# 계층적 캐슁 기법을 이용한 대용량 웹 검색 엔진의 구현

임 성 채
동덕여자대학교 컴퓨터전공
sclim@dongduk.ac.kr

# Implementation of a large-volume Web search engine using the multi-level data caching

Sung Chae Lim
Dept. of Computer Science, Dongduk Women's Univ.

요 약

논문에서는 6,000만개 웹 페이지의 색인 데이터에 대해 일 600만 질의를 처리하는 대용량 웹 검색 시스템을 위해 구현된 계층적 캐슁 기법을 소개한다. 논문에서 설명된 시스템 구조 및 알고리즘은 실제 상용 웹 검색 엔진에서 구현되고 운영 결과를 통해 그 유용성이 입증된 것들로서, 구현된 시스템과 유사성을 가지는 대용량 데이터 처리 시스템에 적용 가능할 것이다. 본 논문에서는 기존에 많이 소개되지 않았던 웹 검색 엔진의 운영 절차 및 웹 질의 처리 시스템에 대한 기술적 내용이 기술되었으며, 기술된 내용을 통해 웹 검색 엔진에 대해 보다 정확한 이해가 가능해 질 것이다.

## 1. 서 론

지난 수십 년간 Web을 이용한 정보 저장 및 정보 공개가 확대됨에 따라 웹 검색엔진의 중요성과 그 사용 빈도도 함께 증가하고 있는 추세이다[4,5]. 일반적으로 웹 검색엔진이라고 함은 크롤러(crawler)라 불리는 웹 로봇을 사용하여 웹 상에서 접근되는 HTML 문서를 자동수집하고 이를 색인한 후, 사용자의 웹 검색 질의를 실시간으로 처리해 주는 검색 시스템을 의미한다. 이런 웹 검색엔진은 HTML 문서에 삽입되어 있는 HTML 태그들이 삭제된 문서 텍스트로부터 stop-word를 제외한 키워드를 추출하고 이를 이용하여 역파일 형태의 색인 파일을 생성하는 것이 일반적이다[1,2,3].

웹 검색 엔진에서 색인한 웹 문서의 수가 빠르게 증가함에 따라 웹 검색 결과에 나타나는 검색 결과 순위(rank)의 결정 방법이 매우 중요시 되었는데, 이는 질의를 만족하는 웹 문서의 수가 많고 출력되는 검색 결과는 웹 화면의 특성상 화면 제약이 있고 표현상의 융통성이 떨어지기 때문이다. 결국 기존 정보 검색(IR: information retrieval)에서 자주 사용된 키워드의 문서 내 빈도나 분포 특성에 따라 결정되는 랭킹 기법은 웹 검색의 경우 원하는 검색 만족도를 제공할 수 없다는 문제점이 있다. 이런 문제점을 해결하기 위해 단순한 텍스트 정보만이 아니라 웹에 존재하는 링크 연결을 분석하여 순위 결정에 반영하는 기법들이 제안되었고, Google.com에서도 이런 방식을 구현하여 기존의 웹 검색 엔진과 차별화된 검색 품질을 보인 바가 있다[3].

웹 검색엔진에 대한 기존 연구에서는 주로 웹 검색 서비스를 구현함에 있어 웹 검색에 사용할 데이터를 준비하는 단계에 대한 기술적인 문제를 다루고 있다고 할 수 있다. 즉, 웹 문서의 효과적인 수집[6,8,10], 링크 분석[9,11], 그리고 역파일 생성에 관련된 기법[1,2]들에 대해서 주로 다루고 있다. 하지만 검색 엔진 서비스를 실현하기 위해서는 이런 데이터 준비 과정 외에 생성된 웹 검색 데이터를 사용하여 실제 사용자 질의를 처리하는 질의 처리 시스템의 구현이 필수적이다. 하지만 이런 웹 검색 엔진의 질의 처리 시스템에 관련된 기술적인 사항이나 문제 해결 방식에 대해서는 연구결과가 미미한 것이 현실이다. 관련된 기술적인 문제들은 웹 검색 시스템의 운영 및 구현을 통해 생각할 수 있는 문제이기에, 연구 기관에서는 이에 대한 경험을 얻기가 현실적으로 어렵다고 생각된다.

