확장된 웹 캐시 서버에서 자원이용률 최적화 기법

김원기°, 김두상, 김성락, 구용완
수원대학교 컴퓨터학과
E-mail : ksr01@osan.ac.kr

An Optimal Technic to Utilize Resource on Extended Web Cache Server

Won-Ki Kim°, Doo-Sang Kim, Seong-Rak Kim, Yong-Wan Koo
Dept. of Computer, The University of Suwon

요 약

대규모 웹 캐시 서버의 자원 이용도는 네트워크와 디스크 I/O 대기 시간에 주로 의존하고 또한 작업 부하 패턴에 있어 네트웍 사용이 폭주하는 시간과 새벽과 같은 한가한 시간간의 변동성이 심하다. 따라서, 한정된 자원범위에서 최상의 서비스를 제공키위해서는 절정기 동안 자원 이용도를 낮추고 이들 작업부하를 비절정기때에 나누어 수행토록 함으로써 자원 활용도를 최대로 끌어올리자는데, 연구의 목적이 있다. 이를 위해 비절정기 동안 캐시 압축 기법을 이용하여 디스크 입출력 작업을 미래예측 기법은 어느 점에서의 실제 작업 세트가 작았다는 것과 페이지 재사용 패턴의 정확한 예측은 물리적 메모리 크기의 캐시에서 높은 히트율을 생산할 것이라는 점을 보여주었다.

1. 서론

웹 캐시란 클라이언트와 서버 사이에 위치해서 대리 서버의 역할을 하는 캐시이다. 본 웹 캐시는 사용자들의 대역폭을 절감해주고, 서버의 부하를 줄여주고, 클라이언트들에게는 더욱 빠른 응답 시간을 제공해주는 역할을 한다. 대규모 웹 캐시 서버의 자원 이용도는 네트워크와 디스크 I/O 대기 시간에 주로 의존한다. 그리고 흔히 예측할 수 있는 접속이 많은 시간(절정기)과 한가한 시간(비절정기)의 주간 작업 부하 패턴 때문에 이들 자원 이용도는 변동이 심하다.

본 연구에서는 절정기 동안 자원 이용도를 비절정기때로 나누어 처리함으로써 전체 웹 캐시 서버의 자원과 네트워크 자원을 효율적으로 사용하자는데 목적이 있다. 이를 위해 비절정기 동안 캐시 압축 기법을 이용하여 디스크 입출력 작업을 분배하는 연구를 수행하고자 한다. 이는, 비절정기 동안 캐쉬 압축을 이용해서, 그리고 캐쉬 구조가 파일시스템 버퍼 캐쉬와 가상 메모리 시스템과 같은 운영체제 서비스를 이용하는 방법을 조심스럽게 설계함으로써, 디스크 I/O를 감소시킨다.

2. 관련연구

2.1 웹 캐시 기법

2.1.1 요구 위주 캐싱(Demand-driven Caching)

요구위주 캐싱은 클라이언트가 요청하는 객체만을 서버에서 가져와 캐시를 하는 수동적인 캐싱 기법을 의미한다. 많은 연구자들이 요구 위주 웹 캐싱의 최대 가능 히트율을 30-50%로 보고 있다. 최근의 연구에 의하면 Hit율이 웹 캐시 서버의 클라이언트 크기와 웹 캐시 서버에 의해 보여지는 요청의 수에 의존한다는 사실을 증명하였다. 작은 크기의 웹 객체에 대한 캐쉬에 있어서 캐쉬 교체법은 히트율을 결정하는 데 있어서의 중요한 요소이다. 가장 일반적으로 구현된 캐시 교체법은 타임 스템프 값을 이용한 LRU 기법이다.

