효율적인 중복제거 메커니즘을 적용한 백업 시스템

정호민*, 김병기*, 유재봉*, 김진*, 고영웅*
*한림대학교 컴퓨터공학과
e-mail:{chorogyi, bkkim, yoojaebong, jinkim, yuko}@hallym.ac.kr

Backup System with Effective File Deduplication Mechanism

Ho-Min Jeong*, Byung-Ki Kim*, Jae-Bong Yoo*, Jin Kim*, Young-Woong Ko*
*Dept of Computer Engineering, Hallym University

요 약

인터넷과 PC 사용의 증대로 개인 사용자, 소규모 그룹의 중요한 파일 백업의 필요성이 증가하고 있다. 그러나 상용 백업 소프트웨어의 규모나 비용이 개인 또는 소규모 그룹에 맞지 않는 문제점을 안고 있다. 본 논문에서는 오픈소스의 범용 시스템을 기반으로 한 백업 시스템을 네트워크를 통해 연결된 백업 서버에 자동으로 백업 및 복구를 해주는 시스템을 제안하고 구현하였다.

1. 서론

사용자의 실수나 천재지변으로 인한 개인 또는 기업의 정보 손실은 경제적인 손실을 발생 시킨다. 만약 기업에서 전산시스템의 고장이나 데이터의 유실로 정지시간이 생긴다면 그 손해는 평균 한 시간동안 10억 원 이상이라고 한다. 특히 전산업무를 많이 하는 기업의 경우에는 그 액수가 2배에 이른다[1]. 이러한 정지시간을 가능한 최소화하기 위해서는 최신의 백업 데이터를 항상 유지하고 복구 할 수 있어야 한다. 따라서 각종[1] 데이터를 보호하기 위해서는 주기적 또는 비주기적인 백업이 필수적이다. 효율적인 백업관리를 위해서는 원하는 시간에 자동 백업 및 복구가 수행되는 백업 소프트웨어가 필요하다[2][3].

유비쿼터스 시대가 도래하여 각 개인마다 모바일, PDA, 노트북과 같은 여러 단말기를 소유함에 따라 중요한 개인 정보를 중소규모의 파일 형태로 광학매체, Zip drive와 USB메모리 같은 저장장치에 따라 저장을 해야 한다. 사용자는 데이터 백업과 복원 시 기존의 상업용 소프트웨어를 사용하거나 데이터가 업데이트 될 때마다 일일이 백업과 복원을 해야만 하는 불편함을 초래한다. 본 연구에서는 이와 같은 문제점을 해결하고자 사용자가 백업을 원하는 디렉터리를 지정하면 원하는 주기마다 디렉터리에 있는 데이터를 백업하기 위해서 서버로 보내며 시스템의 데이터가 손상되더라도 원하는 시간에 백업된 데이터를 서버로부터 받을 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 백업과 복원의 특징에 대해 기술하고, 3장에서는 제안하는 백업 소프트웨어의 구조에 대해서 백업과 복구로 나누어 설명하며, 4장에서는 백업 소프트웨어의 구현 및 평가를 하고, 5장에서는 결론을 맺는다.

2. 백업과 복원

지금까지 구현된 상용 백업 소프트웨어는 각각의 소프트웨어마다 다양한 형태의 백업 기능을 제공하

This research was supported by the Program for the Training of Graduate Students in Regional Innovation which was conducted by the Ministry of Commerce Industry and Energy of the Korean Government, This work was supported by the Industry University Research Institute Consortium grant from the Small &Medium Business Administration.

고 있다[4]. 백업은 형태, 기능, 대상에 따라서 백업 방식을 구분할 수 있다.

모든 파일을 백업할 것인지 또는 변경된 부분만을 백업할 지에 따라 전체 백업과 부분 백업으로 나뉜다. 전체 백업은 수행할 파일시스템이나 장치 내에 존재하는 모든 파일에 대하여 백업을 수행한다. 부분 백업은 이전 백업에 대해서 변경된 파일만 백업을 수행한다. 부분 백업은 이전의 마지막 전체 백업 이후에 변경된 데이터를 백업하는 차등 백업과 바로 이전의 백업 이후에 변경된 데이터를 백업하는 점진 백업으로 나뉜다.

백업할 대상이 파일 시스템 기반인지 장치 기반인지로 구분할 수 있다. 파일 시스템 기반의 백업은 파일 구조를 파악하여 백업장치에 파일과 디렉터리 구조를 저장하기 때문에 백업은 오래 걸리지만 복원은 빠르게 수행할 수 있다. 이에 반해 장치 기반의 백업(device-based backup)은 파일구조를 무시하고 디스크 블록을 백업장치에 기록한다. 따라서 백업 성능은 향상되지만 복원시간이 다소 오래 걸리는 단점이 있다.

