기순위를 유지하고 있어서 프록시에서는 주기적으로 인기 있는 웹 서버에서 인기 있는 웹 오브젝트를 미리 가져다 놓는 방법이다[10].

위에서 기술한 세 가지 방법들에는 각각 문제점들이 있다. 첫 번째 방법은 사용자가 어떤 페이지를 보았을 때 hyperlink 된 오브젝트를 클릭하는 확률의 편차가 크고 또한 한 개의 페이지에 연결된 문서는 한 개에서 수백 개 이상일 수도 있기 때문에 이들 오브젝트들을 전부 prefetching 한다면 네트워과 서버에 큰 부담을 줄 것이다. 두 번째 방법으로 prefetching 할 경우에는 네트웍의 부하가 많은 시간에 prefetch를 한다는 것은 대역폭의 낭비가 될 수 있을 뿐만 아니라, 보통 하루에 웹 캐쉬에 요청되는 요청의 개수는 몇 십만에서 몇 백만 개 이상에 이르는데 이들의 연관 관계에 대한 그래프를 유지한다는 것 자체가 서버에 부담이 될 수 있다. 세 번째 방법은 실제로 모든 웹 서버에서 각 오브젝트의 인기 리스트를 유지하지 않기 때문에 현실에서 적용되기 어려운 방법이다.

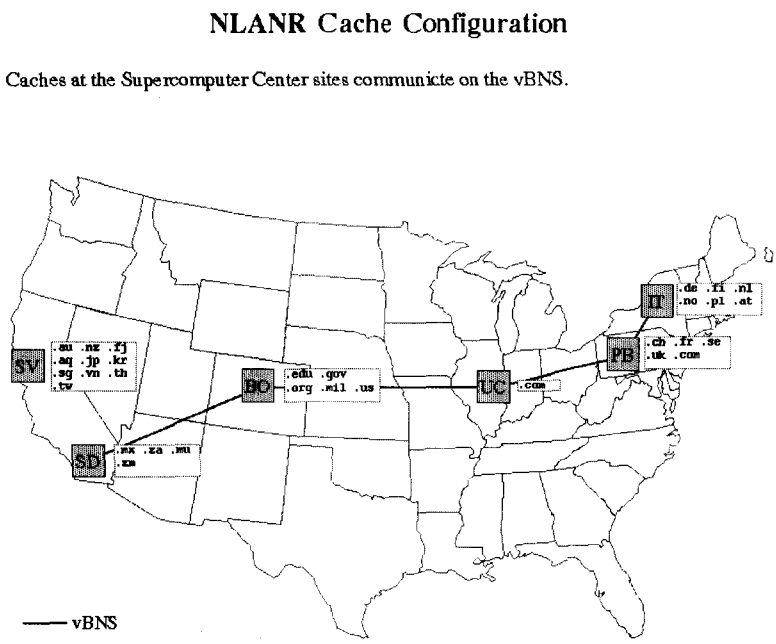
## 3. Prefetching을 위한 웹 캐쉬 프록시 로그 분석

Prefetching이란 서론에서 언급한 것처럼, 사용자가 미래에 요청할 문서를 예측하여 미리 프록시에 가져다 놓는 것이다. 즉, prefetching을 하기 위해서는 사용자들의 웹 사용 패턴을 분석할 필요가 있다. 본 논문에서는 어떤 문서를 언제 어떻게 prefetching 해야할지 결정하기 위하여 prefetching을 실험하기 전에 먼저 캐쉬 프록시 서버의 access 로그를 분석한다. 분석자료로 쓰인 로그는 NLANR의 "A Distributed Tested for National Information Provisioning"이란 프로젝트에서 설치한 vBNS로 연결된

서로 협동하는 캐쉬 프록시 서버의 로그 ([그림 1] 참조)와 서울대 교육망에 설치한 캐쉬 프록시 서버의 로그이다. 두 개의 프록시들은 모두 웹 캐쉬 소프트웨어로 Squid를 사용하고 있으며 이들이 산출한 access.log의 각 항목의 특성([그림 2]참조)을 이용하여 프록시 캐쉬 트래픽의 특성을 파악한다.

