메타 검색 엔진을 위한 인기도 기반 캐쉬 관리 및 성능 평가

(A Popularity-driven Cache Management and its Performance Evaluation in Meta-search Engines)

홍 진 선 † 이 상 호 ††
(Jin Seon Hong) (Sang Ho Lee)

요 약 메타 검색 엔진에서 캐쉬의 사용은 사용자의 응답시간을 향상시킬 수 있다. 본 논문에서는 메타 검색 엔진의 구조와 동작을 보이고, 메타 검색 엔진을 위한 인기도 기반의 새로운 캐쉬 대체 방법을 제안한다. 인기도는 사용자들이 검색 엔진에 요청한 단어들의 출현 빈도수를 정규화한 값으로, 캐쉬 대체를 위한 기준치로 이용된다. 본 논문에서는 인기 검색어 수집 방법, 인기도 산출 방법을 기술하고, 인기도를 기반으로 하는 새로운 알고리즘을 제안한다. 또한 실제 사용자가 검색 엔진에 입력한 자료를 바탕으로, 전통적인 캐쉬 대체 기법인 LRU, LFU 알고리즘과 제안된 알고리즘을 성능 평가하였다. 본 성능 평가에서는 제안된 알고리즘이 대다수의 경우 우수한 성능을 나타내었다.

키워드 : 캐쉬 관리, 캐쉬 대체 알고리즘, 인터넷 데이타베이스, 메타 검색 엔진, 인기도

Abstract Caching in meta-search engines can improve the response time of users' request. We describe the cache scheme in our meta-search engine in terms of its architecture and operational flow. In particular, we propose a popularity-driven cache algorithm that utilizes popularities of queries to determine cached data to be purged. The popularity is a value that represents the normalized occurrence frequency of user queries. This paper presents how to collect popular queries and how to calculate query popularities. An empirical performance evaluation of the popularity-driven caching with the traditional schemes (i.e., least recently used (LRU) and least frequently used (LFU)) has been carried out on a collection of real data. In almost all cases, the proposed replacement policy outperforms LRU and LFU.

Key words : cache management, cache replacement algorithm, internet database, meta-search engine, popularity

1. 서 론

인터넷 사용자들에게 원하는 정보를 쉽게 찾을 수 있게 도움을 주는 검색 엔진들이 서비스되고 있다. 하지만, 하나의 검색 엔진이 제공하는 정보의 양은 사용자의 검색 욕구를 충족시키기에는 충분치 않다[1]. 이를 극복하기 위해 다수의 메타 검색 엔진들이 개발되고 있다 [2]. 메타 검색 엔진은 사용자의 검색 요청을 받아, 다수의 검색 엔진으로부터 해당 정보를 수집하여 정제(refine)한 후, 일괄된 형식으로 사용자에게 제공하는 검색 엔진이다.

사용자가 요청한 정보를 제공하기 위해서, 메타 검색 엔진은 항상 타 검색 엔진에 네트웍을 통한 접근을 해야 한다. 이는 네트웍 부하(traffic)나 원격서버 성능으로 인한 정보 제공 서비스 질(quality)의 저하를 가져올 수 있다. 느린 응답 시간을 해결하기 위한 한가지 방안은 재검색 될 확률이 높은 검색 결과를 저장(caching)하여 네트웍을 통한 타 서버 접근을 최소화하는 결과 캐쉬[3]를 하는 것이다.

현재까지 캐쉬 문제는 많은 사람들에 의해 연구되었으며, 다양한 방법론이 제시되었다. 일반적으로 페이지

· 본 연구는 한국과학재단 목적기초연구 (2000-2-51200-002-3) 지원으로 수행되었음.
† 정 회 원 : (주)메타웨이브
jshong@metawave.co.kr
†† 종신회원 : 숭실대학교 컴퓨터학부 교수
shlee@computing.ssu.ac.kr
논문접수 : 2001년 4월 30일
심사완료 : 2002년 1월 3일

캐쉬(page caching), 튜플 캐쉬(tuple caching), 의미 캐쉬(semantic caching) 등의 캐쉬 관리 기법이 사용되고 있다. 페이지 캐쉬는 자료를 페이지 단위로 캐쉬하는 방법으로, 운영체제나 데이타베이스에서 주로 사용된다. 튜플 캐쉬는 튜플 각각을 전송 단위 및 접근(access) 단위로 사용하는 캐쉬 관리 방법이다.

캐쉬를 사용할 때에는 한정된 캐쉬 공간을 효율적으로 활용하기 위한 캐쉬 대체 정책(cache replacement policy)을 고려해야 한다. 많이 사용되는 대체 기법으로는 LRU(Least Recently Used)와 LFU(Least Frequently Used)가 있다. LRU 대체 기법은 최근에 오랫동안 사용되지 않았던 페이지를 교체하는 방법으로, 미래에 참조되지 않을 페이지는 오랫동안 사용되지 않은 페이지일 가능성이 높다는 페이지 참조의 지역성을 이용한다. LFU 대체 기법은 자주 사용되지 않은 페이지를 교체하는 방법으로 참조 빈도수가 적은 페이지를 우선적으로 교체하는 방법이다. 그 밖에도 FBR (Frequency-Based Replacement)[7], LRU-K(Least Recently Used-K)[8], 2Q(Two Queue)[9], LRFU (Least Recently/Frequently Used)[10] 등이 있다.

이러한 대체 기법은 주로 데이타베이스나 운영체제를 기반으로 연구되었다. 하지만 메타 검색 엔진은 데이타베이스나 운영체제와는 상이한 환경을 가지기 때문에 기존의 대체 기법이 우수한 성능을 발휘하기 어렵다. 메타 검색 엔진이 가진 상이한 환경은 다음과 같다. 첫째, 저장된 자료의 진부성(staleness)을 고려해야 한다. 검색 엔진으로부터 제공된 검색 결과는 시간이 흐름에 따라서 타당하지 않은 정보가 될 수 있다. 외부 검색 엔진에서는 자료 진부성에 대한 정보를 제공하지 않는다. 둘째, 메타 검색 환경에서는 캐쉬된 자료와 서버 자료간의 일치성(consistency)이 요구되지 않는다. 셋째, 자료를 제공하는 원격서버의 연결성(accessibility)을 항상 보장하기 어려우므로 검색 결과에 대한 저장 필요성이 대두된다. 메타 검색 엔진에서의 캐쉬 대체 기법은 이러한 환경을 고려하여 설계되어야 한다.

