무선 인터넷 프록시 서버 클러스터에서 캐시간 협동을 위한 해싱 알고리즘 비교

곽후근, 한경식, 정규식
숭실대학교, 정보통신전자공학부
전화 : 02-821-2757

A Comparison of Hashing Algorithms for Cache Cooperation in a Wireless Internet Proxy Server Cluster

Hukeun Kwak, Kyungsik Han, Kyusik Chung
School of Electronics Engineering, Soongsil University
{gobarian, ihanks, kchung}@q.ssu.ac.kr

Abstract

Caching is one of essential functions in a wireless internet proxy server cluster. To serve best quality of service and choose suitable a proxy server for user, load balancer have to consider cache cooperation between proxy servers. Usually hashing is a simple way to support cache cooperation. In this paper, we compare and analysis static hashing and MD 5 that can be used for cache cooperation.

Ⅰ. 서론

무선 인터넷 프록시 서버를 클러스터로 구성함에 있어 캐시간의 협동성을 보장하는 것은 클러스터 간 중복되는 공간을 줄이고, 효율적인 캐싱을 통해 사용자가 요청한 데이터를 빠르게 얻을 수 있게 한다. 이를 위해 널리 사용되는 캐시 선택 방법은 해싱(Hashing)을 이용한 방법이다. 즉, 사용자 혹은 목적지 URL의 해싱값을 이용하여 캐시를 선택함으로써 동일 URL에 대해 동일 캐시가 선택되는 것을 보장한다. 본 논문에서는 캐시 선택 시에 사용할 수 있는 해싱 기법 중에서 정적 해싱(Static Hashing)[1]과 MD 5 해싱(Message Digest 5 Hashing)[2]기법의 장단점을 비교 분석한다. 이러한 해싱 기법들의 분석을 통해 무선 인터넷 프록시 서버 클러스터에 적용하여 클러스터 구성 상황에 따른 최적의 해싱 기법을 제안한다. 그림 1은 무선 인터넷 프록시 서버 클러스터에서 캐시 협동성을 나타낸다.

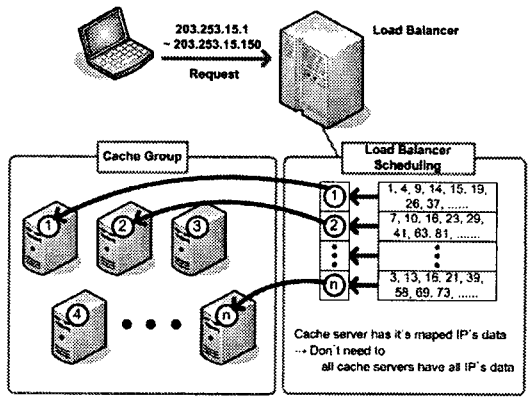

그림 1. 무선 인터넷 프록시 서버 클러스터에서의 캐시 협동성

Ⅱ. 연구배경

캐시간 협동성을 보장하기 위해 사용되는 해싱 기법들을 비교하면 다음과 같다.

• 정적 해싱(Static Hashing) : 256개의 버킷(Bucket)을 생성, 클러스터의 개수로 분할하여 매핑하고, 사용자의 요청 URL중 마지막 자리 (예,

203.253.15.XXX)를 기준으로 캐시를 선택하는 방식이다. 정적 해싱은 초기 버킷 생성 후에는 일정 이상의 요청부터 클러스터 분배를 위한 추가적인 계산 시간이 소요되지 않는다는 장점을 가지고 있다. 반면에 한번 정해진 버킷은 유동적이지 않기 때문에 캐시의 추가나 삭제, 또는 특정 URL로 요청이 집중되어 버킷의 재수정이 필요할 때 유동적이지 못하다는 단점을 가진다. 그림 2는 무선 인터넷 프록시 서버 클러스터 상에서 정적 해싱을 통한 캐시 선택을 나타낸다.

