대표 해시 기반의 FLC를 이용한 파일 유사도 평가 기법

유영준*, 고영웅
한림대학교 컴퓨터공학과
e-mail:willow72@hallym.ac.kr

Similarity Evaluation Scheme Using FLC based Representative Hash

Young-Jun Yoo*, Young-Woong Ko*
*Dept of Computer Engineering, Hallym University

요 약

유사도 평가는 유사 파일 탐색이나 파일의 중복제거에서 필수적으로 수행되어야 하는 모듈이다. 이와 같은 유사도 평가는 파일의 크기가 커지거나 비교할 파일의 수가 많을수록 더 많은 시간이 소요되므로 이 때 발생하는 오버헤드는 시스템 전체의 성능에 영향을 미칠 만큼 중요하다. 특히 사용자의 요구사항에 실시간으로 반응해야 하는 시스템에서는 응답시간을 지연시키는 요인이 된다. 본 논문에서는 파일의 해시 연산과정의 시간을 줄이기 위한 방법으로 대표 해시 FLC를 이용한 유사도 평가 시스템을 제안한다. 실험을 통해서 본 연구에서 제안하는 방식이 기존의 방식에 비해서 빠른 시간 내에 유사 파일을 탐지할 수 있음을 보이고 있다. 또한 해시 리스트의 크기가 줄어들어서 메모리 자원을 효율적으로 사용할 수 있다.

1. 서론

파일의 유사도 평가는 파일의 중복 제거 시스템이나 유사 파일의 탐지 등의 분야에서 사용된다. 중복 제거 시스템에서 파일 저장 시 먼저 파일의 유사한 부분을 찾아 중복을 제거하고, 그 부분을 제외한 나머지 부분을 저장함으로써 컴퓨팅 자원을 절약 할 수 있다. 또한 유사 파일의 탐색은 시스템 내에 유사한 파일을 검색하거나 실행프로그램의 유사도를 파악해 악성코드를 찾아내는 데에 쓰인다. 유사 파일 검색 시스템에서 유사도를 평가하기 위하여 사용되는 모듈의 수행 속도는 시스템 전체에 큰 영향을 미친다. 실시간으로 사용자에게 서버의 유사 파일 리스트를 제공해야 하는 시스템에서 지연 시간은 치명적인 약점일 수 있다.

본 연구에서 제안하는 유사도 판별 시스템은 검색하고자 하는 사용자의 대상 파일과 웹(Web)상에 있는 파일들 간에 유사도를 비교하며, 그 결과 가장 유사도가 높은 파일들을 사용자에게 제공하는 것을 목표로 한다. 본 연구에서는 유사도를 비교하기 위한 파일들의 해시 리스트 연산을 고속화 하는데 있어서 대표 해시 기법을 사용하고 있다. 제안하는 방법은 서버에서 계산된 해시 리스트와 클라이언트에서 요청한 파일의 해시 리스트들 간의 비교연산이 매우 빠르다. 대표 해시 기반 FLC(Fixed Length Chunking)[1]는 블록 단위로 데이터를 읽어 대표 해시가 될 수 있는 블록으로 판단되면 해시 연산을 수행하며, 해당 해시 값을 이용해 서버의 해시 리스트와 비교하고 유사도를 판별해 사용자에게 정보를 제공하는 방식이다. 기존의 순수한 FLC 기법을 사용하여 유사도를 찾는 방법과 본 연구에서 제안하는 대표 해시 기반 FLC 기법을 비교하면 정확도 측면에서는 비슷한 결과가 나오지만 속도면에서는 제안하는 기법이 매우 빠르게 처리되는 장점이 있다.

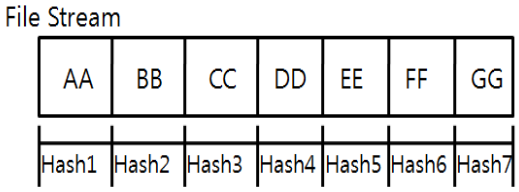
본 논문의 관련 연구에서 본 연구팀이 제안하는 기술의 필요성과 배경 기술이 되는 FLC 및 VLC(Variable Length Chunking)[2] 등에 대한 설명을 한다. 3장의 시스템 설계에서는 대표 해시 FLC의 개념에 대한 설명을 하며, 4장 실험 부분에서 시스템의 성능과 오차를 보이며, 본 연구의 타당성을 주장한다. 마지막으로 결론을 기술한다.

