해시 압축을 이용한 파일 동기화 시스템

박재민, 정호민, 고영웅
한림대학교 컴퓨터공학과
e-mail:{jm-park, hmjung, yuko}@hallym.ac.kr

File Synchronization System Using Hash Compression

Jae Min Park, Ho Min Jung, Young Woong Ko
Dept of Computer Engineering, Hallym University

요 약

본 논문에서는 가변 길이 분할을 사용한 파일 동기화 시스템에서 메타 데이터 교환 오버헤드를 최소화 하기위한 효율적인 접근 방식을 제안한다. 본 논문의 주요 아이디어는 해시 압축 알고리즘을 사용하여 여러 개의 해시키를 하나의 해시키로 치환하여 메타 데이터 교환 비용을 최소화 하는 것이다. 본 논문에서는 제안 알고리즘의 타당성을 증명하기 위해 기존의 파일 동기화 프로그램과 비교 실험하였으며 통신비용과 에너지 소모 실험을 통해 메타 데이터를 10배 이상 줄일 수 있음을 보였다.

1. 서론

최근 디지털 데이터의 양이 기하급수적으로 늘어나고 있다. 따라서 데이터 스토리지 시스템을 설계할 때 데이터를 효율적으로 저장하기 위한 방법을 고려해야 한다. 원본 위치에서 타겟 위치로 파일을 복사하는 파일 동기화는 스토리지를 업데이트 하거나 백업을 하는데 있어 모바일 장치에 특히 유용하게 사용된다. 파일 동기화에서 중복 파일의 전송을 피하기 위해 백업이나 새로 저장하고자 하는 파일을 전송하기 전에 메타 데이터를 교환해야한다. 예를 들어, 일반적으로 중복 제거 기술을 사용하여 파일 동기화를 하는 Rsync와 Duplicati는 동기화 하고자 하는 파일을 블록별로 나누어 해시 키 값을 구하는데, 파일 크기에 비례하는 해시 데이터를 생성하고, 이 해시들을 교환하여 대상 위치의 해시 키 값들과 전체적으로 비교를 해 중복 블록을 제거한다.

이러한 파일 동기화 시스템들은 데이터를 교환 및 처리하기위해 많은 메타 데이터 교환이 필요하다. 이러한 메타 데이터 교환은 통신비용을 증가시키고 에너지를 더욱 소모하는 문제를 야기한다. 본 논문에서는 파일 유사도 정보 및 해시 키 압축 기술을 사용하여 교환하는 메타 데이터의 교환 오버헤드를 최소화하는 방법을 제안한다. 제안된 시스템은 기존의 파일 동기화 시스템에 비해 메타데이터 교환에 필요한 데이터 오버헤드를 크게 줄일 수 있다.

메타데이터 교환 오버헤드 효과를 실험하기 위해 10MByte 파일에서 100MByte 파일까지의 중복을 10%에서 90%변경하고 기존의 중복제거 프로그램인 Duplicati와 본 논문에서 제안하는 방법을 비교하였을 때 10MByte파일에서는 10배의 차이를 보이고 있으며 100MByte의 파일에서는 30배의 차이를 보이고 있음을 보여 파일이 증가할수록 본 논문에서 제안한 방법이 효율적임을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 연구 내용을 살펴보고, 3장에서는 제안하는 시스템의 설계와 구현 내용을 설명하고, 4장에서는 메타데이터의 해시 압축 방식을 사용한 방법의 성능 평가 및 분석을 한다. 5장에서는 결론과 향후연구내용을 설명한다.

본 연구는 교육과학기술부와 한국연구재단의 지역혁신인력양성사업과 기초연구사업(No.2009-0076520)의 지원을 받은 결과물임을 밝힙니다.