2.1.2 선반입(Prefetching)

객체 선반입이란 클라이언트가 요청할 객체를 미리 캐시 서버에 가져온다는 의미를 갖는다. 이에 대한 근거는 클라이언트가 어떤 객체를 요청할지 미리 예측하여 능동적으로 캐싱을 해둔다면 그 만큼 클라이언트들에게 해당 객체들을 빠르게 재공해줄 수 있다는 사실에 따른다. 선반입(Prefetching)은 상충되는 목표를 달성하는 데 이용될 수 있다: 첫번째는 히트율을 높여서 웹 네트웍 대기 시간을 줄이는 것이고, 두 번째 목표는 보다 많은 대역폭이 비절정기에 소비되고 적은 대역폭이 절정기에 사용되는 대역폭 조절이다. 두 가지 상충되는 선반입 메카니즘은 객체가 실제 요구될 때 캐시로부터 전송할 수 있게 하기 위해 웹 객체에 대한 미래의 참조를 예측해야 한다.

2.2 캐시 일치성(Cache Coherency)

웹에서의 캐시에서 일치성 문제는 캐시에 미리 저장해놓은 문서를 사용자들에게 제공하면서 어떻게 하면 가능한 최신 문서를 전송해 줄 수 있는가 하는 것이다. 일관성 문제는 영원히 해결하기 어렵겠지만, 아래와 같은 일관성을 최대한 유지하려는 노력은 여전히 다양하게 이루어지고 있다. 1) 클라이언트 폴링(Client Polling)은 캐시된 객체들에 각각 타임 스템프를 부여하고, 주기적으로 이 객체들을 원래의 객체와 비교하는 방법이다. 2) Invalidation Callbacks은 웹 서버가 웹 캐시에서 캐싱하는 객체들에 대한 기록들을 유지하고 있다가, 그 기록들에 해당하는 객체들이 바뀌면 이를 웹 캐시에게 알려주는(Callback) 방식이다. 3) TTL(Time-To-Live)은 네트워크 상에서 돌아다니는 패킷들의 수명을 나타내는 TTL처럼, 캐시 객체들에 대해서도 TTL을 적용시킬 수 있다[2][3].

3. 웹 캐시 서버 DISK 입출력 개선

빠른 프로세서와 값싼 메인 메모리를 사용할 수 있는 요즘의 대규모 웹 캐시 서버에 있어서 주요한 병목 구간은 디스크 I/O이다

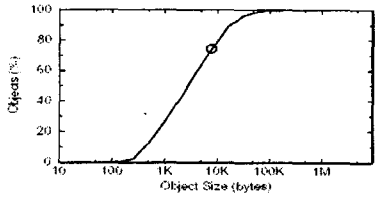
본 논문에서는 두 가지 전략이 디스크 I/O를 크게 감소시킬 수 있다고 제안하였다..

1) 캐시 운용시 HTTP 참조의 국부성 특성을 이용하여 객체를 참조하는 기법

2) 작은 객체에 대해서 파일 시스템보다 메모리에서 관리.

3.1 작업부하 측면에서의 파일시스템

웹 캐시 서버의 기본 기능은 클라이언트로부터 요청 받기, 그 요청이 권한이 있는지 검사하기, 그리고 로컬 디스크 혹은 인터넷에서 요청된 객체를 전송하는 것이다. 일반적으로 인터넷에서 전송되는 객체는 로컬 디스크에도 저장되므로 동일한 객체에 대한 이후의 요청들은 로컬에서 보내질 것이다. 웹 통신 특성과 결합한 이 기능은 파일 시스템 부하에 의해 발생하는 웹 캐시 서버 캐시의 다음과 같은 측면을 암시한다[4].



(그림 1) 객체 크기 대 요청 비율

(그림 1)의 그래프는 객체의 누적 분포를 보여준다. 예를 들면, 참조된 객체 중 74%는 크기가 8k바이트 이하를 의미한다.