### 3.1 Hit rate, 응답 시간, 요청 빈도에 관한 분석

Squid의 로그에서 timestamp항목은 클라이언트가 오브젝트를 요청하고 프록시에서 소켓을 열어 요청에 대한 처리를 마치고 소켓을 닫을 때의 시간을 말한다. 그래서 이 timestamp를 기반으로 하루의 로그를 분리하고, 이것을 해당 지역의 표준시로 환산하여 초당 요청의 개수를 산출하였다.




[그림 1] NLANR의 캐쉬 구성도
[Fig. 1] NLANR Cache Configurations

| 944553571.460 | 45 | 203.237.183.64 | TCP_MISS/200 | 387 | GET |
|---|---|---|---|---|---|
| timestamp | elapsed | client | action/code | size | method |

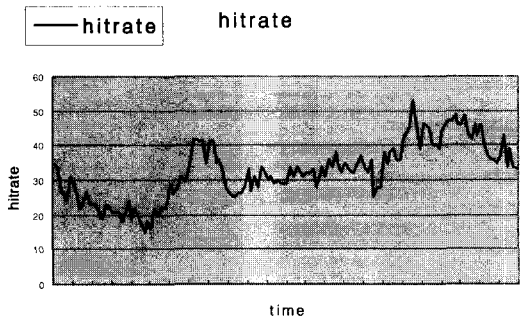
| http://linux.sarang.net/images/n.gif | - | DIRECT/linux.sarang.net | image/gif |
|---|---|---|---|
| URL | ident | hieracy/From | content |

[그림 2] access.log의 형식
[Fig. 2] Format of access.log

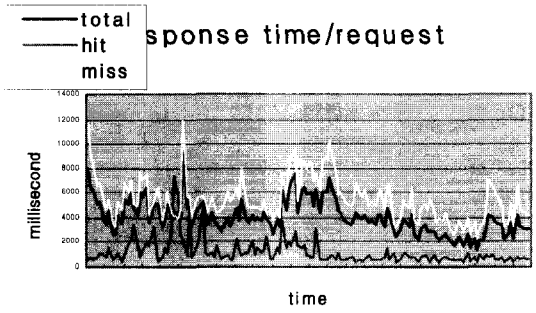
action항목은 클라이언트가 요청한 문서가 프록시에서 HIT되었는지의 여부를 나타내기 때문에 총 요청에 대한 HIT한 요청 수를 구해 hit rate를 산출하였다. elapsed항목은 클라이언트에게서 오브젝트가 요청된 후 소켓을 열고 클라이언트에게 요청된 오브젝트를 전달한 후 소켓을 닫을 때까지 걸린 시간을 나타내기 때문에 이 항목을 클라이언트로 문서가 오기까지 기다린 응답시간으로 사용하였다. 다만 본 논문에서는 클라이언트의 요청이 캐쉬에 도달할 때까지, 또한 캐쉬로부터 응답이 클라이언트에게 도착할 때까지의 네트워크 지연시간은 매번 네트워크의 부하에 따라 달라지기 때문에 응답시간에서 제외하였다.

NLANR에 대해서는 2001년 2월 12일부터 14일까지 주말이 아닌 평일에 대한 로그를 분석하여 각각에 대한 평균을 산출한 것이다. 토요일이나 일요일 등의 휴일의 로그는 분석 결과 트래픽의 분포가 불규칙하고 요청이 평일에 비해 상당히 적어지므로 평균을 산출하는 대상에서 제외하였다. NLANR의 총 6개의 로그 중에 UC와 SV의 로그분석에 대한 결과를 [그림 3]과 [그림 4]로 나타내었다. 각 그림에서 (a)는 하루 중의 hit rate의 변화 나타내었고, (b)는 각 요청에 대한 응답시간의 변화를 (c)는 하루 중 프록시로 요청하는 요청 빈도에 대한 변화를 나타내고 있다. 트레이스 분석 결과로 나타난 그래프를 살펴보면, 하루의 요청 빈도에 대한 경향이 웹 캐쉬의 위치와 역할에 따라 조금 다르다는 것을 알 수 있다. 먼저 [그림 3]에 나타난 UC의 그래프를 보면 낮 시간에 캐쉬에 대한 요청이 많아지는 산 모양의 형태를 하고 있다. PB와 BO등도 이와 비슷한 경향을 나타내었다. 이들은 모두 미국 내나 유럽지역의 요청을 서비스하는 최상위 레벨 캐쉬이다. 반면 [그림 3]의 SV는 낮 시간에 오히려 요청 빈도가 최저 수준을 이루는 골 모양을 하고 있다. SD그래프도 SV와 비슷한 모양으로 나타난다. 이들은 주로 미국과는 생활 시간대가 정 반대인 아시아권의 요청을 서비스한다.

