

# 파일 타입에 의한 프록시 서버의 캐쉬 대체 정책

두현재<sup>o</sup> 박정식 정진하 최상방  
인하대학교 전자공학과  
[sangbang@inha.ac.kr](mailto:sangbang@inha.ac.kr)

## Cache Replacement Policy for Proxy Server using Type-Based Partitioning

Hyun-Jae Doo<sup>o</sup> Chung-Sik Park Jin-Ha Chung Sang-Bang Choi  
Dept. of Electronic Engineering, Inha University

### 요약

전통적 파일 캐쉬나 가상 메모리 시스템과 웹 캐쉬는 다르다. 웹 캐쉬는 WWW상에서 작게는 수백바이트에서 크게는 수십 메가 바이트에 이르는 다양한 크기의 개체를 다루어야 한다. 다양한 크기의 개체를 다루는데 따른 문제점은 캐쉬 성능을 판단하는 매트릭스가 단순한 hit rate가 아니라는 것이다. 기본적인 웹 캐쉬의 성능 매트릭스로는 HR(cache hit rate)와 BHR(byte cache hit rate)가 있으며, 기준에 제시된 캐쉬 정책들은 두 가지 중 하나만을 만족하거나 아니면 어느것도 만족하지 않는 경우가 대부분이다. 트레이스 드리븐 방식을 이용한 시뮬레이션을 통하여, 기준에 우수성이 입증된 캐쉬 대체 정책과 우리가 제시한 TYPE 대체 정책을 HR과 BHR을 기준으로 비교 한다. 우리가 제시한, 파일 타입에 대해 동적으로 할당된 캐쉬 공간을 갖는 캐쉬 대체기법은 각각의 두 성능 매트릭스에 대해서 꼽고루 우수한 성능을 보였다.

### 1. 서론

Martin F. Arlitt의 연구에 의하면 웹상에서 같은 파일 타입의 객체 집합은 낮은 coefficient of variation을 갖는다. 이는 같은 타입의 파일들은 비교적 비슷한 크기의 파일로 구성됨을 시사하는 것이다. 따라서 모든 개체를 저장할 단일 캐쉬공간을 갖는 대신에 파일 타입에 기초한 개체의 클래스들을 저장할 몇 개의 구역을 갖는 분할된 캐쉬구조를 제안한다. 각각의 클래스는 비교적 요구율이 높지만 평균크기가 작은 파일타입이 요구율은 낮지만 사이즈가 매우 큰 파일타입에 의해 회생당하지 않을 일정한 영역을 할당 함으로써 비교적 높은 HR(hit rate)를 보장 받을 수 있다. 이와 함께 우수한 BHR(byte hit rate)를 갖는 LRU 알고리즘도 함께 적용하여 고른 성능의 대체기법을 제시할 수 있었다. 이 논문은 다음과 같이 구성되어 있다. 다음의 2절에서는 HTTP의 특성으로 인한 추가된 성능 평가 매트릭스를 소개하고, 제 3절에서는 제안할 알고리즘을 위해 선 행되어야 할 캐슁 기법과 웹 접근 특성에 대해 알아본다. 제 4절에서는 지금까지 캐쉬 대체 기법의 한계점을 논하고 이를 극복할 수 있는 새로운 캐쉬 대체 기법을 서술한다. 제 5절과 6절에서는 소개된 알고리즘들의 성능을 평가하기 위한 트레이스 드리븐 시뮬레이션 및 그 결과에 대해 논한다. 마지막으로 제 7절에서는 현재까지 수행되어온 본 연구의 성과에 대해 정리하고 앞으로

의 개선 방향에 대해 논한다.

### 2. 선행된 연구들

#### 2.1 HTTP의 한계

WWW개체는 모든 데이터를 개체로 간주하여 한 session당 한 개체 전체를 전송하기 때문에 페이지 단위의 저장 및 대체(replacement)는 의미가 없다. 따라서 프록시 서버에서는 이제까지 자주 이용되어 온 고정 크기의 페이지에 기반 한 캐시 관리 기법 대신 가변 크기의 개체에 기반 한 캐시 관리 기법의 도입이 불가피 하다.

