다단계 메타데이터 관리를 사용한 파일 동기화 시스템

공진산*, 박재민*, 고영웅*
*한림대학교 컴퓨터공학과
e-mail:{kongjs, jmpark, yuko}@hallym.ac.kr

File Synchronization System Using Multi-Level Metadata Management

Jin-San Kong*, Jae-Min Park*, Young-Woong Ko*
*Dept of Computer Engineering, Hallym University

요 약

현재 널리 사용되는 클라우드 스토리지 서비스들의 파일 동기화 기능에 있어 적게 변경된 파일 또는 이름만 변경된 파일에 있어 동기화 시 전체를 전송하는 문제가 있다. 또한 사용자들 간 동일한 파일이 존재함에도 불구하고 전체를 전송하는 문제가 있다. 이러한 문제를 해결하기 위해 본 연구에서는 이중 레벨 메타데이터를 사용한 중복 제거 동기화 시스템을 구현하였다. 주요 아이디어는 VLC(Variable-length Chunking)를 사용하여 중복되지 않은 데이터만 전송하며 서버는 글로벌 메타데이터를 유지하여 사용자 간 중복된 데이터를 검출하는 것이다. 실험 결과로 부분 추가/삭제된 파일 전송 시 제안한 시스템이 네트워크 대역폭과 시간을 감소시키는 것을 보였다.

1. 서론

클라우드 환경이 고속화 되면서 개인 및 기업들은 클라우드 스토리지 서비스에 데이터를 저장하고 사용하는 것이 일반화되었다. 대표적으로 널리 사용하고 있는 서비스는 구글 드라이브, 스카이 드라이브, N 드라이브, 드롭박스가 대표적이다. 이런 클라우드 서비스는 파일 동기화 기능을 제공하여 다수의 클라이언트가 동일한 버전의 파일을 유지할 수 있으며, 수정된 파일에 대해서 일정 기간 동안 구 버전의 파일을 유지하므로 파일에 대한 신뢰성을 높이고 있다. 하지만, 이러한 파일 동기화 기능에 있어서 몇 가지 문제점이 존재한다. 현재 드롭박스를 제외한 클라우드 서비스는 파일 동기화 시 데이터 중복에 대한 처리 기술을 제공하고 있지 않다. 예를 들어, 클라이언트에 있는 파일과 내용은 동일하지만 파일 이름이 상이한 경우에 파일 동기화 과정에서 파일 전체가 서버로 전송되는 문제가 있다. 파일의 일부 부분만 변경이 되더라도 전체 파일이 전송되므로 네트워크 대역폭을 낭비하고 스토리지 자원을 소모하는 문제가 있다.

현재 가장 최선의 기술을 사용하고 있는 드롭박스의 경우에 VLC에 기반을 둔 중복 제거 기술을 사용하기 때문에 중복된 데이터를 제외하고 변경된 데이터만 서버로 전송함으로 매우 효율적으로 파일을 처리하고 있다. 하지만, 사용자들 간의 동일한 파일이나 일부분 변경된 파일에 대해 중복 제거를 처리 못하는 문제가 있다.

본 논문에서는 이러한 문제점을 개선하여 중복 데이터를 효율적으로 처리하는 스토리지 서비스를 설계 및 구현하였다. 제안하는 시스템은 대용량 파일의 경우 사용자들 간의 중복 데이터를 이용하는 글로벌 메타데이터를 유지하고, 작은 파일들의 경우 사용자가 개별적으로 유지하는 로컬 메타데이터를 이용하여 파일 동기화 시간을 감소시켰다. 제안한 시스템의 유용성을 검증하기 위해 각 클라우드 스토리지 서비스들과 동일하게 실험을 진행하였으며 실험 결과 기존 서비스보다 파일 동기화 기능의 네트워크 대역폭과 시간이 감소하는 것을 확인해 본 논문의 아이디어가 유용함을 증명하였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구를 살펴보고, 3장에서는 대표적인 클라우드 서비스의 파일 동기화 비교를 하였고, 4장에서 기존 파일 동기화 기능의 개선을 위해 제안하는 시스템 설계에 대해 설명한다. 5장에서는 드롭박스와 비교 및 성능평가, 6장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

