중복 제거 기술을 이용한 스마트폰 백업 시스템

정호민*, 김병기*, 송창근**, 고영웅*
* 한림대학교 컴퓨터공학과
** 한림대학교 유비쿼터스 컴퓨팅 학과
e-mail:{chorogyi, bkkim, cgsong, yuko}@hallym.ac.kr

Smartphone Backup System Using Deduplication Scheme

Ho Min Jeong*, Byung-Ki Kim*, Chang-Geun Song**, Young-Woong Ko*
* Dept. of Computer Engineering, Hallym University
** Dept. of Ubiquitous Computing, Hallym University

요 약

스마트폰에서 용량을 많이 차지하는 멀티미디어 컨텐츠, 응용 프로그램 데이터가 증가하면서 스마트폰을 위한 백업 프로그램에 대한 요구가 증가하고 있다. 본 논문에서는 중복 제거 기법을 적용한 스마트폰 백업 기술을 제안함으로 네트워크 대역폭을 효율적으로 사용하고 저장 공간을 줄일 수 있는 방안을 제시하고 있다. 제안하는 중복 제거 방식은 스마트폰에서 각 파일을 일정한 크기의 블록 단위로 분할하고 지문을 부여하고, 지문이 동일하지 않을 경우에만 데이터를 전송하는 방식을 사용한다. 실험 결과 제안하는 방식이 네트워크 대역폭 및 저장 공간을 효율적으로 사용함을 보이고 있다.

1. 서론

최근 애플의 아이폰이나 구글의 안드로이드 폰들이 국내에 출시되면서 스마트 폰에 대한 관심이 폭발적으로 증가하고 있으며, 스마트 폰은 본래의 통화 기능 외에 음악 감상, TV 및 동영상 시청, 게임, 다이어리, 사전, 디지털카메라, 웹 브라우징 등의 수많은 부가적인 기능들을 사용할 수 있다. 특히 스마트 폰은 즉시 기억해야 할 사항이 있거나, 즉흥적으로 떠오르는 아이디어, 영상으로 기록할 것들을 바로 저장할 수 있는 강점이 있다. 스마트폰의 활용도가 높아지면서 스마트폰의 저장장치에는 문서 자료, 멀티미디어, 시스템 파일 등과 같은 다양한 형태의 컨텐츠가 저장되고 있다. 이와 같은 데이터를 안전하게 저장하는 것은 스마트폰과 같은 모바일 기기에 있어서 매우 중요한 요소가 된다. 특히 펌웨어 업그레이드나 기타 이벤트로 시스템을 재설치 하는 일이 자주 발생하고 있으므로 백업의 필요성은 PC보다 더욱 강조되고 있다.

스마트 폰은 3G통신을 이용하거나 Wi-Fi 무선 네트워크를 이용하여 네트워크 백업 시스템을 구현할 수 있다. 그러나 대부분의 백업은 데이터의 사본을 다른 공간에 저장하는 형태이므로 백업을 할 때 마다 전에 저장한 파일들을 전송해야 하며 메타데이터를 이용할 경우에도 버전업 되거나 수정된 파일들도 마찬가지로 전부 재전송해야 한다. 따라서 네트워크 비용이 증가하고 저장 공간이 비효율적으로 운영될 수 있다.

본 논문에서는 스마트폰의 백업 비용을 줄이기 위해 네트워크 대역폭을 효율적으로 사용하고 서버의 저장 공간을 줄일 수 있는 중복 제거 시스템을 제안하고 있다. 제안하는 시스템은 안드로이드 플랫폼을 기반으로 구현하였으며, 중복제거 서버는 리눅스 기반의 플랫폼에서 동작된다. 제안하는 기법의 특징은 다음과 같다. 첫째, 소스 기반 중복 제거 방식을 적용함으로 네트워크로 전송되는 데이터를 최소화하였다. 둘째, 중복 제거 작업에 필요한 해쉬 데이터를 스마트폰에서 캐쉬하여 효율적인 중복 제거 시스템이 될 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 중복 제거와 관련된 최근 연구 동향과 스마트 폰에 대해서 알아보고, 3장에서는 전체 시스템의 설계와 구현내용을 설명하고 4장에서는 성능평가에 대해 기술하고 5장에서는 결론 및 향후 연구방향을 제시한다.