#### 2.2 성능 평가 요소

전체 클라이언트 요구중 프록시 서버에 의해 응답되는 URLs의 비율 HR(hit rate)외에, 개체의 가변적인 크기에 의해 BHR(byte hit rate)이라는 [3] 또 다른 성능 평가 요소가 고려 되어야 한다. BHR은 전체 클라이언트의 요구중 프록시 서버에 의해 응답되는 bytes양의 비라고 정의 될 수 있다.

### 3. 대표적 캐쉬 알고리즘 및 WWW의 접근 패턴 특성

#### 3.1 HR에서 우수한 성능을 보이는 SIZE알고리즘

캐쉬 내로 들어온 새 개체를 위해 크기가 큰 파일부터 제거하는 SIZE알고리즘이 높은 HR을 갖는 이유는 다음의 두 가지이다 : 사용자 요청 대부분이 작은 파일을

참조하고, 두 번째로 큰 파일을 제거해 많은 작은 파일을 위한 공간을 만들도록 해서 hit이 일어날 수 있는 캐쉬내의 URL갯수를 증가 시키기 때문이다.

### 3.2 BHR에서 우수한 성능을 보이는 LRU알고리즘

동일한 파일이 가까운 시간 안에 다시 접근될 확률이 높다는 temporal locality특성을 이용하여 개체의 size와 상관없이 접근 된지 오래된 파일부터 캐쉬 내에서 제거하는 LRU(Least Recently Used)알고리즘은 우수한 BHR을 보여준다[1].

### 3.3 분석된 WWW 파일 접근 특성

1996년에 Martin F.Arlitt의 연구[2]는, WWW상의 사용자 파일 요청 특성을 자세히 분석 하였다. 특히 주목할 만한 사항은 요청된 파일이 같은 타입인 경우 낮은CoV(Coefficient of Variation)를 갖는다는 것이었다. 이는 웹 상에서 요청되는 개체가 같은 파일 타입이면 크기 또한 비교적 비슷한 분포를 갖는다는 것을 의미한다. 이상의 특성은 앞으로 분할된 캐쉬를 제안하는데 있어서 매우 중요한 WWW특성이 된다.

### 4. 파일 타입에 의해 분할된 캐쉬 공간을 갖는 대체 알고리즘

3절에서 이야기 된 웹의 특성 및 각각의 성능 매트릭스에서 우수한 성능을 보이는 캐쉬 알고리즘들을 기초로 하여 지금까지 캐쉬 대체 기법의 한계점을 논하고 이를 극복할 수 있는 새로운 캐쉬 대체 기법을 서술하겠다.

#### 4.1 현재까지 제시된 알고리즘의 문제점 및 개선점

이전까지 제시된 캐쉬 대체 기법들은 다음과 같은 문제점을 안고 있다.

1. HR과 BHR 모두를 고려해야 한다.

2. 기존의 hybrid algorithms은 primary sort key에 많은 영향을 받는다.

3. 기존의 value based algorithms은 높은 CPU overhead를 갖는다.

4. 기존의 static algorithms은 cache의 내용이 refresh 될 때 까지 능동적으로 user request를 반영하지 않는다.

따라서 새로 제안될 알고리즘은 낮은 CPU overhead를 갖고, 현재의 user request를 동적으로 반영하면서, 우수한 BHR을 갖는 LRU알고리즘과 우수한 HR을 갖는 SIZE알고리즘의 특성을 모두 수용할 수 있어야 한다.

#### 4.2 제안 알고리즘

파일 타입을 기초로 하여 분할된 캐쉬를 갖는 구조이다. 논리적인 근거는 비교적 비슷한 크기를 갖는 파일 타입별(3.3에서 보여진 것처럼)로 최적의 크기를 갖는 캐쉬 영역을 보장 하고, 분할된 캐쉬 내부에서는 비슷한 크기의 파일을 위한 LRU 알고리즘을 적용 함으로써 우수한 성능을 보장 받는 것이다.

Type-based partitioned cache를 디자인하기 위해서는 다음과 같은 몇 가지 요소들이 포함된다 : 분할된 캐쉬의 갯수, 각 캐쉬 파티션의 크기, 각 파티션에 포함될 파일의 종류, 파티션 내에서 사용될 대체 알고리즘. 이

연구의 목적을 위하여 우리는 다음과 같이 위의 요소들을 정하였다.

