한국컴퓨터정보학회 하계학술대회 논문집 제19권 제2호 (2011. 6)

SA 해쉬 알고리즘을 이용한 동일 데이터 업로드 방지 시스템 설계

황성민°, 석호준*, 김석규*
*°안동대학교 정보통신공학과
e-mail: iris8037@nate.com, rucrazy99@naver.com, sgkion@andong.ac.kr

Design of System for Avoiding Identical-Data Upload using SA Hash Algorithm

Sung-Min Hwang°, Ho-Jun Seok*, Seog-Gyu Kim*
°*Dept. of Information Communication Engineering , Andong National University

● 요 약 ●

본 논문은 클라이언트에서 서버로 파일을 전송할 때, 클라이언트가 보내고자 하는 파일이 서버에 동일한 파일로 있다면 업로드를 받지 않고, 서버의 파일을 재사용함으로써 트래픽을 감소시킬 수 있을뿐더러, 스토리지 용량 또한 절약할 수 있는 시스템 설계이다. 본 논문에서 제안하는 해쉬(Hash) 함수 SA를 사용하여 파일에 해쉬 코드를 생성함으로써 다른 해쉬 함수 보다 키 값의 길이가 길어지고 빠른 속도로 해쉬 값을 얻을 수가 있다. SA Hash Algorithm을 통해 얻어진 해쉬 값을 서버로 전송하여 서버에 동일한 파일이 있다면 클라이언트에서 파일을 전송 받지 않고, 서버 내부의 파일을 사용하는 것으로 자원 절감 효과를 낼 수 있다. 서버에서의 파일 관리도 기존의 날짜, 아이디 등 구별 방식이 아닌 SA Hash Algorithm으로 생성된 해쉬 값으로 파일을 관리 할 수 있음으로 파일 관리의 편의성뿐만이 아니라 빠른 속도로 파일을 접근할 수 있다.

키워드: 동일 데이터(Same data), 해쉬 알고리즘(Hash Algorithm)

I. 서론

기존 PC에 제공되던 웹 서비스가 스마트폰의 보급으로 인해 모바일 시장까지 영역을 넓혀가면서 유/무선 인터넷 사용이 급증함에 따라 웹 서비스 시장도 급증하고 있다. 웹 서비스 시장의 규모가 커지고 있고, 유통되고 있는 자료들은 점점 인스턴트화가 되어가고 있다. 한 가지 이슈에 대한 동일한 컨텐츠가 수많은 사용자로부터 등록되고, 이러한 인스턴트 이슈들은 금방 잊혀지고 또 다른 이슈에 대한 중복된 컨텐츠들이 수없이 등록되고 있다. 이렇게 매일같이 반복되는 중복된 인스턴트-컨텐츠들이 차지하는 서버 storage는 어마어마한 규모가 된다. 더 큰 문제점은, 이러한 중복된 자료들을 각기 다른 저장소에 저장함으로 인해 저장공간을 차지하면서 그러한 컨텐츠들이 계속해서 사용자로 하여금 사용되는 유지성에 대해서는 보장 할 수 없다는 것이다. 그리고 이제는 온라인 시장의 규모가 더 커져버린 음악시장, 이용자간의 자료를 거래하는 웹 하드, 개인 블로그 및 SNS 등등 웹 서비스의 전반적인 부분에서 이러한 중복된 컨텐츠를 각기 다른 저장공간에 별도로 저장을 하고 있다.

위와 같은 현황이 웹 서비스를 제공하는 서버에 큰 부담을 줄 수 있다. 동일한 파일에 대해서도 각기 다른 저장 공간을 차지하면서 서버의 저장 공간을 점유하게된다. 게다가 이용자들이 서버에 파일을 Upload하는 트래픽에 대해서도 서버에 많은 부하를 주게 된다. 이러한 단발적으로 끝나버릴 중복된 컨텐츠 및 많은 사용자가 동일한 파일을 이용하는 서비스에 대해서 서버의 저장공간과 Upload시 트래픽에 대한 대비를 해야하기 때문에 서버 구축비용 및 유지비용에 대한 부담이 증가된다.

본 논문에서는 해쉬(Hash)함수와 시스템의 구성으로 서버의 중복적인 파일을 방지 하고 파일 관리를 하는 방법을 구성함으로써 효율적인 파일 관리가 가능하게 한다.

II. 본론

1. 시스템 구성

기존 데이터 서버에서는 각각의 파일을 각기 다른 별도의 storage에 저장시킨다. 이는 동일한 파일에 대해서도 별도로 저장공간을 차지하고 있어 storage의 저장 공간에 부담을 주게 된다. 이에 본 논문에서는 기존 기술의 자원 낭비에서 비롯된 문제점을 개선하기 위해 자체적인 해쉬 알고리즘을 통한 동일한 파일을 구분하는 check 시스템을 통하여 기존 데이터 서버에 저장되어 있는 파일과 동일한 파일에 대해서는 새로이 Upload를 시키지 않고, 같은 파일에 접근하여 이용자들이 read할 수 있도록 한다.