본 연구는 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임

2. 관련 연구

중복 제거에 관한 연구는 대규모 백업 시스템이나 P2P 시스템 그리고 CDN(Contents Delivery Network)를 중심으로 이루어져 왔으며, 대표적인 연구 결과는 Rysnc[1], Venti[2], LBFS[3], HydraFS[4], DEDE[5] 등이 있다. Rsync는 네트워크로 연결된 디렉토리의 데이터를 동기화 시켜주는 프로그램으로 롤링 체크섬(Rolling Checksum)이라는 중복 데이터를 검색하는 알고리즘을 사용해 새로운 데이터의 복사만 일어나게 한다. Plan9[6]의 Venti는 네트워크 스토리지 시스템에서 중복 데이터를 제거하여 저장하는 스토리지 시스템이다. Venti는 데이터를 저장할 경우 파일을 고정된 크기(8Kbyte)의 블록으로 나누고 각 블록에 SHA1 해시를 적용하여 160bit 크기의 해시를 만들고 전송한다. LBFS는 자주 끊기거나 품질이 좋지 않은 네트워크 환경을 위해 설계된 네트워크 파일 시스템이다. LBFS에서는 파일 전송 효율을 높이기 위해 CDC(Content-defined Chunks) 방식을 사용한다. CDC는 Rabin Fingerprint[7]로 해시하여 특별한 값을 반환하는 앵커(Anchor) 블록을 파일에 삽입하고 데이터 전송 전에 앵커 사이의 블록을 SHA1, MD5같은 해시 함수를 사용해 해시를 만들고 그 해시들을 전송하여 서버에서 캐싱된 해시 룩업(Lookup) 테이블과 비교하여 중복을 검색하는 방법이다.

Rsync는 프로젝트 파일들을 배포할 때 유용하게 쓰이지만 사용자가 동시에 접속하여 같은 파일을 접근할 때 문제가 발생하는 것을 경험하게 된다. Venti와 LBFS는 파일 시스템이기 때문에 안드로이드 폰에 적용하기 어렵다. 본 연구에서는 스마트 폰에 적용하기 위해 애플리케이션 형태로 클라이언트를 구현하였다.

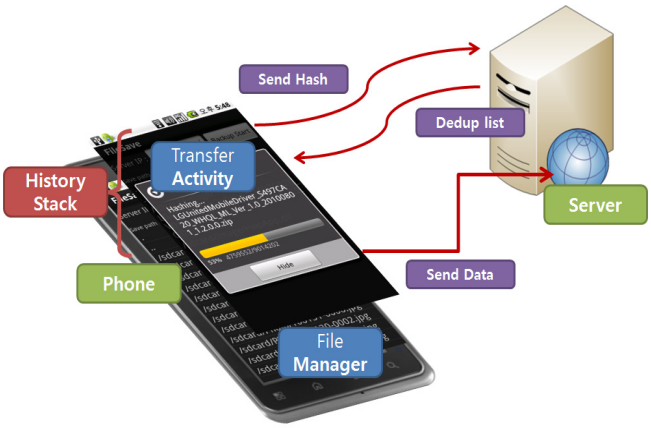
3. 백업 시스템 설계 및 구현

본 논문에서 제안하는 시스템은 여러 사용자가 하나의 백업서버를 사용하는 것을 염두에 두었으며 사용자들 간에 공유하는 데이터가 많을수록 효과적인 백업이 가능해진다. 모바일 단말 에서는 백업하려는 파일들을 해싱하여 파일지문을 부여하고 서버로 전송한다. 전송되는 데이터는 SHA1 해시 데이터이므로 파일 하나에 20byte만을 전송하면 되기 때문에 많은 대역폭을 필요로 하지 않는다.

3.1. 클라이언트

안드로이드는 Activity, Intent Receiver, Service, Content Provider 네 가지의 주요 클래스로 구성되어 있다. Activity는 응용프로그램에서 하나의 화면을 지칭하며, 사용자에게 View와 Event 응답으로 이루어진 인터페이스를 제공한다. window와 같이 여러 창을 동시에 보여주기 힘들기 때문에 그림1과 같이 History Stack을 제공한다.

직접 실행해서 백업하는 대신 백그라운드 형태로 주기적으로 실행할 필요가 있는데 이와 유사한 클래스가 Service이다. 안드로이드에서는 멀티 쓰레딩을 지원하기 때문에 음악 재생이나 사진을 보는 중에도 프로그램을 Service로 만들면 백그라운드로 실행되어 동시에 작업을 할 수 있다.



[그림 1] 클라이언트 구조

안드로이드 프로그램을 만들면서 네 가지 주요 클래스를 다 쓸 필요가 없고 Activity와 Service클래스만으로 백업 프로그램을 완성할 수 있다.

