소규모 기관을 위한 웹 캐쉬 알고리즘

정회원 민 경 훈*, 장 혁 수**, 주 우 석***

A Web Cache Algorithm for Small Organizations

Kyong-Hoon Min*, Hyuk-Soo Jang**, Wou-Seok Jou*** *Regular Members*

요 약

현재 대부분의 웹 프락시 또는 웹 캐쉬는 대규모 기관을 중심으로 사용되고 있다. 하지만 인터넷을 사용하는 이용자 현황을 보면 오히려 대규모 조직에 속하는 이용자보다는 벤처나 PC 방과 같은 소규모 기관에 속해 있는 경우가 많다. 소규모 기관들은 사용 망 및 시스템의 용량 제한으로 인해 필요로 하는 웹 문서를 원활히 제공받지 못하고 있다. 또한 사용자들은 여러 윈도우를 열어 놓고 다양한 종류의 URL에 접속하고 있으며 비교적 짧은 시간 내에 또 다른 URL로 바꾸어 가면서 사용하고 실정이다.

본 논문에서는 근접한 지역에 위치하면서 특성이 유사한 소규모 집단을 연결하는 웹 캐쉬 구성 방안과, 동일한 캐쉬 용량으로도 적중률이 높고 웹 문서 사용자들이 요구하는 URL이 급격히 바뀌어도 적중률 저하를 차단할 수 있는 캐쉬 알고리즘을 제시한다. 제안 알고리즘은 요청된 URL이 속한 네트워크 경로에 따라 웹 문서 사용 특성을 분류하고, 서로 다른 경로에 속하는 웹 문서를 서로 다른 저장 장소에 캐쉬를 하는 형태이다. 제안 알고리즘을 실제 사이트에 적용하여 얻은 실험 결과를 분석하여 보면, 적은 비용으로 기존 알고리즘보다 적중률 및 응답 시간 면에서 현저하게 뛰어남을 알 수 있다.

ABSTRACT

Most of the existing web caches are used in huge organizations. But many internet users belong to small organizations such as a venture company or a PC room. Users are in general in multiple window environments, and use several programs concurrently with rapid preference change within a relatively short period of time.

We develop a network-path based algorithm. It organizes a cache according to the network paths of the requested URLs and builds a network cache farm where caches are logically connected with each other and each cache has its own preference over certain network paths. The algorithm has been implemented and tested in a real site. The performance results show that the new algorithm outperforms the existing algorithms in the hit ratio and response time dramatically with low cost.

I. 서 론

인터넷 이용자 수, 호스트 수 및 인터넷 서비스 공급자(ISP: Internet Service Provider) 수의 증가는 국내뿐만 아니라 전 세계적으로 폭발적이다. 이러한 추세는 인터넷을 쉽게 이용할 수 있는 PC 방 및 인터넷을 기반으로 하는 벤처 회사와 같은 소규모 기관의 증가, 전자 상거래 활성화와 더불어 가속화되고 있다. 인터넷 사용자가 요구하는 정보 및 데이터는 우선적으로 웹 캐쉬(cache)를 검색하여 가져오게 되는데 캐쉬에 원하는 정보가 없으면 원래 요구

* 명지대 정보통신공학과 석사과정(neominga@mju.ac.kr)
** 명지대학교 전자·정보통신공학부(jang-h@mju.ac.kr)
*** 명지대 컴퓨터공학부(wsjou@mju.ac.kr)
논문번호 : 00205-0612, 접수일자 : 2000년 6월 12일

된 서버 또는 이웃 캐쉬 서버로부터 필요한 정보를 받게 된다. 따라서 원하는 정보가 캐쉬에 없으면 없을수록 인터넷망에 걸리는 트래픽 양도 많아지고, 데이터의 이동 거리도 길어지게 되어 망 전체의 성능을 저하시키며, 응답 시간이 길어진다[1][2][3]. 대부분의 웹 서버를 자체적으로 가지고 있지 못한 소규모 기관, 특히 PC 방 및 벤처회사와 같은 기관에서는 외부로 연결된 회선에 병목 현상을 가중시키고, 인터넷 통신 업체에게 지불하는 회선 사용 비용이 회선 증설과 더불어 계속 증가하게 된다.

현재 대규모 기관이나 ISP에서 사용중인 웹 캐쉬 형태는 주로 "CERN"[2][4]과 "Harvest/Squid"[4] 방식으로, 정적인 캐쉬 계층 구조에 기반을 두고 있다. 이러한 구조는 사용자 또는 호스트별로 캐쉬 서버를 미리 지정하도록 되어 있으며, 계층적 캐쉬 상호간 하드 와이어로 연결되어있다. 따라서 캐쉬 서버에 원하는 정보가 없으면, 이 캐쉬 서버와 연결된 또 다른 캐쉬 서버로 요구가 넘어간다. 서버 중 더 이상 다른 서버에 연결되어 있지 않은 최종 부모(parent) 서버의 캐쉬에 원하는 정보가 없으면, 원래의 웹 서버에서 정보를 가져오게 한다. "Harvest/Squid" 방식의 특징은 웹 캐쉬 서버가 동일 망 내부에서는 몇 개의 동료(sibling) 서버에 하드와이어로 연결되어 있고, 망 외부로는 하나의 부모 서버에 연결되어 있다. 검색 속도를 빠르게 하기 위해 부모 및 동료 서버에게 요청하는 메시지를 동시에 멀티캐스트하고, 해결이 안된 요청은 상위 서버로 전달한다. 현재는 "Harvest" 방식이 가장 많이 쓰이고 있다. 또한 "CERN" 및 "Harvest" 방식 모두 정적 계층 구조를 가지고 있으며, 이로 인해 웹 캐쉬 서버 하나의 결함이 전체 웹 캐쉬 서버 결함으로 오동작할 가능성이 있다. 아울러 망 구성 및 트래픽 양의 변화에 따라 웹 캐쉬 구성을 자유롭게 바꾸기 어렵고, 다른 웹 캐쉬에 중복된 정보의 복사본을 가지고 있을 가능성이 있어 자료간 불일치를 초래 할 수 있으며, 결과적으로 기억 장소의 낭비로 이어질 수 있다.