본 논문에서는 실제 상용 웹 검색엔진을 구현하고 운영한 경험을 바탕으로, 웹 검색 질의 처리 시스템의 구조와 중요한 기술 요소인 캐슁 기법을 소개한다. 소개되는 시스템은 주로 한국어 웹 문서를 수집하여 검색하는 시스템이며, 수집되는 웹 문서의 개수는 전체 6,000만개가 넘는 수준이다. 또한, 시스템의 처리 용량은 하루 600만에 건 이상의 웹 질의를 처리할 수 있는 수준이다. 구현된 시스템은 독자적인 웹 사이트에서 입력되는 질의를 처리할 뿐만 아니라, 외부 포털 사이트에서 웹 검색 질의를 입력했을 때 이를 처리해 주는 서비스인 웹 검색 ASP(Application Service Provider)의 형태로 운영되었다.

웹 검색 ASP 서비스는 일정한 수준의 질의 처리의 양과 질의 처리 응답 시간을 제공해야 하는 유료 서비스이므로 운영되는 웹 검색엔진은 안정성 및 응답 시간이 보장되어야 한다. 이런 점을 고려하여 전체 시스템이 고안되었으며, 이미 처리한 웹 질의 결과를 저장하여 재사용하는 캐슁 기법이 적용되었다.

논문에서 기술하는 기술적인 문제들은 웹 검색 엔진과 같이 대용량의 데이터를 처리해야만 하는 다양한 시스템에 적용될 수 있을 것이라 생각한다.

논문의 구성은 다음과 같다. 2장에서는 구현된 웹 검색엔진의 질의 처리 시스템의 전체 구조 및 기능상의 특성에 대해 설명한다. 3장에서는 질의 처리 시스템의 성능 향상을 위해 고안된 계층적 데이터 캐싱 기법에 대해 기술한다. 마지막 4장에서 결론과 함께 성능상의 이슈를 간단히 기술한다.

## 2. 웹 검색 엔진

### 2.1 웹 검색을 위한 색인 작업

웹 검색 서비스를 위해서는 크롤러라는 웹 문서 자동 수집기를 통해 웹상에 존재하는 웹 문서를 수집해야 한다. 웹 문서 수집을 할 때, 크롤러는 HTTP 프로토콜에 따라 웹 서버로부터 자신이 수집하고자 하는 웹 문서의 Url을 요청하는데, 이때 요청되는 Url은 이전 웹 문서 수집 시에 저장해 두었던 Url 집합과 현재의 문서 수집 과정에서 새로 발견된 Url 집합에서 선택된다.

크롤링을 수행하는 시스템은 네트워크 프로그램을 통해 HTTP 요청을 수행하는 프로세스, 현재까지 수집된 웹 문서의 Url을 관리하는 프로세스, 요청될 Url을 결정하고 해당 Url을 HTTP 요청 프로세스에 보내는 프로세스, 수집한 웹 문서를 저장하며 문서 별로 유일한 ID를 부여하는 프로세스로 나눠 볼 수 있다. 이들 프로세스는 성능 향상을 위해 고속 LAN으로 상호 연결된 여러 개의 서버 클러스터에서 동작한다.

구현된 시스템은 6,000만 페이지 이상을 4-5일 정도에 걸쳐 수집할 수 있도록 분산 구조로 구현되었으며, 하나의 프로세스 안에 많게는 64개의 쓰레드(thread)가 동시에 동작하는 멀티쓰레딩 구조를 채택한다. 6,000만 개의 웹 문서를 4일에 걸쳐 수집하는 경우, 초당 약 174개의 웹 문서가 수집되어야 하며 이를 위해 HTTP 요청 서버는 4대의 서버 클러스터로 운영된다. 이와 같은 수집 속도는 구현된 시스템 최대 속도의 50% 정도이며, 최대 속도로 문서 수집을 하지 않는 이유는 상대편 웹 서버의 부하 문제를 고려해야 하기 때문이다. 즉, 각각의 상대 웹 서버에 대해 최소한의 시간 간격을 두고 HTTP 요청을 보내야 하고, 이를 지키기 위해서는 크롤링 시스템의 최대 속도로 문서 수집을 수행할 수는 없다.

문서 수집 작업이 종료된 후에는 색인 작업과 다음번 문서 수집이나 색인 작업을 위한 후 처리 작업과 함께 실제 색인 작업이 진행된다. 본 논문에서는 지면 제약상 색인 작업에 대해서만 기술하기로 한다.

