웹 서버의 참조 특성 분석과 성능 개선

안 효 범[†] · 조 경 산[††]

요 약

웹의 기하급수적인 성장과 클라이언트의 비균일적 요청 특성은 웹 서버의 성능에 큰 영향을 주었으며, 이의 해결책으로 서버 캐쉬가 제안되었다. 본 논문에서는 웹 서버 참조의 반복성, 참조 크기 및 참조적 지역성 등의 특성들을 분석하고, 그 결과를 이용하여 서버 캐쉬의 히트율을 높이기 위한 기존 캐쉬 제거 방식의 분석과 예측을 통한 서버 캐쉬의 선인출 기법을 제안한다. 또한, 웹사이트의 추적 자료에 의한 시뮬레이션을 수행하여 제안된 기법에 의한 성능 개선을 분석 제시한다.

Analysis of Web Server Referencing Characteristics and Performance Improvement of Web Server

Hyobeom Ahn[†] · Kyungsan Cho[††]

ABSTRACT

Explosive growth of the Web and the non-uniform characteristics of client requests result in the performance degradation of Web servers, and server cache has been recognized as the solution. We analyzed Web server accessing characteristics-repetition, size, and locality of access. Based on the result, we analyzed the cache removal policies and proposed a prefetch strategy to improve the hit ratio of server caches. In addition, through the trace-driven simulation based on the traces from real Web sites, we showed the performance improvement by our proposal.

키워드 : 웹서버(Web Sever), 서버 캐쉬(Server Cache), 선인출(Prefetch), 히트율(Hit Ratio), 캐쉬정책(Cache Policy)

1. 서 론

웹은 인터넷으로 연결된 대단위 분산 정보시스템으로 서버, 프록시, 클라이언트 등으로 구성된다. 클라이언트가 정보를 요청하면, 웹 서버(또는 프록시 서버)는 요청된 페이지의 파일을 네트워크를 통해 클라이언트에게 제공한다. 웹 상의 정보는 매월 15% 이상씩 증가하고 인터넷에 연결된 호스트 컴퓨터의 수는 매 9~12개월에 두 배 이상씩 증가하며, 웹에서 제공되는 서비스도 단순 텍스트에서 고용량의 다양한 미디어 정보로 변화하고 있는 등 웹은 기하 급수적으로 성장하고 있다[1, 2]. 하지만, 클라이언트의 정보 요청은 근본적으로 비균일적(non-uniform)이므로 편중된 웹서버로의 접근과 이로 인한 과부하가 웹 참조의 지연을 심각하게 증가시키고 있다[3].

이러한 측면에서 웹의 성능을 개선하려는 여러 연구가 수행되었으며, 통신 기술 및 프로토콜과 네트워크 구조의 지속적인 개선과 캐쉬와 선인출(prefetch) 기법을 활용하여 네트워크와 웹서버의 지연 시간을 감소하고 부하를 경감하기 위한 기법들이 제시되었다[4-6].

인기 있는 페이지를 클라이언트에 인접한 위치에 캐싱하는 것이 웹의 교통량 감소 및 확장성에 유용한 것으로 분석되어 클라이언트 캐쉬 구조가 사용되었다. 또한, 방화벽 내의 같은 도메인(또는 그룹) 상의 클라이언트들에 의해 공유되는 프록시 서버의 효율을 높이기 위한 프록시 캐쉬가 제시되었다.

캐쉬와 더불어 참조 요청들의 순차적 참조 패턴 및 전체 파일들의 연결 구조를 고려한 연결적 지역성, 또는 참조 경로에 대한 프로파일 등을 이용하여 요청을 미리 예측함으로써 캐쉬의 효율을 높이기 위한 선인출(prefetch) 기법들이 사용되었다.

위에서 제시한 바와 같이 캐쉬 및 선인출에 대한 기존 연구는 주로 프록시와 클라이언트에서 이루어 졌다[1]. 그러나, 클라이언트 요청의 비균일성으로 인해 전체 웹 참조 지연 중에서 상당 부분이 웹 서버의 처리로 소요되며, 웹 서버의 성능이 웹 서비스의 질에 보다 큰 영향을 미치게되어 웹서버의 개선에 관한 연구도 활발히 제안되었다[7].

본 연구에서는 서버의 참조 특성을 참조의 반복성, 참조 크기 및 연결적 지역성 측면에서 분석하여 이를 기반으로 기존 웹 캐쉬의 제거 정책 특성을 개선하고, 서버의 유휴 시간(idle time)에 요청되어질 자료를 예측하여 디스크로부터 캐

※ 본 논문은 2000년 단국대학교 대학연구비의 지원으로 연구되었음.
† 정 회 원 : 천안공업대학 정보통신공학과 교수
†† 종신회원 : 단국대학교 전산통계학과 교수, 연구책임자
논문접수 : 2001년 7월 20일, 심사완료 : 2001년 8월 10일

쉬에 미리 가져다 놓는 선인출 방법을 제안한다. 또한 제안된 기법에 의한 성능 개선 정도를 추적 기반 시뮬레이션을 통해 분석하여 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 서버 캐쉬의 제거 정책 및 선인출에 관한 기존연구를 분석한다. 3장에서는 웹 서버의 실제 접근 로그 파일의 추적 자료를 분석하여 웹 참조의 여러 특성들을 제시한다. 제시된 결과를 활용하여 4장에서 제거 정책을 분석하고 선인출 기법을 제안하며, 제안된 기법에 대한 시뮬레이션 분석 결과를 제시하며, 5장의 결론으로 마무리한다.

2. 캐쉬 제거 및 선인출의 기존 연구

캐쉬의 저장 용량에는 한계가 있으므로, 새로운 자료를 캐쉬에 저장하기 위해서 캐쉬에 저장된 자료 중에서 향후에 요청될 확률이 가장 낮은 자료를 제거해야한다. 이와 같이 캐쉬에 저장할 자료와 제거할 자료의 선정이 서버캐쉬의 성능에 큰 영향을 준다.