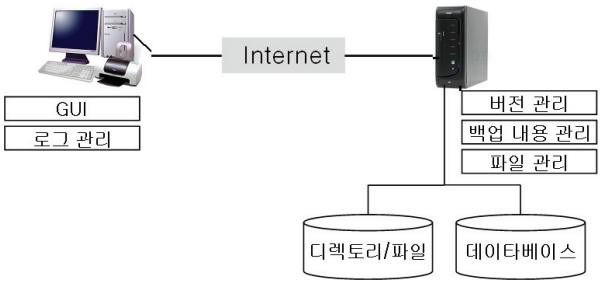
이 외에도 클라이언트들이 작업을 수행하는 중에도 백업을 수행할 수 있도록 지원하는 온라인 백업과 클라이언트의 모든 작업을 중단해야 하는 오프라인 백업(offline backup)이 있다. 또한 데이터의 압축을 지원하는지, 동시에 여러 개의 백업을 수행하는지 그리고 백업 수행 시 데이터에 대한 바이러스 체크를 하는지에 따라 구분할 수가 있다.

3. 설계

본 논문에서 제안하는 백업 소프트웨어는 파일 시스템 기반의 백업에 적합하며 다수 클라이언트의 백업을 지원한다.

- 부분백업과 전체백업 지원
- 부분복구와 전체백업 지원
- 작업 관리 기능 지원
- 동시 다수 클라이언트 백업 지원
- 온라인 백업 지원

언급된 특징들을 지원하기 위해 제안하는 소프트웨어는 [그림 1]과 같이 구성된다. 백업·복구를 클라이언트 프로그램 실행으로 서버에 요청한다.
인터넷을 통해서 접속하기 때문에 원격이나 지역에서 서버에 백업·복구를 요청할 수도 있다. 클라이언트 프로그램은 백업·복구에 필요한 정보를 입력받아 로그를 남겨 관리하고, 백업·복구 요청 시 로그를 바탕으로 디렉터리 구조, 파일들을 서버에 송신한다. 서버 프로그램은 클라이언트 프로그램으로부터 디렉터리 구조와 파일속성들을 받은 후, 데이터베이스에 디렉터리 구조와 파일속성의 정보들을 저장한다. 백업·복구 서버의 파일 관리는 범용 운영체제의 파일시스템을 통해 관리하며 변경되는 파일의 버전관리나 속성변경은 데이터베이스 시스템을 통해 관리한다.

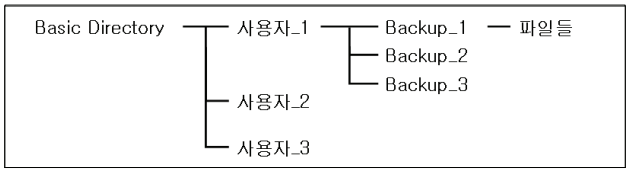


[그림 1] 시스템 구조

3.1 백업

부분백업을 지원하기 위해 중복 파일의 구별기능이 필요하다. 구별방법은 파일의 크기, 파일이름, 수정날짜와 기본 파일속성을 사용하여 비교한다. 파일시스템에서는 기본적으로 디렉터리 내에 중복파일 이름을 허용하지 않기 때문에 파일 이름과 전체 경로만으로도 구별이 가능하지만, 문서파일 같은 경우는 빈번한 업데이트가 일어나기 때문에 백업 주기 사이의 파일 크기와 수정날짜를 비교하여 업데이트 여부를 판단한다.

클라이언트가 서버에 백업을 요청할 때 서버는 새로운 백업인지 아닌지에 따라 수행과정의 차이가 있다. 클라이언트가 요청한 백업이 새로운 백업일 경우 데이터베이스의 테이블을 새로 생성하고 백업하고자 하는 모든 파일을 저장한다.



[그림 2] 디렉터리 구조

기존의 백업본이 있는 경우 테이블을 검색하여 중복 파일의 전송을 피하고 업데이트된 파일을 저장한

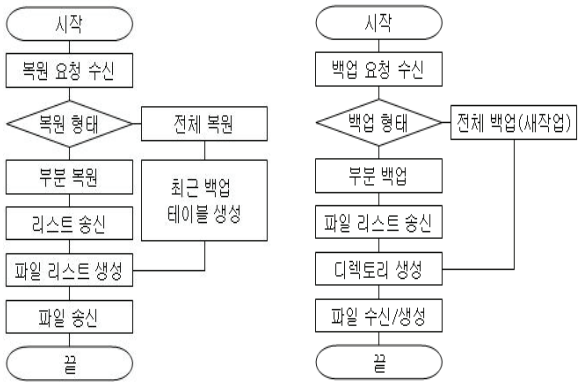
다. 백업 때 마다 디렉터리를 [그림 2]와 같은 형식인 $BasicDir/user/backupNum 으로 생성하고 백업의 디렉터리 구조와 파일들을 저장한다.

3.2 복구

클라이언트는 시스템의 손상 또는 사용자 실수로 인한 데이터의 손실로 인해 백업된 데이터의 일부 또는 전체를 복구하고자 할 것 이다. 백업된 데이터의 부분 복구와 전체 복구 기능을 제공해야 하며 백업된 데이터의 정보를 검색하기 위한 기능을 제공하고 있다.