본 논문에서는 메타 검색 엔진에서 캐쉬를 관리하기에 적합한 인기도 기반의 캐쉬 대체 알고리즘을 제안한다. 숭실대학교에서 개발 중인 메타 검색 엔진의 구조와 동작 과정에 대하여 언급하고, 인기도 산출 방법에 대하여 기술한다. 또한 실제 사용자가 검색 엔진에 입력한 자료를 바탕으로, 전통적인 캐쉬 대체 기법인 LRU, LFU 알고리즘과 제안된 알고리즘을 성능 평가한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 의미 캐쉬와 성능 평가에 사용될 메타 검색 엔진인 EzFinder를 설명하고, 사용자 질의 입력 경향에 대하여 언급한다. 제 3장에서는 인기 검색어에 대한 정의와 인기도를 추출하는 방법에 대하여 기술하고, 인기도 기반의 대체 알고리즘에 대하여 서술한다. 제 4장에서는 기존의 대체 알고리즘과 성능 평가를 수행한다. 제 5장에서는 결론 및 향후 과제에 대해 기술한다.

2. 관련 연구

2.1 의미 캐쉬(Semantic Caching)

의미 캐쉬[4, 5, 6]는 클라이언트 캐쉬를 의미 영역(semantic regions)으로 나누어 관리하는 기법이다. 의미 영역은 캐쉬 대체를 수행하거나, 정보의 접근 단위로 사용되는 것으로 사용자가 요청한 질의에 의해 검색된 결과들을 동일한 의미를 갖는 것끼리 모은 것이다. 또한 동적으로(dynamically) 생성되며, 클라이언트 캐쉬 관리의 수단으로 사용된다.

각 의미 영역은 제한 사항(constraint formula)을 가진다. 제한 사항은 의미 공간(semantic space)에서 의미 영역의 위치정보를 나타내며, 의미 영역이 가질 수 있는 튜플의 수, 캐쉬에 저장된 실제 튜플과의 연결을 나타내는 포인터(pointer), 캐쉬 대체시에 사용될 영역의 순위 등에 대한 추가적인 정보를 포함한다.

의미 캐쉬는 다음과 같은 방법으로 동작한다. 사용자 질의가 클라이언트의 의미 캐쉬에 요청되면 질의는 두 부분으로 나뉜다. (1) 조사 질의(prove query)는 클라이언트 캐쉬 내에서 검색할 수 있는 부분이며, (2) 잔여 질의(remainder query)는 서버로부터 받은 결과 중에서 불일치된 튜플(missing tuples)을 검색하는 부분을 의미한다. 만일 잔여 질의가 빈(null) 값이 아니면(질의에 해당하는 결과가 모두 캐쉬되어 있지 않다면), 잔여 질의는 서버로 전달되어 처리되며, 처리된 결과는 클라이언트로 전달된다.

2.2 EzFinder

EzFinder는 다수개의 국내외 상용 검색 엔진으로부터 검색 결과를 수집하여 사용자에게 제공하는 메타 검색 엔진이다([그림 1] 참조). EzFinder는 검색 엔진에 접근하여 정보를 수집하는 래퍼(wrapper)가 각 검색 엔진별로 독립적으로 존재하며, 질의 관리자(Query Manager)는 사용자가 입력한 질의를 EzFinder에서 정의한 질의 형태로 사상(mapping)하고, 사상된 질의를 래퍼 관리자(Wrapper Manager)로 전달한다. 래퍼 관리자는 각 검색 엔진별로 존재하는 래퍼의 동작을 제어하며, 각 래퍼로부터 받은 검색 결과를 출력 관리자(Output Manager)로 전달한다. 출력 관리자는 래퍼 관리자로부터 받은 검색 결과를 정제하여 사용자에게 제공한다.

EzFinder는 래퍼마다 독립된 캐쉬를 보유하는 래퍼

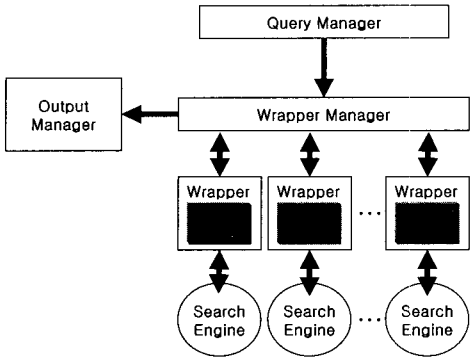

그림 1 EzFinder 캐쉬 구조

단계 캐쉬(wrapper-level cache)를 사용한다. 각각의 래퍼에 존재하는 캐쉬에는 검색 결과(HTML 페이지 집합)와 질의가 저장된다. 캐쉬는 주 메모리와 디스크를 사용하여 구성할 수 있다. 본 논문에서는 주 메모리를 이용한 캐쉬만을 고려한다.

표 1 상용 검색 엔진에서의 사용자 질의 입력 경향

어절 수	질의 수	질의 수/총 질의 수
단일 어절	678,701건	0.86840
두 어절	68,678건	0.08787
세 어절	20,941건	0.02679
네 어절 이상	13,230건	0.01692

[표 1]은 국내 상용 검색 엔진에서 생성된 4일 분량의 로그 파일 중, 유효한 질의 781,550건을 추출하여 분석한 자료이다. 어절의 구별은 질의에 포함되어 있는 공백을 이용하였다. 표를 보면, 단일 어절 질의가 전체 질의의 86% 이상 차지함을 알 수 있다. 또한, 세 어절 이상의 질의는 약 5% 이하의 수치를 나타내었다. 이와 같은 사실은 대다수의 검색 엔진 사용자는 하나 또는 두 개의 어절로 이루어진 질의를 한다는 것을 의미한다. 이에 EzFinder는 사용자가 입력한 검색어 중에 단일 어절 검색어만을 캐쉬에 적재한다. 두 어절 이상의 검색어는 해당 검색 엔진에서 가져온 결과를 사용자에게 표출하며, 캐쉬에 적재되는 경우가 없다.

3. 메타 검색 엔진을 위한 인기도 기반의 캐쉬 구성

이 절에서는 인기도(popularity)의 개념과 산출 방법에 대해서 논하고, 메타 검색 엔진에서 사용되는 캐쉬 대체 기법인 PDR(Popularity-Driven Replacement) 알고리즘에 대해서 기술한다.