3.2 작업 부하의 특징

- **원콜 오퍼란드(Entire Files Only)**
- **인기도(Popularity)**
- **데이터 국부성(Data Locality)**
- **메타 데이터 국부성(Meta-data Locality)**
- **중복성(Redundancy)**

3.3 실험환경

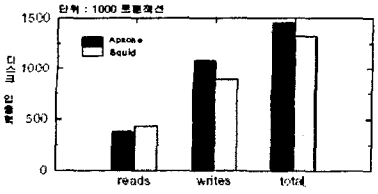
이들 캐시 구조를 실험하기 위해 파일 시스템이나 가상 메모리에 접근하는 웹 캐시의 일부를 시뮬레이트하는 웹 폴리그래프(Web Polygraph)를 이용하였다. 웹 폴리그래프는 캐시의 성능을 평가하는데 공식화된 시뮬레이션 툴이다.[5]

이 모든 실험은 메인 메모리가 512M Byte인 팬티엄 프로세서를 장착한 캐시 서버에서 행하였다. 캐시된 객체를 저장하기 위해 두 개의 4GB 디스크와 하나의 2GB디스크를 사용했다.

본 실험에서는 학교에 설치된 캐시서버에서 로그를 추출하였다. 하나의 웹 캐시 서버에서 2주 동안 선정했고, HTTP 요청이 아닌 것, 응답 코드가 200(ok)가 아닌 것, ?나 또는 cgi-bin을 포함한 요청은 모두 제외시켰다. 그 결과 로그는 첫째 주에는 522,376개의 요청, 둘째 주에는 495,664개의 요청으로 구성되었다. 무한 캐시를 가정하면 그 로그는 히트율 59%에 이른다. 각 실험은 두 가지 단계를 가진다. 첫번째 워밍업(준비) 단계에는 파일 시스템이 새로 초기화되고 웹 폴리그래프가 첫 날의 요청을 실행하도록 디스크가 새로 포맷된다. 두번째 측정 단계는 메인 메모리와 첫번째 단계의 결과로 나온 디스크 상태를 사용하여 다음 날의 요청을 처리하는 과정으로 구성된다. UFS를 사용한 시뮬레이션의 디스크 I/O의 경우는 iostat를 사용했다. iostat rz3 rz5 rz6 1 라는 명령어를 이용하면 바이트 수를 리스트해 주고 매초마다 세 개의 디스크로 한번에 이동한다. 그러나 iostat는 읽기와 쓰기의 수는 구별하지 못한다.

3.4 결과

디스크 I/O는 Apache와 squid 웹 폴리그래프(Web Polygraph)를 사용했을 때 비슷하다. 이 같은 사실은 Apache와 Squid가 디스크 하부시스템을 비슷하게 사용하고 있음을 의미한다. 결과 데이터는 캐시 히트율이 59%임에도 전체 디스크 I/O의 1/3만이 읽기라는 것을 보여준다.



(그림 2) Apache와 Squid 웹 폴리그래프에서의 디스크 입출력

이러한 실험을 통하여 Squid 구조를 조절하여 디스크 I/O를 크게 감소시킬 수 있음을 보았다. 웹 작업부하는 파일 시스템 작업부하와 매우 유사한 참조 특성을 보여주었다. 고성능 어플리케이션을 사용할 때는 파일 시스템 접근 패턴을 적합하게 하는 것이 중요하다. 웹 객체를 파일 시스템에 매핑할 때 첫 번째 레벨의 디렉토리 참조 계층과 장소를 유지하는 것은 디스크 I/O 수를 50%까지 감소시킨다.

웹 객체의 크기와 재사용 방식 또한 비슷하다. 대부분의 인기 있는 페이지들은 크기가 작다. 작은 객체들을 메모리 맵 파일에 캐싱하는 것은 히트의 대부분이 디스크 I/O없이 이루어질 수 있다. 국부성을 보증하는 파일 경로와 메모리 맵 파일을 조합하여 사용하면 본 시뮬레이션에서 70% 이상의 디스크 I/O축약을 이루었다.