<표 1> FileList DTD

```
<!DOCTYPE filelist[
 <!ELEMENT filelist(file*)>
 <!ELEMENT file (dirhash, filehash, blockhash*)>
 <!ELEMENT dirhash (#PCDATA)>
 <!ELEMENT filehash (#PCDATA)>
 <!ELEMENT blockhash (#PCDATA)>
]>
```

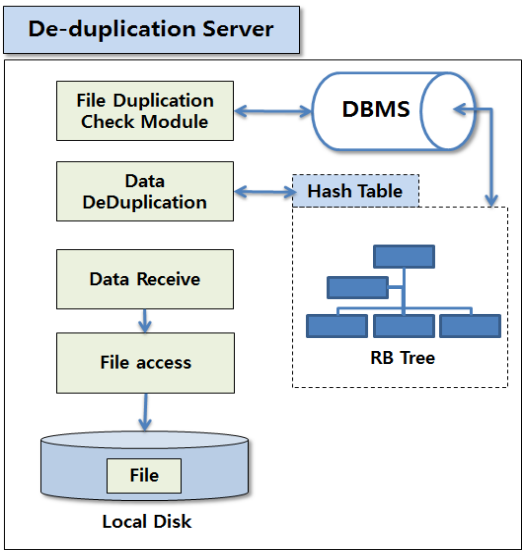
한번 백업 했던 데이터들은 재차 백업이 될 수 있기 때문에 계산했던 해시 데이터들은 전부 보관하여 재계산 되는 것을 막아야 한다. Java에서는 Xml을 효과적으로 지원하고 있으며 해시 데이터 처리를 위해 표와 같은 XML구조로 저장한다. 표1은 백업했던 파일 목록을 나타내고 있으며 파일의 위치와 파일을 해시로 나타내고 파일의 구성들인 블록의 해시들의 집합으로 표현하고 있다.

3.2. 백업 서버

그림 2는 백업 서버의 전체 구성을 보이고 있다. 백업 서버에서는 해시 데이터의 보관과 백업 자료의 관리를 수행한다. 해시 데이터로는 파일 해시와 블록 해시가 있으며 데이터베이스 테이블에 저장한다.

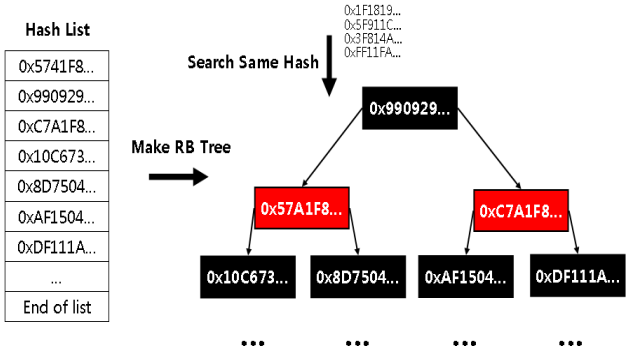
서버는 1 단계로 클라이언트에서 전송한 파일 해시 리스트를 가지고 기존에 저장되어 있는 파일해시들과 비교하여 파일의 중복을 클라이언트에 전한다. 2 단계로 클라이언트에서 전송받은 블록 해시 리스트를 가지고 기존에 저장되어 있는 블록 해시들과 비교하여 블록의 중복을 클라

이언트에 전송한다. 블록 해시의 비교는 파일 해시의 비교보다 처리해야 할 데이터양이 매우 크며 데이터베이스를 사용하여 비교했을 경우 소규모 서버에서는 좋은 성능을 내기 어렵다.