웹에서는 전통적 메모리 시스템의 캐쉬와는 달리 저장되는 자료가 파일 단위이고, 웹 참조는 대부분이 읽기(read)이며, 또한 참조의 인기도에 따라 파일의 참조수가 정해지는 특성을 가지므로 기존 메모리 시스템의 대표적인 LRU 제거 정책은 부적절하다고 분석되었다[8]. 따라서 웹 참조의 특성에 따른 참조 횟수, 저장크기, 캐쉬에 저장된 시간 등을 활용하는 제거 방식들이 제시되었다.

즉, LRU이외에 LFU, SIZE, FIFO 등의 기법들이 연구되었다[9]. 하지만, 한가지 특성만으로는 웹서버 캐쉬의 성능 향상에 한계가 있으므로, 캐쉬에 저장되는 파일의 크기 특성을 다른 특성과 함께 사용하는 혼합정책이 제시되었다.

초기의 혼합 정책들로는 파일의 크기에 우선 순위를 두고, 파일의 크기가 일정한 경우에는 다른 특성 인수(마지막 참조 시간 또는 참조 횟수)를 비교하는 LRU-SIZE(또는 LFU-SIZE)가 있다[9]. 그러나, 이 방법들은 파일 크기의 미소한 차이에 의해서 제거될 파일이 결정되어 두 번째 특성인수의 큰 차이는 고려되지 않는다는 문제점이 있다.

그러한 문제를 완화하기 위해 제시된 LRU-MIN[9], LRU-TH[10]도 파일 크기의 특성이 우선적으로 적용되고 다른 특성은 상대적으로 제거될 파일 선정에 대한 영향이 적게 되어 여러 특성 인수들을 효율적으로 적용하지 못하는 문제점이 있다. 이의 해결책으로 두 참조 특성들이 캐쉬 제거에 동일한 영향을 주는 SLRU(Self-adjusted LRU)[11], 정적 캐쉬(Static Cache) 및 WLFU(Weighted-LFU)[12]등의 제거 기법들이 제시되었다. 이 정책들은 $\frac{\text{이익}}{\text{비용}}$을 각 제거 정책의 측도로 사용한다. 즉, 제거 기법에서 비용에 대한 성능적 이익의 비를 참조 특성들로 수식화하고, 이 값이 가장 작은 파일(결국 비용에 비해 이익이 가장 작은)을 캐쉬에서 제거한다. 이때 비용은 파일의 크기(메모리 비용)를 나타낸다.

SLRU 기법에서는 $\text{이익} = \frac{1}{\text{최종 참조 이후 경과된 시간}}$ 로 설정하므로,

$\frac{1}{(t-t_{last})\cdot S}$ t : 현재 시간, t_{last} : 최종 참조시간, S : 파일크기

의 값을 구해 그 값이 가장 작은 파일을 캐쉬에서 제거한다.

정적 캐쉬 기법과 WLFU정책은 이익=참조수로 설정하되, 주기적(정적)으로 또는 동적으로

$$\frac{\text{참조수}}{\text{파일 크기}}$$

의 값이 가장 작은 파일을 제거한다. 위의 기법들의 비교 연구에 의하면, WLFU가 히트율이 가장 높은 것으로 분석되었다[12].

위의 방법들은 이익과 비용을 표시하는 실제 참조 특성 인수들이 동일한 가중치로 계산되었는데, 각 항목의 실제 가중치는 고려되지 않았다. 예를 들어, SLRU에서 최종 참조된 이후의 시간이 2배가 된 것과 참조 크기가 2배가 된 것이 동일한 가치가 있는지, 또한 WLFU에서 2배의 참조수와 1/2배의 참조 크기인 파일이 동일한 가치가 있는지는 더 분석될 필요가 있다.

앞에서 설명된 제거정책뿐만 아니라, 캐쉬에 저장할 파일의 선정도 캐쉬의 성능에 큰 영향을 준다. 따라서, 캐쉬의 성능을 높이기 위해 요청될 확률이 높은 파일을 미리 예측하여 캐쉬에 저장하는 선인출 방법이 제안되었다. 만약 선인출된 파일이 요청되지 않는다면 선인출을 위한 과부하만 증가된다. 따라서, 참조 요청들의 순차적 참조 패턴, 전체 파일들의 연결구조를 고려한 연결적 지역성, 또는 참조 경로에 대한 프로파일등을 이용하여 요청을 미리 정확히 예측하려는 선인출 기법들이 있었다[5, 6]. 또한, [13]에서는 클라이언트와 서버 사이에서 웹 클라이언트의 과거의 접근 패턴을 통해 사용자의 미래 웹 참조를 예측하는 선인출을 수행하였고, [14]에서는 웹 서버에서 관찰된 참조 패턴을 이용하여 프록시에서 선인출을 수행하였다. 이와 같이 기존 선인출 방법은 프록시 또는 클라이언트가 요청할 가능성이 있는 파일들을 미리 예측하여 서버로부터 가지고 오는 방법을 사용하였다.

3. 웹 서버의 참조 특성 분석

실제 다양한 여러 유형의 웹서버들은 서로 상이한 참조 특성을 가지므로 기존 서버 연구는 해당 서버에서의 참조 특성 분석이 선행되어 왔다. 예를 들면, 대용량 서버에 대한 분석, 소규모 서버의 추적 자료와 대규모 서버의 추적 자료, 또는 Webstone 또는 작업부하 생성기에 의한 합성 벤치마크에 의한 분석이 수행되었다[7]. 또한, [15]에서는 접근 로그로부터 사용자 요청에 의한 트래픽 패턴 모델을 분석하였다. 따라서, 본 연구에서도 웹서버의 작업부하 분석을 제시한다.

3.1 웹서버의 참조 추적 자료

본 연구에서는 웹서버 참조 특성의 분석을 위하여 <표 1>에 보이는 4개의 웹 사이트에서 추적 수집된 접근 로그 파일(access_log file)을 분석하였으며, 이들은 http : //ita.ee.lbl.gov/html/traces.html에서 참조할 수 있다.