색인 작업은 크게 두 개의 작업으로 분리해 진행한다. 먼저 하나는 널리 알려진 것처럼 역파일 형태의 색인 파일을 만드는 과정이다. 이 작업은 수집한 웹 문서에서 HTML을 삭제한 상태에서 문서내의 키워드를 추출하는 작업이 주이다. 이때 추출되는 키워드 정보에는 해당 키워드를 포함한 문서 ID, 키워드의 문서 내 오프셋, 문서 내 중요도를 추정할 있는 속성등이 함께 저장된다. 이런 정보들은 각 키워드 별로 저장되며 이를 통해 키워드에 대한 역파일을 구성한다.

이런 전통적인 역파일 색인에만 의존하여 웹 검색 순위를 결정하는 경우 사용자가 원하는 검색 정확성을 제공할 수 없다. 이는 웹 검색엔진이 색인하는 문서의 양이 매우 많고 검색 순위를 높이고자 웹 문서를 의도적으로 편집하는 경우가 많기 때문이다. 이러한 문제점을 보안하고자 웹 문서간의 링크 연결 정보를 분석하여 웹 문서의 중요도를 정하고자 하는 아이디어가 있었으며, 이를 성공적으로 구현한 것이 Google사의 PageRank 결정 알고리즘이라 할 것이다[9,7,11].

구현한 웹 검색엔진 역시 웹 문서간의 링크 분석 결과에 따라 각 웹 문서의 중요도를 부여할 수 있다. 웹 링크 분석을 통한 중요도 결정은 문서내의 키워드와는 아무런 관계가 없는 작업이기에 역파일을 생성하는 색인 작업과 병렬적으로 진행할 수 있다. 즉, 역파일 생성하는 서버 클러스터와 링크 분석을 통해 각 페이지 별로 중요도를 결정하는 작업을 수행하는 서버 클러스터가 서로 분리되어 운영된다. 현재 구현된 웹 검색엔진의 경우에는 이런 링크 분석을 통한 결과가 역파일 생성 시점에 앞서 나오게 되며, 여기서 얻은 웹 문서 페이지별 중요도 수치는 이후 생성되는 역파일 색인의 한 필드로 삽입된다.

링크 분석을 통한 각 웹 문서 중요도 결정 방식은 여러 논문을 통해 기본적인 이론은 이미 널리 알려져 있으며, 구현 방식에는 서로 간의 차이가 존재할 수 있다. 또한 주제를 고려한 링크 분석 기법[11]의 경우는 다양한 구현 방식이나 알고리즘이 제안될 수 있는 분야이다.

구현된 시스템은 링크 분석 시에 여러 대의 서버를 이용하여 병렬 계산을 수행한다. 링크 분석에 따른 계산에는 기본적으로 $N \times N$의 행렬에 대한 곱셈 연산이 포함되며, 정수 $N$은 분석되는 전체 웹 문서수와 일치한다. $N$이 증가함에 따라 계산의 양이 매우 커질 뿐만 아니라, $N \times N$ 행렬은 4 바이트 부동 소수점 값을 원소로 하는 2차원 행렬이므로 그 데이터 공간 사용도 매우 크다. 행렬 곱셈을 할 때 해당 행렬을 메모리로 모두 올려야 하기 때문에 $N$의 값이 매우 큰 상황에서는 그 계산이 현실적으로 불가능하다. 물론 sparse matrix의 개념으로 행렬의 크기를 좀 더 작게 관리할 수 있지만 $N$의 크기가 매우 큰 경우에는 계산 속도가 느려지고 관리에 어려움이 있다.

이런 문제점에 대해 대용량의 웹 문서에 대해 링크 분석 계산을 수행하는 시스템은 서버 클러스터를 사용하여 병렬 계산을 수행하는 하는 것이 일반적이다. 즉, 서버 클러스터에 속한 서버의 개수를 S라 할 때, $N \times N$의 행렬을 수평적으로 나눠 S개의 $\frac{N}{S} \times N$의 행렬로 나눈 후, 병렬 계산을 수행한다. 이를 위해 S개의 서버가 자신에게 할당된 하나의 $\frac{N}{S} \times N$의 행렬에 대한 페이지 중요도 계산을 수행한다. 각 서버는 병렬적으로 행렬 계산을 수행하며, 계산된 결과는 다음번 계산에 다시 반복적으로 사용된다. 이러한 행렬 계산은 각 웹 페이지의 중요도 값이 일정한 값들로 점차 수렴하여 일정한 수준의 평형 상태에 이를 때까지 반복된다. 이런 행렬 계산이 평형 상태로 가