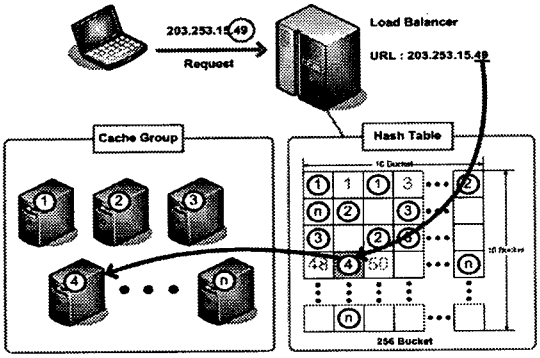

그림 2. 정적 해싱을 통한 캐시 선택

그림 2의 정적 해싱 순서를 살펴보면 다음과 같다

① 프록시 서버내의 로드 밸런서가 사용자로부터 URL요청(203.253.15.49)을 받는다.

② 256개의 버킷을 생성, 캐시 개수인 n 등분한 해싱 테이블을 생성한다.

③ URL의 하위 8비트(49)를 n등분한 해싱 테이블에 맵핑한다.

④ 맵핑된 캐시로 사용자의 요청을 할당한다.

• MD 5해싱(MD 5 Hashing) : 사용자 요청 URL에 대해 고정 길이의 해시값이 나오고, 이를 캐시의 수에 매핑하는 방식으로 동작한다. 다시 말해, 사용자의 요청 URL을 MD 5 Hash를 통해 Message Digest[2]를 생성하고, 생성된 Message Digest를 나머지 연산(mod n)을 통해서 생성된 값에 해당하는 클러스터에 요청을 할당하는 방식이다. 요청이 들어올때 마다 Message Digest를 생성, 클러스터에 할당하기 때문에 버킷을 위한 별도의 메모리 공간 소요가 없다는 이점 가진다. 그러나, 매 요청마다 해시값을 계산하여 클러스터로 분배 하기 때문에 사용자의 요청수 증가에 따른 각 요청의 Message Digest 계산 시간이 커진다는 단점을 가진다. 그림 3은 MD 5 해싱을 통한 캐시 클러스터 선택을 나타낸다.

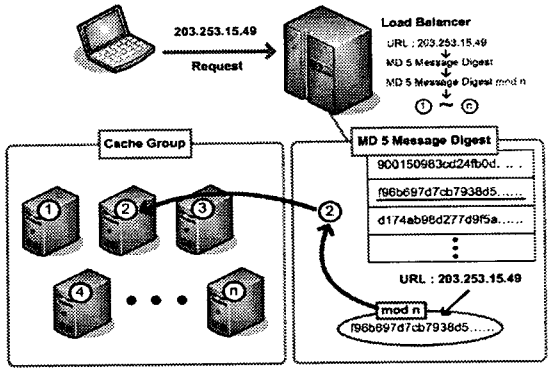

그림 3. MD 5 해싱을 통한 캐시 선택

그림 3의 MD 5 해싱 순서를 살펴보면 다음과 같다.

① 로드 밸런서가 사용자의 요청 URL을 통해 Message Digest를 생성한다.

② 생성된 Message Digest를 캐시 개수(n) 만큼 나머지 연산 한다.

③ 나머지 연산을 통해 나온 값과 동일한 캐시로 요청을 할당한다.

• 동적 해싱(Dynamic Hashing)[3] : 하나의 버킷으로 데이터가 몰릴 경우 버킷을 쪼개고, 합쳐주는 해싱 기법이다. 이 기법은 데이터베이스에서 데이터를 저장할 때 주로 사용되는 방식으로서, 캐시간 협동성을 보장하기 위해 캐시를 선택하는 방식에 사용하기에는 적당치 않다. 그 이유는 해싱 기법 자체가 버킷에 대한 선택이 끝난 후를 대상으로 한 것이고, 캐시에 적용하기 위해서는 버킷을 선택하기 전에 알고리즘이 동작해야 하기 때문이다. 따라서 본 논문에서는 동적 해싱 알고리즘은 고려하지 않았다.

각 해싱기법을 토대로 본 논문에서 실험할 정적 해싱과 MD 5 해싱의 장단점을 비교하면 표 1과 같다.