〈표 1〉 추적된 작업부하

추적 자료 이름	웹 사이트	자료 추적 기간
Calgary	Calgary 대학 전산학과 웹 서버	1년
Clarknet	분주한(busy) ISP 웹 서버	1주
Nasa	분주한(busy) 웹 서버	2개월
Uofs	Saskatchewan 대학 웹 서버	7개월

<표 2>은 분석에서 사용하지 않는 요청을 제거한 각 추적 자료에 대한 요약이다.

〈표 2〉 작업부하 특성

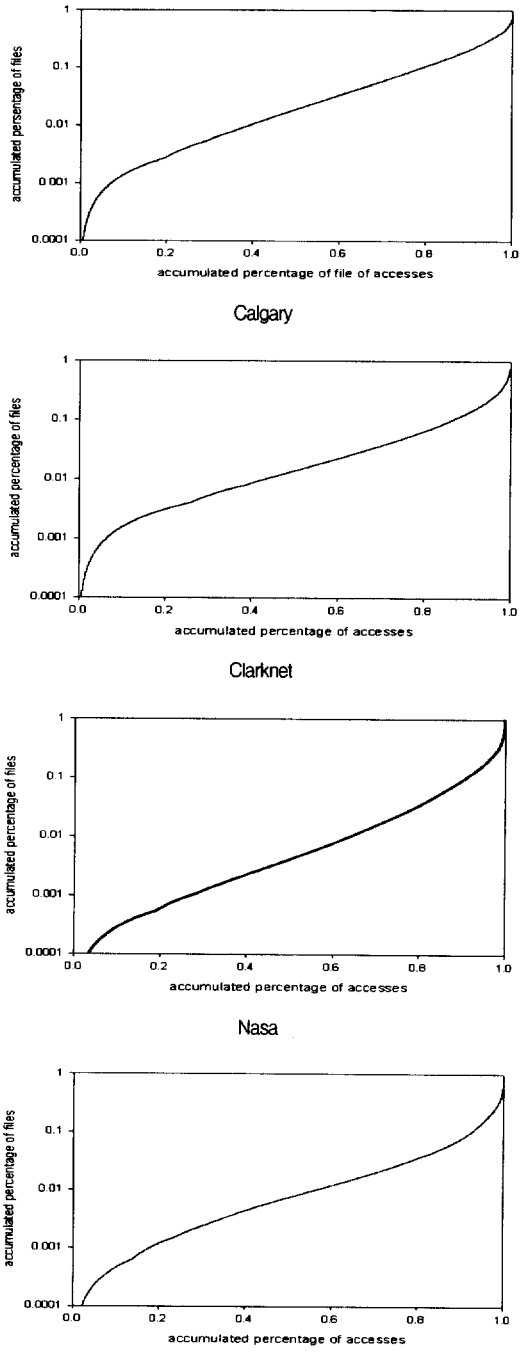
	Calgary	Clarknet	Nasa	Uofs
파일 수	8,018	24,992	7,207	13,552
파일들의 크기 합	294,216,389	325,152,664	223,937,401	190,862,855
참조수의 합	566,649	1,450,499	1,689,827	916,632
참조된 바이트의 합	7,927,044,096	14,357,951,488	38,460,829,696	3,906,178,048
최대히트율 (캐쉬크기무한대)	0.984	0.997	0.995	0.984
최대바이트히트율 (캐쉬크기무한대)	0.978	0.994	0.967	0.958

위의 4개 사이트의 추적 자료 중에서 Clarknet과 Nasa는 1주 및 2개월의 짧은 추적 기간 동안에 각기 145만과 169만의 참조가 요청된 분주한(busy) 웹사이트로 평균 참조수는 각각 20.7만/일 및 2.8만/일이다, Calgary와 Uofs는 참조수에 비해 작업 부하의 추적기간이 1년과 7개월로 길며 평균 참조수는 각각 1552개/일 및 4364개/일이다. 본 논문에서 제시할 캐쉬의 정책은 서버의 과부하에 의한 지연과 성능 저하를 막기 위한 것이므로, 요청이 단 시간 내에 많은 참조 횟수를 갖는 웹서버의 작업부하가 더 적합하다. 그러므로 본 논문에서는 Clarknet과 Nasa에서의 성능 향상이 더 중요한 의미를 갖게된다.

3.2 참조의 반복 특성

웹 서버 참조의 반복 특성은 [16-18]에서 분석되었으며, 웹서버에서 요청되는 문서들은 전체 문서들 중에 일부분만이 많이 참조된다는 특성이 제시되었다. 또한, 웹서버에서 참조되는 파일들의 참조가 Zipf의 법칙을 따른다는 분석이 있었다[22]. Zipf의 법칙은 인기 있는 파일의 참조수가 그렇지 않은 파일의 참조수 보다 월등히 높음을 보인다.

(그림 1)은 앞 절에서 제시된 작업부하에서 파일들의 참조 분포를 로그 눈금으로 보인다. 기존 연구에 의하면, 서버에서는 전체 파일의 10%가 전체 참조수의 90%를, 프록시 서버에서는 전체 파일의 25%~40%가 전체 참조수의 70%를 나타낸다[19].



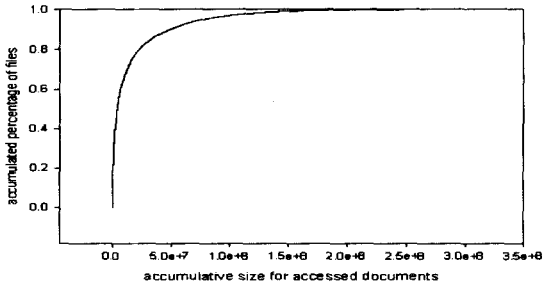
(그림 1) 참조 수에 따른 파일 분포

(그림 1)에서 보는 바와 같이 분석된 작업부하들은 전체 파일의 10%가 전체 참조수의 90%이상을 차지하여 기존 서버의 분석과 일치함을 알 수 있다. 이것은 비균일적 참조 특성에 의한 편중된 서버의 인기도(popularity)때문에 나타나는 성질이다. 따라서, 전체 파일의 10%만 캐쉬에 저장한다면, 최대 90%의 캐쉬 히트율을 얻을 수 있게 된다. 만약 참조 특성이 시간적으로 급속히 변경되지 않는다면, 과거의 참조수는 미래의 참조 확률을 나타낸다. 또한, 최종 참조 시간보다는 참조수가 캐쉬의 제거뿐만 아니라 선인출을 위

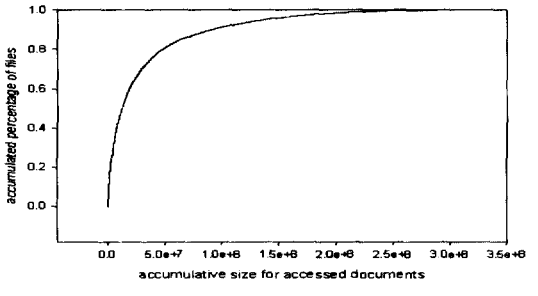
한 예측 기법에도 활용될 수 있다.