4. 교체 전략

4.1 미래 예측 교체 전략

본 전략은 모든 미래 참조를 포함한 전체 추적으로부터 추론되어 미리 계산된 배치 테이블을 사용하는 구조이다. 그 전략은 최적에 가까운 할당 정책을 수립하는 데 있다. 그 배치 테이블은 참조가 미스인지 히트인지 결정하고, 객체가 캐싱되어야 하는지, 그리고 캐쉬에 위치해야 되는지 여부를 결정하기 위해 사용된다. 본 연구에서는 그 배치 테이블을 만들기 위해 다음의 경험적 학습법을 사용한다.

(1) 작업부하에서 발생하는 모든 객체는 그 인기도에 따라 저장되고 단 한번만 참조된 객체들은 캐쉬에서 재참조되지 않을 것이기 때문에 모두 삭제된다.

(2) 남겨진 객체들은 그 인기도 순으로 저장된다.

(3) 두 번째 단계동안 캐쉬와 일치하지 않는 객체들은 그것들이 가장 인기 있는 객체를 대체하듯이 배치된다. 그리고 새로운 객체의 첫번째와 마지막 참조 사이의 시간은 대체된 객체의 첫번째와 마지막 참조 사이의 시간과 중복되지 않는다.

4.2 LRU 교체 전략

이 전략의 단점은 인기 있는 객체를 캐쉬에 가지고 있다는 것이다. 메모리 맵 파일의 문맥에서 LRU의 단점은 하나의 페이지에 인기 있는 객체를 배치하는 개념이 없다는 것이다. 그래서 로딩될 것 같지 않은 페이지에 대한 목적 객체들을 선택하는 경향이 있다. 이는 두 가지 영향을 미친다. 첫째, 그것은 목적 객체의 많은 비율이 8K보다 작은 세그먼트 크기이기 때문에 많은 페이지 장애(page fault)를 유발시킨다. 둘째, 페이지 장애의 많은 수가 큰 페이지 클러닝 오버헤드를 발생시키는 더티 페이지를 많이 생성하고, 또한 교체가 두 가지 디스크 입출력 트랜잭션을 발생시키는 최악의 경우가 일어날 가능성이 증가한다. LRU 대체의 세 번째 단점은 목적 페이지의 선택이 보다 연속적인 접근 스트림 대신에 거의 불규칙한 접근 스트림을 생성할 가능성이 있다는 것이다.

4.3 빈도 기반 순환 교체 전략

FBC(Frequency-based Cyclic Replacement)는 각 캐싱된 객체의 접근 빈도수와 대체될 첫번째 객체를 가리키는 목적 포인터를 유지한다. 어떤 객체가 실제 대체될지는 객체의 참조 빈도에 달려 있다. 만일 참조 빈도가 최대치보다 같거나 크다면 목적 포인터는 동일한 세그먼트 크기의 다음 객체로 이동한다. 참조 빈도가 최대값보다 작다면 그 객체는 교체의 목적 객체가 된다. 객체가 교체된 후에 목적 포인터는 다음 객체로 이동한다. 목적 포인터가 캐쉬의 마지막에 이르면 다시 처음부터 시작한다. 빈도수 계산은 모든 객체의 평균 참조수가 최대값보다 커질 때마다 누적된다. 평균값이 이 값에 이르면, 각 빈도수는 줄어든다. 그래서 안정된 상태에서 모든 참조수의 총계는 최대와 최대값 사이에 있다. 기능은 최소 참조수를 유지하기 위해 필요하다.

웹 캐싱은 히트율이 낮기 때문에 대부분의 캐싱된 객체들은 재참조되지 않는다. 이는 대부분의 시간동안 목적 포인터가 가리키는 첫번째 객체가 목적 객체가 된다는 것을 의미한다. 그 결과는 더티 페이지의 연속적 생성과 연속적 접근 스트림을 만들 수 있는 페이지 장애이다. 인기있는 객체의 간과는 두 가지 영향을 미친다. 첫째, 인기 있는 객체의 대체를 피하

고, 둘째, 주기적 교체와 나이 요소의 결합은 단기간동안만 인기 있는 객체의 참조로 나타난다.