최근에는 웹 캐쉬 서버들 간의 통신에 관한 연구[2][5][6] 및 분산된 캐쉬 서버간 동적 구조와 운영 방법[7]등에 관한 연구가 활발히 진행되고 있다. 하지만 이러한 연구들의 취약점은, 실제 인터넷을 사용하는 특정 기관이나 조직 특성 및 사용자의 웹 문서 요구 형태를 반영시키지 않고, 어느 기관이든 일률적인 웹 캐쉬 구조 및 방식을 적용하고 있어, PC 방과 벤처 회사와 같은 소규모 기관에 그대로 적용하는 것은 어려운 일이다. 사용자 수가 적은 소규모 기관에서는 웹 캐쉬의 필요성에 대한 인식 부족과 설치 및 유지에 따른 비용 부담으로 인해 사용하고 있지 않은 상황이다. 또한 기관의 규모가 작아 여러 대의 서버를 설치할 필요가 없는 점을 감안하면, 대규모 조직에서 사용되는 계층별 캐쉬 연결 구조를 그대로 소규모 기관에 적용하는 것도 불합리하다.

일반적인 캐쉬 서버의 운영 형태는 사용자 요구(request)에 의한 캐싱으로, 최근에 요구율이 높지 않은 정보는 저장 능력의 한계 때문에 새로운 것으로 교체된다. 다양한 웹 문서의 폭발적 증가와 한 명의 사용자가 윈도우를 여럿 열어 놓고 동시에 다양한 웹 문서를 이용하는 지금의 상황에서, 최근의 요구율이 높은 웹 문서만을 캐싱 하는 기존 방법은 대규모 기관뿐만 아니라 소규모 기관에 적용하는 것 역시 적절한 캐쉬 방안이 되지 못할 것으로 생각된다. 이는 각 사이트의 요구율이 비슷할 경우, 전반적으로는 요구율 차이가 없음에도 불구하고, 사이트 요구 순서의 역동적 변화로 인해 우선 순위에서 밀려 제거된 웹 문서에 대해 다시 요구하게 되면, 원래 서버로 가서 데이터를 가져 와야 하는 상황이 발생하기 때문이다. 이렇듯 기존 캐쉬 방법으로는 사용자의 URL(Uniform Resource Locator) 선호도에 대한 역동적 변화를 반영하는 웹 캐쉬 서버를 구축하는데 어려움이 있었다. 본 논문에서는 소규모 기관에서 인터넷을 이용하는 사용자의 웹 문서 요구 형태에 대한 데이터를 수집 분석하여, 요구되는 URL을 네트워크 경로에 기초하여 분류하고, 캐쉬별 담당 네트워크 경로를 할당하였다. 소규모 기관의 웹 문서 요구를 라우터가 캐쉬 서버로 연결시키는 방법을 사용하여, 기존에 사용되는 여러 개의 캐쉬보다 하나 또는 단지 몇 개의 캐쉬 서버로도 적중률을 높일 수 있는 알고리즘을 제시하였다.

Ⅱ절에서는 본 논문에서 제안하는 웹 캐쉬 알고리즘을 설명하고, 제 Ⅲ절에서는 제안된 캐쉬 알고리즘을 실제 사이트에 적용한 실례를 보여 주며, 제 Ⅳ절에서는 성능 분석 및 기존 방식과의 결과 비교를 시행하고, 제 Ⅴ절에서 결론을 맺는다.

Ⅱ. 웹 캐쉬 알고리즘 제안

웹 캐쉬를 두는 목적이 캐쉬의 적중률을 높여 원거리에 있는 원래(original) 웹 서버까지 가야 할 확률을 줄이고, 이를 통한 네트워크 트래픽 양을 감소시켜 통신비용을 절감 하고자 하는데 있다. 대부분