3.1 인기 검색어(Popular Queries)

검색 엔진에 입력된 질의들 중에 출현 빈도수가 높은 단어는 사용자에게 인기 있는 검색어이다. 이러한 인기 검색어는 각 검색 엔진의 추출 정책에 따라, 공시되는 인기 검색어 수, 선별작업(filtering) 여부, 공시주기(posting cycles) 등이 상이하다. 국내의 대다수 상용 검색 엔진들은 5개에서 100개의 인기 검색어를 매일 또는 매주 제공하고 있다.

일반적으로 인기 검색어는 시대적 흐름과 사회 전반적인 경향 및 유행(public trends/interests)을 잘 반영한다. 예를 들면, 수학능력시험에 즈음하여 각 검색 엔진의 인기 검색어 목록에는 입시, 대학교 등의 수학능력시험과 관련된 단어들이 상위에 공시되었다.

인기 검색어가 가지는 다른 특징은 많은 중복을 포함하고 있다는 것이다. 인기 검색어의 중복은 수집 기간대별로 나타난다. 예를 들면, 50개의 인기 검색어를 매일 제공하는 상용 검색 엔진의 인기 검색어를 일주일 동안 수집한 결과, 중복이 없는 유일한(distinct) 인기 검색어 60-70개만이 수집되었다. 만일 중복을 제거하지 않았다면, 총 350개의 인기 검색어가 수집되었을 것이다. 인기 검색어의 중복은 수집 검색 엔진간에도 나타난다. 공시주기가 다른 세 개의 검색 엔진(각각 50개/매일, 50개/매주, 100개/매주)에서 일주일 동안 수집된 인기 검색어를 살펴보면, 중복을 제거하지 않았을 경우에는(단, 인기 검색어를 매일 공시하는 경우에는 중복이 제거되어 60-70개의 인기 검색어를 제공한다고 가정한다.) 210-220개의 인기 검색어가 수집되어야 한다. 하지만 중복을 제거한 결과, 유일한 인기 검색어의 수가 140개 내외로 나타남을 알 수 있었다. 이는 서로 다른 검색 엔진에서 동일 기간에 제공되는 인기 검색어 목록은 매우 유사함을 나타낸다.

많은 중복이 일어난 검색어일수록 다수의 사용자에 의해 질의된 검색어로 여길 수 있다. 차후 기술할 인기도는 인기 검색어의 이러한 특성을 충분히 반영하도록 하였다[11].

3.2 인기도(Popularity)

검색 엔진에서 제공하는 인기 검색어는 검색어간에 순위(ranking)를 가진다. 검색어간의 순위는 해당 검색어의 출현 빈도수에 의해 정해지며, 상위 순위를 갖는 몇 개만이 사용자에게 공시되고 있다.

본 논문에서는 사용자가 입력한 임의의 단일 어절 검색어에게 0과 1사이의 소수값을 갖는 인기도를 부여한다. 인기도는 해당 검색어의 순위를 공시주기, 사전에

정의된 최대 인기 검색어 수(W), 외부 검색 엔진의 수로 정규화하여 산출된다. 인기도가 높을수록 많은 사람에 의해 질의된 검색어임을 의미하며, 만일 임의의 검색어가 0의 인기도를 갖는다면, 해당 검색어는 비인기 검색어(non-popular queries)임을 의미한다. 인기도는 산출 방법에 따라 단일 인기도(unit popularity; UP), 누적 인기도(accumulated popularity; AP), 병합 인기도(fused popularity; FP)로 구분된다.

W는 사전에 정의되는 최대 인기 검색어 수를 나타내는 값으로 인기도를 정규화할 때 사용된다. W는 검색 엔진에서 발표하는 인기 검색어 수와는 무관하게 결정되며, 각 검색 엔진에서 공시하는 인기 검색어 수보다 항상 큰 값을 가진다. 정의된 W는 지속적으로 변하지 않는다.

단일 인기도는 하나의 인기 검색어 목록을 대상으로 각 검색 엔진의 공시주기마다 산출한다. $R_i^j(X)$는 i번째 검색 엔진에서 j번째 공시되는 인기 검색어 X에 대한 순위를 나타낸다. i번째 검색 엔진에서 j번째 공시되는 인기 검색어 X의 단일 인기도를 나타내는 $UP_i^j(X)$는 다음과 같이 산출된다.

$$UP_i^j(X) = \frac{W - R_i^j(X) + 1}{W}$$

인기 검색어 X의 순위를 나타내는 $R_i^j(X)$는 항상 1 이상의 값을 가진다. 단일 인기도는 본질적으로 하나의 인기 검색어 목록에 포함되어 있는 인기 검색어들의 순위를 정규화하여 산출하며, $0 < UP_i^j(X) \leq 1$의 범위를 갖는다. 만일 서로 다른 검색 엔진에서 추출한 인기 검색어의 순위가 동일하다면, 각각의 인기 검색어는 동일한 단일 인기도를 부여받는다. 이는 각 검색 엔진에서 제공되는 인기 검색어 수와 상관없이 각 인기 검색어에 단일 인기도를 부여함을 의미한다.

누적 인기도는 선결된(predetermined) 기간동안에 동일한 검색 엔진의 단일 인기도를 누적하여 산출한다. 선결된 기간동안 단일 인기도를 누적한 횟수를 차수(order)라고 한다. 차수는 인기도 값을 조정(adjustment)할 때 사용되며, 공시주기가 상이한 검색 엔진들 간의 누적 인기도 병합을 용이하게 한다.

n_i는 i번째 검색 엔진의 차수를 나타낸다. i번째 검색 엔진에서 제공하는 인기 검색어 X의 누적 인기도를 나타내는 $AP_i(X)$는 다음과 같이 산출한다.

$$AP_i(X) = \frac{1}{n_i} \sum_{j=1}^{n_i} UP_i^j(X)$$

누적 인기도는 단일 인기도의 합을 해당 검색 엔진의 차수로 나누어 산출된 값으로, $0 < AP_i(X) \leq 1$의 범위를 갖는다. 만일 인기 검색어 X가 해당 검색 엔진에 자주 출현된다면, 해당 검색어는 높은 누적 인기도를 부여받는다. 하지만 출현 빈도수가 적은 검색어는 차수가 높아질수록 누적 인기도 값은 감소하게 된다.

