파일 타입을 이용한 웹 캐싱

임 재 현[†] · 이 준 연[††]

요 약

본 논문에서는 웹상의 높은 가변성을 고려하여 웹 캐시 공간을 관리하는 새로운 접근 방법을 제안한다. 모든 문서를 저장하는데 하나의 캐시를 사용하는것 대신에, 캐시를 분할하여 파일 타입에 따라 저장한다. 제안된 방법은 LFU, LRU와 SIZE 기반 알고리즘을 사용하여 현재의 캐시 관리 정책과 비교한다. 2가지의 서로 다른 실제 작업부하를 사용하며, 시뮬레이션을 통해 파일 타입별 캐싱이 적중률과 바이트적중률을 개선시킴을 보인다.

Web Caching using File Type

Jae Hyun Lim[†] · Jun Yeon Lee[††]

ABSTRACT

This paper proposes a new access method which is to considered the high variability in World Wide Web and manage the web cache space. Instead of using a single cache, we divide a cache and store all documents according to their file types. Proposed method was compares with current cache management policies using LFU, LRU and SIZE base algorithm. Using two different workload, we show the improvement hitting ratio and byte hitting ratio through simulating on the file type caching.

키워드 : 인터넷(Internet), 캐싱(caching), 교체(replacement), 적중률(hit ratio), 바이트적중률(byte hit ratio)

1. 서 론

인터넷에서 웹 사용의 증가는 수많은 요구를 발생하고 있으며, 이로 인해 웹이 직면하고 있는 규모의 문제는 다음과 같다. 첫째, 증가하는 사용자의 요구를 수용하기 위해 웹 서버의 규모를 증가시키는 것이다. 둘째, 인터넷 그 자체가 계속해서 용량을 증가시키고 새로운 망(network) 기술을 수용하는 것이다. 셋째, 클라이언트에 초점을 맞춰 웹 브라우저와 프락시 서버를 계층적으로 확장하는 것이다. 비용 면에서 인터넷의 규모를 확장하는 가장 좋은 정책은 클라이언트 가까이에 데이터를 캐시해서, 액세스 지연을 개선시키고 망과 서버의 부하를 줄이는 것이다.

지금까지 대부분의 많은 웹 서버가 텍스트와 이미지 객체를 주로 서비스하였다. 하지만, 웹은 텍스트 기반에서 멀티미디어 기반으로 급격히 발전하고 있다. 최근의 연구에서 연속 미디어인 오디오와 비디오 객체가 웹상에서 1998년 3배 가까이 증가하였다. 현재 웹 서버 상에 저장되어 있는 연속 미디어의 양이 상대적으로 적지만, 2003년에는 50%이상으로 증가할 것이라고 한다[6]. 뿐만 아니라 이들 데이터를 액세스하는 클라이언트는 손에 들고 다니는 PDA부터 워크스테이션에 이르기까지 다양해지며, 이들 각각은 성능과 서비스 요구가 다르다. 현재의 웹 캐싱은 텍스트와 이미지 데이터를 위해 설계되었기 때문에 연속 미디어를 캐시하거나, 사용자의 다양한 서비스 요구를 효과적으로 처리하지 못한다. 이와 같은 환경의 변화를 수용하기 위해서는 서버에 대한 사용자의 액세스 패턴을 정확히 분석할 필요가 있으며 새로운 캐시 전략이 필요하다.

웹 캐싱에는 여러 가지 접근 방법이 있다. 클라이언트 캐싱, 서버 캐싱, 프락시 캐싱, 계층적 캐싱과 협동적 서버 캐싱이 있다[1, 3]. 모든 캐싱의 공통적인 과제는 웹 캐시 공간을 효율적으로 사용하고, 높은 적중률을 얻기 위해 제한된 기억장소 공간을 관리하는 방법에 있다. 캐시 교체 정책은 새로운 문서를 저장하고 기존 문서를 제거하기 위해 사용하며, 교체 정책의 성능은 캐시 구조의 효과에 매우 중요하다. FIFO, LRU, LFU, SIZE 같은 교체 정책이 많은 연구[1]에서 분석되고 평가되었으며, 기본적으로 캐시의 성능은 적중률과 바이트적중률 2가지로 측정한다. 적중률은 사용자의 전체 요청 수에서, 캐시에 파일이 존재하여 적중된 수의 비율을 의미한다. 인터넷 캐시에서 적중률은 서버에 도착하

† 종신회원 : 공주대학교 영상정보공학부 교수
†† 정 회 원 : 동명정보대학교 멀티미디어공학과 교수
논문접수 : 2002년 4월 26일, 심사완료 : 2002년 8월 19일

는 사용자 요청의 감소 비율을 나타낸다. 인터넷 캐시는 고정된 크기의 페이지 단위로 캐시를 하는 것이 아니라 가변 길이의 파일 단위로 캐시를 하기 때문에 단순히 적중률만으로 평가하기가 어렵다. 이를 위해 바이트적중률을 이용한다. 바이트적중률은 사용자가 요청한 전체 바이트량에서, 캐시에서 적중된 바이트량의 비율을 나타낸다. 인터넷 캐시에서 바이트적중률은 캐시를 사용함으로써 얻어지는 망 통신량 감소를 나타내는데 사용한다. 기존 연구에 기술된 웹 캐싱 전략은 적중률 또는 바이트적중률 가운데 오직 한가지만을 최적화 하는데 그치고 있지만, 본 논문에서는 적중률과 바이트적중률을 모두 개선하고자 한다.

웹 캐싱은 현재 대부분의 교체 정책을 포함하여 전통적인 파일 캐싱 구조와 가상메모리 시스템과는 다르다[2]. 웹 캐시는 가변적 크기의 문서를 처리하며, 캐시 해야 할 문서의 개수가 매우 많고, 웹 자체의 변화가 매우 높다는데 있다. 예를 들어 웹 문서 크기는 수백 바이트에서 수십 메가 바이트에 이르며 문서의 참조 경향도 매우 다르다. 결과적으로, 크기가 크면서 사용 가능성이 적은 문서를 저장하기 위해, 교체 정책은 앞으로 자주 사용되는 수천 개의 작은 문서를 교체하는 상황이 있을 수 있다[5]. 이와 같은 상황이 웹 캐싱의 성능을 감소시킨다.