의 웹 캐쉬는 정적인 계층 구조를 가지고 있으며 원하는 데이터가 없을 때 물리적으로 연결된 캐쉬 서버에게 요청을 넘겨주는 형태를 취하고 있다. 캐쉬에 있을 확률을 높이기 위해서는 캐쉬에 있는 정보의 내용을 체계적으로 분류하여 적중률이 가장 높은 캐쉬로 요청을 라우팅 해주는 캐쉬 시스템 구축이 필요하다. 하지만 인터넷에 존재하는 URL과 그에 따른 객체의 종류 및 수가 워낙 많아 현실적인 분류 방법을 찾기에 어렵고, 소규모 조직인 경우 여러 대의 캐쉬 서버를 두고 가능한 많은 종류의 URL을 저장하는 방법은 설치비용 부담 등으로 인해 현실적으로 더욱 힘들다. 이러한 문제를 해결하고자 조직 또는 기관의 임의의 호스트에서 출발하여 목적지 URL에 이르는 네트워크 경로를 조사해 보면, 대략 어느 시점까지 몇 개의 네트워크 경로를 공통으로 이용한다는 사실을 알 수 있다. 또한 인접한 유사 기관들 역시 같은 ISP를 사용할 가능성이 크고, 이에 따라 유사한 경로를 나타내게 된다. 이는 원래 서버에 있는 문서의 속성이나 물리적인 위치 정보를 알 수 없는 현재의 인터넷 상황에서 간단한 명령어, "traceroute"로 쉽게 파악 가능한 네트워크 경로를 바탕으로 사용자의 인터넷 이용 현황을 분석한 결과이다. 특히 국내에서는 그림 1에 나타났듯이 네트워크 경로별 특성이 대략 ISP별로 구분 할 수도 있는데, 이는 국내 ISP 수가 미국등 선진국에 비해 상대적으로 많지 않아서 몇 개의 그룹으로 구분하기에 용이한 것으로 생각된다. 또한 인터넷 사용자 한 명에게 주어지는 환경 및 취향을 조사해 보면, 다수의 원도우를 통하여 성격이 전혀 다른 URL들을 수시로 바꿔가며 사용하고 있어, 캐쉬 시스템에 저장하는 URL 종류를 다양화하는 것이 캐쉬 적중률을 높일 수 있을 것으로 생각된다. PC 방과 같이 비슷한 특성을 가지는 기관들을 이용 목적, 사용자 계층 및 기관 위치를 고려하여 사용자의 행동 유형을 실제로 조사해 본 결과, 거의 비슷한 웹 문서들을 요구하고 있음을 알 수 있었다[8].

본 논문은 URL이 몰리는 대표적인 몇 개의 네트워크 경로별로 URL을 구분하고, 이를 서로 다른 장소에 캐싱하는 알고리즘을 제안한다. 이로 인한 기대 효과는 캐쉬공간별 저장 정보가 특성화되어 있고 실제 캐쉬에 저장되어 있는 정보의 종류가 많아질 수 있기 때문에 사용자 취향의 급격한 변화에도 적중률의 급격한 저하를 차단하는 효과를 얻을 수 있다. 또한 소규모 기관에 적용 될 수 있도록 비슷한 특성을 가지는 소규모 기관을 묶어 단일 캐쉬

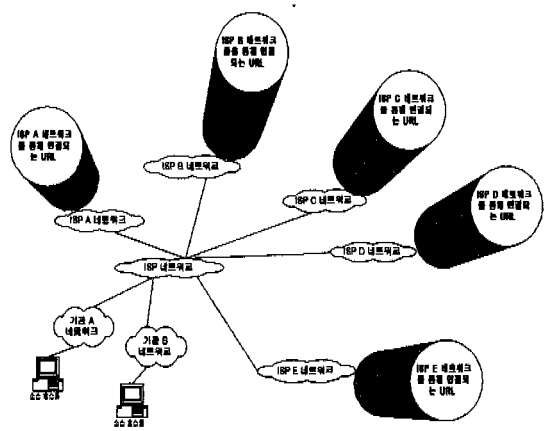


그림 1. 네트워크 분류에 의한 분류 방법

서버를 사용하도록 한다. 이로 인해 적은 비용으로도 캐쉬 적중률의 저하 없이 사용이 가능하다는 것이다. 소규모 기관에 적용되는 캐쉬 구조는 그림2와 같이 하나의 캐쉬 서버가 여러 기관을 담당하는 형태의 캐쉬 시스템을 이용한다. 캐쉬 서버는 저장하는 정보를 네트워크 경로별로 특성화하여 동일한 정보를 서로 다른 저장 공간에 두어 중복 존재를 막고, 서버에 다양한 종류의 정보를 저장하게 하여 사용자 취향이 급격히 변해도, 웹 캐쉬 서버내의 특정 저장 공간에 원하는 정보가 있을 확률을 높이도록 시도하였다.

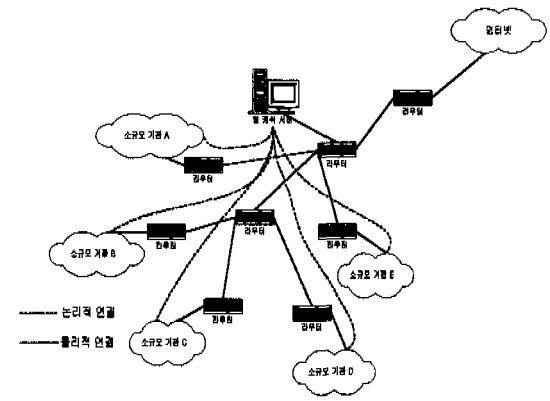


그림 2. 웹 캐쉬 서버와 소규모 기관의 연결 구조

본 논문에서 제안하는 웹 캐쉬 알고리즘은 그림 3과 같다. 그림 3의 URL 경로수는 알고리즘을 설명의 편의를 위해 3개로 나누어 나타낸 것 뿐으로 경우에 따라 다를 수 있다. 알고리즘의 1 단계는 정보 수집 단계로, 소규모 기관내 사용자들이 사용하는 URL을 알아내기 위해 각 사용자의 컴퓨터에서