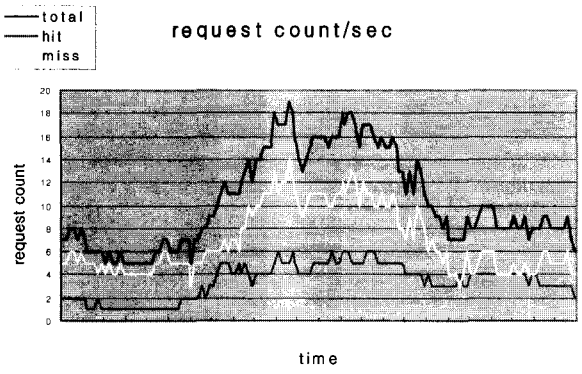
위에서 분석된 세 가지의 성능 지수의 연관관계를 살펴보자. UC와 같은 경우에는 요청 빈도의 변화가 hit rate에 별로 영향을 주지 않으며 응답시간도 hit rate에 크게 좌우되지 않는다. 그보다는 MISS되는 요청의 절대 개수가 많아지면, hit rate과는 관계없이 응답시간은 길어진다. SV의 경우에는 밤과 새벽에 MISS되는 요청이 현저히 늘어나지만 그에 대한 응답시간은 거의 일정한 수준을 보인다. 이것은 미국 전역에 인터넷 트래픽이 많은 낮 시간에 발생되는 요청과 그렇지 않은 밤 시간에 발생되는 요청 - 특히 MISS된 요청 - 에 대한 응답 시간에 차이가 있을 수 있음을 시사하며 웹 캐쉬, 특히 최상위 레벨 캐쉬의 응답 시간을 개선하기 위해서는 트래픽이 많은 피크시간에 MISS되는 요청의 수를 줄이는 것이 하나의 방법이 될 수 있을 것으로 보인다.