표 1 정적 해싱과 MD 5 해싱 기법의 비교

		Static Hashing	MD 5 Hashing
요청 URL의 분포	균일	하나의 캐시로 몰리지 않음	하나의 캐시로 몰림
	불균일	하나의 캐시로 몰림	하나의 캐시로 몰림
캐시 추가 & 삭제		매번 버킷을 갱신해야함	영향을 받지 않음
계산량		적다 (버킷 갱신시 계산)	많다 (매 요청마다)

표 1을 보면 MD 5 해싱의 경우 요청 URL에 따른 로드밸런서의 요청 분배가 하나의 캐시로 몰리는 현상이 나타나는 것을 말하고 있는데, 이것은 MD 5 해싱

의 특성에 의한 것으로 URL에 대한 해시, 즉 Message Digest 값이 항상 같기 때문에 요청 URL의 분포가 균일하다 할지라도 고정 길이의 해시값의 분포가 균일하지 못하기 때문에 발생하는 현상이다.

Ⅲ. 실험 및 토론

3.1 실험환경

본 논문에서는 정적 해싱과 MD 5 해싱을 실제 무선 인터넷 프록시 서버 클러스터에 적용하여 실험을 수행한다. 실험 수행 결과는 캐시 클러스터 상황에 따른 최적의 해싱 기법 적용에 대한 기초 자료로 활용될 수 있을 것으로 사료된다. 실험에는 Client 1대, 로드밸런서 1대, 동일 사양의 Host 15대를 사용하였다. 여기서 사용된 로드밸런서는 LVS(Linux Virtual Server)[4]로서 부하 분산기로 동작, Hashing을 통해 Host를 선택할 수 있다. Host는 Hashing비교에 영향을 미치지 않도록 하기 위해 확장성 면에서 가장 뛰어난 성능을 가진 CD-A[5] 구조를 사용하였다. 표 2는 실험에 사용된 하드웨어 사양을 나타내며, 그림 5는 LVS와 Client, 그리고 Host와의 연결상태를 나타낸다.

표 2 실험에 사용된 하드웨어 사양

	하드웨어		개수
	CPU (Hz)	RAM (MB)	
Client	P-III 700 M	128	1
Load Balancer	P-IV 2.4 G	512	1
Host	P-II 400 M	256	15

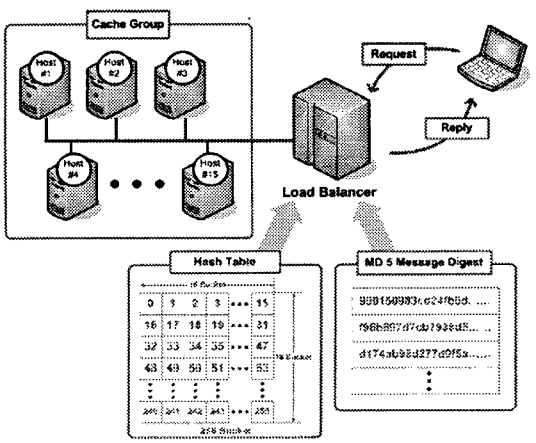

그림 6. 정적 해싱과 MD 5 해싱의 단순 성능비교

3.2 실험결과

정적 해싱과 MD 5 해싱간의 비교를 위하여 성능 비교 부분, URL수에 따른 분포 부분, 그리고 해싱 연산 소요시간 계산의 3부분으로 구분하여 실험 하였다.

① 단순 성능 비교 : 정적해싱과 MD 5 해싱의 현재 구성된 실험장비 상에서 단순 성능 비교를 통해 기본적인 요청/응답에 대한 각 해싱의 성능을 파악하였다. 그림 6은 두 해싱의 단순 성능 비교 그래프 이다.