이용되는 웹 브로우저가 저장한 캐쉬 파일을 수집하고, 이를 "traceroute"를 사용하여 각 URL이 위치한 네트워크 경로를 생성한다. 2 단계는 정보 분석 단계로, 생성된 네트워크 경로를 사용하여 URL을 분류하는데 이를 위해 각 URL이 지나는 공통경로를 찾고, 공통경로 및 공통경로 이후 경로별 특성을 분류 기준으로 사용한다. 이 과정을 통하면 URL들이 공통적으로 경유하는 특정 네트워크 경로에 따라 URL을 분류할 수 있다. 다음으로, 네트워크 경로 중 요청이 많은 네트워크 경로를 선정한다. 이는 웹 문서 사용자의 URL 요청이 많다는 것으로 주요 캐싱 대상이 되어야 하기 때문이다. 몇 개의 경로가 선정이 되면 서버 내에 이를 캐싱 할 저장 공간을 할당한다. 3 단계는 운영 단계로, 사용자가 URL을 요청하면 라우터가 요청 URL을 원래 서버가 아닌 캐쉬로 연결 설정 바꾸어주는 것이다. 캐쉬 서버는 특정 경로에 대해서 서로 다른 공간에 캐싱을 하게 되며, 사용자의 URL 요청은 URL이 속한 경로에 따라 해당 저장 공간 또는 캐쉬로 보내지게 된다. 이때, 요청 URL중, URL이 속한 경로에 해당하는 캐쉬 서버가 없는 경우는 디폴트 저장 공간으로 보내진다. 캐쉬 대체 방법은 통상 쓰이는 LRU(Least Recently Used) 방식을 사용하였다.

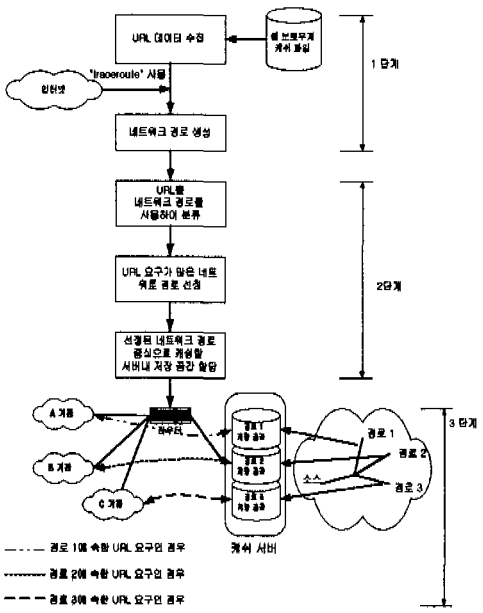


그림 3. 제안 웹 캐쉬 알고리즘

그림 4는, 편의상 서버 내 4개의 저장 공간을 사용한 사례에 대한 논리 흐름을 보여 주고 있다.

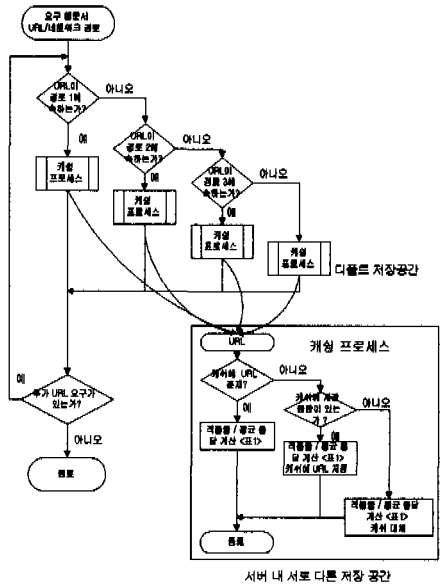


그림 4. 제안 알고리즘의 논리 흐름도

표 1. 적중률과 평균 응답 시간 계산 방법

적중률 (%)	▪ 적중된 URL 및 URL 객체 개수 / 전체 요구 URL 및 객체 개수 ×100
평균 응답 시간 (홉 수)	▪ (적중되었을 때 홉 수 + 실패했을 경우 홉 수) / 전체 요구수

URL 요청이 들어오면 라우터가 요청을 캐쉬 서버로 보내며 캐쉬 서버는 저장된 URL과 요청 URL을 비교하여 정보가 있을 경우 요청에 응답을 하고 없을 경우 원래 서버로 요청을 보낸 후 받은 응답을 캐쉬 서버에 일단 저장한 후 요청한 사용자에게 보낸다. 제 Ⅳ절에 설명될 성능 평가를 위해 표 1과 같은 계산 방법이 사용되었다. 적중이라는 것은 요청한 URL이 서버에 있을 경우를 말하며, 적중률은 전체 요청 URL중 적중한 URL의 비율이다. 응답 시간은, 적중할 경우는 캐쉬 서버까지 갔다온 시간이 되고, 실패했을 경우는 캐쉬 서버까지 간 시간에 원래 서버까지 갔다온 시간, 다시 캐쉬 서버에서 사용자까지 간 시간의 합이 된다. 본 본문에서는 응답 시간을 편의상 홉(hop) 수로 사용하였다.

Ⅲ. 제안 알고리즘 적용 사례

알고리즘 적용 대상으로 서로 인접하고 인터넷 사용 특성이 비슷한 소규모 기관 2개를 선택하고,

각 기관에 있는 3대 컴퓨터의 웹 브로우져에 저장된 캐쉬 데이터를 분석한 후 알고리즘을 적용하였다. 그림 5는 알고리즘 적용 대상인 소규모 "N"과 "S" 기관의 문서 URL을 기준으로 공통적인 네트워크 경로를 조사한 그림이다.