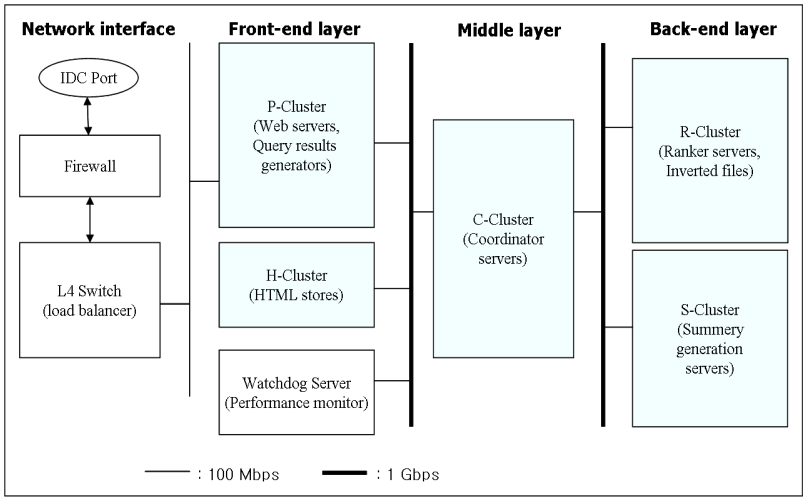


그림 1. 웹 질의 처리 시스템의 구성도.

기 위해 웹 페이지 링크를 표현하는 행렬 값을 조정할 필요가 있다.

이렇게 계산된 각 웹 문서별 중요도는 웹 문서간의 상대적인 중요도를 나타내는 중요한 척도가 되며, 이 중요도 값은 사용자의 입력 질의와는 무관하게 각 웹 문서가 가지는 고유의 값이다. 이런 값은 사용자 질의에 포함된 키워드의 수가 적을 때 그 유용성이 크다. 예를 들어 사용자 질의가 w1, w2, w3, w4의 네 개의 키워드로 구성된 경우를 생각해 본다.

이 경우는 단어의 수가 많은 경우이므로 이들 단어를 모두 포함한 웹 문서의 개수가 적을 뿐만 아니라 단어 간의 근접도나 단어의 문서 내 중요도 등을 반영하는 등 문서간의 중요도를 측정할 요소들이 많다고 할 수 있다. 하지만 만약 w1이란 단일 키워드로 이루어진 질의어의 경우라면 질의어를 만족 시키는 문서 수가 많을 뿐만 아니라 w1이라는 하나의 단어만으로 웹 문서간의 중요도를 결정하기 매우 어렵다. 이때 유용하게 사용되는 것이 링크 분석을 통한 웹 문서의 중요도이다. 이러한 사실은 구현한 웹 검색 엔진의 운영을 통해서도 알 수 있었고 Google과 같은 웹 검색엔진을 통해서도 입증되었다고 할 수 있다.

### 2.2 구현된 질의 처리 시스템

앞 절에서 웹 문서 수집 및 색인 작업에 대해 간단히 기술하였다. 이렇게 생성된 웹 데이터와 색인 데이터를 가지고 사용자로부터 입력되는 웹 검색 질의를 실시간으로 처리하는 시스템을 웹 질의 처리 시스템이라 칭하기로 한다. 설명할 웹 질의 처리 시스템은 일 600만 질의를 처리하도록 구현되었으며 이를 위해 LAN 상의 분산 처리 구조를 채택하고 있다.

웹 질의 처리 시스템을 구현함에 있어 중요한 요소는 질의 처리의 평균 시간 및 시스템 안정성이다. 웹 검색 ASP를 위해서는 피크 타임 때에 일정한 수준의 평균 응답시간을 유지해야 하며, 웹 검색엔진 운영 시에 발생하기 쉬운 서버의 하드웨어 고장에 대해서 전체 시스템의 장애는 발생하지 않아야 한다. 이를 고려한 분산 시스템이 필요하며 fail-over 기능을 고려하여 동일한 기능을 하는 다수의 서버로 하나의 논리적 서버 군으로 구성하는 서버 클러스터 기술이 요구된다.