2. 관련연구

자료를 교환하는 방법과 절차를 동기화 방식이라고 하며 현재 개발된 파일동기화는 ActiveSync[1], HotSync[2]와 같이 전체 데이터를 전송하는 방식이 있으며 CPISync[3]와 같이 데이터의 변경 사항을 특정 형태로 구조화 하여 관리하는 방식이 있다. 그러나 이러한 파일동기화들은 파일간의 중복을 구분하는 것이 아니기 때문에 버저닝(versioning) 시스템 같이 데이터가 자주 변경되고 중복이 많은 시스템에서 사용하기 어렵다. 중복을 처리하면서 파일 동기화하는 대표적인 연구로는 Rsync[4], LLRFS[5], tpsync[6]가 있으며 데이터 중복제거 기술에 관한 연구로는 Venti[7], LBFS[8] 등이 있다. Rsync는 롤링 체크섬(Rolling Checksum)이라는 중복 데이터를 검색하는 알고리즘을 사용해 새로운 데이터의 복사만 일어나게 한다. LLRFS는 CDC(Contents-define Chunking) 방식과 set reconciliation 기술을 사용하여 중복 제거량과 네트워크 트래픽을 감소시켰다. tpsync는 CDC 방법으로 중복이 있는 부분을 대략적으로 찾아내고 롤링 체크섬을 사용해 중복이 있는 부분에 대해서 세밀하게 중복 제거하는

기법이다. Venti는 중복 데이터를 제거하여 저장하는 스토리지 시스템이며 파일을 고정된 크기의 블록으로 나누고 각 블록에 SHA1[9] 해시를 적용하여 중복을 제거한다. LBFS는 품질이 좋지 않은 네트워크 환경을 고려해 설계된 네트워크 파일시스템이며, 파일전송 효율을 높이기 위해 CDC 방식을 사용하며 Rabin Fingerprint[10]로 해시하여 특별한 값을 경계로 하고 데이터 전송 전에 경계 사이의 블록을 SHA1 또는 MD5[11]같은 해시로 만들어 중복을 제거한다.

3. 시스템 설계 및 구현

본 논문은 파일 동기화 작업에서 파일 유사도의 개념을 적용하여 파일을 블록단위로 나누워 각 블록별로 해시 키 값을 구해 서버와 클라이언트의 비교를 통해 파일 유사성을 알아낸다. 만약 서버가 클라이언트의 파일과 동일한 해시 키 값을 가지고 있다면 서버에 동일한 데이터 블록이 있다고 가정한다. 따라서 클라이언트가 서버로 중복된 데이터를 전송하는 것을 피할 수 있다.

그리고 서버와 클라이언트간의 메타데이터를 교환 및 비교 절차를 수행 할 때 여러 개의 해시 키 값을 압축하여 하나의 해시 키 값으로 만들어 비교 하는 방식을 사용하여 오버헤드를 줄인다.

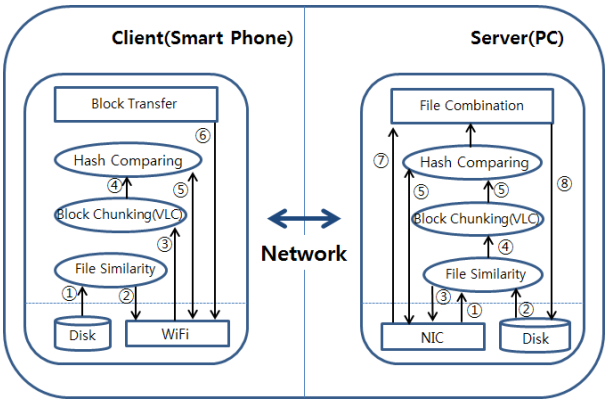
3.1 유사도 비교

유사도 비교는 바이트 단위로 연속 해시가 가능한 Rabin해시를 사용하여 데이터에 대한 해시 키 값을 생성한다. 결과 값은 오름차순 또는 시스템 구성에 의한 내림차순으로 해시 키 값에 여러 숫자를 포함 할 수 있다. 이렇게 구해진 해시 키 값들은 파일 유사성 검색을 위해 사용된다. 서버와 클라이언트간의 해시 값 비교에서 유사도를 발견하였을 때 데이터 블록 간에 중복이 있는 것으로 간주하여 본 연구에서는 총 10 MByte의 데이터에 대해 500KByte당 1개의 해시 키 값을 추출하여 총 20개의 해시 값을 구했다.

메타데이터 교환 모듈에는 해시 리스트와 파일 오프셋 정보들이 들어있다. 해시 리스트에는 각 블록에 대한 해시 값들이 포함되어 있고 파일 오프셋정보에는 파일 블록마다 상응되는 위치 정보들이 포함되어 있고 이것을 파일 동기화 모듈에서 활용을 한다.