중복 제거는 현재 데이터 백업 시 스토리지 사용량의 감소를 위해 사용된다. 뿐만 아니라 네트워크 트래픽을 줄이거나 시스템에서 중복된 메모리를 줄일 때 사용되기도

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

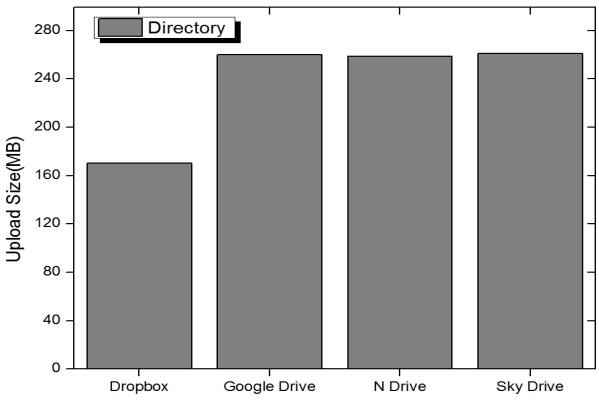
한다.[1][2]

LBFS[3]는 네트워크 환경이 좋지 않은 상황을 고려하여 만든 네트워크 파일 시스템이며, 네트워크 트래픽을 낮추려는 아이디어로 VLC 방식을 사용하여 파일 간의 중복을 제거한 후 네트워크로 전송한다. 먼저 청크(Chunk)의 해시(Hash) 값을 목적지 노드에 전송하고 목적지 노드에서는 해시 값의 유무를 바탕으로 청크 데이터의 전송을 결정한다. Shark[4]는 LBFS의 VLC 방식을 분산 파일 시스템에 적용하였으며, 클라이언트가 파일을 저장하길 원할 때 파일을 청크로 나누어 각 노드에서 데이터를 전송하는 방식이다. TAPER[5] 연구역시 복제 노드와의 동기화 단계에서 중복 제거 기술을 사용하였다. TAPER는 각 청크들의 해시 값을 노드들에 전달하고 각 노드들은 라빈핑거프린트를 통해 전달된 해시 값이 있는지 찾는다. 해시 값이 발견된 노드들은 이에 응답하고 TAPER는 네트워크를 통해 중복이 아닌 청크들을 각 노드들에 전송한다.

Mogul의 연구[6]는 HTTP 전송에서 중복 제거 기술을 적용한 연구이다. 각 HTTP 페이로드 마다 MD5 해시를 하고 MD5 해시 값이 브라우저의 캐시에 있는지 검색 한다. 이 기술은 프락시 서버에 저장된 웹 캐시를 검사하여 중복된 HTTP 페이로드를 줄이는데 도움을 주고 있다.

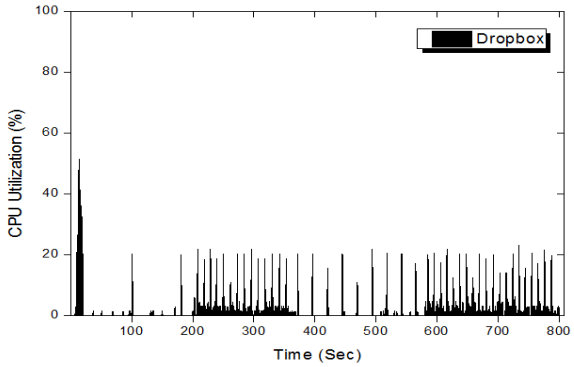
3. 클라우드 시스템 비교

기존 클라우드 스토리지 서비스에 대해서 파일 동기화의 성능을 알아보기 위해 실험을 수행하였다. 성능 분석의 대상은 대표적으로 널리 사용되고 있는 구글 드라이브, 스카이 드라이브, N드라이브, 드롭박스이다. 실험 데이터는 5개의 파일이 존재하고, 각각의 파일은 0%(원본), 20%, 40%, 60%, 80%의 중복률을 가지며 크기는 50MB인 텍스트 파일이다. 즉, 원본 파일에 각각 20%, 40%, 60%, 80%의 패치를 적용하여 4개의 파일을 생성한 것이다. 따라서 이상적인 중복 제거 시스템의 경우에 전체 250MB의 파일 데이터 중에서 100MB의 중복된 부분을 제외하고 150MB의 데이터를 전송해야 한다.