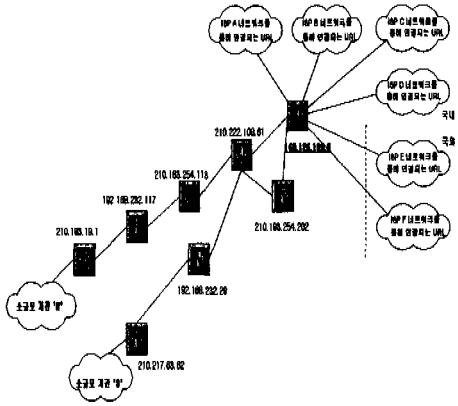


그림 5. 소규모 "N" 기관과 "S" 기관에서의 웹 문서에 따른 네트워크 경로

그림 5에서 볼 수 있듯이 크게 국내와 국외로 구분이 가능하며, 특히 거의 같은 경로를 이용하던 트래픽들이 외부 ISP 라우터 "168.126.109.9"에서부터 여러 경로로 나누어진다는 것을 알 수 있다. 따라서 이 라우터를 기준으로 연결된 네트워크들을 분류 대상으로 이용을 할 수 있다. 그림 6은 그림 5에서 판단한 네트워크 경로의 분기점, "168.126.109.9"를 기준으로 사용하여, 실제 "N" 기관 사용자들의 웹 문서 URL 리스트를 가져와 분류한 경로를 나타낸 것이다. 보다 자세한 분류가 필요하다면, 그림 6의 끝점으로부터 이어지는 경로를 따라 보다 더 많은 홉 수를 이용하면 더 자세하게 분류 할 수 있다. 우리가 실험한 실제 사이트에서, "N"과 "S" 기관 경로 유형을 조사해 보면, 주 관심 노드 "168.126.109.9"까지는 거의 같은 경로를 따라가고, 그 이후에 한 홉 정도 떨어진 서로 다른 노드로 가고, 그 이후에는 다양한 경로로 나뉘어짐을 알 수 있다. 이를 통해서 주 관심 노드에서 한 홉 정도 떨어진 노드까지의 경로의 수를 조사해 보면 대략 44개 정도로 압축됨을 알 수 있다.

네트워크 경로에 따라 실제 "N" 기관과 "S" 기관에서 사용자들이 요구한 웹 문서 요구 경향을 그림 7과 8에 나타내었다. 'x' 축에 표시된 네트워크 경로는 "168.126.109.9"를 기준으로, 이에 연결된 "168.126.16.4"에서 "168.126.16.126"까지 중에서

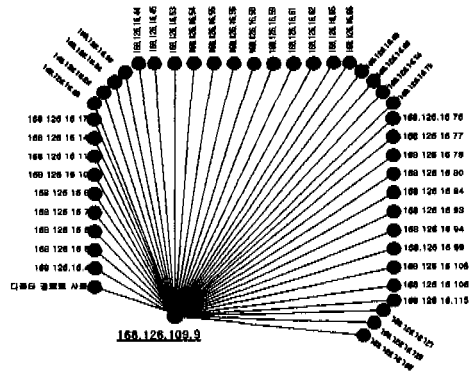


그림 6. "N" 기관의 URL에 따른 네트워크 경로

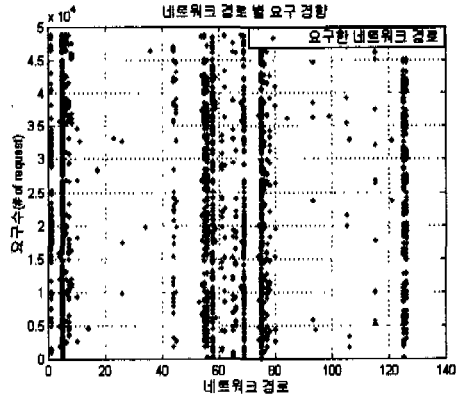


그림 7. "S" 기관 웹 문서에 대한 네트워크 경로별 요구 수

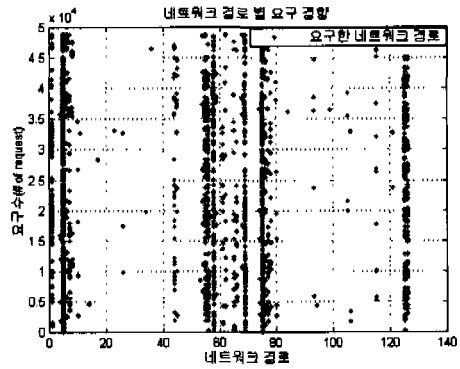


그림 8. "S" 기관 웹 문서에 대한 네트워크 경로별 요구 수

호스트 주소 부분만을 나타낸 것이다. 'y'축에 표시된 요구 수는 특정 네트워크 경로를 몇 번째 URL이 요구했는가를 보여주는 것으로 총 URL 개수는

대략 50,000개 정도 된다. 그림 7과 8에서와 같이, 사용자의 요구 문서를 찾아가는 경로가 일정 유형을 보이고 있으며 특정한 몇몇의 경로에 집중되어 있음을 발견할 수 있다. 이러한 경로에 집중되는 요구율은 표 2 및 그림 9와 같으며, 전체 요구의 68%가 특정 경로 4개에 집중됨을 알 수 있다.