3.2 동기화 시스템의 설계

(그림 1)은 스마트폰 클라이언트와 서버로 구성이 되어있는 시스템 구성을 나태내고 있다. 시스템을 구성하는 주요 모듈로는 파일들의 해시 키 값 생성과 유사도를 계산하는 유사도 검색 모듈(File Similarity), 파일을 중복감지 할 수 있도록 가변길이 중복제거 기법으로 블록들을 나누고 각 블록을 해시하는 중복제거 모듈(Block Chunking), 클라이언트에서 서버로 보내고자 하는 파일의 중복된 부분을 감지하고 중복 된 부분을 기존의 파일에서 복사시키는 중복감지 모듈(Hash Comparing), 클라이언트에서 중복이 아닌 데이터를 서버로 전송하는 블록전송모듈(Block Transfer), 서버에 존재하는 파일에서 복사한 내용과 클라이언트에서 전송한 내용을 합쳐 클라이언트에서 보낼 파일과 같은 파일을 합성하는 파일합성모듈(File Combination)로 구성되어있다.

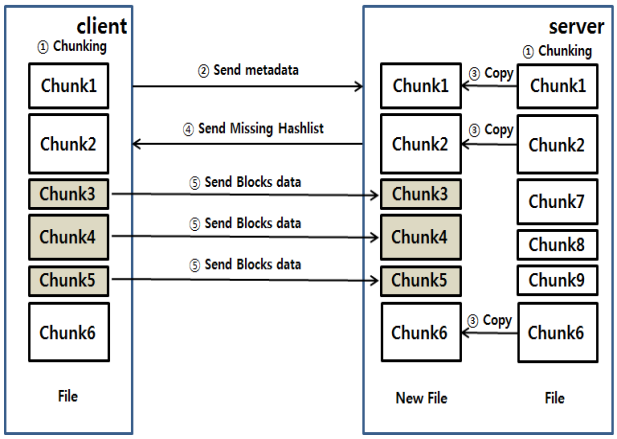


(그림 1) 시스템 구성 및 흐름

전체 흐름을 기술하면 다음과 같다. 수행 순서는 원호안의 숫자로 표시되어 있다. 클라이언트에서 ①, ②단계를 통해 동기화 하려는 파일들을 Rabin함수를 사용하여 해시 키 값과 파일내의 오프셋 정보들을 서버로 전송한다. 서버는 ①, ②를 통해 클라이언트로부터 수신한 해시 키 값들과 서버 파일들의 해시 키 값들을 본 논문에서 제안하는 해시 압축방식으로 비교하고 파일 유사도가 높은 파일을 선택하여 ③, ④를 통해 클라이언트로 유사도 결과를 보내주고 중복제거 알고리즘을 이용해 블록들로 나누고 각 블록들을 해시한다. 클라이언트는 ③, ④을 통해 서버로부터 유사도 결과를 받고 중복제거 알고리즘을 이용해 파일을 블록들로 나누고 각 블록들을 해시한다. 각 블록들이 해시가 되면 서버와 클라이언트에서 ⑤를 통해 해시 키 값들을 비교하여 중복된 데이터를 찾고 마지막 단계에서 서버는 클라이언트로부터 수신한 중복이 아닌 데이터와 기존에 중복되는 데이터를 합하여 동기화 작업을 끝낸다.

3.3 파일합성 과정

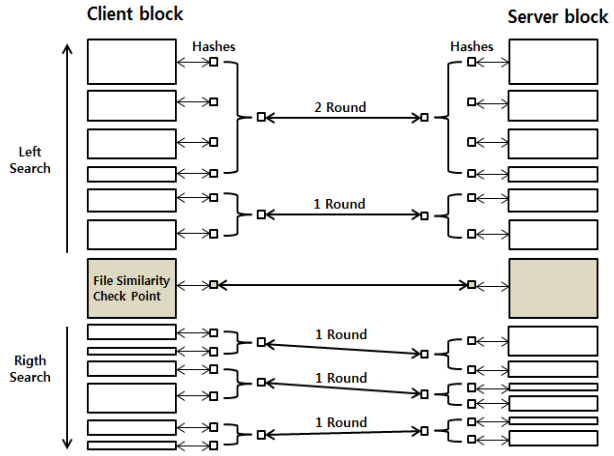
(그림 2)는 파일을 합성 하는 단계를 나타낸 것이다. ① 클라이언트에서 가변길이 분할 알고리즘을 사용하여 동기화 하고자 하는 파일을 여러 개의 블록으로 나누어 해시 키 값을 구한다. 서버는 유사도가 높은 파일에 대해 클라이언트와 동일하게 가변길이분할 알고리즘을 사용하여 블록으로 나눠진 데이터들의 해시 키 값을 구한다. ②클라이언트는 서버에 해시 리스트를 전송한다.