(그림 1) 중복 제거 방식을 판별하기 위한 실험 결과

(그림 1)은 각 클라우드 스토리지 서비스마다 동기화 완료 후 총 업로드 크기를 측정한 것이다. 드롭박스를 제외한 시스템들은 약 250MB, 즉 모든 데이터를 전송하는 반면에 드롭박스는 약 170MB 데이터를 전송하였다.

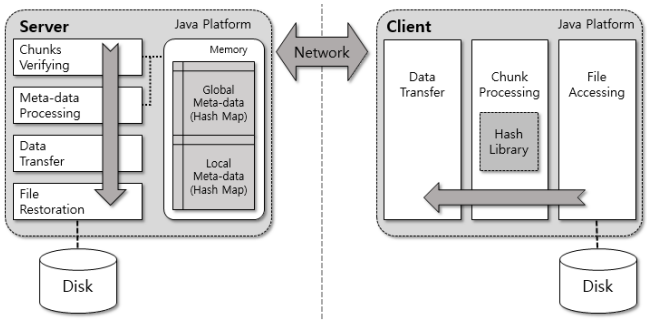


(그림 2) 드롭박스 CPU 사용률

(그림 2)는 드롭박스의 파일 동기화 시 CPU 사용률을 WMI(Windows Management Instrumentation)로 측정한 것이다. 이는 메타데이터 정보를 포함하여 VLC 방식의 중복 제거를 수행하여 전송하는 것으로 판단할 수 있다.

4. 시스템 구조

클라우드 서비스들의 비교를 통하여 드롭박스를 제외한 다른 서비스들은 중복된 데이터를 찾지 못하며, 동일한 내용에 이름만 상이한 파일조차 찾지 못하였다. 드롭박스 또한 압축 파일에 대한 문제점이 있음을 확인하였다. 이러한 문제를 해결하기 위해 다단계 메타데이터 관리를 사용한 파일 동기화 시스템을 설계 및 구현하였다.

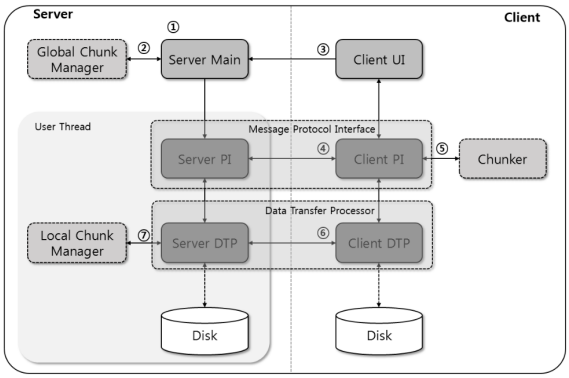


(그림 3) 제안하는 시스템의 전체 구조

(그림 3)은 제안하는 시스템의 전체 구조 및 파일 백업의 흐름을 나타낸다. 중복 제거 방식은 소스 기반 중복 제거를 사용하여 데이터를 전송한다. 서버는 클라이언트로부터 전송된 메타데이터를 서버가 메모리에 유지하고 있는 메타데이터들과 비교하고, 관리하는 역할을 수행한다. 또한 비교를 통해 생성된 메타데이터를 기반으로 클라이언트의 파일을 서버에 재구성하여 저장한다. 클라이언트는 서버로 파일을 백업/복원이 가능하며, 파일 백업 시 (그림 3)의 해시 라이브러리를 통해 네트워크 사용량을 줄일 수 있다.