본 논문에서 웹상의 여러 문제점을 고려하여 웹 캐시 공간을 관리하는 새로운 접근 방법을 제안한다. 모든 문서를 저장하는데 하나의 캐시를 사용하는것 대신에, 캐시를 분할하여 파일 타입에 따라 저장한다. 문서를 수용하는 분할은 매우 동질적이며 전통적인 교체 정책에 의해 적절하게 관리한다. 제안된 방법은 LFU, LRU와 SIZE 기반 알고리즘을 사용하여 현재의 캐시 관리 정책과 비교한다. 시뮬레이션에서는 2가지의 서로 다른 실제 작업부하를 사용하며, 캐시 크기에 따른 파일 타입별 적중개수와 바이트량을 평가하여 그 효과를 입증하고, 파일 타입별 캐싱이 적중률과 바이트적중률을 모두 개선시킴을 보인다.

2. 액세스 패턴 분석

웹 작업부하의 분석은 여러 가지 특성을 파악할 수 있다. 사용자가 액세스 하는 파일의 크기와 타입 분포의 특징, 각 파일 타입에 대한 파일 크기 분포의 특징, 그리고 다른 그룹의 사용자도 같은 액세스 패턴을 가지는가를 분석할 수 있다. 또 다른 것은 특정 서버의 액세스에 관한 것으로 서버에 대한 사용자의 액세스 분포가 어떤 형태로 발생하는지를 알 수 있다.

본 논문에서는 웹 작업부하 분석을 통해 인터넷 캐시에서 캐시 성능을 좌우하는 요소를 파악하고 효율적인 캐싱 전략을 제안한다.

2.1 작업부하

웹의 통신량 특성을 파악하기 위해 본 연구에서는 2개의 작업부하를 분석하였다. 한 가지는 서울대(SNU)에서 운영했던 프락시 서버의 로그이고, 나머지 하나는 제이씨현의 엘림네트(Elimnet)에서 운영하고 있는 프락시 서버의 로그이다.

〈표 1〉 작업부하

파일 타입	SNU 요청횟수	SNU 전송바이트 (MB)	Elimnet 요청횟수	Elimnet 전송바이트 (MB)
image	302,798 (67.1%)	2,514 (45.1%)	123,970 (72.8%)	1,209 (70.7%)
text	130,619 (28.9%)	628 (11.3%)	42,599 (25.0%)	203 (11.9%)
video	512 (0.1%)	293 (5.2%)	89 (0.0%)	36 (2.0%)
audio	1,274 (0.3%)	490 (8.8%)	394 (0.2%)	15 (0.9%)
application	7,928 (1.8%)	1,621 (29.1%)	881 (0.5%)	235 (13.7%)
multipart	87 (0.0%)	3 (0.0%)	5 (0.0%)	4 (0.3%)
unknown	8,117 (1.8%)	28 (0.5%)	2,574 (1.5%)	9 (0.5%)
전 체	451,335 (100%)	5,577 (100%)	170,512 (100%)	1,711 (100%)

<표 1>은 작업부하별로 파일 타입에 따른 사용자 액세스 횟수와 전송 바이트 비율을 나타낸 것이다. 텍스트 타입에는 "html, htm, txt"와 같은 확장자를 가진 파일이 포함되고, 이미지 타입에는 "gif, jpeg, xbm" 등의 확장자를 가진 파일이 포함된다. 오디오 타입에는 "au wav, mp3" 등이 포함되고 비디오 타입에는 "mpeg, mpg, qt, mov, avi, movie" 등이 포함된다. 알려지지않은(unknown) 타입으로 분석된 파일은 대부분이 CGI에 의해 생성된 것이다.

2.2 파일 크기

사용자가 요청한 파일의 평균(mean) 크기는 10K~13K이지만, 파일의 중간(median) 크기는 3K정도이다. 또한 파일의 최대 크기는 수십 메가이며, 이는 평균치의 수백 배에 이른다. 파일 크기에 대한 액세스 분포는 평균치보다 작은 것에 집중되어 있고, 몇몇의 아주 큰 파일들이 존재한다. 캐시에 크기가 큰 파일을 저장하지 않고 사용자가 많이 사용하는 크기가 작은 파일을 저장한다면 캐시 적중률은 상당히 증가할 것이다. 그러나 크기가 큰 파일이 캐시에서 적중되면 바이트적중률이 많이 증가한다. 파일의 평균 크기, 중간 크기는 모든 작업부하에서 유사한 경향을 갖는다.

<표 1>에서 파일 타입별 요청 및 전송 바이트율를 살펴보면, 이미지와 텍스트가 요청 횟수와는 다르게 절대적인 비율을 차지하지 않는다. 왜냐하면 텍스트 파일의 크기는 다른 타입의 크기보다 작기 때문에 요청 횟수가 많더라도

전체 전송 바이트량은 크기가 큰 다른 타입의 파일에 비해 적게 나타난다. 이미지 타입이 전체 바이트 전송량의 45%~70%를 차지하고, 텍스트는 10% 내외를 차지한다. 또한 타입별 요청 횟수와는 달리, 오디오와 애플리케이션 타입을 가진 파일의 전송 바이트량이 상대적으로 높은 비율을 차지한다.