4.4 Cache 압축

FBC 교체의 유용한 부수적 효과는 캐싱된 객체에 대한 인기도 정보를 보유하고 있다는 것이다. 메모리 맵 파일의 내용을 재구성하기 위해 이 정보를 사용할 수 있다. 인기 있는 웹 객체는 계속 인기 있는 상태일 가능성이 많다. 그래서 캐시 압축은 보다 낮은 디스크 입출력으로 시간이 지남에 따라 변환되는 작업 세트(working set)를 표현하기 위해 필요한 페이지 수를 줄일 수 있다.

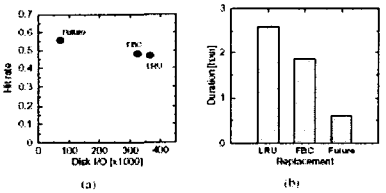
캐시 압축은 분명히 많은 양의 디스크 I/O를 초래한다. 그러나 캐시 압축은 비절정기 동안 수행될 수 있다.

본 연구에서 구현한 캐시 압축은 Cmax와 Cmin 이라는 두개의 리스트를 다음과 같은 방법으로 생성한다. 우선 처음부터 끝까지 캐시를 스캔하고 인기 있는 객체 리스트에 Cmax와 같거나 큰 참조 회수인 객체를 첨가하고, 인기 있는 객체 리스트에 Cmin보다 작거나 같은 참조 회수인 객체를 첨가한다. 양 리스트는 참조 회수를 기준으로 해서 역순으로 정렬된다. 인기 있는 객체 리스트는 인기도 순으로 저장된다. 두 개의 리스트는 비인기 객체의 장소와 인기 객체의 장소를 스왑 하는데 사용된다. 결과는 인기 객체가 캐시의 처음 페이지들에 집중된다.

캐시 압축의 두 파라메터인 Cmax 와 Cmin은 다음과 같은 의미를 갖는다. Cmax는 인기 객체의 상한선을 결정한다. Cmax가 높을수록 더 적은 객체들이 하루가 끝날 때 캐시의 처음으로 이동할 것이다. 이는 캐시 처음에 인기도를 더 많이 집중시키고 캐시 재정렬로 인해 1회의 디스크 I/O를 감소시킨다. Cmax가 낮으면 인기도의 집중은 낮아지지만 많은 객체들이 캐시의 처음으로 이동한다. Cmin는 객체 인기도의 하한선을 결정한다. 한번만 참조된 페이지들의 대규모 집합이 주어지면 Cmin은 항상 1이 될 것이다.

4.5 실험

본 논문에서는 메모리 맵 파일의 성능을 파악하기 위해 사용자 접근 로그에서 8KB 이상의 객체에 대한 참조는 모두 제외시켰다. 또한 64MB 주메모리와 1.6GB 디스크를 가진 작은 시스템을 사용했다. 메모리 맵 파일 크기는 160MB로 설정했다. 각 실험은 준비 단계와 측정 단계로 이루어지며, 범용으로 사용되는 iostat 로 디스크 I/O를 측정했다. 캐시 재정렬은 작업 세트를 크게 변화시키기 때문에 캐시 압축은 비절정기 동안 별도의 디스크 I/O를 발생시킨다. 본 연구에서는 캐시 압축을 준비 단계의 마지막에 수행했다. 측정 단계의 시작에 증가된 디스크 I/O 요소를 캐치하기 위해 측정 단계의 로그를 분리하고 버퍼 캐시를 준비하는 부분을 사용했다. 실험 시작은 자정이고 절정기의 시작은 오전 4:30으로 일간 트래픽 패턴을 가정한다. 자정과 오전 4:30사이의 추적 구간을 버퍼 캐시 준비 단계를 위해 사용한다. 캐시 압축의 효과를 비교하기 위해서 나머지 측정 추적(오전 4:30 이후의 요청들)만을 이용하여 캐시 압축을 했을 때와 안 했을 경우의 측정 단계의 전체 디스크 I/O를 비교했다. FBC의 경우 파라메터로 Cmax=3 과 Amax=100을 사용했다. 캐시 압축의 경우에는 Cmax=3, Cmin=1을 사용했다.