병합 인기도는 정기적으로(periodically) 인기 검색어 목록을 제공하는 이종의(heterogeneous) 검색 엔진에서 수집된 누적 인기도를 병합하여 산출한다. 수집 대상이 되는 검색 엔진의 수는 병합 인기도를 정규화할 때 사용된다. N은 인기 검색어를 제공하는 검색 엔진의 수를 나타낸다. 임의의 검색어 X에 대한 병합 인기도 FP(X)는 다음과 같이 산출된다.

$$FP(X) = \max\left(0, \frac{1}{N} \sum_{i=1}^{N} AP_i(X)\right)$$

본 논문에서 언급되는 모든 검색어는 병합 인기도를 부여받는다. 만일 검색어의 병합 인기도가 0이라면, 해당 검색어는 인기 검색어 목록에 포함되어 있지 않은 비인기 검색어임을 의미한다.

본 논문에서, 임의의 검색어가 높은 인기도를 갖는 경우는 다음과 같다. (1) 인기 검색어 목록에서 높은 순위를 가질 경우, (2) 동일 기간에 다수의 검색 엔진에 출현될 경우, (3) 서로 다른 기간에 동일 검색 엔진에서 자주 출현될 경우이다. 본 논문에서 제안한 인기도 산출 방법은 공시되는 인기 검색어 수, 외부 검색 엔진의 수, 각 검색 엔진의 공시주기에 충분히 유연하게(flexible) 적용된다.

[예제] 인기도 계산

본 예제에서는 두 개의 검색 엔진(각각을 1, 2로 명명한다)을 고려한다. W는 200으로 정의한다. 검색 엔진 1은 매주 100개의 인기 검색어를 공시하고, 검색 엔진 2는 매일 50개의 인기 검색어를 공시한다. 따라서 병합 인기도의 차수(n_i)는 각각 1, 7을 부여받는다.

각 검색 엔진에서 공시된 검색어 "Napster"의 순위는 다음과 같다. 검색 엔진 1에서는 2위에 랭크되었고, 검색 엔진 2에서는 일주일동안 각각 15, 10, 5, 1, 8, 9, 12위에 랭크되었다. 검색 엔진 1에서 "Napster"의 누적 인기도는 차수가 1이기 때문에 단일 인기도와 동일하다.

$$AP_1(Napster) = UP_1^1(Napster) = \frac{200 - 2 + 1}{200} = 0.995$$

검색 엔진 2에서의 "Napster"의 누적 인기도는 다음과 같이 산출된다.

$$\begin{aligned} AP_2(Napster) &= \frac{1}{7} \sum_{j=1}^{7} (Napster) \\ &= \frac{1}{7}\left(\frac{(200-15+1)+(200-10+1)+(200-5+1)}{200}\right) \\ &+ \frac{1}{7}\left(\frac{(200-1+1)+(200-8+1)+(200-9+1)+(200-12+1)}{200}\right) \\ &= 0.962 \end{aligned}$$

검색어 "Napster"의 병합 인기도 FP(Napster)는 다음과 같이 산출한다.

$$FP(Napster) = \frac{1}{2}\sum_{i=1}^{2} AP_i(Napster)$$
$$= \frac{1}{2}(AP_1(Napster) + AP_2(Napster))$$
$$= \frac{1}{2}(0.995 + 0.962) = 0.9785$$

따라서, 검색어 "Napster"는 0.9785의 병합 인기도를 부여받는다.

3.3 PDR(Popularity-Driven Replacement) 알고리즘

본 절에서는 인기도 기반의 캐쉬 대체 알고리즘인 PDR에 대하여 기술한다. PDR 알고리즘은 기본적으로 검색어의 병합 인기도를 이용하여 캐쉬 대체를 수행한다. 또한 비인기 검색어를 캐쉬 내에 일정 비율 적재함으로써 캐쉬 성능을 향상시킨다. 이를 위해 참조 빈도수(reference counter)를 사용한다. 참조 빈도수는 캐쉬 내에서 특정 검색어가 요청된 수를 나타내며, 항상 1 이상의 값을 유지한다.

캐쉬는 동일한 크기를 가진 다수의 슬롯(slot)으로 구성된다. 각각의 슬롯의 5개의 요소(X, FP, C, T, R)를 가진다. X는 사용자가 입력한 검색어를 나타낸다. FP는 검색어 X의 병합 인기도를 나타내며, C는 현재까지의 참조 빈도수를 나타낸다. T는 검색어에 해당하는 검색 결과가 수집된 시간을 나타내며, R은 수집된 검색 결과를 나타낸다. 본 논문에서 R은 HTML 페이지의 집합을 의미한다.

일반적으로 상용 검색 엔진들은 저장된 정보를 특정 시간 주기마다 갱신한다고 알려져 있다. 본 논문에서는 캐쉬에 적재되어 있는 검색 결과 중에 진부한 검색 결과(stale search result)를 사용자에게 반환하지 않기 위해서, 수집된 검색 결과의 시간을 항상 점검한다. 검색 결과의 진부성을 관리하기 위해서, 각 검색 엔진마다 특정 변수를 할당한다. 이 변수는 해당 검색 엔진의 갱신 주기를 나타내며, 캐쉬에 적재되어 있는 검색 결과의 진부성 여부를 판단하는 기준으로 사용된다. 만일 검색 결과가 진부하다면, 해당 검색 엔진에 접근하여 새로운 검색 결과(fresh search result)를 받아온다. 새롭게 받아온 검색 결과는 해당 슬롯에 저장되며, 해당 검색어의 참조 빈도수는 하나 증가된다.

PDR에서 캐쉬 대체 시에 가장 중요한 기준이 되는 인기도는 인기도 갱신 주기(popularity update cycle)에 맞추어 갱신된다. 만일 인기도 갱신 주기에 해당되면, 현재까지 수집된 모든 인기 검색어들의 병합 인기도와 캐쉬에 적재되어 있는 검색어들의 병합 인기도의 갱신을 수행한다. 따라서 캐쉬에 적재되어 있는 비인기 검색어(0의 인기도를 가진 검색어)들이 인기도 갱신 주기가 경과하면 인기 검색어로 변경되는 일이 발생하기도 한다. 반대로 인기 검색어들이 비인기 검색어가 되는 경우도 존재하게 된다. 하지만 인기도 갱신 주기가 경과하여도 캐쉬에 적재되어 있는 검색어의 참조 빈도수는 변경되지 않는다.