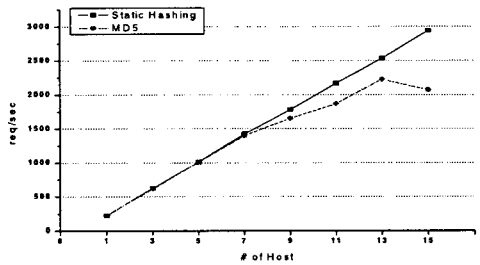

그림 6. 정적 해싱과 MD 5 해싱의 단순 성능비교

그림 6에서 보는 것과 같이 MD 5 의 경우 호스트의 개수가 증가함에 따라서 하나의 캐시로 요청이 몰림에 따라 시스템의 전체 성능이 저하 되는 것을 볼 수 있다.

② 요청 URL수를 증가 할 때의 비교 : 사용자로부터 요청되는 URL의 수가 늘어남에 따라 MD 5 해싱 기법의 경우 하나의 클러스터로 집중되는 현상이 줄어들어 병목 클러스터로 인한 성능저하가 없게 된다. 그림 7은 URL수를 증가 하였을때 MD 5 의 데이터 분산을 나타낸 그래프 이다.

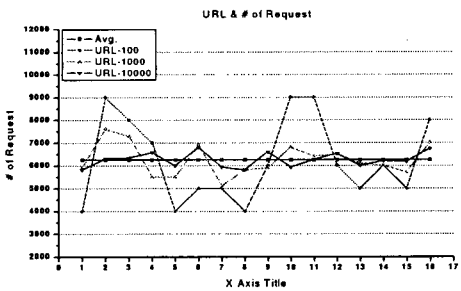

그림 7. URL수 증가에 따른 각 해싱의 성능 비교

그림 7의 그래프를 통해 10만번의 요청이 각 호스트에 6250번씩 분배되는 것을 기준으로 할때, URL수가 100에서 10000으로 점차 증가 할때 평균에 가까워 지는, 즉 각 호스트에 골고루 분산 됨을 육안으로 확인 할 수

있다.

③ 정적 해싱과 MD 5 해싱 소요시간 계산 : 15개의 호스트에 1개의 요청을 하여 해싱 테이블이 생성된 후에 추가 요청을 통해 두요청간에 시간차를 계산, 각 해싱에서 해싱연산을 위한 소요시간을 계산할 수 있다. 표 3은 각 해싱에 처음 1개의 요청 및 소요시간과 다음 추가 요청때의 소요시간을 나타낸 것이다.

표 3 해싱연산 소요시간 계산

	Static Hashing		MD 5 Hashing	
	1st req.	2nd req.	1st req.	2nd req.
request	1	1	1	1
sec	0.006	0.006	0.006	0.006
req/sec	6ms	6ms	6ms	6ms

표 3의 데이터를 보면 각 해싱간에 차이가 없는 것을 볼 수 있는데, 이것은 실험에 2.4Ghz 로드밸런서를 사용했기 때문에 정적 해싱에서의 버킷 생성 시간과 MD 5 해싱 에서의 계산 시간이 줄어든 것이다.

Ⅳ. 결론

본 논문에서 비교한 해싱 기법들의 경우 캐시간 협동성을 보장하나 하나의 URL로 요청이 몰릴 경우 (사용자 요청 URL의 분포가 불균일 할 경우) 이를 해결할 방법이 없다는 단점을 가진다. 이러한 경우에 대해서는 2D 해싱(2D Hashing)이나 이중 해싱(Double Hashing)의 적용을 통해 특정 URL에 의한 클러스터의 병목을 감소 시키는 것을 고려할 수 있다.

Ⅴ. 참고문헌

[1] Static Hashing,
http://computer.org/computer/co1988/rx045abs.htm
[2] D. Rivest, "The MD 5 Message Digest Algorithm", RFC 1321, 1992
[3] Dynamic Hashing,
http://www.eecg.toronto.edu/~ece1762/hw/dhash.pdf
[4] Linux Vitual Server,
http://www.linuxvirtualserver.org
[5] 곽후근, 한경식, 정규식 "클러스터링 기반의 무선인터넷 프록시 서버 성능 개선", 한국정보과학회 봄 학술발표논문집, Vol. 31, No. 1, pp. 406-408, 2004.