3.3 참조의 크기 특성

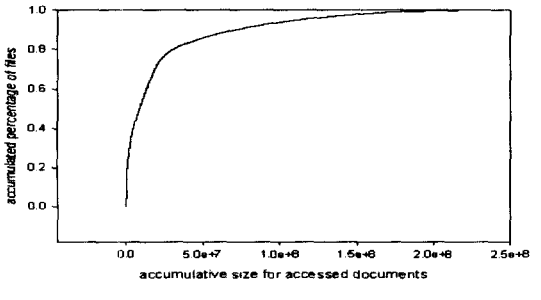
전통적인 캐쉬에 저장하는 고정된 크기의 블록과는 달리, 웹 서버 캐쉬는 가변적 크기의 파일 단위로 저장한다. 따라서 요청된 문서의 크기는 고정되지 않고 매우 큰 편차를 가질 수 있다.



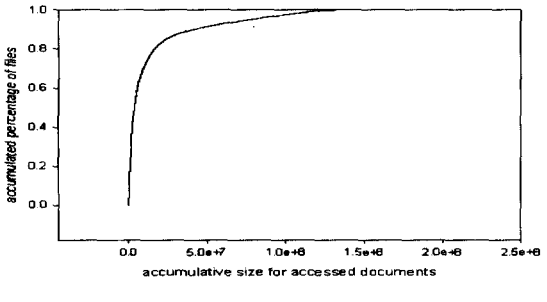
Calgary



Clarknet



Nasa



Uofs

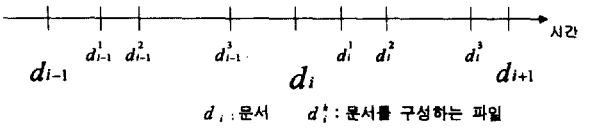
(그림 2) 파일의 크기 분포

(그림 2)는 참조되는 파일의 크기에 따른 분포도이다. 파일의 크기는 6바이트에서 약 11M바이트(11855882바이트)까지 다양한 크기 분포를 가지지만, 대부분의 파일은 10K바이트 이하의 파일이 각 작업부하에서 72%이상을 차지함을 알 수 있다. 또한, 크기가 작은 파일들이 참조수가 많음을 알 수 있는데 10K바이트 이하의 파일들이 각 작업 부하에서 71%이상의 참조수를 가지며, Uofs에서는 총 참조수의 92.8%가 10K바이트 파일들에서 발생했다. 이러한 사실로 볼 때 파일의 크기가 10K바이트 이하의 파일들이 실제 참조되는 파일들의 대부분임 알 수 있다. 따라서, 너무 큰 크기의 파일을 캐쉬에 저장하게되면 캐쉬의 한정된 크기 때문에 많은 작은 크기의 파일들이 캐쉬에 저장되지 못하여 캐쉬의 활용도 및 성능이 감소할 수 있다. 하지만, 동일한 참조수를 갖는 경우에는 큰 파일을 저장하는 것이 제공하는 정보량에 대하여 유리하다. 따라서, 파일의 크기가 캐쉬의 관리 정책에 고려되어야 한다.

3.4 웹 참조의 지역성 특성

전통적인 메모리 시스템의 참조는 지역성 특성이 있으며, 전통적인 캐쉬는 이 특성을 이용하여 블록의 크기나 캐쉬의 관리 정책이 정해진다. 즉, 공간적 지역성 특성에 의해 인접한 메모리에 저장되는 정보를 하나의 블록으로 캐쉬에 저장하고, 시간적 지역성 특성에 의해 다시 참조될 블록을 캐쉬에 계속 저장하여 히트율을 높이게 된다. 웹 서버에서는 3.2절에서 보인 바와 같이 시간적 지역성 특성이 있으며, 전통적인 메모리 시스템과는 다른 형태의 공간적 지역성 특성을 발견할 수 있다. 즉, 요청된 문서가 여러 파일들로 구성된다면, 그 문서를 구성하는 파일들은 문서가 요청된 이후 곧 요청될 확률이 매우 높다는 특성이 있다. 자주 접근되는 문서를 구성하는 파일들과의 관계에서 캐쉬에 활용할 수 있는 공간적 지역성을 발견할 수 있다.

일반적으로 웹 클라이언트는 웹 서버에게 문서를 요청한 후 그 문서를 구성하고 있는 파일들을 곧 요청하게되는 참조 패턴을 유지한다. 이는 문서와 문서를 구성하는 파일들 간에 지역성을 가진다는 것을 의미하며, 본 논문에서는 이를 참조적 지역성이라 정의한다. (그림 3)은 문서와 이를 구성하는 파일들간의 참조 패턴을 보여준다.



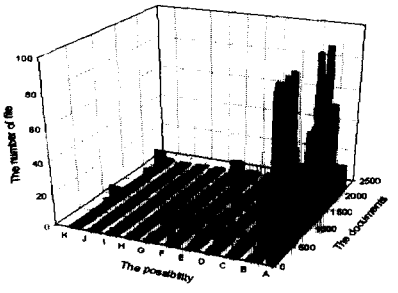
(그림 3) 문서의 참조 패턴

한 문서(d_i)가 참조된 후 다른 문서가 요청되기 전까지 그 문서를 구성하는 파일들(d_i^k)이 뒤이어 요청되는 횟수($\|d_i^k\|$)는 문서가 요청되는 수($|d_i|$)보다 작거나, 크거나 또는 같게 된다.