(a) UC의 하루 중 hit rate 변화



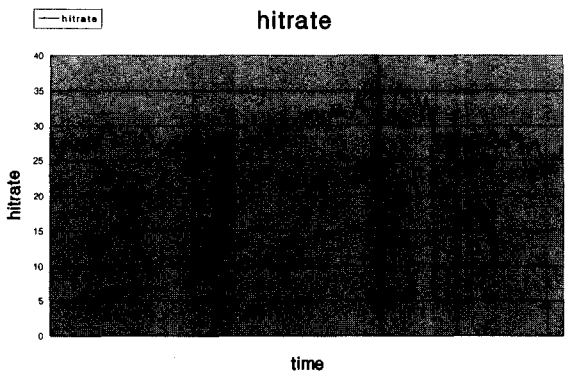

(b) UC의 하루 중 응답시간 변화



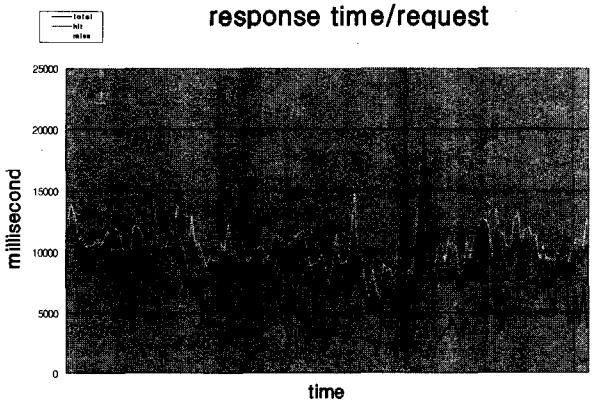

(c) UC의 하루 중 요청 빈도 변화

[그림 3] UC의 특성 그래프

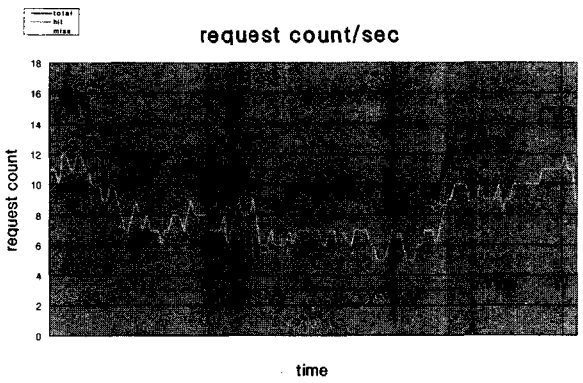
[Fig. 3] Traffic characteristics of UC




(a) SV의 하루 중 hit rate 변화



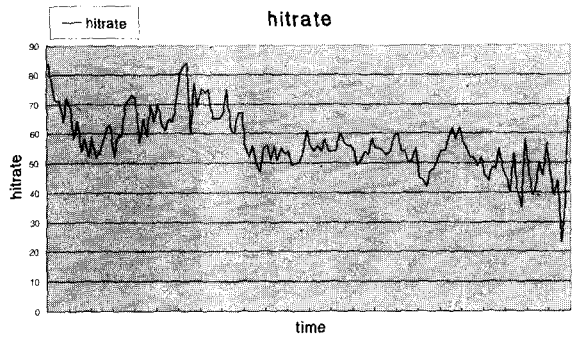

(b) SV의 하루 중 응답시간 변화



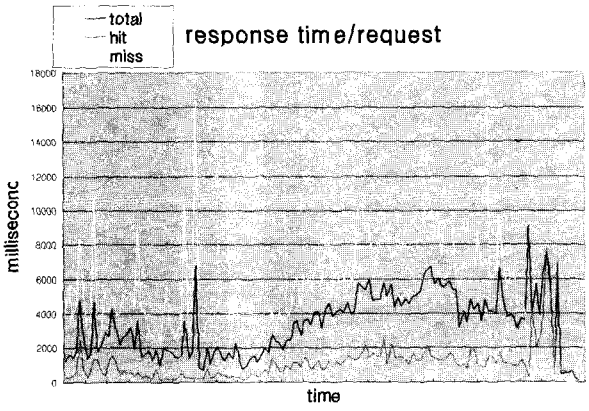

(c) SV의 하루 중 요청 빈도 변화

[그림 4] SV의 특성 그래프

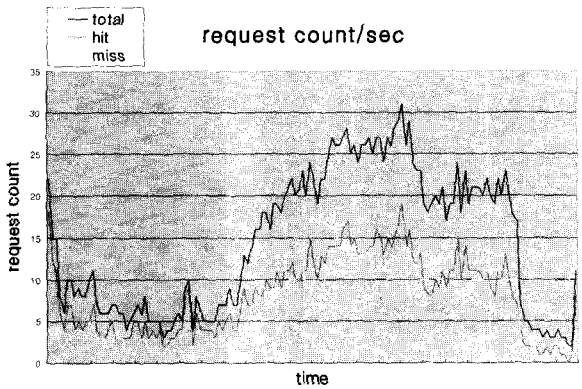
[Fig. 4] Traffic characteristics of SV




(a) 교육망의 하루 중 hit rate 변화




(b) 서울대 교육망의 하루 중 응답시간 변화




(c) 교육망의 하루 중 요청 빈도 변화

[그림 5] 교육망의 특성 그래프

[Fig. 5] Traffic characteristics of Education network

서울대에 위치한 교육망의 캐쉬 프록시 로그를 분석한 결과는 [그림 5]에 나타나 있다. 1997년 11월 1일부터 5일까지의 로그를 분석한 결과 서울대 교육망의 캐쉬 프록시 로그도 역시 하루 중 활동 시간대인 오전 9시부터 오후 9시까지에 높은 요청 횟수를 보였다. 이에 따라 peak 시간대의 응답시간도 길어졌고, 웹 오브젝트에 대한 요구가 많아짐에 따라 상대적으로 HIT되는 요청보다 MISS된 요청이 증가해서 hit rate도 감소하였다.

이 두 가지의 로그를 분석한 결과 Squid의 로그를 이용해 요청의 수가 많은 문서 중에 MISS 응답시간이 긴 문서들을 비교적 요청이 적은 시간대에 서버에 부담을 주지 않는 범위 내에서 미리 가져다 놓는 prefetching을 사용하면 개선 효과가 별로 없는 경우에도 prefetching하지 않은 경우만큼의 평균 응답시간은 보장할 것을 예상할 수가 있고, 만약 prefetching이 peak 시간대에 발생하는 MISS된 요청의 수를 조금이라도 줄이게 된다면 이것은 평균 응답시간의 개선으로 이어질 수 있다는 가능성을 확인하였다.