이 논문은 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2012R1A1A2044694), 또한 정보통신산업진흥원의 IT/SW 창의연구과정의 연구결과로 지식경제부와 NHN의해 지원된 과제로 수행되었음.

2. 관련 연구

유사도 비교를 위해서 사용을 하는 청킹(chunking) 기법은 크게 고정 분할 방식(FLC)과 가변 분할 방식(VLC)으로 구분된다. VLC 연산 모듈을 사용해서 파일의 유사도를 추출하는 방법은 정확도가 높지만 서버의 파일들을 해

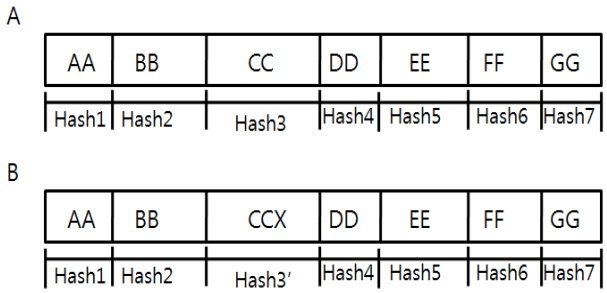
시 연산하는 과정과 해시 리스트를 서로 비교하는 과정에서 많은 오버헤드가 발생된다.



(그림 1) FLC 개념도

고정길이 기반의 연산방식은 그림 1처럼 파일을 일정 크기의 블록으로 나누고, 각 블록을 해시 함수(MD5, SHA1 등)을 통해 해시 값을 구하는 방법이다. 이 해시 리스트를 가지고 비교하고자 하는 파일의 해시 리스트와 값을 상호 비교하여 유사도를 판별한다. 고정 길이 방식은 일정 크기 블록단위로 해시연산을 수행하기 때문에 다른 해시 연산에 비해 수행속도가 빠르다. 특히 데이터의 변경이 파일의 후반부에서 이루어질 경우 전반부의 해시 값은 변화가 없기 때문에 비교적 정확하게 유사 파일을 찾을 수 있다. 하지만 데이터의 변경이 전반부나 중간에 이루어지게 되면 “바이트 쉬프트(Byte Shift)” 현상에 의해서 이후에 중복된 데이터들을 찾기가 어려운 문제가 있다.

가변 길이 기반의 유사도 탐색은 라빈 핑거 프린트(Rabin Fingerprint)와 같은 함수를 이용해 파일에서 각 블록을 구분할 경계(Anchor)를 찾아 블록으로 나눈 후 해시 값을 구하는 방식이다.