그림 1은 구현된 웹 질의 처리 시스템의 구성도이다. 시스템은 네트워크 인터페이스 뒷단에 존재하여 전체 3개의 레이어로 구성되며, 각 레이어에 속한 서버들은 서로 다른 LAN 연결을 이용하여 통신한다. 그림에서 굵은 라인으로 표시된 LAN은 1 Gbps의 속도를, 그렇지 않은 라인은 100 Mbps LAN을 나타낸다. 서버들 간의 통신 대역폭 경쟁을 최소화하기 위해 LAN을 분리하였으며, 서버 간 이동되는 데이터의 크기가 큰 레이어 간에는 보다 고속의 LAN을 사용한다.

구현된 서버 중 유일하게 단일 서버로 운용되는 watchdog 서버는 처리된 누적 질의 개수, 시간 구간별 질의 처리 개수, 시간 구간 별 질의 응답시간, 서버의 정상 작동 여부 등을 실시간으로 체크하며, 이상이 발견되는 경우 이를 알리는 메시지를 관리자에게 자동 발송하는 역할을 수행한다. 이외의 서버들은 CPU나 하드디스크 오류와 같은 하드웨어 오류에 대비하여 동일한 작업을 수행하는 여러 개의 서버를 동시에 운영한다.

구현된 시스템의 클러스터 기능을 간단히 요약하면 표 1과 같다.

| 클러스터 | 클러스터 동작 요약 |
|---|---|
| P-Cluster | 검색 결과 화면의 생성 및 웹 서버를 통한 사용자와의 통신을 수행. |
| R-Cluster | 색인 데이터를 사용하여 join 연산과 질의어에 대한 랭크 값 계산. 색인데이터는 여러 대의 서버에 분할 저장됨. |
| S-Cluster | 검색 결과에 보이는 웹 문서 요약 데이터를 생성. |
| C-Cluster | R-클러스터와 S-클러스터 서버와의 네트워크 연결을 담당. 이 두 클러스터로부터 결과를 수집하여 최종 검색 결과를 생성하며 이를 P-클러스터로 전달함. |
| H-Cluster | 크롤링된 HTML 문서를 저장하고 사용자 요청시 저장된 HTML 문서를 하이라이팅하여 보여줌. |

표 1. 서버 클러스터의 기능 요약.

## 3. 계층적 데이터 캐싱 기법

본 장에서는 질의 처리 시스템의 속도를 크게 향상시키기 위해 디자인되고 구현된 계층적 캐싱 기법에 대해 기술한다.

### 3.1 데이터 캐싱의 필요성

웹 질의 처리에는 사용되는 CPU 시간과 디스크 대역폭의 양의 매우 큰 것이 특징이다. 디스크 자원의 경우는 그림 1의 R-클러스터와 S-클러스터에 속한 서버에서의 사용량이 매우 크다고 하겠다. R-클러스터의 경우는 키워드에 대한 조인 연산을 수행하기 위해 각 키워드 별로 보통 40-50KB 정도의 색인 데이터를 읽어야 하고, 랭킹 계산 시에는 이보다 10배 이상 되는 데이터 공간에서 디스크 상에 흩어져 존재하는 데이터 블록을 읽어 들어야하는 어려움이 있다. 이 경우는 디스크 탐색시간이 커짐으로 해서 사용되는 디스크 자원이 매우 커질 수 있다.

또한 S-클러스터의 경우는 각 질의어에 대한 결과 화면을 위해, 100-120 여개의 디스크 블록을 읽어 들이는 작업을 한다. 각 디스크 블록에는 웹 문서에서 추출한 텍스트 데이터, 문서의 타이틀, Url 정보들이 저장되어 있으며, S-클러스터는 읽어들인 텍스트 데이터 안에서 키워드들이 위치한 구역을 추출하여 C-클러스터로 보내는 역할도 수행한다. 이런 텍스트 구역 추출은 이후 검색 결과 출력 시에 보이는 하이라이팅을 위한 것이다. 하나의 질의어에 최소 100개가 넘는 디스크 블록을 읽어야 하고, 이들 블록은 디스크 상에 비연속적으로 위치하기 때문에 사용되는 디스크 대역폭의 크기가 매우 클 수 밖에 없다. 일반적인 예측으로는 R-클러스터에서 사용하는 디스크 대역폭이 가장 클 것이라 생각될 수 있겠지만 실제 환경에서는 S-클러스터에서 사용되는 디스크 대역폭이 가장 크다.