### 3.2. 프록시에 요청되는 오브젝트의 크기 분석

두 번째로 로그를 가지고 분석한 것은 프록시에 요청되는 웹 오브젝트의 크기와 응답시간의 관계에 관한 것이다. 분석 결과는 <표 1>에 나타나 있다. 이를 통해 보면 프록시에 요청하는 오브젝트 중 교육망의 경우는 약 70%가, SV와 UC의 경우는 약 75%이상이 3Kbyte 미만의 오브젝트이다. HIT 와 MISS 요청의 응답시간을 살펴보면 3Kbyte 미만의 문서들은 MISS 되었을 경우보다 hit 되었을 때가 약 4 ~ 7배 가량 빠르게 클라이언트에게 전달되었다. 이것으로 보아 상대적으로 많은 부분을 차지하는 이들 적은 오브젝트를 요청이 적은 한가한 시간대에 가져다 놓음으로써 peak시간대의 응답시간의 개선을 얻을 수 있으리라 판단된다.

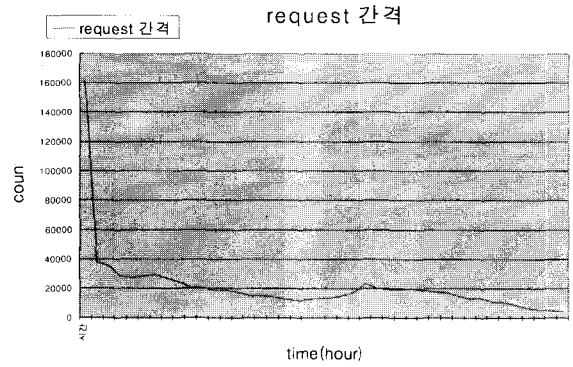
### 3.3. 반복 요청들 사이의 시간 간격에 관한 분석

세 번째로 프록시에 요청되는 오브젝트 들 중에서 한번 요청된 오브젝트가 또다시 프록시에 요청될 때까지의 시간의 분포를 분석하였다. 분석 결과는 [그림 6, 7, 8] 에 나타나 있다.

<표 1> 오브젝트의 크기와 평균응답시간에 대한 분석

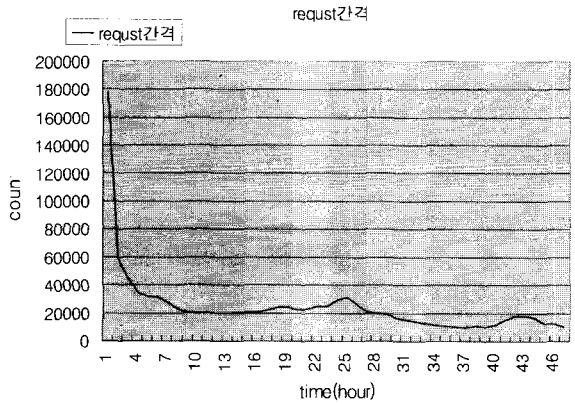
<Table 1> Analysis on the object size and response time

| 구 분 | HIT request | | | MISS request | | |
|---|---|---|---|---|---|---|
| | 문서길이 (Kbyte) | 요청 갯수 비율 (%) | 평균응답시간 (millisecond) | 문서길이 (Kbyte) | 요청 갯수 비율 (%) | 평균응답시간 (millisecond) |
| 서울대 교육망 | 0 미만 | 36 | 490 | 0 미만 | 20 | 2,704 |
| | 1 ~ 3 | 9 | 802 | 1 ~ 3 | 9 | 3,375 |
| | 4 ~ 10 | 3 | 2,203 | 3 ~ 4 | 3 | 5,327 |
| | 11 이상 | 10 | 130,105 | 11 이상 | 10 | 412,085 |
| SV 와 UC | 0 미만 | 16 | 405 | 0 미만 | 31 | 1,776 |
| | 1 ~ 3 | 4 | 525 | 1 ~ 3 | 25 | 2,427 |
| | 4 ~ 10 | 2 | 834 | 4 ~ 10 | 2 | 3,678 |
| | 11 이상 | 10 | 247,946 | 11 이상 | 10 | 458,926 |