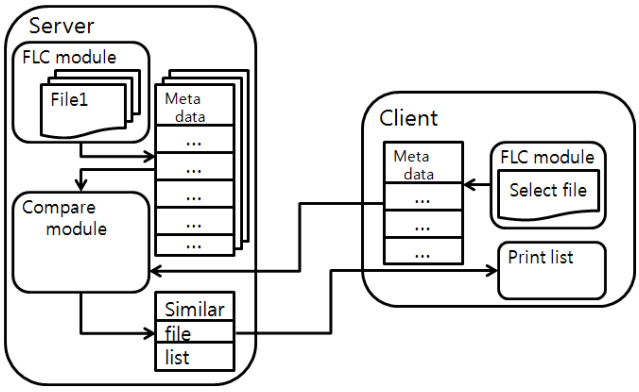


(그림 2) VLC 개념도

그림 2와 같이 각 경계를 기준으로 청킹을 수행하기 때문에 각 블록의 사이즈는 가변적인 크기를 갖게 된다. 이 때 데이터가 중간에 삽입이 되더라도 경계로 구분된 각 블록들의 해시 값은 변화가 없다. 따라서 이 방법을 이용하면 높은 정확도로 동일한 블록들을 찾을 수 있다는 장점이 있다. 하지만 라빈 핑거프린트로 경계를 찾고 경계를 찾은 후 해당 블록을 다시 해시 연산하는 작업을 수행하기 때문에 오버헤드가 매우 높다. 이 밖에도 Zhu[3]의 블룸필터를 이용한 연구가 있으나 블룸필터의 크기가 다양하며 해시 함수를 두 개 이상 사용하기 때문에 그에 따른 오버헤드가 증가하는 등의 문제가 발생한다.

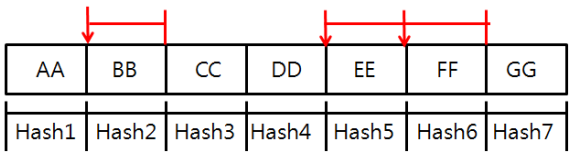
3. 시스템 설계 및 구현

본 연구팀에서 제안하는 유사도 판별 시스템은 사용자가 제공하는 대상 파일을 입력받아서 기존에 데이터베이스에 등록된 파일들의 메타데이터 정보를 이용하여 가장 유사한 파일의 목록을 제공하는 시스템이다. 본 연구에서는 사용자에게 최소한의 지연시간 내에 응답을 줄 수 있는 시스템을 목표로 하고 있다. 현재 유사도 판별 시스템은 파일의 크기가 크거나, 서버에서 탐색해야 하는 파일의 종류가 많을수록 서버의 파일을 해시 연산하는 과정이나, 파일의 해시 값을 비교해 유사도를 판별 하는 모듈에서 큰 시간 지연이 발생한다. 다음 그림 3은 본 연구에서 제안하는 유사도 판별 시스템의 구조도를 보이고 있다.



(그림 3) 유사도 판별 시스템 구조도

따라서 본 연구에서는 대상 파일에 대해서 빠른 해시 연산을 수행하기 위하여 대표 해시 기반 FLC 기법을 제안한다. 대표 해시 기반 FLC는 파일을 구성하는 각 청크들의 해시 값 중에서 대표의 성격을 갖는 일부 해시 값으로 전체 해시 값을 대표하는 방식이다. 본 연구에서는 각 경계의 시작 바이트 값이 동일한 블록을 해시한 값을 대표 해시로 사용하였다. 즉, 일정한 크기의 블록으로 청킹을 수행한 후에, 블록의 시작 바이트 값이 ‘a’인 블록들의 해시값을 대표 해시로 정하는 방식이다. 따라서 모든 블록에 대해서 해시를 생성하고 이 중에서 상위의 값을 추출하는 기존의 FLC에 비해서 더욱 작은 오버헤드로 파일을 대표하는 해시값들을 추출할 수 있는 것이다.



(그림 4) 대표 해시 FLC 개념도

그림 4는 대표 해시 기반 FLC의 개념을 보여주고 있다. 대상 파일에 대해 동일한 블록 크기로 나눈 후 블록의 맨 앞에 경계를 나눌 수 있는 경계 바이트(Anchor Byte)를 기준으로 블록 분할을 수행한다. 그리고 이 경계를 기준으

로 블록 해시 연산을 수행한 후 이 값을 대표 해시로 사용하기 위해 리스트에 저장한다. 다음 그림 5는 제안하는 대표 해시 기반 FLC의 의사 코드를 보이고 있다.

```
List FLC(InputFileStream in)
    offset ← 0
    anchor_byte ← 'a'
    fdlength ← Length(in)
    while offset < fdlength do
        offset ← seek(in,seek_cur)
        block ← read(in, offset, blocksize)
        if(block[0] == anchor_byte) then
            hash ← hashing_FLC(block)
            HashList += (hash, offset, true)
```