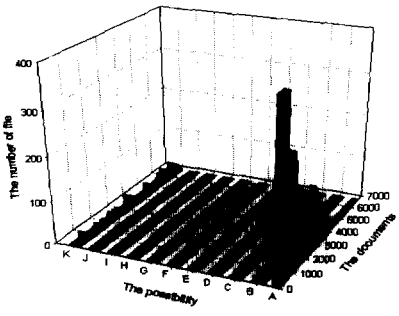
따라서, 문서를 구성하는 파일들이 요청될 확률(p_i)은 다음과 같이 정의 할 수 있다.

$$p_i = \mathrm{Min}\left(\frac{\| d_i^k \|}{| d_i |}, 1\right)$$

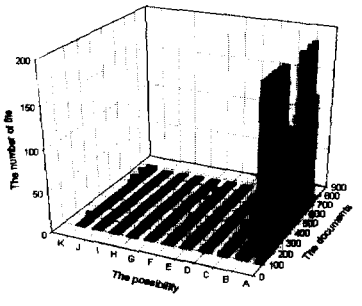
p_i가 1이면 d_i^k가 d_i 이 후에 참조됨을 의미하며, p_i가 클



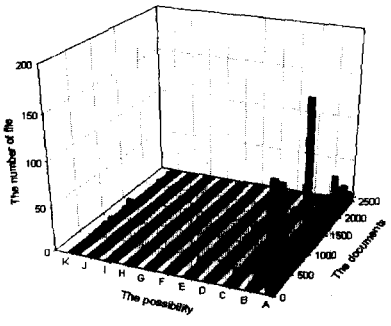
Calgary



Clarknet



Nasa



Uofs

(그림 4) 각 작업부하의 참조적 지역성- A : 0%-9%, B : 10%-19%, C : 20%-29%, D : 30%-39%, E : 40%-49%, F : 50%-59%, G : 60%-69%, H : 70%-79%, I : 80%-89%, J : 90%-99%, K : 100%이상

수록 문서가 참조된 후, 그를 구성하는 파일이 요청될 가능성이 크다는 참조적 지역성의 정도를 나타낸다고 볼 수 있다. (그림 4)는 접근 로그 파일을 통해 각 문서와 그를 구성하는 파일에 대한 p_i의 그래프이다.

(그림 4)는 문서를 구성하는 파일들이 그 문서에 뒤이어 요청되는 참조적 지역성을 보이며, 파일들을 미리 예측하여 캐쉬에 가져오는 선인출 기법에 사용할 수 있다.

4. 웹 캐쉬의 성능 개선을 위한 제안

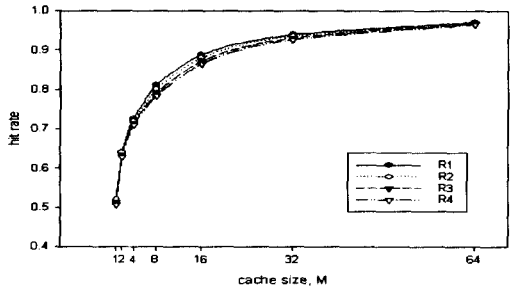
가변적 저장 크기로 인하여 웹 서버 캐쉬의 성능은 전통적인 캐쉬와는 다른 측도로 평가된다. 즉, 두 가지 히트율(참조 히트율과 바이트 히트율)로 서버 캐쉬의 성능을 정의하는데, 전체 참조 수에 대한 서버 캐쉬에서 히트된 참조 수의 비율이 히트율(hit rate, 전통적인 캐쉬의 히트율과 같다)이고, 전체 참조되는 정보의 양에 대한 서버 캐쉬에서 히트된 정보의 비율이 바이트 히트율(byte hit rate)이다.

웹 캐쉬에서는 일반적으로 히트율이 높으면 디스크의 접근 횟수가 줄게되며, 바이트 히트율이 높으면 디스크에서 읽어 올 정보의 양이 줄게된다. 따라서, 본 장에서는 캐쉬의 히트율 또는 바이트 히트율을 높이는 기법들이 연구된다.

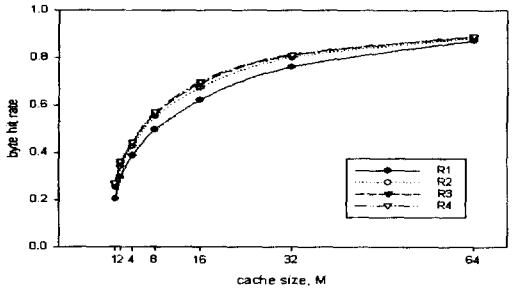
4.1 가중치를 부여한 캐쉬 제거 정책

캐쉬의 크기는 제한되므로, 캐쉬 미스 시에는 가장 참조될 확률이 낮은 항목만을 선정하여 캐쉬에서 제거해야 한다. 웹의 참조 특성을 고려하여 사용되는 제거 방법은 2장에서 설명된 바와 같다. 현재까지 알려진 가장 효율적인 제거 방법은 WLFU이며, 이 방법에서는 $\frac{\text{참조수}}{\text{파일크기}}$의 값이 가장 적은 파일을 캐쉬에서 제거한다. 하지만, 이 기법에서 사용된 참조 특성 인수인 참조수와 파일 크기가 동일한 가중치를 갖는지는 검증되고 있지 않다. 예를 들어, 10K의 저장 크기가 있을 때 100번씩 참조되는 1K 크기의 파일을 10개 저장하는 것과, 1000번 참조되는 10K 크기의 파일을 하나만 저장하는 것은 어느 것이 유리한가? 저장되는 파일만을 고려하면, 두 크기의 파일은 모두 동일한 비교값을 갖으며 히트율은 양쪽이 모두 같으나, 바이트 히트율은 10K바이트의 파일이 더 높게 된다. 따라서, 본 연구에서는 특성 인수들의 값에 따라 서로 다른 가중치가 성능에 미치는 영향을 분석하였다.