CPU 사용 측면에선 R-클러스터에서의 join 연산 및 랭킹 계산, S-클러스터에서의 요약 데이터 생성을 위한 하이라이팅 구역 선택, C-클러스터에서의 검색 결과 수집 및 중복 제거 작업에 CPU 사용이 크다고 할 수 있다. C-클러스터의 질의 처리 방식을 설명하면 다음과 같다. 먼저 C-클러스터는 웹 질의를 R-클러스터로 보낸다. R-클러스터에는 색인 파일이 분할(partition)되어 분산 저장되어 있으며, 하나의 질의 처리를 위해는 동시에 여러 대의 R-클러스터 서버로 동작한다. 각각의 R-클러스터의 서버는 해당 질의어에 대해 join 연산 및 랭킹 작업을 수행한 뒤, 웹 문서 ID와 랭크값으로 이루어진 결과 리스트를 C-클러스터로 보내고, C-클러스터에서 이를 수집하여 랭크값에 따라 다시 정렬한다. 정렬된 랭크값이 높은 웹 문서에 대해서는 문서 요약 데이터를 S-클러스터에게 요청하며, 이때에도 여러 대의 S-클러스터 서버에 나눠서 요청한다. 이후 S-클러스터로부터 받은 문서 요약 데이터를 이용하여 중복된 웹 문서를 찾아내어 삭제한다. 이런 삭제 연산은 사용자가 느끼는 검색 품질에 많은 영향을 준다.

하나의 질의를 처리하기 위해서는 매우 많은 네트워크 연결과 디스크 및 CPU 비용이 필요하며 이들 자원의 사용을 최소로 할 수 있는 데이터 캐싱 기법이 반드시 필요하다. 이런 데이터 캐싱 기법을 통해 시스템 운영에 필요한 전체 서버 수와 질의 처리 시간의 평균값을 크게 줄일 수 있다.

### 3.2 구현된 계층적 데이터 캐쉬

구현된 데이터 캐쉬는 그림 1에 보인 분산 형태의 클러스터에 나뉘어 설정되며, 그 특성에 따라 크게 3단계로 구분된다. 3단계의 계층은 아래와 같이 나눠지며 구현한 시스템에서는 1단계의 캐쉬 데이터가 가장 높은 캐쉬 적중률을 보였다.

- 1단계 캐쉬: 메모리 저장 캐쉬. P-클러스터에 위치. 검색 결과의 1, 2번째 결과를 보이는 HTML 문서를 저장.
- 2단계 캐쉬: 디스크 저장 캐쉬. C-클러스터에 위치. 중복 웹 페이지를 삭제한 후의 상의 랭크 웹 페이지 정보 및 요약 데이터가 저장됨.
- 3단계 캐쉬: 디스크 저장 캐쉬. C-클러스터에 위치. 랭크 값으로 소팅된 결과 문서 ID 값이 저장됨.

위의 1단계 캐쉬는 주기억 장치에 저장되며 웹 서버가 동작하는 P-클러스터 서버의 주기억 장치를 이용한다. 이들 서버의 주 메모리 공간은 2 GB 크기를 가지며, 이중 70%인 1.4 GB의 공간이 캐쉬 저장에 사용된다. 1단계 캐쉬 공간에는 웹 검색 결과를 표현하는 HTML 문서 자체가 저장되며, LRU 알고리즘의 변형을 캐쉬 재배치(replacement) 알고리즘으로 이용된다. 검색 결과를 표현하는 HTML 페이지 데이터는 P-클러스터 단에서 생성되며, 이는 C-클러스터 단으로부터 받은 웹 문서 요약 데이터를 이용하여 생성되는데, 여기서 생성된 HTML 문서에는 검색 결과에 보이는 Url 정보, 하이라이팅 결과, 타이틀 내용, 화면 표현을 위한 추가적인 HTML 태그 등이 포함된다. 이런 HTML 문서는 1.4 GB 크기의 메모리를 일정한 크기로 분할한 공간에 저장되며, 해당 검색 결과 페이지를 나타내는 Url을 키로 하여 저장된 HTML 문서를 찾는다.