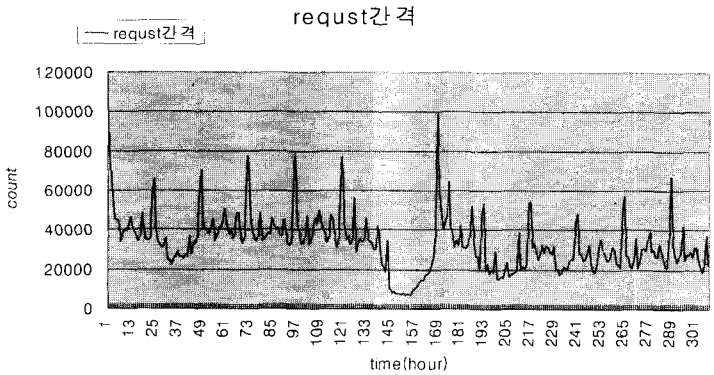

[그림 6] SV 의 request 간의 간격

[Fig. 6] Request interval of SV




[그림 7] UC 의 request 간의 간격

[Fig. 7] Request interval of UC




[그림 8] 교육망의 request 간의 간격

[Fig. 8] Request interval of Education network

[그림 6]과 [그림7]을 살펴보면 한 오브젝트에 대한 요청 간격이 3 일 이상 되는 오브젝트는 거의 1% 미만으로 나타났다. [그림 8]의 교육망의 요청 패턴은 SV와 UC와의 다른 양상을 보이기 때문에 더 정확한 결과를 위해 11월 1일부터 13일까지 13일의 로그를 분석하였다. SV와 UC는 사용자들의 그룹이 일정하지 않고 전 세계에 펼쳐져 있는 임의의 사용자이기 때문에 광범위한 오브젝트들을 요청하지만, 서울대에 위치한 교육망의 경우 사용자들은 각 학교에 있는 학생들로 제한되었고 매주 수업과 관련한 웹에 대한 접근이 있을 수 있기 때문에 24시간, 48시간, 72시간 등 하루, 이틀 등의 단위로 반복되는 요청 패턴이 나타난다. 일주일이 되는 168시간대에는 가장 많은 오브젝트 들이 재 요청이 된다. 이것은 학교라는 그룹의 특성상 일주일 단위로 학습 활동이 반복되기 때문으로 보인다.

## 4. Prefetching을 하는 캐쉬 프록시

### 4.1. Prefetching 방법

2장에서 살펴본 기존 prefetching의 문제점을 없애고 3장에서 분석한 결과를 바탕으로 본 논문에서는 새로운 prefetching방법을 제안하였다. 기존의 연구에서 웹 서버나 클라이언트에서 어떤 자료구조를 유지하거나 추가 모듈을 설치하는 것이 사실상 어렵고, 사용자가 요청을 할 때마다 prefetch를 실시하면 상대적으로 네트웍의 부하가 많은 시간에는 오히려 더 부담을 줄 수 있다는 문제점이 있었기 때문에 서버와 클라이언트의 수정이 없고 오직 프록시에서만 수행하여 문제를 해결하는 방법을 고안하였다.

제안하는 방법에서는 먼저 로그를 분석하고 상대적으로 요청의 빈도가 적은 시간대에 서버에 부담을 주지 않는 범위 내에서 웹 문서를 미리 가져오는 prefetching이 이루어진다. 프록시는 prefetching을 수행하는 prefetching 서버를 가지고 있다. prefetching 서버는 각 로그의 특성을 바탕으로 지난 몇 일간의 access 로그들을 분석하여 전제 문서의 70%이상을 차지하는 3 Kbyte 이하의 오브젝트 중 가장 접근 횟수가 높은 오브젝트의 순서로 Top list를 작성한다. 지난 access 로그를 바탕으로 만든 이 리스트를 이용해 하루 중 가장 한가한 시간대를 선택하여 서버에 부담을 주지 않는 범위(요청의 개수가 가장 많았던 시간과 가장 적은 시간의 평균) 내에서 1초에 몇 개씩 prefetching 할 것인지 개수를 설정하여 Top list의 순서대로 오브젝트들을 시간이 허락하는 한 웹 캐쉬에 요청해 프록시로 가져온다. prefetching의 절차를 다시 요약하면 다음과 같다.