[그림 2] 백업 시스템 구성도

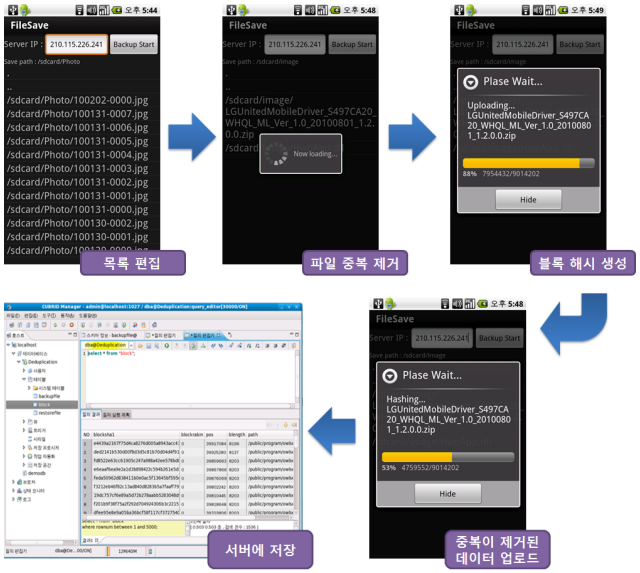
효율적으로 블록 해시를 비교하기 위해 블록해시들을 메모리에서 꺼내 써야 하고 되도록 파일 I/O를 피해야 한다. 수많은 비교에 매우 효율적인 자료구조로 레드 블랙트리가 쓰이고 있으며 실제 구현에서도 데이터베이스에서 쿼리를 이용한 비교보다 좋은 성능을 나타냈다.



[그림 3] RB Tree를 이용한 해시 데이터 관리

3.3 구현

모바일에서 실행과정은 그림 4와 같다. 첫째, 모바일에서 저장하려는 파일들을 편집한다. 둘째, 데이터를 저장하기 위한 백업 서버를 선택하기 위해 IP를 입력한 다음 시작 버튼을 눌러 백업을 시작한다. 셋째, 파일 단위의 중복을 제거 하고 중복이 아닌 파일들의 블록에 대한 해시를 생성한다. 넷째, 블록 해시목록을 서버에 전송하여 중복 데이터를 판별한 다음 중복이 아닌 데이터를 서버에 전송하고 있다.



[그림 4] 클라이언트 실행 과정

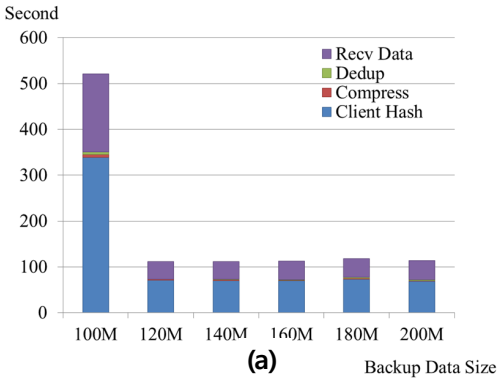
4. 성능 평가

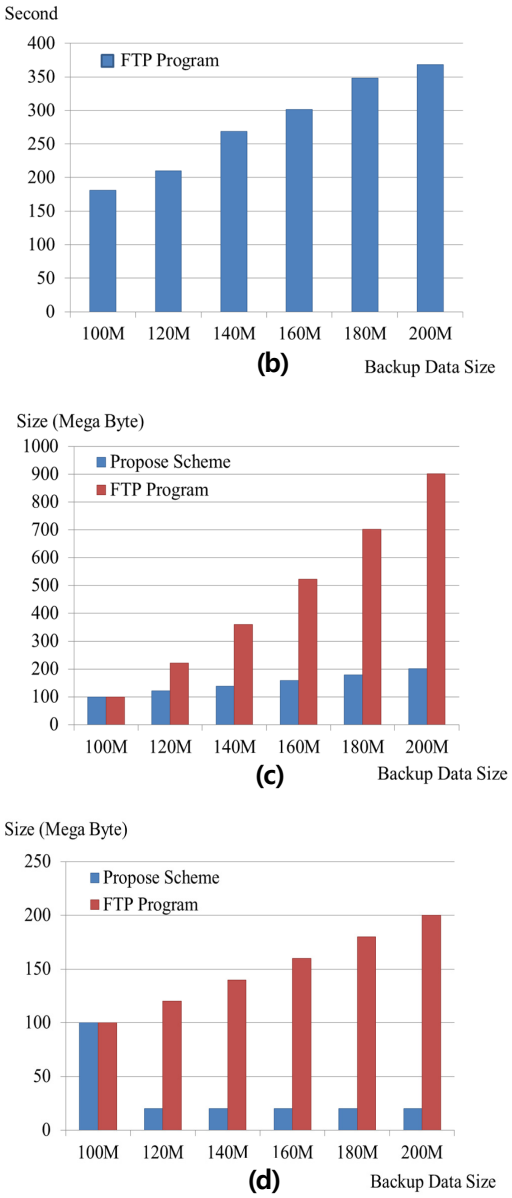
본 연구에서는 표2와 같은 환경에서 개발 및 실험하였다. 안드로이드에서 지원되는 언어는 Java이며 이를 위해 개발도구로 이클립스를 사용하였다. 서버 환경도 Java 플랫폼으로 개발하였다.