<표 2>에서 타입별 파일의 평균 크기를 보면, 타입에 따라 많은 차이가 있다. 이미지 파일의 평균 크기는 8K~9K 정도이고, 텍스트 파일은 5K 정도로 나타난다. 반면에 오디오 타입의 파일은 40K~400K, 비디오 타입의 파일은 400K~570K의 평균 크기를 보인다. 이처럼 파일 타입은 파일 크기와 밀접한 관계가 있다. 비디오나 오디오의 파일의 크기는 텍스트나 이미지 파일의 크기보다 상당히 크다. 캐시 크기가 제한되어 있을때, 비디오나 오디오 파일 대신에 텍스트나 이미지 타입의 파일을 저장한다면 보다 많은 파일을 저장할 수 있을 것이다. 또한 하나의 큰 비디오 파일은 수많은 텍스트 파일을 제거할 수도 있다. 만약 캐시에 크기가 큰 몇몇 파일만 존재한다면 적중률은 상당히 감소할 것이다. 그러나 크기가 큰 파일이 캐시 내에서 적중된다면, 바이트 적중률은 상당히 증가한다. 캐시는 적중률과 바이트적중률을 상호 보완하여 파일들을 저장할 필요가 있다.

〈표 2〉 타입별 파일의 평균 크기(단위 : 바이트)

파일 타입	SNU	Elimnet
image	8,301	9,574
text	4,807	5,409
video	572,773	402,359
audio	385,023	38,223
application	204,466	266,606
multipart	33,908	895,671
unknown	3,482	4,785

2.3 액세스 분포

<표 3>은 사용자 액세스의 분포도(distribution)를 보여준다. 첫 번째 항목인 액세스가 요청된 서버 비율은 3% 미만으로 나타난다. SNU의 경우, 전체 45만여 건의 액세스가 약 8,900개의 서버에 요청되기 때문에 2.0%의 비율을 갖는다. 이는 하나의 서버당 50개 정도의 액세스가 있었다는 것을 의미한다. 비율이 낮을수록 서버 집중도(concentration)가 높다는 것을 의미하는 것으로, 적은 개수의 서버에 모든 액세스가 집중하여 서버당 평균 액세스 횟수가 많다는 것을 나타낸다.

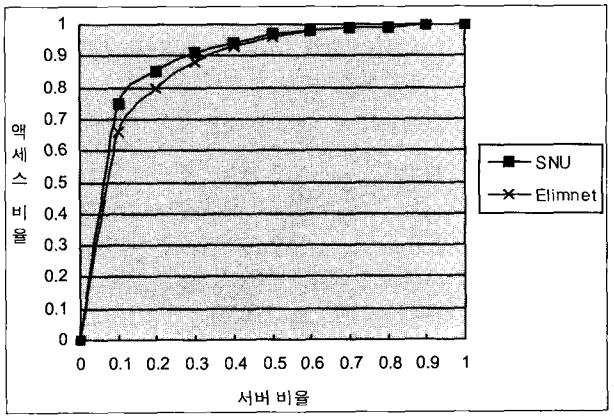
〈표 3〉 서버와 URL에 대한 액세스 분포도

작업량	액세스가 요청된 서버 비율	한번 액세스된 서버 비율	한번 액세스된 URL 비율
SNU	2.0%	0.4%	38.6%
Elimnet	2.7%	0.5%	47.3%

두 번째 항목인 한번 액세스된 서버의 비율은 액세스가 요청된 서버중 한번만 액세스가 발생하는 서버 비율을 나타낸 것으로 0.2%~0.5% 정도이다. 마지막 항목인 한번 액세스된 URL의 비율은 31%~48%정도로 실제 캐시할 필요가 없는 파일의 비율이다.

2.3.1 서버에 대한 액세스 분포

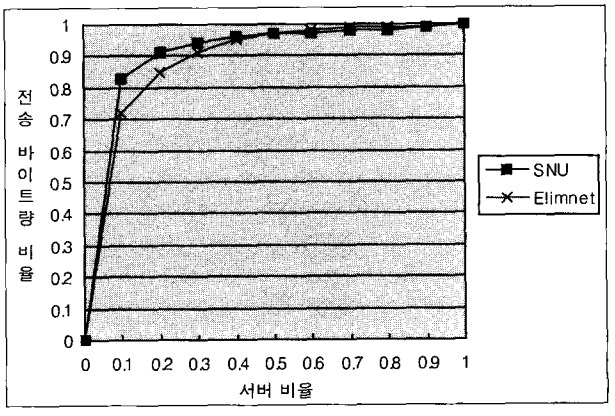
서버에 대한 액세스 집중도를 조사하면, 사용자의 액세스가 특정 서버에 집중됨을 알 수 있다. (그림 1)의 X축은 서버의 비율, Y축은 사용자 액세스 비율을 나타낸다. 사용자가 액세스한 모든 서버를 액세스 횟수를 기준으로 내림차순으로 정렬한 후, 서버 개수의 누적 비율에 따른 사용자 액세스 횟수의 누적 비율을 나타냈다.



(그림 1) 서버에 따른 액세스 집중도

전체 서버의 10%에 사용자 액세스의 70%가 집중되어 있음을 알 수 있다. 또한 전체 서버의 30%에 액세스의 90%이상이 집중되고 있다. 이는 인터넷 사용자들이 이용하는 서버가 전체 중에서 일부의 서버에 집중되고 있음을 의미한다. 인터넷 캐시에서는 이렇게 액세스가 집중되는 파일을 저장함으로써 성능을 증가시킬 수 있다.

(그림 2)는 서버 개수의 누적비율에 따른 전송 바이트량의 누적 비율을 나타낸다. (그림 1)에서 몇몇 서버에 액세스



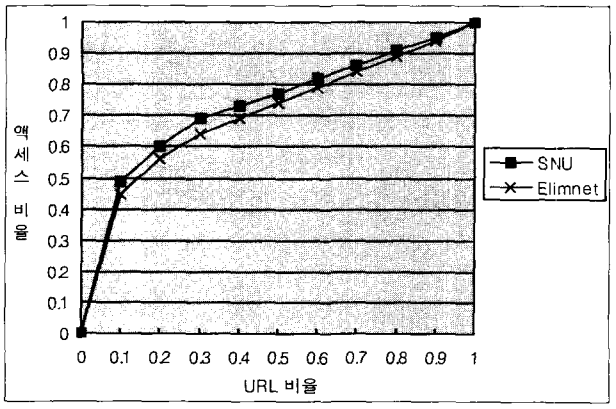
(그림 2) 서버에 따른 전송 바이트량 집중도

가 집중되는 것과 마찬가지로 전송되는 바이트량도 특정 서버들에 집중됨을 알 수 있다. 바이트량은 액세스보다 집중도가 더 높은 것으로 나타난다. 70%~85%의 전송 바이트량이 10%의 서버의 집중되었다.