데이터 서버에 이용자들이 파일을 Upload하기 직전, 자체적인 해쉬 알고리즘을 통하여 데이터 서버에 동일한 파일 있을 경우 재

Upload를 시키지 않음으로써, Upload시 발생할 수 있는 트래픽은 물론, 시간 및 데이터 서버의 저장공간 절약을 도모할 수 있다. 과도한 서버 트래픽에 의한 유지비용 및 데이터 접근 속도의 문제로 인해 요즘 많이 사용하고 있는 캐쉬서버에 대해서 본 논문의 기술을 대입 시키면, 기존 캐쉬서버의 문제점이었던 적은 저장 공간을 효율적으로 사용함은 물론, 원 서버의 트래픽에 대한 부담을 줄일 수 있어 서버 유지비용의 절감 및 데이터 접근 속도의 향상 및 캐쉬 서버의 활용률의 증가를 기대할 수 있다.

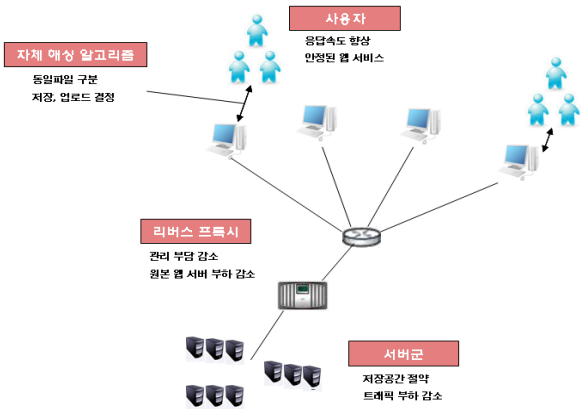


그림 1 중복 데이터 방지 서버 구조

2. 데이터 중복 비율

표 1. 이미지 데이터 중복 개수

	Naver	Nate	Google
키워드 A	8개	7개	7개
키워드 B	5개	9개	5개
키워드 C	13개	13개	12개

3가지의 키워드로 이미지 검색하여 100개 중 몇 개가 중복된 데이터인지 나타낸 표이다. 동일한 이미지가 다수가 검색 되었고, 평균적으로 8.7%로 조사되었다. 동일한 이미지의 3~5개 정도가 저장공간을 중복 점유하고 있다. 이를 본 논문의 기술로 저장공간을 절약한다면 평균 7.1%의 절약을 가져 올 수 있는 수치이다.

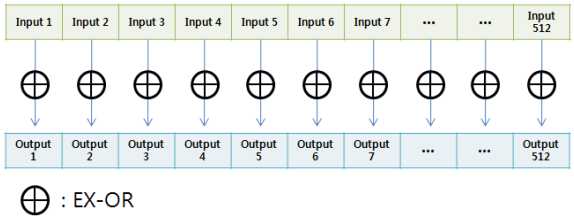
표 2. 동영상 데이터 중복 개수

	A 웹하드	B 웹하드
동영상 컨텐츠	26.4%	27.5%

웹하드 표본에서는 동영상 컨텐츠가 많이 보관되어 있는 사이트 중 2사이트를 선정하고 측정 1시간 동안 업로드 되는 컨텐츠 중 가장 많이 업로드 되는 중복된 동영상 컨텐츠가 업로드된 컨텐츠중 얼마의 비중을 차지하고 있는지 조사하였다. 이는 각각 26.4%와 27.5%로 조사되었다. 이는 동일한 인코딩 방식과 컨텐츠 이며 동일한 용량을 차지하고 있는 파일이다. 이를 본 논문의 기술로 적용한다면, 동일한 컨텐츠로 인한 서버의 접유를 방지 할 수 있다.

3. 해쉬(Hash) 함수

본 논문에서 사용하려는 해쉬 함수는 파일을 구별하고자하는 식별 코드이기 때문 해쉬함수의 공격에 있어서는 그렇게 중요하지 않다. 그리하여 기존에 사용되는 해쉬 함수 SHA와 MD 시리즈가 아닌 본 논문에 최적화 된 해쉬 함수를 사용하고자 한다. 이를 SA Hash 라고 칭하겠다.



SA 해쉬 함수는 시간 복잡도가 O(n)으로 다른 해쉬함수 보다 연산과 반복이 적은 것이 특징이고, 해쉬 값을 가변 적으로 쉽게 변환된다는 것이 또 다른 특징이다.