- 3.1절의 결과에 따라 prefetching 할 시간대를 상대적으로 요청이 적은 기간을 선택한다.
- 3.2절의 결과에 따라 전체의 70% 이상 차지하는 문서들인 3Kbyte 이하가 되는 오브젝트 들만을 prefetching 대상으로 하고 가장 요청의 빈도가 높은 순서대로 Top list를 만들어 놓는다.
- 1초에 몇 개의 오브젝트를 가져올 것인지 결정하기 위해 가장 요청이 적은 경우와 와 가장 많은 경우를 기준으로 평균 요청의 개수를 산출해 prefetching으로 인해 추가적으로 발생하는 요청과 기존에 발생되는 요청의 수를 합쳐도 평균을 넘지 않도록 prefetching의 요청 빈도를 설정한다.
- prefetching 서버는 하루에 설정된 시간동안 Top-list에 있는 오브젝트를 시간이 허락하는 한 설정된 빈도만큼 서버에 요청을 해서 프록시로 가져온다.

### 4.2. Prefetching 시뮬레이션

Prefetching의 실험을 위해 NLANR에서 제공한 UC의 2001년 7월 3일부터 12일까지 12일간의 로그와 서울대 교육망의 1997년 11월 1일부터 12일까지 12일간의 로그를 사용하였다. 두 개의 로그의 하루 평균 요청의 개수는 100만 ~ 120만 사이이고, 하루의 요청한 총 오브젝트의 크기는 약 7 ~ 10Gbyte 이다. Prefetching을 하는 프록시 시뮬레이션을 위하여 Perl을 사용하였다. 먼저 프록시의 캐쉬를 오브젝트 들로 초기화하기 위해 1주일의 access 로그에 있는 오브젝트 들을 캐쉬에 공간의 여유가 있으면 요청된 오브젝트를 캐쉬하고 더 이상 공간이 없을 경우에는 Squid에서 기본적으로 사용하는 교체 방법인 LRU(Least Recently Used) 알고리즘을 사용하여 오브젝트를 캐쉬 밖으로 내보낸다. 캐쉬의 초기화를

실시한 이후에 앞 절에서 언급한 방법으로 LRU 교체 정책을 사용하여 prefetching을 수행하였다. UC의 로그의 경우에는 [그림 3]에서 분석한 결과 가장 요청이 적은 시간인 0시에서 5시까지 5시간을 선택하여, Top list에 있는 오브젝트를 초당 3개씩 각 웹 서버로부터 prefetching 하였고, 서울대 교육망의 로그의 경우에는 [그림 5]에 근거하여 오전 3시에서 8시까지 5시간을 prefetching 시간으로 선택하고 Top list에 있는 오브젝트를 초당 3개씩 시간이 허락하는 만큼 웹 서버에서 prefetching 하였다.

Top list는 요청 간의 간격을 고려하여 UC의 경우는 지난 3일의 로그를 이용하여 만들었고, 서울대 교육망의 경우는 지난 1주일의 로그를 이용하여 Top list를 만들었다. 응답 시간에 대해서는 HIT 했을 경우에는 0으로, MISS 했을 경우에는 로그의 elapsed time을 사용하였다.

<표 2> UC의 로그를 사용한 prefetching 결과
<Table 2> Prefetching with UC access log

| 캐쉬 크기 | 날짜 | Prefetching | | No Prefetching | |
|---|---|---|---|---|---|
| | | hit rate (%) | response time (millisecond) | hit rate(%) | response time (millisecond) |
| 10G | 11.08 | 43.91 | 4066 | 42.41 | 4285 |
| | 11.09 | 47.01 | 2947 | 47.03 | 2950 |
| | 11.10 | 45.54 | 4396 | 39.40 | 4659 |
| | 11.11 | 44.41 | 4288 | 43.30 | 4363 |
| 20G | 11.08 | 46.03 | 4002 | 43.96 | 4206 |
| | 11.09 | 49.04 | 2865 | 49.04 | 2865 |
| | 11.10 | 47.03 | 4354 | 47.05 | 4352 |
| | 11.11 | 46.32 | 4342 | 46.41 | 4356 |

<표 3> 서울대 교육망의 로그를 사용한 prefetching 결과
<Table 3> Prefetching with Education Network(SNU) access log

| 캐쉬 크기 | 날짜 | Prefetching | | No Prefetching | |
|---|---|---|---|---|---|
| | | hit rate (%) | response time (millisecond) | hit rate (%) | response time (millisecond) |
| 10G | 7.10 | 24.23 | 6892 | 25.12 | 6884 |
| | 7.11 | 27.54 | 6325 | 27.34 | 6331 |
| 20G | 7.10 | 26.01 | 6876 | 26.12 | 6870 |
| | 7.11 | 28.91 | 6237 | 28.78 | 6237 |

실험 결과는 <표 2>와 <표 3>에 나타나 있으며 hit rate와 response time에 각각 성능의 개선을 가져온 것을 볼 수 있다.