(그림 1)과 (그림 2)를 종합해 보면, 사용자의 액세스는 모든 서버에 분산되어 있지 않고 일부 서버에 집중되는 것을 알 수 있다. 이렇게 집중적으로 참조되는 서버는 쉽게 과부하가 발생할 수 있기 때문에 캐싱과 부하 분산이 필요하다.

2.3.2 URL에 대한 액세스 분포

(그림 3)은 URL에 대한 액세스 집중도를 보여준다. 의 X축은 URL의 누적비율을, Y축은 사용자 액세스 비율을 나타낸다. 20%의 URL에 전체 액세스의 50%~60%가 집중된다. 즉 사용자들은 같은 URL을 중복해서 액세스하고 있고, 캐시는 이러한 중복 액세스된 URL을 캐시에 저장해야 한다. URL 누적 비율이 30% 이상 되면 그래프의 경사가 완만해지는데, 이는 30% 상위 부분에 속한 URL은 단 한번만 액세스된 것이기 때문에 그래프가 URL의 비율과 액세스 비율이 정비례해서 증가한다.



(그림 3) URL에 따른 액세스 집중도

3. 파일 타입별 캐싱

액세스 패턴 분석을 기반으로 새로운 캐싱 전략을 제안한다. 파일 타입에 따라 파일 크기는 매우 다르고, 웹 캐싱은 멀티미디어 파일을 연속으로 저장하기 때문에, 기존의 통합된 캐시 대신에 파일 타입별로 캐시를 구성한다.

3.1 캐시 성능

캐시 크기를 제한하지 않고 실험함으로써 실험 작업부하의 최고 적중률과 최고 바이트적중률을 구하였다. <표 4>를 보면, 실험 작업부하의 적중률은 39%~49%까지 나타나며 바이트적중률은 30%~46%이다. 적중률이 바이트적중률보다 높게 나타난 이유는 실제 캐시에서 적중된 파일의 평균 크기가 사용자가 액세스한 파일의 평균 크기보다 작기 때문이다. 즉 사용자는 평균 파일 크기보다 작은 파일을 더 많이 중복 액세스함을 나타낸다.

〈표 4〉 최고 캐시 적중률 및 바이트적중률

	적중횟수	적중률	총 적중 크기(MB)	바이트 적중률
SNU	220,815	48.9%	2,554.08	45.8%
Elimnet	66,343	38.9%	504.53	29.5%

이 결과는 다른 연구 결과[7]와 마찬가지로 사용자의 액세스 횟수가 높으면 적중률과 바이트 적중률이 증가함을 알 수 있다.

<표 5>는 SNU 서버에서 사용자에 의해 두 번 이상 액세스된 파일의 수와 크기를 파일 타입별로 분석한 것이다. 파일이 사용자에 의해 두 번 이상 액세스된다는 것은 첫 번째 액세스 때 파일이 캐시에 저장되고, 다음 번 액세스 때에는 캐시에서 적중됨을 의미한다. 캐시는 이러한 파일을 저장해야만 그 효과가 나타난다[5]. 본 논문에서는 캐시에 저장할 필요성이 있는 두 번 이상 참조된 파일의 총 크기를 구하여 캐시의 전체 크기로 결정하였으며, 앞으로 이 크기를 "유효 캐시 크기"라는 용어로 사용한다. SNU의 경우 유효 캐시 크기는 640MB이며, Elimnet의 경우 유효 캐시 크기는 200MB이다.

〈표 5〉 타입별 유효한 파일의 수와 크기

파일 타입	유효한 파일 수	비 율	유효한 파일 크기(MB)	비 율
image	39,355	69.8%	340.21	53.0%
text	14,607	26.0%	64.15	10.0%
video	88	0.2%	41.01	6.4%
audio	217	0.4%	67.70	10.5%
application	1,082	1.9%	125.92	19.6%
multipart	17	0.0%	0.78	0.1%
unknown	974	1.7%	2.41	0.4%
전 체	56,340	100%	642.22	100%

3.2 캐싱 전략

제2장의 액세스 패턴 분석을 보면, 인터넷에서는 다양한 타입의 파일들이 존재하고 타입마다 갖고 있는 특성이 다르다. 그러나 지금까지 인터넷 캐시의 전략은 파일을 타입에 따라 구분하지 않고 모든 파일을 동일하게 취급함으로써 비효율성 문제를 발생시켰다. 예를 들어 캐시 안에서 크기가 큰 비디오 파일이 크기가 작은 수십 또는 수백 개의 텍스트 파일을 교체한다면, 캐시 적중률을 감소한다. 반대로 크기가 작은 텍스트 파일이 크기가 큰 비디오 파일을 교체한다면, 상대적으로 바이트적중률은 감소한다. 본 논문에서는 이 같은 문제점을 해결하기 위해 파일 타입에 따라 분할된 캐싱 전략을 제안하여, 캐시 적중률과 바이트적중률

모두를 향상 시킨다.

〈표 6〉 타입별 캐시의 크기 비율

캐시 타입	SNU	Elimnet
image cache	58.8%	73.1%
text cache	20.1%	19.1%
application cache	13.1%	7.1%
etc cache	8%	1.7%

파일 타입별 캐싱 전략을 사용할 때, 파일 타입에 따른 캐시 크기를 결정하는 것이 중요한 문제이다. 이를 위해 사용자의 액세스 패턴을 분석해 본 논문에서 적용하려는 캐싱 전략은 다음과 같다.