#### 1. Cache partitioning

각각의 파일타입별 분류를 class라 부를 수 있다. 6개의 class(graphic, text/html, audio, video, unknown)를 생성 한다. 각각의 class가 일정한 영역을 보장 받을 수 있도록 캐쉬를 분할한다. 이때 분할 비율 결정 할 요소로 class별 % of Bytes Transferred(Number of request \* Mean Transfer Size)를 사용한다.

#### 2. TYPE algorithm 적용; HR 보장

항상 큰 파일이 작은 파일로 대체되지는 않지만, 작은 크기를 갖는 파일 타입(같은 타입의 파일은 낮은 CoV를 갖으므로)을 위해 적절한 크기의 캐쉬 공간이 일정하게 할당 되므로 SIZE알고리즘에 준하는 HR을 보장 받을 수 있다.

#### 3. LRU algorithm 적용; BHR 보장

비교적 동일한 크기의 집합으로 볼 수 있는 class 내에서는 우수한 BHR을 갖는 LRU 알고리즘을 적용한다.

#### 4. Dynamic decision of partitioned cache

Cache의 분할 비는 주기적으로(하루 또는 일주일 단위) 혹은 일정한 임계치에 다다르면 이전까지 누적된 분할 비 결정요소에 의해 새로이 결정된다.

이 알고리즘의 구현은 쉽고 직관적이다. 캐쉬 관리자는 캐쉬된 개체의 목록을 관리한다. 만일 새로운 요청이 오면 관리자는 요구된 개체를 목록에서 찾는다. 만약 목록에 존재하지 않으면(a miss) 관리자는 웹 서버로부터 직접 그 개체를 요청해야 한다. 이 동작은 파일 전송이 될 것이고 그 파일의 타입을 알 수 있게 된다. 전송된 파일은 반드시 그 파일이 속한 클래스를 책임지는 캐쉬 파티션 내에 위치하게 된다. 만약 캐쉬가 새로운 개체를 위해 공간을 비워야 한다면, LRU정책이 그 파티션 내에서 적용된다.

#### TYPE algorithm:

*Set each partition size by % of Bytes Transferred.*

*Initialize each partition in cache.*

*Initialize thePartition ← 0*

*Initialize AccessedTime ← 0*

*Initialize temp ← 0*

*process each document in turn.*

*the current request is for document p:*

*thePartition ← partitionNumber(documentType(p))*

*if p is already in thePartition,*

*{ AccessedTime of p ← current time }*

*if p is not in thePartition,*

*{ While there is not enough room in thePartition for p,*

*{ Let temp ← MAX<sub>q ∈ thePartition</sub>AccessTime(q)*

*Evict q such that AccessTime(q)=temp }*

*Bring p into thePartition and set*

*AccessedTime of p ← current time }*

*End*

그림 1. TYPE algorithm.

#### 5. 시뮬레이션 환경 및 가정

이 절에서는 위에서 소개한 알고리즘을 실제 운용중인

프락시 서버에서 수집된 request sequence에 대해 적용한다. 또한 HR과 BHR에서 높은 성능을 보이는 SIZE와 LRU알고리즘을 제안된 알고리즘과 비교한다.

### 5.1 시뮬레이션에서 사용된 트레이스 데이터

신퇴성을 높이기 위해 인터넷 상에 공개된 데이터[4]를 사용하였다. Virginia Tech의 5개의 트레이스 자료 중 프락시 서버의 트레이스 데이터로 문제점이 없는 workload U만을 시뮬레이션에 사용하였다.

*Undergrad(U)* : 약 30대의 컴퓨터가 학부생 컴퓨터실에 있다. 1995년 4월부터 10월 까지 190일간 2.19GB

SIZE와 의 정적인 웹 페이지를 요청했던 173,384개의 유익한 접근을 포함한다.

### 5.2 시뮬레이션 방법

실험은 제시된 workload에 대해 HR과 BHR에서 우수함이 입증된 알고리즘과 우리가 제시한 알고리즘(TYPE)을 비교 평가 하는 것이다.

모든 실험은 캐쉬를 완전히 비우고, 연속적으로 들어오는 사용자 요구에 대한 hit rate의 변화를 그래프로 보여준다. 이때 캐쉬 크기는 10%의 MaxNeeded(새로운 개체에 의해 캐쉬내의 어떤 개체도 제거되지 않을 최소한의 캐쉬 크기)로 한다.