(그림 3) 교체 전략간 디스크 I/O와 히트율

그래프 (a)는 세 가지 교체 실험에 의한 히트율과 디스크 I/O를 나타낸다.

본 그래프에서 x축 값이 낮은 것이 높은 값보다 좋은 것이다. 즉 3 가지 교체 알고리즘에서 Future 기법이 가장 좋은 결과를 나타내었다. 그래프 (b)는 각 실험의 지속 기간을 나타낸다.

5. 결론 및 향후 연구 과제

각 교체 전략의 성능을 디스크 I/O의 양과 캐시 히트율에 의해 평가했다. 예상대로 LRU 교체 방법은 측정 단계에서 높은 디스크 트랜잭션을 유발했다. 미래예측 기법은 어느 점에서의 실제 작업 세트가 작았다는 것과 페이지 재사용 패턴의 정확한 예측은 물리적 메모리 크기의 캐시에서 높은 히트율을 생산할 것이라는 점을 보여준다. (그림 3)과 (표 1)은 빈도 기반 주기적 교체가 히트율을 변화시키지 않는 lru 교체보다 적은 디스크 I/O를 발생시킴을 보여준다. 또한 감소되는 디스크 I/O에 의해 발생된 시간 단축도 보여준다. 시간 단축은 디스크 I/O 단축보다 더 크다. 디스크 I/O 절약은 접근 스트림이 보다 연속적으로 일어나서 보다 많은 트랜잭션이 동일한 실린더에 접근함으로써 디스크 부분 재배치가 필요 없게 됨을 의미한다.

전략	디스크 전송회수	전송률	경과시간 (Seconds)
LRU	364617	0.47	9371
FBC	323563	0.48	6646
Future	70351	0.56	2171

(표 1) 세 가지 교체전략의 디스크 입출력 시간과 히트율

(표 2)는 캐시 압축으로 인한 절감 효과를 보여준다. 여기서 추측 컨데, 기간이 더욱 길어지면 그 효과는 더욱 증가하리라고 예상된다.

전략	디스크 전송회수	경과시간 (Seconds)
FBC	305443	6275
FBC/C	300631	6198

(표 2) FBC 전략에서 압축/비압축 비교

실제 작업부하는 다른 밴치마크 지향의 연구들에서는 보여지지 않았던 많은 중요한 성능 관련 이슈들을 확인할 수 있었다. 디스크와 네트웍 I/O로 인한 대기시간이 자원 이용도에 중요한 영향을 미친다는 사실을 알아냈다. 웹 캐시 서버를 도입하여 리소스 이용도를 향상시키기 위해 절정기때의 디스크 입출력 작업부하를 비절정기때로 분배해주는 작업과 객체 선반입으로 인한 네트웍 트래픽을 절약하는 연구가 필요하다

[참고문헌]

[1] Li Fan, Quinn Jacobson, and Pei Cao. Potential and Limits of Web Prefetching Between Low-Bandwidth Clients and Proxies. In to appear in SIGMETRICS99, 1999

[2] Adam Dingle and Tomas Partll. Web Cache Coherence. In Fifth International World-wide Web Conference, Paris, France, May 6-10 1996. W3C

[3] Stewart Forster. Personal Communication. Description of disk I/O in Squid 1.2, August 14 1998

[4] Ari Luotonen. Web Proxy Servers. Prentice Hall, Upper Saddle Rivber, Nj, 1998

[5] Duane Wessles. Ssquid Internet Object Cache. Available on the World-Wide Web