(그림 4)는 제안하는 시스템의 모듈들과 중복 제거 흐름을 나타낸다. 제안하는 시스템은 크게 청크 매니저

(Chunk Manager), 청커(Chunker), 프로토콜 인터페이스로 분류된다. 이 모듈들은 클라이언트에서 파일 백업 시 네트워크 사용량을 줄이고 더 빠른 데이터 전송을 위해 사용된다. 서버는 클라이언트의 사용자마다 개인 폴더와 개인 로컬 청크 매니저를 통해 기존의 스토리지 시스템보다 더 빠른 파일 백업이 가능하다.



(그림 4) 전체 시스템 모듈과 중복 제거 흐름

서버는 대용량의 파일들의 청크 정보(메타데이터)를 저장하는 글로벌 청크 매니저와 저용량 파일들의 청크 정보를 사용자 개별로 유지, 관리하는 로컬 청크 매니저가 있다. 청크 매니저는 청크의 정보들(파일 ID, 해시 값, 오프셋, 크기 등)을 트리로 유지하고 있다. 또한 compare() 함수를 통하여 클라이언트 파일의 청크들과 비교를 수행한다.

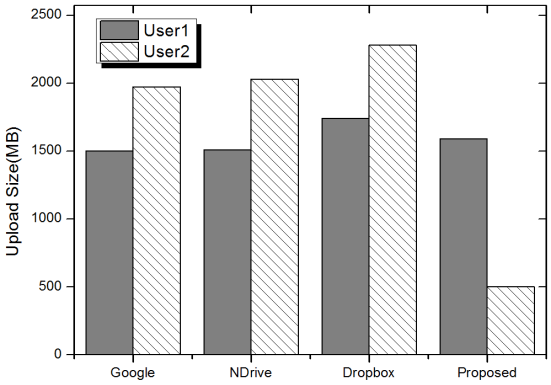
5. 성능 평가

실험에 사용한 컴퓨터는 쿼드코어 3.4GHz, 4GB RAM의 하드웨어 스펙을 가지며, 윈도우7에서 실험을 진행하였다. 파일 동기화 시 사용자들 간의 중복 제거 여부를 확인하기 위해 실험을 진행하였다. 실험 데이터는 <표 1>과 같으며, 파일 내용은 mp3 파일을 zip 확장자로 압축한 것이다. User2의 데이터는 User1의 데이터에 임의의 mp3 파일을 추가한 파일이다.

<표 1> 글로벌 파일 동기화 실험 데이터

	User1	User2
용량(GB)	1.49	1.95
파일 수	130	197

(그림 6)은 클라우드 서비스에서 User1이 파일 동기화를 완료한 후에 User2가 파일 동기화를 실시하였을 때 클라이언트에서 서버로 업로드 크기를 측정한 것이다. 구글 드라이브와 N드라이브는 파일 크기에 맞게 약 1.5GB, 2.0GB 전송됨을 확인했다. 드롭박스의 경우는 메타데이터 정보를 포함하기 때문에 약 100MB이상의 업로드 차이를 보였고 사용자들 사이에서 중복된 데이터를 찾지 못하는 것을 알 수 있었다. 제안하는 시스템에서 글로벌 메타데이터를 사용하기 때문에 사용자 간 중복 제거가 가능하고, 압축된 파일의 중복 데이터를 발견하기 때문에 User2는 실험 데이터에서 중복 되지 않은 500MB만을 전송하였다.

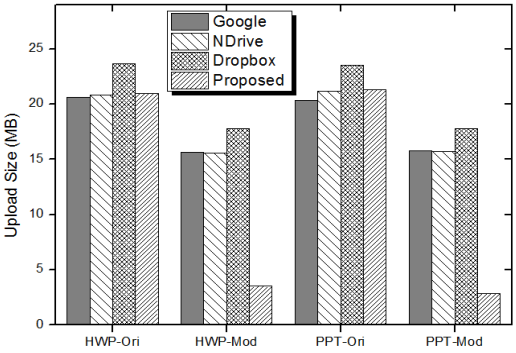


(그림 5) 사용자 간 데이터 중복 제거 실험

사용자 개인 파일들 간의 중복 데이터 처리 유무를 위해 문서 파일로 실험을 진행하였다. 실험 데이터는 <표 2>와 같으며 원본 파일에 대해 임의의 구간을 삭제하여 수정된 파일을 생성하였다.