제거될 파일의 우선 순위를 결정하기 위해 WLFU에서 제시된 $\frac{\text{참조수}}{\text{파일 크기}}$를 $\frac{\text{참조수}^m}{\text{파일크기}^n}$으로 수정하였다. 이 기법을 VWLFU(Variable WLFU)라 정의한다. 여기서, n, m은 변화되는 가중치이다. 파일의 크기와 참조수에 각각 다른 가중치 값을 부여하여 Clarknet은 참조수의 가중치를 변화시켰고, Nasa는 파일의 크기를 변화시켰다. (그림 5)의 (a)의 결과에서 Clarknet에 참조수에 가중치를 주었을 경우에는 히트율은 감소하였으나, 바이트 히트율은 증가하는 결과를

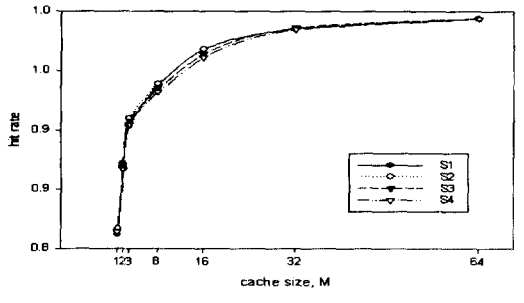


Clarknet

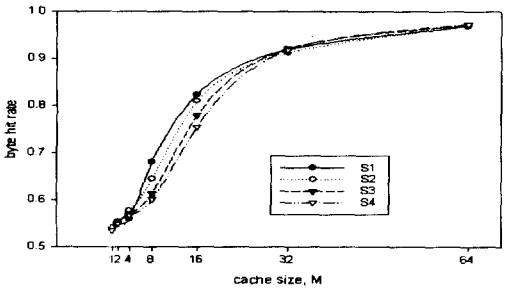


Clarknet

(a) 참조수에 가중치를 부여한 경우



Nasa



Nasa

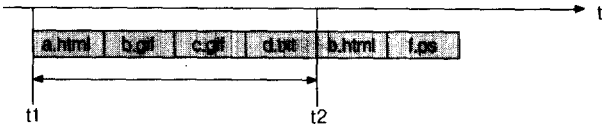
(b) 파일크기에 가중치를 부여한 경우

(그림 5) 가중치의 변화에 따른 캐쉬 히트

보였다. 그리고 (그림 5) (b)의 결과에서 Nasa에 파일크기에 2배(S2)의 가중치를 주었을 경우에는 히트율이 증가되고 바이트 히트율이 감소함을 보였다. 이로부터, 제거정책에 특성인수의 가중치를 변화시키는 방법이 성능에 영향을 끼치는 것을 보였고, 캐쉬 정책의 선정 시에 적절한 가중치를 부여함으로서 성능의 향상을 얻을 수 있음을 보였다.

4.2 선인출(Prefetch)기법

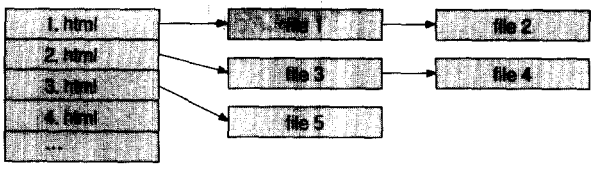
캐쉬의 크기가 한정된다면 가장 참조될 확률이 높은 파일만을 선정하여 캐쉬에 저장하여야 하며, 이러한 목적으로 선인출 기법이 사용될 수 있다. 파일을 선인출하기 위해 접근되는 파일들에 대한 참조 트리를 이용하는 방법이 제안되었으며, 온라인 분석적 모델링 기법을 기반으로 파일 접근 패턴을 사용하여 선인출이 수행되었다[20]. 이와 유사하게, 3.4절에서 보여진 것처럼 HTML문서에 뒤이어 요청되는 파일들에 대한 참조 확률(p_i)을 구할 수 있고 이를 이용하여 문서에 뒤이어 요청될 확률이 높은 파일을 미리 캐쉬에 저장할 수 있다.



(그림 6) HTML문서의 요청과 그 외 파일들에 대한 요청열

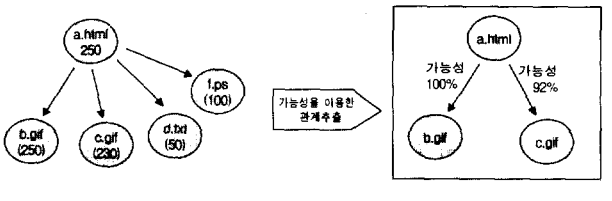
웹 서버는 (그림 6)에서 보는 것과 같이 일련의 요청을 받아들인다. 브라우저에서 HTML문서를 받은 뒤, 그 문서를 구성하는 파일을 GET명령을 사용하여 또는 GETLIST 명령을 사용하여 연속적으로 호출을 하게된다[21]. 그러므로, (그림 6)처럼 요청이 받아들여졌다면, 문서 a.html과 문서 b.html의 요청이 발생한 시각 t1과 t2사이의 시간에 요청되는 파일 b.gif, c.gif, d.txt들은 a.html 문서를 구성하는 파일일 확률이 높다. 따라서, 이들이 a.html에 뒤이어 요청될 확률이 높을 수 있다.

(그림 7)은 각 문서에 이어 접근될 파일들을 기록한 자료구조이다. 이 자료구조에는 문서에 뒤이어 요청되는 파일들의 참조 횟수를 기록한다.



(그림 7) 선인출 정보를 저장하기 위한 자료구조

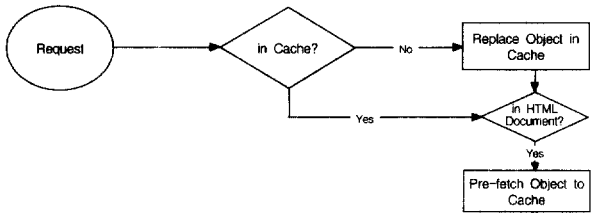
이렇게 구해진 횟수는 각 요청된 문서에서 선인출 될 파일들을 결정하기 위하여 (그림 8)에서 보는 것과 같이 문서와 파일들간의 관계를 추출하는데 사용된다.