<표 2> 개발 및 실험 플랫폼

S/W 플랫폼	Client	운영체제	Andorid 1.6
		개발도구	eclipse
	De-Duple. Server	운영체제	Fedora Core 9
		Kernel	2.6.18
		개발도구	eclipse, cubrid dbms
H/W 플랫폼	CPU	Pentium 4 3.0 GHz	
	Memory	512 MB	
	Hard Disk	WD-1600JS(7200/8MB)	
	Network	myLGnet(WiFi)	
	Mobile	Optimus Q (LG LU2300)	

실험 데이터는 백업 자료가 순차적으로 늘어난다는 가정을 하여 초기 자료 100Mbyte를 백업하고 다음 차례부터 20Mbyte씩 늘려 총 5차례 백업을 하는 것을 테스트 하였다. 비교 대상은 기존의 FTP프로그램과 본 논문에서 제안한 시스템이다





[그림 5] 실험 결과 (a) 중복제거 기술을 이용한 저장 (b) FTP 프로그램 (c) 총 백업 용량 (d) 네트워크 트래픽

실험 그래프는 수행 시간, 저장용량, 네트워크 트래픽 세 가지 내용을 관찰한 결과이다. (a), (b) 그래프는 제안된 시스템과 FTP 프로그램의 수행시간을 측정한 그래프이다. (a) 에서는 수행시간을 해싱, 데이터 전송, 압축 그리고 해시 비교의 4 단계로 분류하였으며 초기 백업에서는 클라이언트 해싱 시간이 오래 걸려 전체적으로 수행시간이 증가하지만 다음 단계의 백업부터는 새로운 데이터만을 해싱 하기 때문에 백업 시간이 단축되는 것을 볼 수 있다. (a)와 (b)를 비교하면 (b)는 점점 데이터가 늘어남으로써 백업 시간도 데이터에 비례해서 늘어나지만 (a)는 새로운 데이터만큼만 작업하므로 전체 적인 백업 사이클에서 볼 때 (b)보다 매우 효율적이다. (c)그래프는 백업 서버에 누적된 용량인데 제안된 시스템이 FTP 프로그램보다 완만하게 증가하는 것을 볼 수 있다. (d)는 네트워크 트래픽을 보이고 있다. 초기 백업에서는 데이터의 전송량이 동일하지만 다음 단계부터는 제안된 시스템의 경우 20Mbyte 정도의 네트워크 전송 트래픽만 발생하게 된다. 하지만 FTP 프로그램에서는 중복 제거 없이 전송이 되므로 전송량이 매우 크다. 따라서 백업 데이터의 크기가 커지는 경우에는 전송 데이터 용량의 차이가 급격해지는 현상이 발생한다. 따라서 제안하는 시스템의 효과가 매우 높음이 자명하다.

5. 결론 및 향후연구

본 연구에서는 구글의 모바일 플랫폼인 안드로이드에서 백업 클라이언트와 이를 지원하는 백업 서버의 설계와 구현에 대해 기술하였다. 본 논문의 주요 내용은 중복제거 기술을 사용하여 모바일에서 백업을 할 경우 중복된 데이터를 제외하고 전송하는 것이며 이를 구현하고 실험으로 검증하였다. 기존의 전통적인 백업을 이용하는 경우에 네트워크 트래픽이 매우 높아져서 3G 네트워크를 이용하는 경우에 과금이 증가할 위험이 높다. 제안하는 중복 제거 기술을 이용한 시스템을 사용한 백업은 네트워크 대역폭을 효과적으로 줄일 수 있다.

향후 제안된 기술의 성능을 개선하기 위해 중복 제거 알고리즘을 최적화하는 방안을 연구할 계획이며, 백업 과정에서의 배터리 소모량을 관찰하여 효율적인 에너지 사용을 지원할 수 있는 백업 방안에 대해 연구할 것이다.

참고문헌

[1] A. Tridgell. Efficient algorithms for sorting and synchronization. PhD thesis, The Austrailian National University, 1999.

[2] 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.

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

[4] C. Ungureanu, B. Atkin, A. Aranya, S. R. et al. HydraFS: a High-Throughput File System for the HYDRAstor Content-Addressable Storage System. In FAST, 2010.

[5] C. Ungureanu, et al. HydraFS: a High-Throughput File System for the HYDRAstor Content-Addressable Storage System. In FAST, 2010.

[6] plan9 home page, http://plan9.bell-labs.com/plan9/

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