(그림 2) 파일 합성 과정

③서버는 클라이언트와 동기화 될 파일의 크기와 동일한 공간을 디스크에 만들고, 클라이언트로부터 수신한 해시 리스트와 서버의 해시 리스트를 비교하여 중복이 되는 블록은 서버상의 데이터에서 새롭게 만든 파일공간에 바로 복사를 한다. ④중복이 되지 않은 블록에 대해서 리스트를 만들어 클라이언트로 전송한다. ⑤클라이언트는 중복이 아닌 블록 리스트를 수신하여 서버에는 없을 데이터 블록들을 서버로 전송한다. ⑥서버는 클라이언트가 보내온 블록들을 수신하여 준비되어 있는 파일 공간에 복사하여 클라이언트에서 동기화 하려는 파일과 동일한 파일을 생성하여 동기화를 끝낸다.

3.4 해시 압축 방식

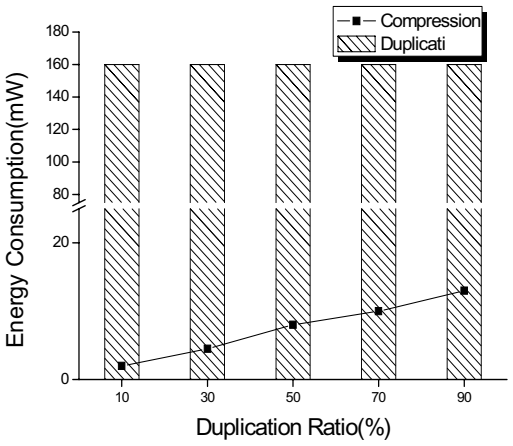


(그림 3) 해시 압축 비교 개념도

(그림 3)은 해시 압축 비교 개념을 나타낸 그림으로 파일 유사성 체크 포인트(File Similarity Check Point)는 파일 유사성 처리 절차 동안 습득되어 질수 있는 같은 블록 위치를 의미한다. 실제로 이 포인트의 근처의 클라이언트와 서버의 데이터 블록은 서로 중복되었다. 본 논문에선, 최초 파일 유사성 체크 포인트로부터 양 방향으로 해시 키 값을 비교하면서 파일 동기화 처리 절차를 시작한다. 클라이언트와 서버 각각의 블록 해시 키 값을 비교하는 과정에서 만약 블록이 동일하다면 두 번째로 압축된 해시 키 값을 비교하는 처리 절차를 시작한다. 여기서 두개의 해시 키 값을 간단히 연결하고, 압축된 하나의 해시 키 값을 생성한다. 계속해서 서버와 클라이언트 해시 키 값이 동일하다면 클라이언트와 서버 사이의 해시 키 값이 동일하지 않을 때 까지 압축 시키는 해시 키 값들의 개수가 기하급수 적으로 증가한다. 이와 다르게 클라이언트와 서버 사이의 해시 키 값이 동일하지 않다면 다시 하나의 해시 키 값으로 비교를 시작한다. 이 같은 방법의 알고리즘 사용으로 클라이언트와 서버 사이의 교환 되는 해시 키 값들의 개수를 줄일 수 있다.

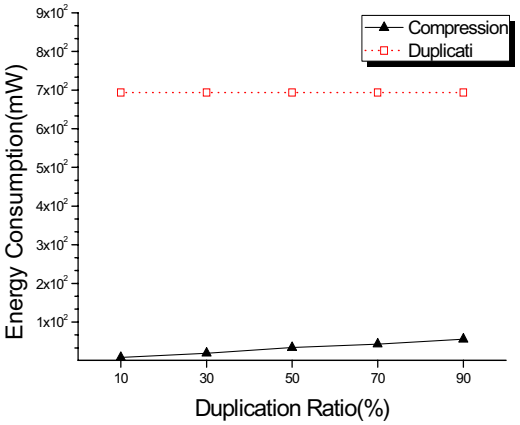
4. 성능 평가

본 연구의 실험을 위해서 클라이언트와 서버를 구현한 컴퓨터는 CPU 3.0 GHz, 1024MB RAM의 하드웨어 스펙을 가지며, 100Mbps 네트워크로 구성되어 있다. 실험 데이터는 20 MByte 크기이며 무작위로 생성을 하였다.