- 인터넷에는 다양한 타입의 파일이 존재하고, 각 타입은 캐시의 성능에 영향을 미치고 있다. 크기가 작은 텍스트, 이미지 파일 타입은 캐시의 적중률에 유리하게 작용하고, 상대적으로 크기가 큰 오디오, 비디오 파일 타입은 캐시의 바이트적중률에 유리하게 작용한다. 지금까지의 인터넷 캐시 전략은 모든 파일을 동일하게 취급하였지만, 본 논문에서는 파일 타입별로 캐싱을 수행한다.
- <표 1>과 <표 4>에서 알 수 있듯이 액세스 횟수가 많으면 적중률이 상대적으로 높다. 따라서 파일 타입별로 크기를 결정할 때 타입별 요청 횟수와 전송 바이트량을 중요한 팩터(factor)로 이용한다.
- <표 5>에서 보듯이 적중된다는 것은, 두 번 이상 액세스된 파일을 캐시에 저장해야만 그 효과가 있기 때문에 파일 타입별로 유효한 파일의 수와 크기 비율을 중요한 팩터로 이용한다.

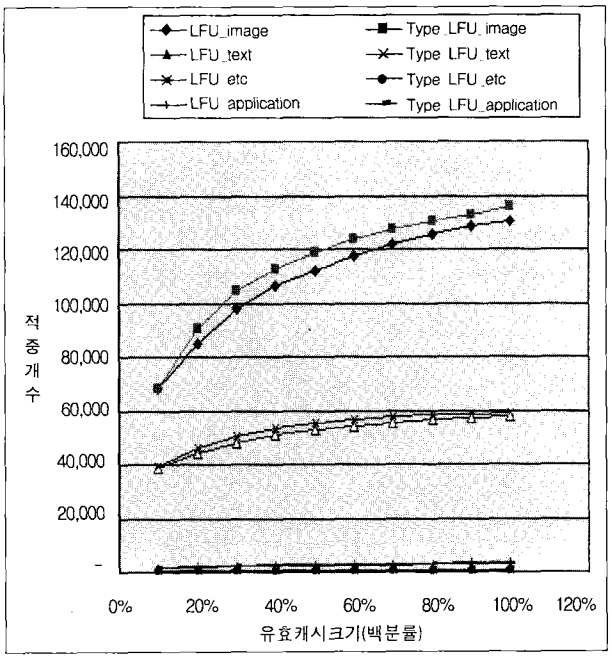
파일 타입별로 캐시를 분리할때 캐시의 크기는 4가지 팩터의 비율을 산술 평균하여 사용하였다. <표 6>은 타입별 캐시의 크기 비율이다. 캐시를 총 4개의 부분으로 구분하였는데, 이미지 캐시, 텍스트 캐시, 응용(application) 캐시 그리고 나머지(etc) 캐시로 분리하였다. 전체 캐시 크기는 유효 캐시 크기의 10%에서 100%까지 성능을 측정하였다.

4. 성능 평가

본 논문의 성능평가는 SUN Enterprise 3500을 이용하였으며, 액세스 패턴 분석 및 시뮬레이션 프로그램은 C언어로 구현하였다. 각 작업부하별로 LFU, LRU, SIZE 캐시 교체 정책을 기존 방법대로 통합 캐시에 적용했을 때의 성능과 ,타입별로 구분한 캐시에 적용한 결과를 평가한다. 적중개수와 적중바이트량을 기준으로 비교하였으며, 통합 캐시의 전체 적중개수와 적중바이트량도 파일 타입별로 분석하여 비교한다. 최종적인 캐시 성능 평가는 적중률(hit ratio)과 바이트적중률(byte hit ratio)로 평가한다[4, 8].

4.1 LFU 평가

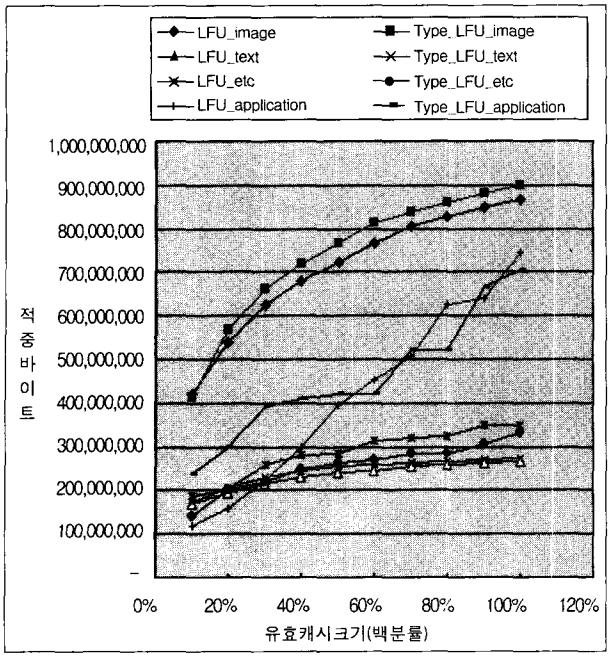
LFU는 빈도수 위주의 교체 정책을 수행하므로, 자주 참조되는 파일이 캐시에 존재할 확률이 높다. (그림 4)는 SNU에서 파일 타입에 따른 적중개수를 분석한 것이다. X축은 유효캐시크기 640MB의 10%에서 100%까지의 설정했으며, Y축은 실제 적중개수를 나타낸다. 실제 유효 캐시크기의 30%에서 80% 구간에 적중 개수 차이가 많이 생기며, 30%이하와 80%이상에서는 큰 차이가 없다. image와 text를 제외한 나머지 파일 타입은 적중개수가 적기 때문에 전체 적중률에 큰 영향을 미치지 않는다. image와 text를 대상으로 비교해 보면 새로 제안한 파일 타입별 캐시의 적중개수가 많이 늘고 있다. 유효캐시 크기 50%일때 image 타입의 적중개수는 6,000개 이상 차이가 나며, text 타입에서는 3,000개의 차이가 난다. 이것은 파일 타입별로 캐시를 분할하였기에 상대적으로 크기가 작은 image, text타입이 크기가 큰 데이터 파일에 의해 교체되는 현상이 적기 때문이다.