<표 2> 로컬 파일 동기화 실험 데이터

	HWP-Ori	HWP-Mod	PPT-Ori	PPT-Mod
용량(MB)	20.0	15.0	20.3	15.1



(그림 6) 개인 파일 동기화 실험

(그림 6)은 클라우드 서비스 별 개인 파일 동기화 실험 결과이다. 실험 내용은 원본 파일 동기화 후 수정된 파일을 차례로 동기화하였다. 구글 드라이브와 N드라이브, 드롭박스는 중복을 찾지 못하여 수정된 파일 전체를 동기화하는 것을 볼 수 있다. 드롭박스의 경우 메타데이터 전송으로 인해 업로드 크기가 조금 더 크게 측정됨을 알 수 있다. 제안하는 시스템은 5MB 미만의 업로드 크기를 보였다. 제안하는 시스템은 로컬 메타데이터를 이용하여 사용자 개인파일에 대해서 중복 제거를 실시하기 때문에 수정된 파일에서 중복되지 않은 부분만 전송되었다.

두 가지 실험을 통하여 기존 클라우드 서비스와 다르게 사용자 전체에 대한 메타데이터와 개인에 대한 메타데이터를 사용하여 중복 제거 시 네트워크 대역폭과 시간이 감소됨을 확인하였다.

6. 결론 및 향후연구

본 논문에서는 기존 클라우드 스토리지 서비스에서 파일 동기화 시 발생하는 네트워크 대역폭과 시간을 줄이기 위해, 다단계 메타데이터 관리를 이용하는 중복 제거 동기화 시스템을 설계하고 구현하였다. 주요 아이디어는 VLC 기반 소스 기반 중복 제거 기법을 사용하고, 서버는 글로벌 메타데이터와 로컬 메타데이터를 유지함으로써 중복되지 않은 데이터만 전송하는 것이다. 실험 결과 대표적으로 구글 드라이브, N드라이브, 드롭박스와 동일한 파일로 실험을 진행하였고 제안하는 시스템이 기존 클라우드 서비스들보다 네트워크 대역폭 및 시간이 중복된 데이터양에 따라 감소함을 보였다. 향후 연구로는 제안하는 시스템을 클러스터 시스템에 적용에 있어 글로벌 메타데이터 관리의 병렬처리 부분에 대해 연구할 계획이다.

참고문헌

[1] F. Guo and P. Efstathopoulos “Building a high-performance deduplication system” In Proceedings of the 9th USENIX conference on File and stroage technologies (FAST), 2011.

[2] D. Gupta, S. Lee, M. Vrable, S. Savage, A.C. Snoeren, G. Varghese, G.M. Voelker, and A. Vahdat “Dierence engine: Harnessing memory redundancy in virtual machines” Communications of the ACM, 53(10):85-93, 2010.

[3] A. Muthitacharoen, B. Chen, and D. Mazieres “A low-bandwidth network file system” ACM SIGOPS Operating Systems Review, 35(5):174-187, 2001.

[4] S. Annapureddy, M.J. Freedman, and D. Mazieres “Shark: Scaling file servers via cooperative caching” In In Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI), pages 129{142. USENIX Association, 2005.

[5] N. Jain, M. Dahlin, and R. Tewari “Taper: Tiered approach for eliminating redundancy in replica synchronization” In Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST), pages 21. USENIX Association, 2005.

[6] J.C. Mogul, Y.M. Chan, and T. Kelly “Design, implementation, and evaluation of duplicate transfer detection in http” In Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI), pages 4{4. USENIX Association, 2004.