표 2. 특정 경로에 대한 요구율

요구가 많은 상위 경로 수	1	2	3	4
요구율 (해당 경로 요구수/전체 요구 수, %)	30 %	47 %	58 %	68 %

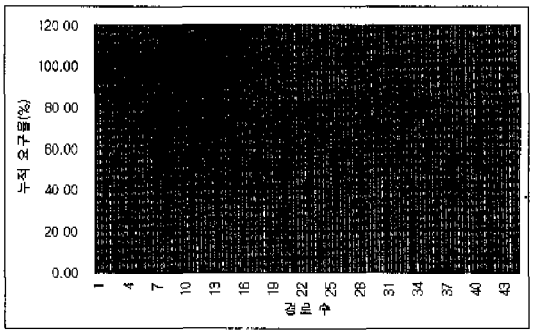


그림 9. 전체 경로 수 별 누적 요구율

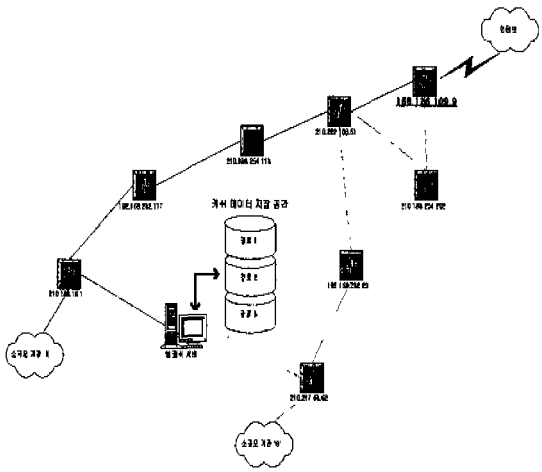


그림 10. 제안 알고리즘을 적용한 웹 캐쉬 구조

본 논문에서 제안된 알고리즘을 적용한 웹 캐쉬 서버 구성은 그림 10과 같으며 서버 저장 공간을 4개로 분류했을 경우의 예이다. 캐쉬 서버는 각 기관의 게이트웨이와 연결되어 있으며, 네트워크 경로를 기초하여 요구 URL을 분류하고 이를 이용하여 서버 내 다른 공간에 저장하였다. 즉, 서버 내 각 저장 공간은 특정 경로에 속한 사이트에 대해서 캐싱을 하도록 하여 캐쉬 서로 간에 중복 저장을 피하고 저장 내용이 다양화되어, 웹 요구 경향이 빠르게 바뀌어도 캐쉬 적중률이 나빠지는 것을 막을 수 있었다.

Ⅳ. 알고리즘 성능 분석

성능 분석을 위해 "N" 및 "S" 기관으로부터 수집한 웹 브로우져의 기록(history) 파일 내용 중, 각각 16,000개, 14,000개 정도를 입력 데이터로 사용하였다. 기록 파일에서 우리가 필요로 하는 것은 사용자가 요청한 웹 URL 및 URL에 속한 문서와 이미지 파일과 같은 객체 목록이기 때문에 이것을 추출하여 입력으로 사용하였다. 추출 및 분석 과정은 그림 11에 나타나있다. 성능 분석을 위해 다음과 같은 3가지 시나리오를 만들어 각각을 비교하였다; ① 캐쉬를 사용하지 않는 경우, ②소규모 조직이 각자의 캐쉬를 따로 두고 이용하는 경우, ③모든 소규모 조직이 하나의 캐쉬를 공동으로 사용하는 경우. 캐쉬를 사용하지 않을 경우는 PC 자체 지역(local) 캐쉬를 웹 브로우져가 설정한 캐쉬로 이용하는 경우이고, 캐쉬 서버를 설치해서 이용하는 경우는 제안된 알고리즘이 적용된 캐쉬를 이용하는 경우이다. 제안된 알고리즘을 사용하기 위해서는, 네트워크 경로가 파악되어야 하며 이를 위해서 "traceroute"를 이용하여 네트워크 경로 및 원래 소스가 위치한 서버까지의 홉(hop) 수가 추출된 데이터 파일을 만들었다. 성능 비교를 위해 3가지 시나리오에 대한 요청 웹 문서에 대한 캐쉬 적중률과 응답시간을 구하

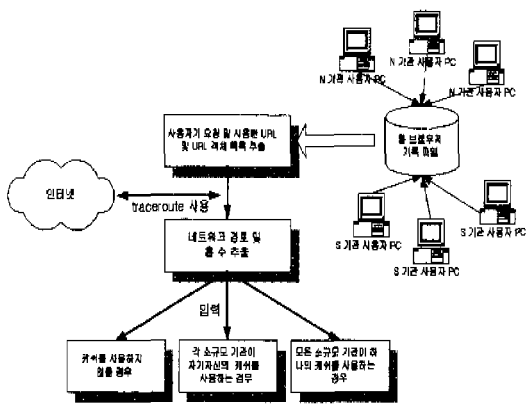


그림 11. 성능 분석 체계도

였다. 제안 알고리즘이 적용된 캐쉬 구조에서 서버 내 저장 공간 분류는 요구율이 많이 몰려 있는 네트워크 경로를 중심으로 하여 4개 정도로 구분하여 적용하였다.