(그림 4) 파일타입별 적중개수(SNU)

(그림 5)는 SNU에서 파일 타입에 따른 적중바이트량을 분석한 것이다. (그림 4)의 파일 타입별 적중개수와는 달리 각 파일 타입별 적중바이트량이 전체 바이트적중률에 큰 영향을 미친다. application, etc 파일 타입의 적중바이트량이 text 파일 타입보다 많으며, 통합 캐시의 적중바이트량이 파일 타입별 캐시의 적중바이트량보다 우수한 구간도 있다. 하지만 전체 바이트적중률에 가장 큰 영향을 미치는 image 타입에서는 파일 타입별 캐시가 우수하다. 유효캐시 크기 50%면 실제 캐시 크기는 320MB인데, image 타입의 적중

바이트량에서 45MB이상 차이가 난다. 이 크기는 타입별 캐시 40%일때와 통합 캐시 50%일때의 적중바이트량이 유사한 결과이다.

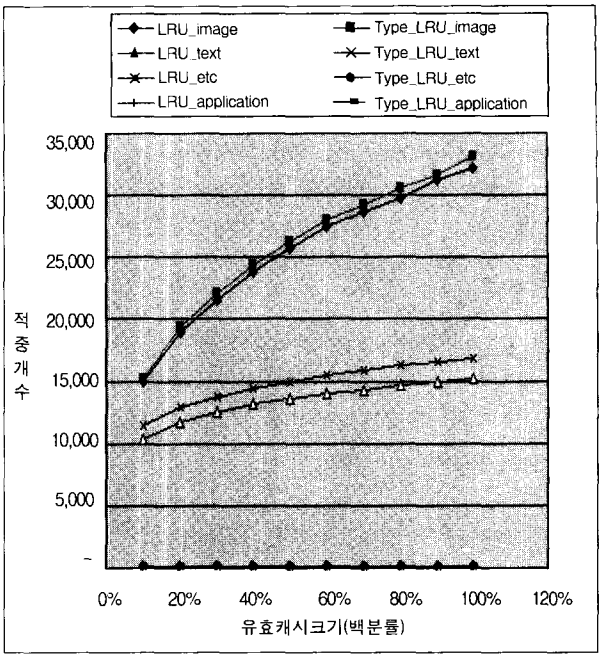


(그림 5) 파일타입별 적중바이트(SNU)

(그림 4)와 (그림 5)를 종합해보면 파일 타입 캐시가 크기는 작지만 요청 횟수의 대부분을 차지하는 image, text 파일 타입이, 크기가 큰 파일 타입에 의해 교체되는 현상이 적게 발생하기 때문에 적중률과 바이트적중률이 모두 안정적으로 유지된다. 이와 같은 결과는 Elimnet에서도 유사하게 나타난다.

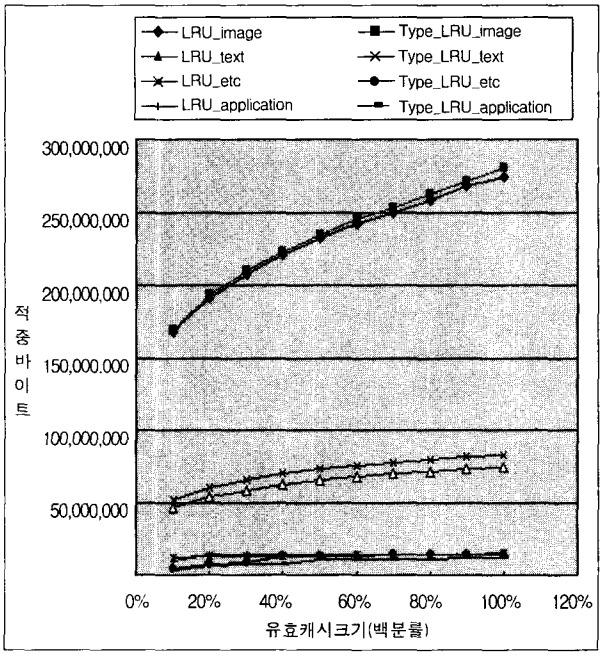
4.2 LRU 평가

LRU는 참조된 시간 위주의 교체 정책을 수행하므로, 최근에 참조된 파일이 캐시에 존재할 확률이 높다. (그림 6)은 Elimnet에서 파일 타입에 따른 적중개수를 분석한 것이다. X축은 유효캐시크기 200MB의 10%에서 100%까지의 설정했으며, Y축은 실제 적중개수를 나타낸다. SNU와 마찬가지로 실제 유효캐시크기의 30%에서 80% 구간에 적중개수 차이가 많이 생기며, 30%이하와 80%이상에서는 큰 차이가 없다. image와 text가 대부분을 차지하며, 다른 파일 타입의 참조는 극히 미비하다. 유효캐시 크기 50%일 때 image와 text 타입에서 1,000개씩의 차이가 난다. 전체 요청 횟수가 SNU에 비해 작지만, 그래프의 모습은 LFU 결과와 매우 흡사하다. 역시 image와 text를 대상으로 비교해 보면 새로 제안한 파일 타입별 캐시의 적중개수가 많이 늘고 있다. 이것은 파일 타입별로 캐시를 분할하였기에 상대적으로 크기가 작은 image, text가 크기가 큰 데이터 파일에 의해 교체되는 현상이 적기 때문이다.