[그림 2]는 PDR 알고리즘을 나타낸다. 입력 값은 검색어 X이고, 출력 값은 해당 검색어의 검색 결과 R이다. N_S는 캐쉬 전체 슬롯수를 의미하며, N_P는 현재 캐쉬에 적재되어 있는 인기 검색어 수를 의미한다. O_{NP}는 캐쉬 내에 적재되어야 할 최소 비인기 검색어의 비율을 의미하며, 캐쉬 내에 비인기 검색어의 비율을 일정치 이상으로 유지하기 위한 임계값(threshold)으로 사용된다. 만일 사용자로부터 받은 검색어가 캐쉬에 저장되어 있다면, 해당 검색어의 인기도에 상관없이, 검색어의 참조 빈도수를 증가시킨 후에 검색 결과를 반환한다.

사용자 검색어가 캐쉬에 저장되어 있지 않다면, 질의에 대한 검색을 수행하고 캐쉬에 검색 결과를 저장한다. 이때 캐쉬가 비어 있다면, 비인기 검색어의 비율과 사용자 검색어의 종류(인기 검색어 또는 비인기 검색어)에 따라 캐쉬 대체(인기도 기반) 수행 여부를 결정한다. 캐쉬에 빈 슬롯이 존재하지 않는다면 비인기 검색어의 비율과 사용자 검색어의 종류에 따라 캐쉬 대체(인기도 기반 또는 참조 빈도수 기반)를 수행한다.

일반적인 캐쉬 대체 알고리즘은 캐쉬에 빈 슬롯이 없을 경우에만, 캐쉬 대체가 수행된다. 하지만 PDR 알고리즘은 캐쉬에 빈 슬롯이 존재하여도, 입력된 값이 인기 검색어일 경우에는 캐쉬 대체를 수행한다. 이는 캐쉬 내의 비인기 검색어를 일정 비율이상 유지하기 때문이다. 이를 위해, 항상 입력 값의 인기 검색어 여부를 검사하여, $(1-N_P/N_S) > O_{NP}$를 만족시키도록 한다. N_P/N_S은 캐쉬내의 인기 검색어 비율을 나타낸다. 따라서 $1-N_P/N_S$은 캐쉬내의 비인기 검색어의 비율을 나타내며, 사전에 정의된 O_{NP}보다 큰 값을 유지하도록 한다.

PDR는 크게 두 가지 척도(인기도와 참조 빈도수)를 이용하여 캐쉬 대체를 수행한다. 인기도를 이용하여 캐쉬 대체를 수행하는 경우는 다음과 같다. (1) 캐쉬 내에 빈 슬롯이 없을 때, 입력 값이 인기 검색어이고, $(1-N_P/N_S) \le O_{NP}$ 경우, (2) 캐쉬 내에 빈 슬롯이 존재 할 때, 입력 값이 인기 검색어이고, $(1-N_P/N_S) \le O_{NP}$ 경우이다. 두 가지 경우에는 현재 캐쉬에 적재되어 있는 값들 중에 최소 인기도를 가진 인기 검색어를 선발하여, 요청된 검색어의 인기도와 비교한 후 퇴거 여부를 결정한다.

```
Function Popularity-base replacement()
        Let EVICT be the slot number such that FP is minimum and not zero;
        If (FP of slot EVICT ≥ FP(X)) then
            Return R and exit;
        End if
        Flush slot EVICT into disk;
        Store (X, FP, 1, T, R) into slot EVICT and return R and exit;
End function
Function Reference-base replacement()
        Let EVICT be the slot number such that C is minimum and FP is zero;
        Flush slot EVICT into disk;
        Store (X, FP, 1, T, R) into slot EVICT and return R and exit;
End function
If X is already in the cache (say slot i) then
    Fetch slot i in the cache;
    If slot i is not stale then
        Increment C by one;
        Return R of slot i and exit;
    End if
    Construct the search result (R) of X by accessing search engines;
    Increment C by one;
    Store (X, FP, C, T, R) into slot i and return R and exit;
End if
Construct the search result (R) of X by accessing search engines;
If cache is not full then
    If FP(X) is not zero then
        If ((1-NP/NS) > ONP) then
            /* The ratio of non-popular queries is greater than threshold */
            Get the next available slot (let the slot number be FREE);
            Store (X, FP, 1, T, R) into slot FREE and return R and exit;
        End if
        /* The ratio of non-popular queries no greater than threshold */
        Popularity-base replacement();
    End if
    /* When X is a non-popular query */
    Get the next available slot (let slot number be FREE);
    Store (X, FP, 1, T, R) into slot FREE and return R and exit;
End if
/* When the cache is full */
If FP(X) is not zero then
    If ((1-NP/NS) > ONP) then
        /* The ratio of non-popular queries is greater than threshold */
        Reference-base replacement();
    End if
    /* The ratio of non-popular queries is no greater than threshold */
    Popularity-base replacement();
End if
/* When X is a non-popular query */
Reference-base replacement();
```

그림 2 PDR 알고리즘 의사 코드

두 번째로 참조 빈도수가 사용되는 경우는 다음과 같다. (1) 입력 값이 비인기 검색어이고, 캐쉬에 빈 슬롯이 존재하지 않을 경우, (2) 입력 값이 인기 검색어이고, $(1-N_P/N_S) > O_{NP}$ 경우이다. 두 가지 경우에는 현재 캐쉬에 적재되어 있는 비인기 검색어들 중에 참조 빈도수가 가장 적은 것을 퇴거시키고, 새롭게 요청된 검색어를 적재한다.