2단계와 3단계 캐쉬 데이터와 모두 C-클러스터에 속한 서버에 위치한 캐쉬 데이터이며 디스크에 저장된다. C-클러스터의 서버는 S-클러스터 서버들로부터 받은 요약 데이터를 이용하여 중복 페이지로 판단되는 페이지를 삭제한다. 이처럼 중복 제거된 후의 결과에서 상위 랭크 값을 가지는 50개 웹 문서에 대한 요약문 데이터를 2단계 캐쉬 데이터로 저장한다. 이런 데이터는 하나의 웹 검색 결과 페이지에 10개의 검색 결과를 출력할 경우 연속적인 5개의 결과 페이지를 생성할 수 있음을 의미한다. 실제 검색 결과의 85% 이상이 검색 결과 앞선 3개의 페이지를 넘어가지 않기 때문에 이 정도의 캐쉬 데이터만으로도 충분하다고 할 수 있다.

3단계 캐쉬는 C-클러스터 서버가 R-클러스터 서버로부터 받은 랭크값 순으로 정렬된 웹 문서 ID 리스트이다. 이 중에서 상위 랭크된 일부 웹 문서에 대해서는 2단계 캐쉬 데이터에도 중복되어 있다고 할 수 있으며, 3단계 캐쉬 데이터의 경우는 2단계 캐쉬 데이터를 벗어난 하위 검색 결과에 대한 캐쉬 데이터라고 할 수 있다. 3단계 캐쉬 데이터는 요약문 데이터를 포함하고 있지 않기 때문에, 해당 웹 문서의 요약문 데이터를 얻기 위해서는 S-클러스터로 요청을 보내야 하며, 2단계 캐쉬 데이터가 제공하는 검색 결과 범위를 벗어나는 경우에는 중복 제

거를 수행하지 않는다. 이는 하위 검색 결과에는 중복 웹 페이지가 보일 수 있음을 의미하며, 이를 통해 S-클러스터 단의 지나친 디스크 사용을 방지할 수 있다.

### 3.3 계층적 캐쉬 관리 알고리즘

본 절에서는 앞서 기술한 캐쉬 데이터를 참조하고 생성하기 위한 알고리즘을 설명한다. 기술된 알고리즘은 특정 클러스터 단에서만 수행되는 알고리즘이 아니고, 전체 질의 처리 시스템의 범위에서 수행되는 알고리즘이다.

그림 2는 캐쉬 관리 알고리즘이 수행되는 과정을 표현하고 있다. 아래 알고리즘에는 지면 제약으로 재배치 알고리즘은 생략하였다. 또한 실제 구현에서는 C-클러스터에서 관리되는 캐쉬 데이터가 2가지 계층으로 관리됨에도 그림 2에서는 이를 별도로 나눠 기술하지는 않았다. 재배치 알고리즘은 LRU 알고리즘의 변형을 기본 알고리즘으로 하며, 각 계층별 캐쉬 데이터의 특성에 따라 서로 다른 알고리즘이 적용된다.

```
/* in P-Cluster */
1. Get the Url that requests for a query result page.
2. Check whether a cached page exists for this requested Url.
3. If existence is true, then return the cached page to the user and
     quit.
   Otherwise, perform the steps below
4. Parse the requested URL so as to extract keywords from the Url.
5. The packet containing the keywords of the user query is sent to
     a server in C-Cluster. Wait for the result from the server.
/* in C-Cluster */
6. Check whether a cached data exists for the received keywords.
7. If that exists, return the cached data toward P-Cluster.
   Otherwise, perform query processing using the servers in R-
     Cluster and S-Cluster. Save the query result into the disk
     managing the cache data.
8. Return the query result toward P-Cluster and quit.
/* in P-Cluster */
9. Save the query result received from C-Cluster into the cache data
     storage.
10. Return the query result to the user and quit.
```

그림 2. 캐쉬 관리 알고리즘.

웹 검색의 경우는 이전 기간 동안 처리된 질의어 DB를 통해 통계적인 정보를 얻을 수 있기 때문에, 이를 이용하여 보다 높은 캐쉬 적중률을 보장할 수 있다. 구현된 시스템에서도 이를 이용하기 위해 이전 질의어 DB에서 상위 인기 검색어를 추출한 뒤, 추출된 질의어에 대해서 사용자 질의를 처리하기 전에 이들 인기 질의어에 대한 캐쉬 데이터를 미리 생성하여 사용하는 방식을 채용한다. 이렇게 생성된 캐쉬 데이터는 2단계와 3단계 캐쉬 데이터로 저장되어 사용된다.