개념은 입력되는 값을 각 자리별로 EX-OR을 해주면 된다. 이는 파일을 구별하기 위한 본 논문에 있어 최적화된 알고리즘이라 할 수 있다. SA로 생성된 코드는 서버 storage의 파일 경로로 사용한다.

표 3. SA Hash Algorithm 특성

알고리즘	해쉬값 크기	내부 상태 크기	블록 크기	워드 크기
SA	512	512	1024	64

SA 알고리즘은 본 논문에서 동일 파일인지 검사할 때 사용된다. 해쉬 함수의 특성상 동일한 해쉬 값 H를 갖는 M과 M'가 존재함으로 유일성을 보장하지 못한다.

SA 해쉬값 H가 같아질 확률이 1/2이 되려면 서명문 M의 수가 몇 개나 있어야 하는가 하는 문제는

$$K = 1.17\sqrt{M}$$

이다 SA는 512비트의 해쉬값을 가지기 때문에 안전성을 보장할 수 있다. 생성된 해쉬 값으로 서버의 폴더 경로가 생성이 되고, 그 폴더 안에 파일이 저장된다. 만약 동일한 해쉬 값을 가지는 다른 파일이 서버에 전송되게 되면 같은 폴더에 저장 하게 되지만 파일 크기로 각각의 파일을 구별할 수 있게 되는 것이다. 해쉬 값 512비트와 파일의 크기로 구별하게 된다면

1/1.375*10^77 으로 0에 수렴하는 경우의 수 즉, 다른 파일간의 동일 해쉬코드가 나올 확률이 된다.

그리고 만약 동일한 해쉬값을 가지는 파일이 있더라도 파일의 용량을 통한 최종 비교를 하기 때문에 중복될 확률은 지극히 작다고 할 수 있다.

4. Retain Count

본 논문의 시스템은 한 파일에 여러 사용자가 접근하기 때문에 파일 관리에 있어서도 중요하다. 사용자가 컨텐츠를 올리고자 할 때, 컨텐츠에 포함되는 파일들의 해쉬 값을 서버에 보내게 된다. 해쉬 값과 파일의 파일 크기가 동일 하다면 Upload를 받지 않고 서버에 있는 파일에 링크만 걸어주게 된다. 링크를 걸 때 Retain Count를 증가하게 되고 상용자가 컨텐츠를 삭제하거나 파일을 삭제하게 되면 Retain Count를 감소 시키게된다. Retain Count가 0이 되면 서버에서 그 파일은 삭제 하게된다.

III. 결론

1. 해쉬 알고리즘 성능 분석

Algorithm	Mib/Sec
SA	267
MD5	88
SHA-1	52
SHA-256	38
SHA-512	34

성능 분석 환경은 Intel Core2 Duo E8400 3.00Ghz 32비트 Windows7 으로 테스트를 하였다.

같은 길이의 해쉬 값을 발생하는 SHA-512와 SA Hash Algorithm의 중복코드를 발생할 확률은 같지만, 속도 적인 측면에서는 월등한 차이를 보이고 있다.

SA는 다른 해쉬 알고리즘에 비해 단순한 연산을 가지고 있어 보안 성능은 기대하기 어렵지만, 본 논문에 있어서 SA 해쉬 알고리즘은 서명 인증에 사용되는 보안 기술이 아니기 때문에 보안 성능은 고려하지 않았다.

2. 중복 제거 효과

	일반 웹사이트	자료공유 웹사이트
데이터 중복율	평균 8.7%	평균 26.9%
중복 제거 효과	7.1.% 절감	21.8% 절감

일반 적인 웹사이트에서는 데이터 중복이 많이 일어나지는 않는다. 동일한 이미지 파일이더라도 크기와 내부 정보 들이 제각기 다 다르기 때문에 중복의 확률이 낮아 본 논문의 기술을 적용하는 것 보다 일반 적인 방법으로 서버의 파일을 보관하는 것이 더 효율적이다. 하지만 자료 공유를 목적으로 하는 웹사이트나 동영상 서비스 등 파일 용량이 크고 파일의 중복률이 높은 웹사이트에서는 본 논문의 기술을 적용하게 되면 평균 21.8%의 저장 공간 및 업로드 트래픽을 감소할 수 있다.

향후 클라우드 컴퓨팅 시스템에도 본 논문의 기술을 적용하게 된다면 서버 저장공간 및 트래픽에 있어서 큰 효과를 나타낼 것으로 보여 진다.

참고문헌

[1] SHA, MD5 wikipedia, http://en.wikipedia.org/wiki/Sha-2

[2] S. Kelly, S. Frankel "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec" RFC4868

[3] Crypto++ http://www.cryptopp.com/

[4] 원동호 저 “현대 암호학” 2005

[5] 이만영, 원동호, 이민섭, 송주석, 임종인, 박춘식 공저 “현대 암호학 및 응용” 2002