표 2 LRU, LFU, PDR 알고리즘의 비교

	LRU	LFU	PDR
대체 기준이 되는 값은 무엇인가?	시간	참조수	인기도 + 참조수
알고리즘 수행을 위한 사전 작업이 필요한가?	없음	없음	인기도 산출
시간이 지남에 따라 데이타의 우선 순위가 변하는가?	고정	고정	주기적으로 갱신

[표 2]는 LRU, LFU, PDR 알고리즘의 특성을 비교한 자료이다. LFU는 참조 빈도수를 이용하여 캐쉬 대체를 수행하는 알고리즘으로 과거의 참조 정보를 기억하지 않기 때문에, 참조 가능성이 높은 값을 제거하는 단점을 가진다. LRU는 캐쉬에 적재된 시간을 사용하여 대체를 수행하는 알고리즘으로, 적은 정보만을 가지고 캐쉬 대체를 수행하기 때문에 우수한 성능을 나타내지 못한다. 이에 반해 PDR에서 사용하는 인기도는 여러 검색 엔진들에서 추출한 광범위한 사용자 입력 경향을 반영한 값으로, LFU에서 사용하는 참조 빈도수와는 다르다. 또한 인기도뿐만 아니라 참조 빈도수를 사용하여 캐쉬 대체를 수행함으로써 LRU의 단점을 보완한다. 이는 다음 장에서 실험을 통하여 살펴본다. 하지만 PDR 알고리즘은 사전 작업(인기 검색어를 수집하여 중복을 제거한 후 인기도 산출)을 해야한다. 이는 PDR 알고리즘을 위한 필수 작업으로, 기존의 알고리즘에는 불필요

한 오버헤드라고 할 수 있다.

4. 성능 평가

성능 평가는 256MB의 주 메모리를 가진 펜티엄 III 650 PC에서 수행하였다. 성능 평가를 위해 상용 검색 엔진에서 인기 검색어를 수집하였으며, 상용 검색 엔진에서 생성된 실제 로그 파일의 질의를 추출하여 성능 평가를 수행하였다. 자바 개발 도구(Java Development Kit version 1.3; JDK1.3)를 이용하여 LRU, LFU, PDR 알고리즘을 구현하였다.

4.1 자료 수집

본 논문에서는 인기 검색어를 제공하는 여러 검색 엔진 중에서, 다수의 인기 검색어를 공시하는 세 개의 검색 엔진을 선정하여 인기 검색어를 수집하였다. 주 단위의 공시주기를 갖는 두 개의 검색 엔진과, 일 단위의 공시주기를 갖는 한 개의 검색 엔진을 선정하였으며(각각 100개/매주, 50개/매주, 50개/매일), 최대 인기 검색어 수를 나타내는 W는 300으로 정의하였다.

본 논문에서는 인기 검색어의 누적 효과를 알아보기 위해서, 세 가지 종류의 인기 검색어 목록을 생성하였다. 사전에 선정된 검색 엔진으로부터 1주, 2주, 4주의 수집 기간을 통하여 생성되었으며, 각각 144, 178, 194개의 인기 검색어가 수집되었다. 수집된 인기 검색어의 중복은 모두 제거하였다.

사용자 입력 값은 상용 검색 엔진에서 생성된 4일 분량의 로그 파일을 사용하였다. 로그 파일은 사용자 질의, 요청 날짜 및 시간, 사용자 주소(IP) 등의 내용을 포함하고 있으며, 그 중 유효한 사용자 질의 781,550건을 추출하였다. 추출 과정에서 사용자 입력 오류로 인한 잘못된 질의는 모두 제외하였다.

4.2 성능 평가 및 분석

본 절에서는 구현된 LRU, LFU, PDR 알고리즘의 성능을 비교 분석한다. PDR 알고리즘은 인기 검색어의 수집 기간에 따라 세 가지로 구분된다. 본 논문에서는 각각을 PDR-1, PDR-2, PDR-4라고 명명하며, 접미사인 1, 2, 4는 인기 검색어의 수집 기간을 주 단위로 나타낸 값이다.

본 논문에서 성능을 평가하기 위하여 캐쉬 적중률(hit ratio)과 성능 향상률(gain ratio)을 사용한다. 캐쉬 적중률은 일반적으로 많이 사용되는 캐쉬 성능 평가 요소이다. 본 논문에서는 캐쉬에 적재되었던 검색어들 중에 캐쉬 적중이 일어난 빈도수를 총 입력된 단일 어절 수로 나누어 산출한다. 성능 향상률은 LRU 알고리즘에 대비한 각 알고리즘의 성능 향상 수준을 나타낸다. 성능 향상률은 LFU 알고리즘과 PDR 알고리즘의 캐쉬 적중률을 LRU 알고리즘의 캐쉬 적중률로 나누어 산출되며, 백분율로 표시한다.

$$LFU_{gain-ratio} = \frac{LFU_{hit-ratio}}{LRU_{hit-ratio}} \times 100,$$

$$PDR_{gain-ratio} = \frac{PDR_{hit-ratio}}{LRU_{hit-ratio}} \times 100$$

본 논문에서는 PDR 알고리즘의 세 가지 관점에서 평가를 수행한다. 첫 번째는 캐쉬 크기에 따른 PDR 알고리즘의 성능 변화이다. 이를 위해 50, 75, 100, 125, 150, 180, 200으로 캐쉬 슬롯수를 증가시키면서 평가를 수행한다.

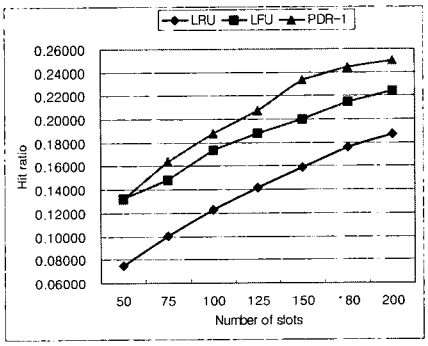

그림 3 LRU, LFU, PDR-1 알고리즘의 캐쉬 적중률

[그림 3]은 LRU, LFU, PDR 알고리즘의 캐쉬 적중률을 보인 그래프이다. 단, 비인기 검색어의 최소 적재 비율은 0%를 사용하였다. 그래프에서 볼 수 있듯이 슬롯수가 증가할수록 세 알고리즘 모두 캐쉬 적중률이 향상됨을 보였다. 슬롯수가 50개인 경우를 제외하고는, PDR 알고리즘이 가장 좋은 성능을 보였다. LRU 알고리즘의 경우 거의 일정한 비율로 적중률이 향상된 반면, LFU 알고리즘의 증가율은 조금씩 감소됨을 보였다. 대부분의 경우에 PDR 알고리즘이 LRU 알고리즘에 대비하여 6~8% 정도의 향상된 캐쉬 적중률을 보였다.