## 4. 결론 및 토의

기술된 웹 질의 처리 시스템은 전체 60여대의 서버로 구성되어 있으며, 각 서버는 두 개의 Xeon CPU를 탑재하고 있는 SMP 서버이다. 사용된 대부분의 서버가 2 GB 크기의 주기억장치를 가지고 있다. 질의 처리 시스템은 발생할 수 있는 하드웨어 고장에 대비하여 여러 개의 서버를 동일 서버 클러스터로 관리하고 있으며, 클러스터간의 통신은 고속 LAN을 통해 구현된다. 웹 질의 처리 시스템에서 하나의 사용자 질의를 처리할 때 사용되는 디스크 및 CPU 자원의 크기는 매우 크며, 대용량 질의를 처리해야 하는 경우 그 비용으로 인해 구축이 어려울 수 있다. 이런 문제점을 피하기 위해 한번 처리된 질의 처리 결과를 메모리나 디스크에 저장하여 사용하는 캐쉬 기법이 사용되어야 한다.

본 시스템에서도 이런 캐쉬 기법이 구현되었으며, 시스템 효율을 고려하여 크게 3개의 계층으로 캐쉬 데이터가 관리된다. 구현된 캐쉬 기법을 통해 얻을 수 있는 성능의 향상은 캐쉬 적중률에 크게 의존하며 이는 사용자 질의의 특성과도 관련을 가진다. 구현한 시스템의 운영을 통해서 살펴본 바로는 1계층 캐쉬 적중률은 45%, 2계층 캐쉬를 통한 캐쉬 적중률은 42%, 3개층 캐쉬를 통한 캐쉬 적중률은 16% 정도의 크기를 보였다. 이런 적중률에 따르면 전체 웹 질의에 대해 73% 정도의 캐쉬 적중률을 기대할 수 있었다. 이를 통해 성능향상 비율은 300% 정도이며, 이는 하루 100만 웹 질의를 처리를 할 수 있는 시스템을 구축한 뒤 캐쉬 기법을 적용한다면 하루 400만 질의를 처리할 수도 있음을 의미한다. 이런 사실을 통해 웹 검색 엔진과 같이 대용량의 데이터 처리 시스템에서 캐쉬 기법이 매우 유용함을 확인할 수 있었다.

### [참고문헌]


[1] Sergey Melnik, Sriram Raghavan, Beverly Yang, and Hector Garcia-Molina. Building a Distributed Full-text Index for the Web, In Proc. of the Tenth International World Wide Web Conference. pp. 396-406, 2001.

[2] Maxim Lifantsev and Tzi-Cker Chiueh. Implementation of a Modern Web Search Engine Cluster. http://citeseer.ist.psu.edu/561246.html

[3] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, 30(1-7), pp. 107-117.

[4] 이주남, Google과 함께 떠오르는 검색엔진, 소프트웨어진흥원 시장이슈 보고서, 2004.

[5] Search Engine Report, Http://www.searchenginewatch.com, 2005.

[6] Soumen Chakrabarti, Kunal Punera, and Mallela Subramanyam., Accelerated focused crawling through online relevance feedback. In Proc. of the 11th international conference on World Wide Web, pp. 148-159, 2002.

[7] Luiz Andre Barroso, Jeffrey Dean, and Urs Holzle. Web Search for a Planet: The Google Cluster Architecture, IEEE Micro, 23(2), pp. 22-28, 2003.

[8] Sriram Raghvan and Hector Garcia-Molina. Crawling the Hidden Web. In Proc. of VLDB, pp. 129-138, 2001.

[9] C. Lee, G. Golub and S. Zenios. A Fast Two-Stage Algorithm for Computing PageRank, Technical report, Stanford University, 2003.

[10] Krishna Bharat, Andrei Z. Broder, Jeffrey Dean, and Monika Rauch Henzinger. A comparison of techniques to find mirrored hosts on the WWW. Journal of the American Society of Information Science. 51(12), pp.1114-1122, 2000.

[11] Taher H. Haveliwala. Topic-sensitive PageRank, In Proc. of the 11th International World Wide Web Conference, 2002.