서울대 교육망의 로그는 prefetching을 하지 않았을 경우에도 hit rate가 평균 45% 정도로 지역성이 높다. 반면 UC의 로그는 hit rate이 25%정도로 로그들에 대한 지역성이 상당히 작은 편이다. 이러한 지역성의 차이는 prefetching을 통해 성능이 향상되는 정도에 영향을 준다.

## 5. 결론 및 향후 연구

본 논문에서는 웹 캐쉬 prefetching 적용의 근거를 마련하기 위한 기본 실험으로 NLANR의 최상위 캐쉬와 서울대 교육망의 access 로그를 분석하였다. 분석 결과 요청의 빈도에 대한 분포 경향이 존재하는 것을 발견하였고, 전체 로그 중 약 70% 가 3Kbyte 미만의 오브젝트에 대한 요청이며, request 사이의 간격도 일정 패턴을 보이는 것을 발견하였다. 이것을 바탕으로 지난 로그를 가지고 Top list를 만들어 한가한 시간에 서버에 부하를 주지 않는 범위 내에서 prefetching한 결과 hit rate의 경우는 3%, 응답시간의 경우는 5% 정도의 성능 개선을 보였다.

향후 본 결과가 보다 실제적으로 효용이 있음을 보이기 위해 요청이 몰리는 peak 시간대에 응답시간을 얼마나 개선하는가에 대한 실험이 이루어질 예정이며 신문기사와 같이 특정한 주기를 가지고 내용이 변하는 오브젝트에 관한 prefetch가 얼마나 효과가 있는가하는 실험도 진행될 예정이다.

## ※ 참고문헌


[1] Abrams M., Charles R. Standrigde, Ghaleb Abdulla, Stephen Williams and Edward A. Fox. "Caching Proxies: Limitations and Potentials". *In Proceedings of Fourth International World Wide Web Conference*, December 1995.

[2] Balachander Krishnamurthy and Craig E. Wills, "Proxy Cache Coherency and Replacement-Towards a More Complete Picture," *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, 1998.

[3] Dilley, J. and M. Arlitt, "Improving Proxy Cache Performance: Analysis of Three Replacement Policies," *Internet Computing*, Vol. 3, No. 6, IEEE, November 1999.

[4] Hyokyung Bahn, Sam H. Noh, Sang Lyul Min, and Kern Koh, "Using Full Replacement History for Efficient Document Replacement in Web Caches," *Proceeding of USITS'99: The 2nd USENIX Symposium on Internet Technologies & Systems*, Colorado, USA, October 11-14, 1999.

[5] Duska, B., D. Marwood and M. J. Feeley, "The Measured Access Characteristics of WWW Client Proxy Caches," *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997.

[6] Rousskov, A., "A performance Study of the Squid Proxy on HTTP/1.0," *WWW Journal 99*, 1999.

[7] Dan Duchamp, "Prefetching Hyperlinks," *Proceedings of USITS '99: The 2nd USENIX Symposium on Internet Technologies & Systems*, October 11-14, 1999.

[8] Ken-ichi Chinen and Suguru Yamaguchi, "An Interactive Prefetching Proxy Server for Improvement of WWW Latency," *In Proceedings of 7th Annual Conference of the Internet Society*, Kuala lumpur, June 1997.

[9] Venkata N. Padmanabhan and Jeffrey C. Mogul, "Using Predictive Prefetching to Improve World Wide Web Latency," *Proceedings of SIGCOMM 96*, 1996.

[10] Evangelos P. Marcatos and Catherine E. Chronaki, "A Top-10 Approach to Prefetching the Web," *In Proceedings of 8th Annual Conference of the Internet Society*, Geneva, July 1998.



백 윤 철

1998 서울대학교 계산통계학과 이학사
1990 서울대학교 전산과학 이학석사
1995 서울대학교 전산과학 이학박사
1996 ~ 현재 상명대학교 소프트웨어학부 조교수