(그림 8) 문서와 포함된 파일의 관계 추출

파일들을 선인출하는 제안 알고리즘은 (그림 9)와 같으

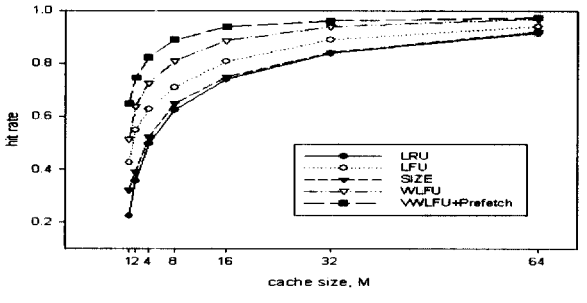
며, 구현에서는 선인출 버퍼와 캐쉬를 구분하지 않고 같이 사용하였다.



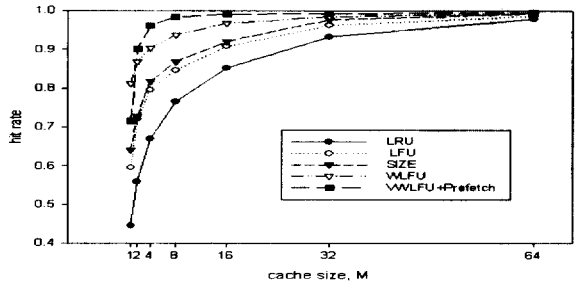
(그림 9) 선인출 알고리즘

제시된 선인출 방법은 (그림 9)에서 보는 것처럼 요청된 파일이 캐쉬에 없을 경우에는 디스크로부터 읽어오고, 요청을 받은 파일이 HTML문서일 경우에만 연관된 파일들을 선인출한다. HTML문서가 캐쉬에 있을 경우에는 문서와 연관된 파일들을 선인출을 한다. HTML문서일 경우에만 선인출을 수행한 이유는 일반 파일들에서는 관계를 추출하기가 불가능하기 때문이다.

본 연구에서 제시된 기법들의 성능영향을 검증하기 위해, 기존연구에서 제시된 캐쉬 제거 정책들을 사용한 경우와 4.1절에서 제안된 VWLFU와 선인출 기법(VWLFU + prefetch)을 병행하여 사용한 경우를 시뮬레이션 하였다. 작업부하는 3장에서 제시된 작업부하 중 의미가 큰 Clarknet와 Nasa을 사용하였고, 예측을 위한 관계를 추출하기 위해 HTML문서에 뒤이어 참조될 확률(p_i)이 30%이상을 가지는 파일에 대하여 선인출을 수행하였다. 시뮬레이션 결과는 히트율 (그림 10)과 바이트 히트율 (그림 11)의 두 가지로 분석되었다.



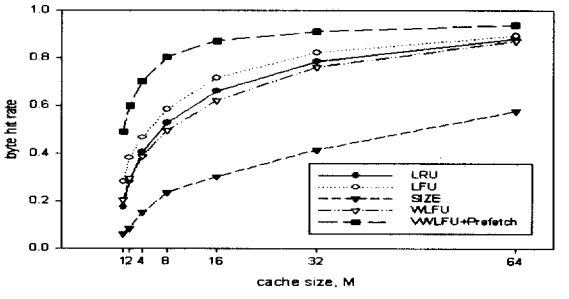
Clarknet



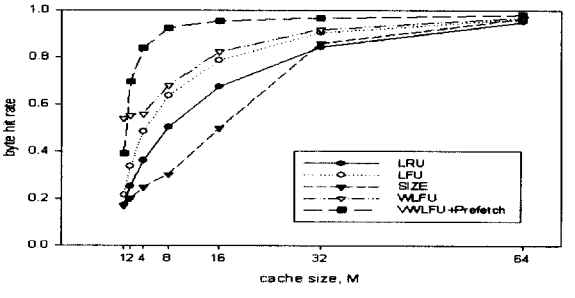
Nasa

(그림 10) 선인출 기법과 기존의 정책과 비교(히트율)

제시된 기법은 기존의 캐쉬 제거 정책만을 사용했을 경우보다 히트율과 바이트 히트율이 10%이상의 향상(Nasa작업부하에서 캐쉬의 크기가 1M일 경우를 제외하고)을 보이고 있다. Nasa작업부하에서 캐쉬크기 1M에서 히트율과 바이트 히트율이 낮은 이유는 선인출로 인하여 참조될 확률이 높은 파일이 작은 크기의 캐쉬에서 제거되었기 때문이다.



Clarknet



Nasa

(그림 11) 선인출 기법과 기존의 정책과 비교(바이트히트율)

5. 결 과

웹에서 지연을 줄이고 성능을 개선하는 방법으로 캐쉬 제거 정책과 선인출 방법이 제안되고 연구되었다. 웹 캐쉬는 클라이언트, 서버, 그리고 프록시에서 구현될 수 있는데, 본 논문에서는 서버의 지연과 과부하를 줄여 성능을 향상시키기 위한 서버 캐쉬에 대하여 연구되었다.

캐쉬의 용량에는 한계가 있으므로, 캐쉬에 가장 참조될 확률이 높은 정보를 저장하기 위한 기법과 가장 참조될 확률이 낮은 정보를 제거하는 정책이 캐쉬에 큰 영향을 준다. 기존 연구에 의하면 서버 캐쉬의 제거정책 중에서 참조수와 파일의 크기에 동일한 가중치를 부여하는 WLFU가 제일 우수하다고 알려져 있다. 본 논문에서는 특성 인수들에게 가중치를 달리하여 작업부하에 따라 적응하는 제거 정책을 제시하고 이에 의한 히트율 또는 바이트 히트율이 영향을 보였다. 또한, 캐쉬 제거 정책만에 의한 서버 캐쉬의 성능 향상에는 제한이 있음을 분석하고, 캐쉬에 저장할 자료를 과거의 참조적 지역성 이용하여 예측하는 선인출 방법을 제안하였다.