[그림 4]는 LFU와 PDR 알고리즘의 성능 향상률을 나타낸다. 슬롯수가 증가할수록 두 알고리즘 모두 성능 향상률이 감소하였다. 하지만, PDR 알고리즘에 비해 LFU 알고리즘이 더 급격히 성능이 감소된다는 것을 알 수 있다. 이와 같은 사실은 슬롯수가 증가할수록 (캐쉬 크기가 증가할수록) LRU 알고리즘의 성능이 차츰 향상됨을 의미한다[8]. 위 두 실험 결과에 근거하여, 캐쉬 크기는 캐쉬 성능에 매우 큰 영향을 준다는 사실을 알 수 있다.

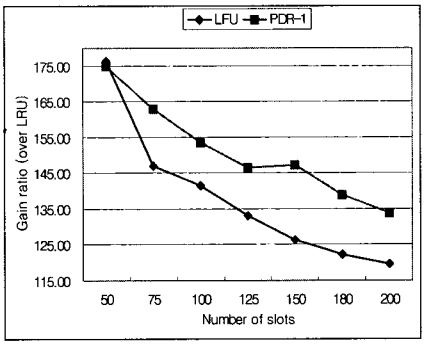

그림 4 LFU, PDR-1의 성능 향상률

두 번째로, 캐쉬 내의 비인기 검색어의 최소 적재 비율 변화에 따른 성능 평가를 수행하였다. 이를 위해 O_{NP}를 0%, 5%, 10%, 15%, 20%, 25%로 증가시키면서 성능을 측정하였다. [그림 5]는 비인기 검색어의 최소 적재 비율 변화에 따른 PDR 알고리즘의 캐쉬 적중률을 나타낸다.

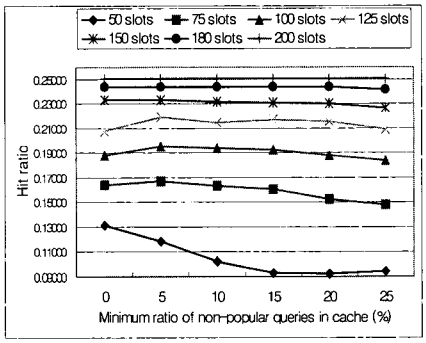

그림 5 PDR-1 알고리즘의 캐쉬 적중률

[그림 5]에서 볼 수 있듯이 50개의 슬롯을 제외하고는 비인기 검색어의 점유율이 최소 5%일 경우가 가장 좋은 적중률을 나타내었다. 슬롯수가 200일 경우에는 비인기 검색어의 점유율에 상관없이 동일한 적중률을 나타내었다. 이는 캐쉬 슬롯수가 수집된 인기 검색어보다 많음에 기인한다. 즉, 비인기 검색어의 점유율에 상관없이, 144개의 인기 검색어가 모두 캐쉬에 적재되고, 나머지 56개의 슬롯을 대상으로 비인기 검색어가 경쟁을 하는 현상이 발생한다. 이와 같은 현상은 슬롯수 180과 슬롯수 150의 경우에도 부분적으로 나타난다.

[그림 6]은 PDR 알고리즘의 성능 향상률을 나타낸다. 대부분의 경우에 비인기 검색어의 최소 비율이 5% 이상으로 증가할 경우 성능 향상률은 대체적으로 감소하는 경향을 보였다. 이는 과다한 비인기 검색어의 캐쉬 적재는 성능 향상에 도움을 주지 못함을 나타낸다.

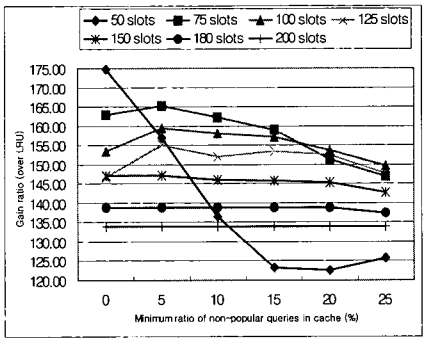

그림 6 PDR-1 알고리즘의 성능 향상률

세 번째로, 수집된 인기 검색어 누적 여부가 캐쉬 성능에 미치는 영향에 대하여 평가하였다. 이를 위해 1, 2, 4주 단위로 나누어 수집된 인기 검색어를 사용하여 PDR 알고리즘의 성능 평가를 수행하였다.

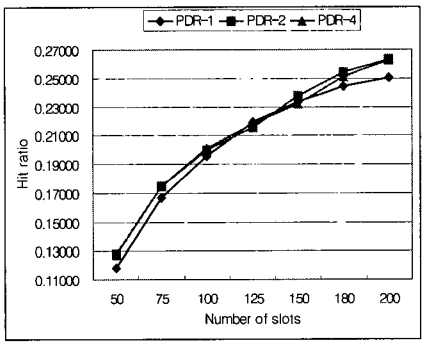

그림 7 PDR-1, PDR-2, PDR-4의 캐쉬 적중률

[그림 7]은 PDR-1, PDR-2, PDR-4 알고리즘의 캐쉬 적중률을 나타낸 그래프이다. 단, 비인기 검색어의 최소 적재 비율은 5%를 사용하였다. 대부분의 슬롯에서 각 알고리즘의 캐쉬 적중률이 거의 유사함을 알 수 있다. 슬롯수 50일 경우와 슬롯수 200일 경우를 보면 PDR-2와 PDR-4가 PDR-1에 비해 약 1% 정도 높은 캐쉬 적중률을 보였을 뿐, 나머지 슬롯은 거의 동일한 성능을 보였다. 이와 같은 결과는 인기 검색어의 누적은 캐쉬 적중률 향상에 별다른 도움을 주지 못함을 나타내며, 1주 또는 2주 동안의 기간이 적절한 누적 기간으로 사료된다.

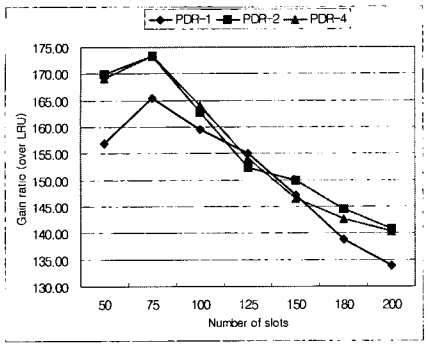

그림 8 PDR-1, PDR-2, PDR-4의 성능 향상률

[그림 8]은 PDR-1, PDR-2, PDR-4의 성능 향상률을 보인 그래프이다. [그림 7]과 동일하게 5%의 최소 비인기 캐쉬 적재 비율을 사용하였다. 슬롯수 125를 보면, 세 가지 알고리즘이 거의 유사한 성능 향상률을 나타냄을 알 수 있다. 이는 수집된 세 종류의 인기 검색어 목록 상위 120개가 거의 유사함에 기인한다. 이들 상위 120개의 인기 검색어는 캐쉬에 적재된 후 퇴거되지 않고 계속해서 캐쉬에 잔여한다.