(그림 5) 대표 해시 기반 FLC 알고리즘

그림 5는 앞에서 설명한 대표 해시 기반 FLC 알고리즘을 보이고 있다. 파일 스트림을 이용해 파일의 처음부터 끝까지 동일한 크기의 블록을 읽어 이 블록의 첫 번째 바이트를 비교한다. 이 바이트가 경계를 구분하는 anchor_byte일 경우 이 블록은 대표 해시 값으로 지정하고 리스트에 저장한다. 이 방법을 사용하면 모든 블록에 대해 해시 연산을 할 필요 없이 대표 해시 값이 될 수 있는 특정 블록에 대해서 해시 연산을 수행하기 때문에 해시 연산을 하는 횟수와 수행속도가 줄어들고, 파일 비교를 위한 해시 값의 개수도 줄어든다. 만약 블록 단위가 아니라 기존의 롤링 체크섬과 같이 바이트 단위로 이동하면서 해시를 생성하는 경우에는 마찬가지로 시간이 더욱 지연될 수 있다.

4. 실험

(표 1) 실험을 위한 플랫폼

OS	Windows 7
CPU	Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz
RAM	4GB
Platform	Java JDK 7

본 실험은 표 1과 같은 플랫폼에서 진행 됐으며 각 실험 데이터는 리눅스 상에서 *dd* 명령어를 통해 1GB 크기의 원본 파일과 파일의 일부가 수정된 수정 파일을 하나의 쌍으로 생성했다. 각 파일은 원본 파일과 90%, 80%, 그리고 70%가 일치한다. 이 파일을 이용해 기존의 FLC 기법과 제안하는 대표 해시 기반의 FLC 유사도 모듈의 성능을 측정 및 비교하였다. 또한 실제 유사도와 대표 해시 FLC를 이용했을 때 유사도를 비교함으로써 오차의 크기를 측정했다.

(표 2) 일반 FLC와 대표 해시 FLC의 성능비교

	FLC	대표 해시 기반 FLC
해시 개수	128000	479
실행 속도(초)	0.305	0.003

표 2은 일반 FLC방식과 대표 해시 FLC를 비교한 내용이다. 동일한 파일로 해시 연산을 수행했을 때 일반 FLC는 128,000개의 해시 값이 생성된 반면 대표 해시 FLC는 500개 미만의 대표 해시가 만들어졌다. 또한, 각 방법의 해시 값 비교 모듈의 성능을 측정했을 때 일반 FLC는 유사도 연산과정에서 0.305초가 걸린 반면 대표 해시 FLC는 0.003초가 소요됐다.

(표 3) 대표 해시 FLC의 수행 결과 오차

실제 파일 유사도	수행 결과(%)	오차율(%)
90(%)	88.4	1.6
80(%)	81	1
70(%)	73.8	3.8

표 3는 대표 해시 FLC의 수행 결과 오차를 나타낸다. 실제 90%만 일치하는 두 개의 파일을 비교했을 때 수행 결과는 88.4%로 1.6%의 오차를 보인다. 80%가 일치하는 파일에 대해서도 1%의 적은 오차를 보였고 70%의 파일에 대해서도 5% 미만의 오차를 나타내고 있다.

5. 결론

본 연구에서는 기존의 FLC 기법의 오버헤드를 줄이면서 대등한 성능을 보이는 대표 해시 기반의 FLC 방법을 제안하고 있다. 제안하는 방법은 각 블록의 경계 바이트의 값이 특정한 값에 해당하는 경우에 대표 해시로 처리함으로 빠르게 해시 값을 구할 수 있다. 성능 비교를 통하여 제안하는 방법이 작은 오차율을 보이면서 빠르게 유사 파일을 탐지함을 보였다.

참고문헌

[1] Quinlan, S. and Dorward, S., "Venti: a new approach to archival storage", Proceedings of the FAST 2002 Conference on File and Storage Technologies, 2002.

[2] Athicha M., Benjie C., and David M.. "A Low-Bandwidth Network File System." In Proceedings of the Symposium on Operating Systems Principles (SOSP'01), pp. 174 - 187, 2001.

[3] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," in Proceedings of the Seventh USENIX Conference on File and Storage Technologies (FAST), pp. 269 - 282, 2008.