(그림 4) 중복율에 대한 메타데이터 크기 변화

(그림 4)은 데이터 중복율 10%, 30%, 50%, 70%, 90% 일 때의 메타데이터 크기 변화에 대해 실험한 결과이다. 본 논문에서는 제안하는 알고리즘과 Duplicati를 사용하여 비교해 보았다. Duplicati는 byte shift 중복제거 알고리즘을 사용하고 18 Byte 사이즈의 해시키를 사용하고 있다. 제안한 알고리즘은 중복율이 90%일 때 13 KByte 크기의 메타 데이터를 생성하여, 160KByte 크기의 메타데이터를 생성한 Duplicati보다 12 배 더 적게 메타 데이터 크기를 감소 시켰다.



(그림 5) 메타데이터 크기에 비례한 중복 율별 에너지 소모 측정

(그림 5)는 두 알고리즘의 메타데이터 교환에 필요한 에너지 소모량을 측정한 그래프이다. 실험 결과와 같이 Duplicati를 사용했을 때와는 달리 본 연구에서 제안한 해시 압축 기법 알고리즘을 사용했을 때 확실히 에너지 소모량이 적은 것을 알 수 있다.

5. 결론 및 향후연구

중복제거를 지원하는 파일 동기화 프로그램은 클라이언트와 서버사이 메타 데이터를 교환해야 한다. 기존의 파일 동기화 알고리즘은 메타 데이터 교환에 대해 별다른 고민 없이 모든 해시키를 교환했지만 본 논문에서는 해시 압축 방식을 통해 교환 하는 메타 데이터의 크기를 줄이고 동시에 오버헤드 및 에너지 소모까지 줄일 수 있다. 이를 실험을 통해 증명하였으며 제안하는 알고리즘은 기존의 파일 동기화 알고리즘에 비해 네트워크 동기화 과정에서 성능 향상을 보여준다. 이는 앞으로 클라우드 시스템에서 파일 동기화를 할 때 더욱 효과적으로 사용 될 수 있다.

참고문헌

[1] P. Meunier, S. Nystrom, S. Kamara and S. Yost, “ActiveSync, TCP/IP and 802.11 b Wireless Vulnerabilities of WinCE-based PDAs“, WET ICE 2002. Proceedings, pp. 145-150, 2002.

[2] P. HotSync, “Palm Developer Online Documentation “, 2002.

[3] D. Starobinski, A. Trachtenberg, and S. Agarwal, “Efficient PDA Synchronization“, IEEE Transcations on Mobile Computing, Vol. 2, No. 1, pp.40-51, 2003.

[4] A. Tridgell, “Efficient algorithms for sorting and synchronization“, PhD thesis, The Australian National University, 1999.

[5] Yan, H., Irmak, U. and Suel, T., "Algorithms for Low-Latency Remote File Synchronization", INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, pp. 156-160, 2008.

[6] Xu, D., Sheng, Y., Ju, D., Wu, J. and Wang, D., "High Effective Two-round Remote File Fast Synchronization Algorithm", Jisuanji Kexue yu Tansuo, vol. 5, no. 1, pp. 38-49, 2011.

[7] QUINLAN, S. and DORWARD, S., "Venti: a new approach to archival storage", In Proceedings of the 1st USENIX Conference on File and Storage Technologies (FAST), 2002.

[8] A. Muthitacharoen, B. Chen, and D. Mazieres. “A Low-Bandwidth Network File System,“ In Proceedings of the Symposium on Operating Systems Principles (SOSP'01), pages 174 - 187, 2001.

[9] Secure Hash Standard. Fips pub 180-1. In National Institute of Standards and Technology, volume 17, page 15, 1995.

[10] M. O. Rabin. Fingerprinting by random polynomials. Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[11] R.Rivest. RFC 1321: The md5 message-digest algorithm. In Internet activities board, volume 143, 1992.