위의 세 가지 관점의 캐쉬 성능 평가를 통하여 다음과 같은 결론을 얻을 수 있다. (1) 캐쉬 크기는 캐쉬 성능에 영향을 미친다. 세 알고리즘 모두 캐쉬 크기가 클수록 우수한 성능을 나타내었다. (2) 인기 검색어가 캐쉬에 모두 적재되었을 경우에 가장 좋은 성능을 나타낸다. (3) 과다한 비인기 검색어의 적재는 캐쉬 성능 향상에 도움을 주지 못한다. 캐쉬 내의 비인기 검색어의 최소 적재 비율이 5%을 기점으로 캐쉬 대체 성능이 감소하는 경향을 보였다. (4) 인기 검색어의 누적은 캐쉬 성능에 큰 영향을 미치지 못한다.

5. 결론 및 향후 과제

본 논문에서는 메타 검색 엔진을 위한 인기도 기반의 캐쉬 대체 알고리즘에 대하여 기술하였다. 또한 인기 검색어의 수집 방법과 인기도 산출법에 관하여 언급하였다. PDR 알고리즘의 의사 코드를 서술하였으며, 다방면의 성능 평가를 수행하였다. 본문에서 서술한 바와 같이 대부분의 경우에 PDR 알고리즘이 우수한 성능을 나타내었다.

본 논문에서는 주 메모리만을 사용한 캐쉬를 고려하였다. 메타 검색 엔진은 주 메모리뿐만 아니라, 디스크를 캐쉬로 활용할 수 있다. 주 메모리와 디스크를 사용할 때의 차이점은 단지 디스크가 주 메모리보다 많은 용량의 자료를 캐쉬에 적재할 수 있다는 것이다. 본 논문에서 제안한 알고리즘은 주 메모리를 고려하여 설계되었지만, 디스크를 사용할 때에도 충분히 적용할 수 있다고 사료된다.

본 논문에서는 외부에서 수집된 인기 검색어만을 이용하여 성능평가를 수행하였다. 현재 상용되고 있는 검색 엔진들의 대다수는 인기 검색어 추출 정책(예를 들면, 선정적인 단어 제외)에 의하여 사용자에게 공시하는 인기 검색어를 선별(filtering)한다. 따라서, 실제로 많이 입력되는 검색어들이 인기 검색어로 채택되지 않은 경우가 많다. 만일 내부적으로 사용자 입력 검색어를 받아 정제를 하지 않고 인기 검색어 목록을 만들어 PDR 알고리즘에 적용한다면, 현재보다 더 좋은 성능을 가질 것이라 사료된다.

향후 과제로는 외부 검색 엔진뿐만 아니라 내부적으로 인기 검색어를 수집하여 인기도 산출 방법을 확장한다. 또한 수집 대상이 되는 검색 엔진에 가중치(weight)를 부여하여 캐쉬 성능을 향상시키는 연구가 요구된다.

참 고 문 헌

[1] S. Lawrence, and C. Giles, "Accessibility of Information on the Web," Nature, 400: 107-109, 1999.

[2] A. Scime, and L. Kerschberg, "WebSifter: An Ontology-based Personalizable Search Agent for the Web," Proceedings International Conference on Digital Libraries: Research and Practice, Kyoto Japan, 2000.

[3] J. Cheong, and S. Lee, "A Boolean Query Processing with a Result Cache in Mediator Systems," Advances in Digital Libraries, 2000, Proceedings, IEEE, 2000, pages 218-227, 2000.

[4] S. Dar, M. Franklin, B. Jonsson, D. Srivastava, and M. Tan, "Semantic Data Caching and Replacement," Proceedings of the 22nd VLDB Conference, India, 1996.

[5] B. Chidlovskii, C. Roncancio, and M. Schneider, "Semantic Cache Mechanism for Heterogeneous Web Querying," WWW8, 1999.

[6] D. Lee, and W. Chu, "Semantic Caching via Query Matching for Web Sources," Proceedings of the 8th International Conference on Information Knowledge management, pages 77-85, 1999.

[7] J. Robinson, and M. Devarakonda. "Data Cache Management Using Frequency-Based Replacement," In Proc. of the 1990 ACM SIGMOD Int'l Conf. on management of data pages 134-142, August 1990.

[8] E. O'Neil, P. O'Neil, and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering," Proc. 1993 ACM SIGMOD, 1993.
[9] T. Johnson, and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm," Proc. of the 20th International Conference on VLDB, pages 439-450, 1994.
[10] D. Lee, J. Choi, J. Kim, S. Noh, S. Min, Y. Cho, and C. Kim, "On the Existence of a Spectrum of Policies that Subsumes the Least Recently Used (LRU) and Least Frequently Used (LFU) Policies," Proc. of the International Conference on Measurement and Modeling of Computer Systems, 1999.
[11] S. Lee, J. Hong, and L. Kerschberg, "A Popularity-driven Caching Scheme for Metasearch Engines: An Empirical Study," Springer-verlag Lecture Notes in Computer Science, Vol. 2113, pages 877-886, 2001.

홍 진 선

1999년 숭실대학교 전자계산학과(학사). 2001년 숭실대학교 대학원 컴퓨터학과(석사). 2001년 ~ 현재 (주)메타웨이브 소프트웨어 개발팀 연구원. 관심분야는 인터넷 데이타베이스, 정보검색, XML, 유/무선 통합

이 상 호

1984년 서울대학교 전산공학과(학사). 1986년 미국 노스웨스턴대 전산학과(석사). 1989년 미국 노스웨스턴대 전산학과(박사). 1990년 ~ 1992년 한국전자통신연구원, 선임연구원. 1999년 ~ 2000년 미국 조지메이슨 대학교 소프트웨어 정보공학과 교환 교수. 1992년 ~ 현재 숭실대학교 컴퓨터학부 부교수. 관심분야는 인터넷 데이타베이스, 데이타베이스 시스템 성능평가 및 튜닝