캐쉬 서버를 사용하지 않는 경우, PC 자체 웹 브로우져가 설정한 지역 캐쉬 사용에 따른 적중률과, 캐쉬 서버를 따로 설치해 사용하는 경우에 대한 각 기관별 적중률을 그림 12에 나타냈다. 입력 데이터로 URL 및 URL 객체 목록을 이용한 관계로, 캐쉬 저장 용량은 목록을 저장할 수 있는 개수로 나타내었고, 100개 목록을 저장할 수 있는 공간을 초기 크기로 하여 100개 목록 공간씩 증가 시켜가며 적중률을 구하였다.

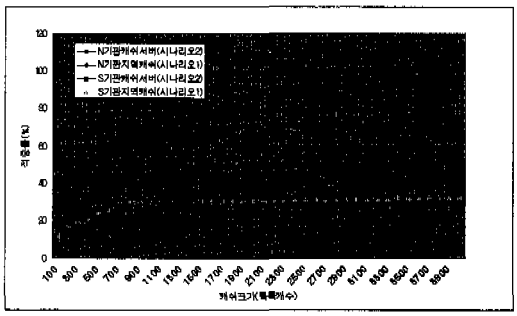


그림 12. 시나리오 ①, ②에 대한 적중률

지역 캐쉬를 이용할 경우 "N" 기관에서는 캐쉬 크기를 아무리 크게 해도 44% 이상의 적중률 향상을 기대하기 어렵다. 캐쉬 서버를 이용할 경우, 캐쉬 크기가 2000이상에서 적중률의 향상을 보이며, 캐쉬 크기가 3300에서 69% 정도의 적중률 향상이 되고 이후 93% 까지 적중률이 향상됨을 볼 수 있다. "S" 기관에서는 지역 캐쉬만을 이용할 경우 32% 정도에서 적중률 향상을 기대하기 어렵다. 캐쉬 서버를 이용할 경우 캐쉬 크기 3200에서 49% 정도 향상이 되는 것을 알 수 있다. 이후 캐쉬 크기가 커질수록 계속 향상이 되다가 57% 정도에서 둔화를 보인다. 두 기관의 적중률 차이는 입력 데이터의 크기 및 중복 정도에 따른 데이터 특성에 따라 다르다.

응답 시간에 대한 성능 분석은 홉 수로 나타냈으며 결과는 그림 13과 같다. 응답 시간은 적중률과 밀접한 관련이 있는데 적중률이 좋아지면 원래 서버로 가야 하는 요구 수가 줄고 이로 인해 응답 시간은 빨라지게 된다. 성능 분석 결과도 마찬가지로 적중률에 비례해서 응답 시간이 빨라짐을 알 수 있다.

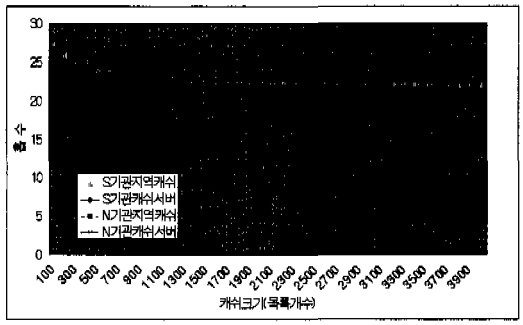


그림 13. 시나리오 ①, ②에 대한 응답시간

지역 캐쉬를 이용할 경우 "N" 기관은 캐쉬 크기를 크게 해도 평균 14 홉 정도의 응답 시간을 보인다. 캐쉬 서버를 사용할 경우는 캐쉬 크기 2000에서부터 응답시간이 향상이 되며, 3300에서는 9홉, 그 후 5홉 정도까지 향상됨을 알 수 있다. "S" 기관은 지역 캐쉬인 경우 22 홉 정도에서 일정한 값을 가지며, 캐쉬 서버를 이용할 경우 캐쉬 크기 3300에서 성능 향상을 보이며 3800에서 15홉 정도를 보이며 이후 15홉 정도까지 계속 향상이 됨을 알 수 있다.

다음으로 각 기관이 각자 독립적인 캐쉬 서버를 사용할 때와, 두 기관이 하나의 서버를 공동으로 사용할 때에 따른 성능분석을 하였고, 결과는 다음 그림 14에 나타내었다. 두 기관이 하나의 서버를 이용하는 경우에 대한 성능평가를 위해서, 두 기관의 데이터를 무작위로 선택하고 이를 입력 값으로 사용하였다.

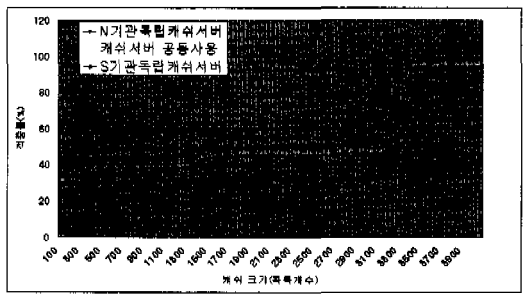


그림 14. 시나리오 ②, ③에 대한 적중률

두 기관이 캐쉬 서버를 공동으로 이용할 경우 "N"기관과는 비슷한 적중률을 보이며, "S" 기관에 비해서는 대략 10%이상 적중률이 높게 나타나며, 캐쉬 크기 3100에서 두 기관 모두에 비해 적중률 향상이 나타난다. 이는 두 기관이 독립적인 캐쉬 서

버를 사용하는 것에 비해 공용으로 캐쉬 서버를 사용하여도, 성능 저하가 없이 사용이 가능하다는 것으로 소규모 기관 여러 개를 묶어 사용할 경우 캐쉬 사용에 따른 통신비용 이득뿐만, 아니라 설치에 대한 부담 등을 줄일 수 있다는 것을 보여 준다.