(그림 6) 파일타입별 적중개수(Elimnet)

(그림 7)은 Elimnet에서 파일 타입에 따른 적중바이트량을 분석한 것이다. 파일 타입별 적중개수와는 달리 각 파일 타입별 적중바이트량이 전체 바이트적중률에 큰 영향을 미친다. 하지만 SUN와는 다르게 text 파일 타입의 적중바이트량이 많으며, 통합 캐시의 적중바이트량보다 파일 타입별 캐시의 적중바이트량이 약간씩 우수하다. 특히 전체 바이트 적중률에 가장 큰 영향을 미치는 image, text 타입에서는 타입별 캐시가 우수하다. 유효캐시 크기전 구간에서 text 타입의 적중바이트량은 8MB이상 차이가 난다. 이 크기는 타입



(그림 7) 파일타입별 적중바이트(Elimnet)

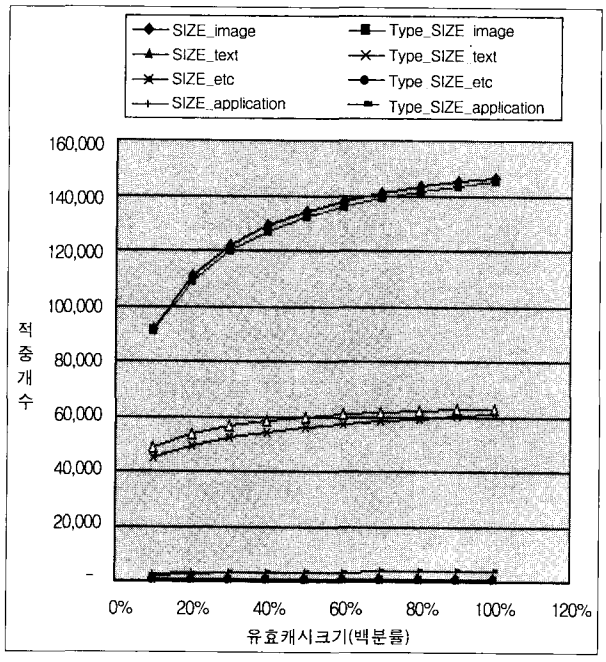
별 캐시가 통합 캐시보다 10% 이상 캐시 용량을 절약하면서, 우수한 바이트적중률을 유지한다는 결과이다.

(그림 6)과 (그림 7)를 종합해 보면 LFU와 마찬가지로, 파일 타입 캐시가 통합 캐시보다 적중개수와 적중바이트량에서 우수한 결과를 보인다. 크기는 작지만 요청 횟수의 대부분을 차지하는 image, text 파일 타입이, 크기가 큰 파일 타입에 의해 교체되는 현상이 적게 발생하기 때문에 적중률과 바이트적중률이 모두 안정적으로 유지된다. 수치적으로는 파일 타입별 캐시가 10%이상의 캐시 용량 절감 효과가 나타나며, 이와 같은 결과는 SNU에서 더욱 크게 나타난다.

4.3. SIZE 평가

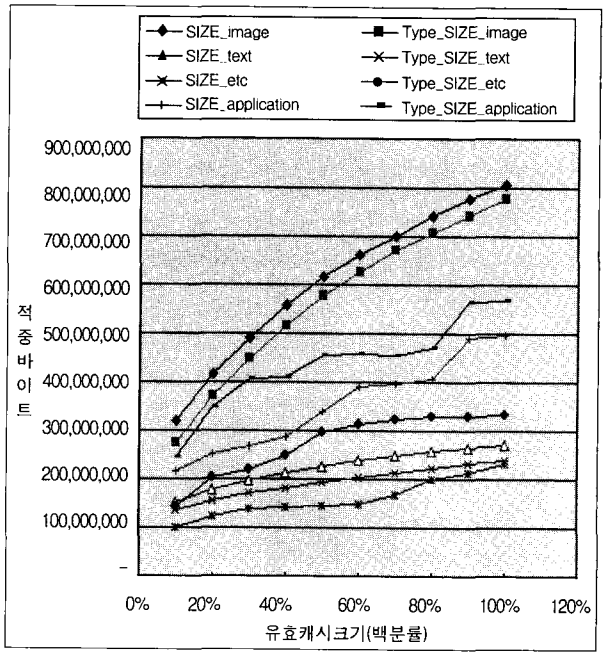
SIZE는 크기 위주의 교체 정책을 수행하므로, 크기가 작은 파일이 캐시에 존재할 확률이 높다.

(그림 8)은 SNU에서 파일 타입에 따른 적중개수를 분석한 것이다. LFU, LRU와는 다르게 실제 유효캐시 크기의 전 구간에서 큰 차이가 없다. LFU, LRU와 마찬가지로 image와 text를 제외한 나머지 파일 타입은 적중개수가 적기 때문에 전체 적중률에 큰 영향을 미치지 않는다. image와 text를 대상으로 비교해 보면 통합 캐시의 적중개수가 약간씩 우수하다. 이것은 SIZE 알고리즘 자체가 크기가 큰 파일을 교체 대상으로 하기 때문에 크기가 작은 image, text 파일에 상대적으로 유리하다. 하지만 본 논문에서 제안한 파일 타입별 캐시도 그와 유사한 적중개수 분포를 보인다.



(그림 8) 파일타입별 적중개수(SNU)

(그림 9)는 SNU에서 파일 타입에 따른 적중바이트량을 분석한 것이다. 역시 각각의 파일 타입별 적중바이트량이 전체 바이트적중률에 큰 영향을 미친다. LFU, LRU와는 다르게 application, etc의 적중바이트량이 text 파일 타입보다 많으며, 파일 타입별 캐시의 적중바이트량이 모든 구간에서 통합 캐시보다 우수하다. 이것은 크기가 작은 image와 text 타입에서는 통합 캐시가 약간 우수하지만, 상대적으로 크기가 큰 etc와 application 타입에서는 파일 타입별 캐시가 매우 우수하기 때문이다.



(그림 9) 파일타입별 적중바이트(SNU)

예를 들어 유효캐시 크기 50%에서 application의 적중바이트량은 150MB이상 차이가 나며, etc에서는 70MB이상 차이가 난다. 이와 같은 결과는 SIZE 알고리즘의 단점인 작은 파일 위주의 보호 정책이, 오히려 파일 타입별 캐싱에서는 작은 파일에 의해 크기가 큰 파일이 교체되지 않기 때문이다.

(그림 8)과 (그림 9)를 종합하면, LFU, LRU와는 다르게 SIZE 알고리즘에서는 파일 타입 캐싱이, 작은 파일에 의해 큰 파일이 교체되는 현상이 적어지기 때문에 적중개수와 적중바이트량에서 우수한 결과를 보인다. 이와 같은 결과는 Elimnet에서도 유사하게 나타난다.