### 6. 시뮬레이션 결과

#### 기존 대체정책과 제시된 정책의 성능 비교

시뮬레이션의 결과는 HR과 BHR 이란 두 성능 매트릭스로 요약된다. 그림2는 트레이스내의 연속된 요청의 개수에 대한 함수로서 HR을 보여준다. 최상의 알고리즘은 SIZE이다. TYPE의 HR 역시 SIZE에 매우 근접하였다. 그러나 LRU는 두 알고리즘에 비해 분명히 가장 나쁜 결과를 보였다. 이는 대부분의 파일 요청이 작은 개체에 집중하기 때문이다. 그림3은 트레이스내의 연속 요청의 개수에 대한 함수로서 BHR을 보여준다. 이 경우 LRU가 최상의 정책이다. TYPE은 다시 최적 값에 가까운 결과를 보인다. 이때 SIZE는 분명히 가장 나쁜 정책이다. SIZE알고리즘은 BHR을 증가시킬 큰 파일들을 버린다는 사실이 낮은 성능을 보이는 이유이다.

우리는 TYPE이 HR과 BHR 모두에서 잘 동작함을 알 수 있으며 이것이 주요한 잇점이다.

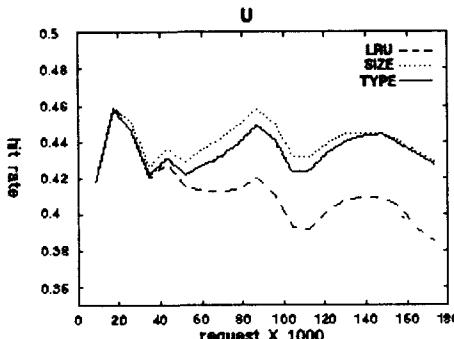


그림 2. Hit rate and cache size of 10% of MaxNeeded in workload U

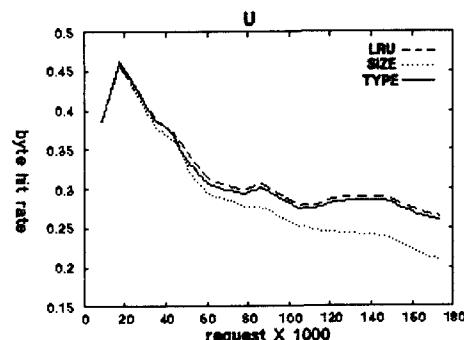


그림 3. Byte hit rate and cache size of 10% of MaxNeeded in workload U

### 7. 결론 및 앞으로의 방향

본 논문에서 우리는 파일 타입에 대해 분할된 캐쉬 구조를 갖는 캐쉬 대체 기법을 제안하고 그 성능을 분석 하였다. 이 대체 알고리즘은 개체의 크기와 요구율의 높은 변화폭을 잘 고려한다. 기존에 제시된 다른 정책들과는 달리, 시뮬레이션에서 알 수 있듯이 캐쉬 관리 정책의 성능을 평가하는 두 가지 성능 매트릭스(HR과 BHR) 모두에서 좋은 결과를 보여주었다.

앞으로의 연구에서는 다음과 같은 문제점들이 개선 되어야 할 것이다.

- 좀 더 다양한 트레이스 자료에 대한 성능 분석
- 하이브리드 알고리즘과의 비교 분석
- 보다 효율적인 파일타입별 클래스 분류
- 주기적 캐쉬 분할비 결정시 최적의 주기 및 임계치 도출

### 참고 문헌

- [1] Pei Cao and Sandy Irani "Cost-Aware WWW Proxy Caching Algorithms," *2nd Web Caching Workshop*, Boulder, Colorado, June 1997.
- [2] M. F. Arlitt and C. L. Williamson "Web server workload characterization: The search for invariants," *In Proc. SIGMETRICS*, Philadelphia, PA, Apr. 1996. ACM.
- [3] S. Williams, M. Abrams, C. Stanbridge, G. Abdulla, and E. Fox "Removal Policies in Network Caches for World-Wide Web Documents," *In Proceedings of ACM Sigcomm96*, Stanford, USA, August 1996.  
<http://ei.cs.vt.edu/~succeed/96WAASF1/>
- [4] Stephen Williams, Marc Abrams, C. R. Standridge, Ghaleb Abdulla, and Edward A. Fox "Removal policies in network caches for world wide web documents"  
<ftp://ei.cs.vt.edu/pub/succeed/Sigcomm96/>