제안된 제거 정책과 선인출 기법을 병행한 결과, 대부분의 경우에서 히트율과 바이트 히트율에서 10%이상이 개선

됨을 보였다. 따라서, 캐쉬에서 제거정책과 선인출 기법의 혼합 사용이 웹 서버의 성능 개선에 효과가 있음을 제시하였다. 다양한 캐쉬 제거 정책과 여러 선인출 기법들과의 상세한 연관성에 관하여 연구중이다.

참 고 문 헌

[1] Jia Wang, "A survey of web caching schemes for the Internet," ACM Computer Communication Review, Vol.29, No.5, pp.36-46, October, 1999.

[2] 조경산 역, 컴퓨터 네트워크와 인터넷, 도서출판 그린, 2000.

[3] Mohammad S. Raunak, Prashant Shenoy, Pawan Goyal, and Krithi Ramamritham, "Implications of proxy caching for provisioning networks and server," In Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2000), pp. 66-77, Santa Clara, CA, June 2000.

[4] E. P Markatos and C. E Chronaki, "A Top-10 Approach to Prefetching on the Web," In Proceeding INET 98. July, 1998. http : //www.ics.forth.gr/proj/arch-vlsi/html_papers/INET98_prefetch/paper.html.

[5] Dan Duchamp, "Prefeching Hyperlinks," In Proceedings of the USENIX Symposium on Internet Technologies and Systems, 1999. http : //www.usenix.org/events/usits99/ duchamp.html.

[6] T. Ibrahim and C. Xu, "Neural Net Based Pre-fetching to Tolerate WWW latency," In Proc. of the 20th Int'l Conf. on Distributed Computing Systems (ICDCS2000), April, 2000.

[7] Erich Naham, Tsipora Barzilai, and Dilip Kandlur, "Performance Issue in WWW Servers," In Proc. ACM SIGMETRICS 99, Vol.26, No.1, pp.216-217, May, 1999.

[8] Luigi Rizzo and Lorenzo Vicisano, "Replacement Policies for a Proxy Cache," IEEE/ACM Transactions on Networking, Vol.8, No.2, pp.159-170, April, 2000.

[9] Stephen Williams, Marc Abrams, Charles Standridge, Ghaleb Abdulla, and Edward A. Fox, "Removal Policies in network caches for world wide web documents," In Proc. of ACM SIGCOMM 96, pp.293-305, August, 1996.

[10] M. Abrams, C. Stanbridge, G. Abdulla, S. Williams, and E. Fox, "Caching Proxies : Limitations and Potential," In Proc. of 4th Int'l Conference on WWW, Boston, USA, Dec. 1995. http : //ei.cs.vt.edu/~succeed/WWW4/.

[11] C. Aggrawal, M. Epelman, J. Wolf, P. Yu, "On Cache Policies for Web Objects," IBM Research Report 20619, 1996. http : //www.research.ibm.com/.

[12] Igor Tatarinov, "Performance Analysis of Cache Policies for Web Servers," In proc of 9th Intl Conf on Computing and Information, ICCI'98 Winnipeg, June, 1998. Available from http : //www.cs.ndsu.nodak.edu/~tatarino/pubs/ICCI98-final.ps.

[13] C. R. Cunha, and C. F. B. Jaccoud, "Determining WWW user's next access and its application to pre-fetching," Proceedings of ISCC'97 : The 2nd IEEE symposium on Computers and Communications, July, 1997.

[14] A. Bestavros and C. Cunha, "Server-initiated document dissemination for the WWW," IEEE Data Engineering Bulletin, Sept. 1996.

[15] M. Arlit and C. Williamson, "Web Server Workload Characterization : The Search for Invariants," In Proceeding of the Fourth International Conference on the WWW, Boston, USA, December, 1995.

[16] L. Cherkasova, G. Ciardo, "Characterizing Temporal Locality and its Impact on Web Server Performance," HP Laboratories Report No.HPL-2000-82, July, 2000.

[17] Virgilio Almeida, Azer Bestavros, Mark Crovella, and Adriana de Oliveira. "Characterizing Reference Locality in the WWW," In Proceedings of 1996 International Conference on Parallel and Distributed Information Systems (PDIS'96), December, 1996.

[18] S. Jin and A. Bestavros, "Temporal locality in web request streams," In Proc. 2000 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, pp.110-111, June, 2000.

[19] V. Padmanabhan and L. Qiu, "The Content and Access Dynamics of a Busy Web Site : Finding and Implications," Procs. of SIGCOMM '00, pp.111-123, 2000.

[20] Hui Lei and Dan Duchamp, "An Analytical Approach to File Prefetching," In Proceeding of the USENIX 1997 Annual Technical Conference Anaheim, Califonia, pp.6-10, Jan. 1997.

[21] Venkata N. Padmanabhan and Jeffrey C. Mogul, "Improving HTTP latency," Computer Networks and ISDN Systems, 28(1/2), pp.25-35, December, 1995.

[22] L. Breaslau, P. Cao, L. Pan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions : Evidence and implications," In IEEE INFOCOM'99, pp.126-134, Mar. 1999.

안 효 범

e-mail : adapter@dragon.cntc.ac.kr

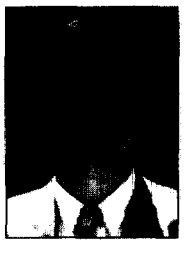
1992년 단국대학교 전자계산학과(학사)

1994년 단국대학교 전산통계학과(석사)

1997년 단국대학교 전산통계학과 대학교 박사과정

1997년~현재 천안공업대학 정보통신과 조교수

관심분야 : 클라이언트/서버 구조, 컴퓨터네트워크, 컴퓨터 구조



조 경 산

e-mail : kscho@dankook.ac.kr

1979년 서울대학교 전자공학과 졸업(학사)

1981년 한국과학원 전기및전자공학과 졸업 (공학석사)

1988년 텍사스 대학교(오스틴소재) 전기 전산공학과 졸업(Ph.D.)

1988년~1990년 삼성전자 컴퓨터부문 책임연구원

1990년~현재 단국대학교 전산통계학과 교수, 연구책임자

관심분야 : 컴퓨터 구조, 컴퓨터네트워크, 시뮬레이션