4.4 종합 평가

SNU 적중률은 LFU, LRU 알고리즘에서 파일 타입별 캐싱이 2%이상 적중률이 증가하였고, 캐시 크기로 보면 최소 10%에서 최대 50%가까이 절점 효과를 보인다. 다만 SIZE 알고리즘에서는 큰 변화가 생기지 않는다. 이것은 SIZE 알고리즘이 통상적으로 적중률에 있어서 매우 높은 결과를 갖기 때문이다. 바이트적중률은 LFU, SIZE 알고리즘에서 파일 타입별 캐싱이 2%이상 적중률이 증가하였고, 캐시 크기로 보면 LFU에서는 10%, SIZE 에서는 20%이상 절감 효

과를 보인다. 다만 LRU 알고리즘에서는 큰 변화가 생기지 않았다.

Elimnet의 적중률은 LFU, LRU 알고리즘에서 파일 타입별 캐싱이 2%이상 적중률이 증가하였고, 캐시 크기로 보면 최소 10%에서 최대 30%가까이 절점 효과를 보인다. SNU와는 다르게 SIZE 알고리즘에서도 1%이상 적중률이 증가하였고, 캐시 크기는 유사하게 유지된다. 바이트적중률은 LFU, SIZE, SIZE 알고리즘에서 파일 타입별 캐싱이 1%이상 적중률이 증가하였고, 캐시 크기로 보면 최소 10% 에서 최대 30% 가까이 절감 효과를 보인다.

5. 결 론

기존 연구에서 적중률과 바이트적중률은 서로가 양립하기 어려운 결과를 보여 왔지만, 본 논문에서 제안한 파일 타입별 캐싱은 적중률과 바이트적중률 모두에서 향상된 결과를 보인다. 예를 들어 통합 캐시의 SIZE 알고리즘 적중률은 최고의 결과를 갖지만, 바이트적중률에 있어서 최악의 결과를 갖는다. 하지만 파일 타입별 캐시를 사용하면 적중률과 바이트적중률 모두에서 향상된 결과를 갖는다. 결론적으로 하나의 통합된 캐시를 사용하는 것보다 파일 타입별로 캐시를 분할하여 캐싱 하는것이 성능을 향상시키는 것으로 나타났다.

적중률 면에서 LFU, LRU 교체 정책을 사용할때 타입별 캐시가 2% 가까이 성능 증가를 보였다. 그러나 SIZE 교체 정책을 적용하면 약간 감소하거나 유사한 성능을 보였다. 파일 타입별 캐시는 바이트적중률 면에서도 우수한 성능 향상을 보이는데, 유효캐시 크기에 따라 1%정도 성능이 증가하였다. 바이트 적중률은 모든 캐시 교체 정책에서 성능 향상을 보였으며, SIZE 교체 정책의 경우 가장 높은 성능 증가가 나타났다. 작업 부하에 따라 성능 향상 비율에 차이가 나는데, 비디오나 오디오와 같이 크기가 큰 파일의 비율이 낮은 Elimnet의 경우 낮은 성능 증가를 보였고, 크기가 큰 파일이 높은 비율을 차지하는 SNU의 경우 상대적으로 높은 성능 증가를 나타냈다. 또한 캐시 크기면에서 살펴보면 통합 캐시 보다는 파일 타입별 캐시가 최소 10%에서 최대 50% 가까이 기억공간을 절약하면서, 유사한 성능을 보이고 있다. 파일 타입별 캐시는 비디오나 오디오처럼 크기가 큰 파일의 액세스가 많은 경우 더욱 우수한 성능 향상을 나타낼 것으로 판단된다.

참 고 문 헌

[1] Jia Wang., "A survey of Web caching schemes for the Internet," ACM Computer Communication Review, 29(5), pp.36-46, October, 1999.

[2] Edith Cohen and Haim Kaplan, "Exploiting Regularities in Web Traffic Patterns for Cache Replacement," In Proceedings of the 31st Annual ACM Symposium on Theory of Computing, ACM, 1999.

[3] Greg Barish and Katia Obraczka, "World Wide Web Caching : Trends and Techniques," IEEE Communications Magazine Internet Technology Series, May, 2000.

[4] Ghaleb Abdulla, "Analysis and Modeling of World Wide Web Traffic," PhD thesis, Computer Science Department, Virginia Tech., 1998.

[5] Cristina Duarte Murta, Virgilio A. F. Almeida, and Jr. Wagner Meira, "Analyzing performance of partitioned caches for the WWW," In Proceedings of the Third International WWW Caching Workshop, Manchester, England, June, 1998.

[6] Sambit Sahu, Prashant Shenoy, Don Towsley, "Design Considerations for Integrated Proxy Servers," Proc. of IEEE NOSSDAV'99, pp.247-250, June, 1999.

[7] B. M. Duska, D. Marwood and M. J. Feeley, "The Measured Access Characteristics of World-Wide-Web Client Proxy Caches," Proc. of the USENIX Symposium on Internet Tech. and System, DEC., 1997.

[8] M. Abrams, S. Williams, G. Abdulla, S. Patel, R. Riber, and E. A. Fox, "Multimedia Traffic Analysis Using CHITRA95," ACM Multimedia 95-Electronic Proceedings, Nov., 1996.

임 재 현

e-mail : defacto@kongju.ac.kr

1986년 중앙대학교 전자계산학과 졸업 (학사)

1988년 중앙대학교 일반대학원 전자계산학과(석사)

1998년 중앙대학교 일반대학원 컴퓨터공학과(공학박사)

1998년~현재 공주대학교 영상정보공학부 조교수

관심분야 : 클라이언트/서버 시스템, 정보검색, 분산운영체제

이 준 연

e-mail : jylee@tmic.tit.ac.kr

1990년 중앙대학교 전자계산학과 졸업 (학사)

1992년 중앙대학교 일반대학원 전자계산학과(석사)

1992년~1994년 (주)Microsoft

2000년 중앙대학교 일반대학원 컴퓨터공학과(박사)

2000년~현재 동명정보대학교 멀티미디어공학과 조교수

관심분야 : 멀티미디어, 운영체제, 분산시스템