성능 분석 결과, 실험 대상 "N" 기관과 "S" 기관은 본 논문에서 제안한 캐쉬 서버를 사용할 경우, 적중률 및 응답시간의 획기적 개선이 이루어지고, 이에 따라 외부로 나가는 트래픽 양을 줄일 수 있음을 알 수 있다. 아울러 두 기관이 각자의 캐쉬 서버를 가지는 것 보다 두 기관을 묶어 하나의 캐쉬 서버를 설치하여 공동으로 사용하여도, 적은 비용으로도 성능 저하 없는 캐쉬 운영이 가능하다는 것을 보여 주었다. 또한 보다 빠르게 변화하는 사용자 경향에도 충분히 적응 할 수 있다는 것을 알 수 있다.

V. 결 론

현재 인터넷 사용자 층은 벤처 회사나 PC 방과 같은 기관을 중심으로 두터운 층을 이루고 있으며, 인터넷을 이용하는 형태 역시 비슷한 면을 보여 주고 있다. 대다수의 인터넷 이용자들은 여러 윈도우를 동시에 열어 놓고 여러 인터넷 사이트를 동시에 접속하여 다양한 종류의 웹 문서를 요구하고 있고, 요구하는 문서 종류 또한 수시로 빠르게 바꾸며 사용하고 있다. 이런 인터넷 이용 환경에 기존의 캐쉬 알고리즘으로는 적중률 및 응답 시간의 개선을 기대하기 어렵다. 이러한 환경에서도 적중률의 저하를 줄일 수 있는 네트워크 경로에 의거한 웹 캐쉬 알고리즘을 제안하였고, 성능 분석을 통하여 소규모 기관에서 자체 웹 브로우져 내의 지역 캐쉬를 사용할 경우 보다 적중률 및 응답 시간에 향상이 있음을 알 수 있다. 사용자 요구 내용이 빠르게 변하는 환경에도 적중률 저하 없이 능동적으로 대처 할 수 있음을 보였다. 또한 인접한 거리에 있으며 특성이 비슷한 기관을 묶어 캐쉬 서버를 공유할 경우, 각 기관이 캐쉬 서버를 독립적으로 사용하는 경우와 비교하여도 캐쉬 적중률이 나빠지지 않음을 보였다. 이와 같은 결과는 캐쉬 서버 설치 사용에 있어서 부담을 가지는 소규모 기관이 서로 캐쉬 서버를 공유해 사용할 수 있음을 보였다.

참 고 문 헌

[1] Lee Breslua, Pei Cao, Li Fan, Graham Phillips, Scott Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implication", *IEEE INFOCOM*, 1999

[2] Duane Wessels, K. Claffy, "ICP and the Squid Web Cache", *IEEE Journal*, Apr. 1998

[3] H. Sun, X. Zang, K.S. Trivedi, "The effect of Web caching on the network planning", *Elseviser Science* B.V., 1999

[4] A. Lutonen, H. F. Nielsen, T. Berners-Lee, "Cern httpd", July 1996

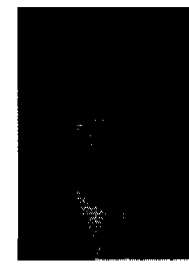
[5] Keith W. Ross, "Hash Routing for Collections of Shared Web Cache", *IEEE Network*, Nov. 1997

[6] Li Fan, Pei Cao, Jussara Almeida, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", *ACM*, 1998

[7] Samrat Bhattacharjee, Kenneth L. Calvert, Ellen W. Zegura, "Self-Organizing Wide-Area Network Caches", *IEEE INFOCOM*, 1998

[8] 민경훈, 장혁수, "네트워크 경로에 근거한 웹 캐쉬 알고리즘", *정보처리학회논문지*, 2000 제7권 7호, 2000

민 경 훈(Kyung-hoon Min) 준회원



1998년 : 명지대학교 정보통신공학과(공학사)
1998년~현재 : 명지대학교 정보통신공학과 석사 과정
<주관심 분유> 인터넷, 컴퓨터통신

장 혁 수(Hyuk-soo Jang) 정회원

1975년~1983년 : 서울대학교 산업공학과(공학사)
1986년 : 미국 Ohio State University 산업 및 시스템 공학석사
1990년 : 미국 Ohio State University 전산학박사
1990년~1992년 : 경북대 전자공학과 교수
1998년~1998년 : 미국 O.S.U 전산학과 교환 교수
1992년~현재 : 명지대 전자 · 정보통신 공학부 교수
<주관심 분야> 인터넷, 컴퓨터통신, 컴퓨터구조

주 우 석(Woo-suk Jou) 정회원

1976년~1983년 : 서울대학교 공과대학 전자공학과 (공학사)
1983년~1985년 : 한국 IBM, 데이콤 정보통신연구소
1985년~1987년 : University of Florida, 컴퓨터공학(석사)
1987년~1991년 : University of Florida, 컴퓨터 공학(박사)
1992년~현재 : 명지대학교 컴퓨터학부 교수
<주관심 분야> 컴퓨터그래픽스, 알고리즘, 데이터베이스