복구 과정에서 클라이언트는 필요에 따라 부분복구, 전체 복구를 요청한다. 부분 복구는 서버가 이전에 수행했던 백업내용을 검색하고 요청사용자의 데이터를 선택하여 복구 리스트를 작성한다. 작성된 복구 리스트를 클라이언트로 보내면 클라이언트는 그 리스트 중에 파일이나 백업날짜를 선택하여 원하는 파일들을 전송 받는다.

시스템이 파괴되어 새로운 시스템에서 백업된 데이터가 필요할 경우 전체 복구가 필요할 것이다. 전체 복구는 사용자가 최초 전체 백업을 기준으로 데이터베이스에 저장된 최신 파일들만 검색하여 리스트 즉, 최근 백업본을 생성하여 서버는 클라이언트인 새로운 시스템으로 보낸다.



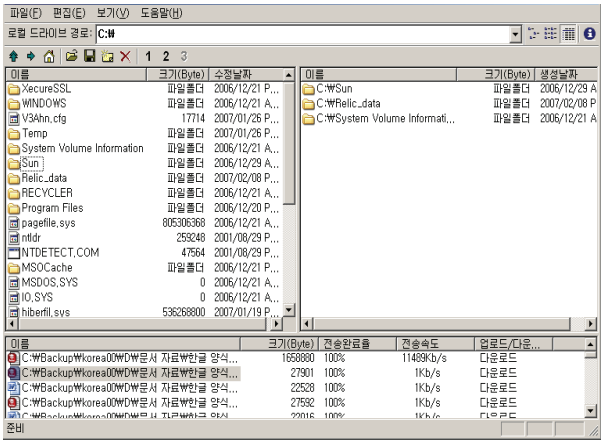
[그림 3] 백업과 복구 순서도

4. 구현과 평가

4.1 클라이언트 프로그램

클라이언트 프로그램은 사용자의 명령을 받고 백업·서버와 필요한 정보를 송수신하는 역할을 한다. 백업 서버와 클라이언트 프로그램은 소켓을 통해 명령을 전달한다. 클라이언트가 백업 또는 복구를 하기 위해 서버에 접속해야 하며 접속을 위해서는 로그인절차가 필요하다.

클라이언트 프로그램의 주요 작업은 백업과 복구 명령 수행이다. [그림 4]는 클라이언트 프로그램의 동작 모습이다. 클라이언트 시스템의 파일탐색기이고 오른쪽 상단 리스트 뷰는 백업을 실행할 디렉터리 목록을 보여주고 있다. 하단의 리스트 뷰는 파일의 송수신의 상태를 나타낸다.



[그림 4] 클라이언트

4.2 서버 프로그램

서버 프로그램은 클라이언트의 요청에 따라 백업·복구 기능을 동작한다. 클라이언트에서 전송된 명령을 처리하고 명령의 성공실패 여부를 전송한다.

```
Backup 과정
입력 : filelist, backup_title
출력 : 성공 1, 실패 0
{
    // 백업이름으로 검색 성공 1, 실패 0
    if( !search_backup_title(backup_title) ) {
        create_backuptable(backup_title);     // 백업테이블 생성
    }else {
        update_backuptitle(backup_title);  // 백업테이블 속성 업데이트
        compare_filelist(filelist); // 비교 후 새로운 파일리스트 생성
    }

    create_directory(backup_title, filelist);//디스크에 디렉터리 생성
    // 클라이언트로부터 파일 다운로드
    if( download(backup_title, filelist) ) {
        return 1;
    } else {
        return 0;
    }
}
```

[그림 5] 백업 알고리즘

[그림 5]는 백업을 수행하는 알고리즘이다. 클라이언트에서 전송받은 백업 이름으로 테이블을 검색한다. 백업이름이 검색되지 않았다면 전체백업으로 백업을 하고 검색이 되었다면 부분백업을 한다.

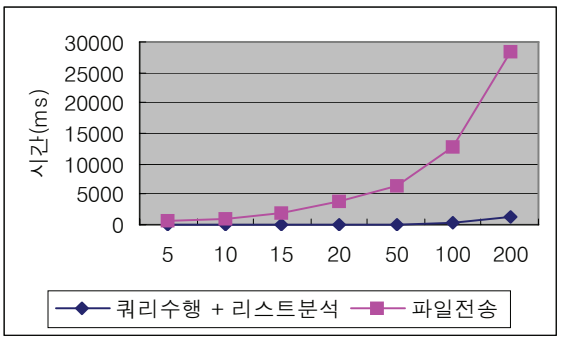
```
Restore 과정
입력 : selectformat, backup_title, filelist
출력 : 성공 1, 실패 0
{
    // 전체복구(1), 부분복구(0) 인지 나뉨
    if( selectformat ) {
        find_recentlist(backup_title, filelist);// 최근 백업 리스트를 찾음
    }else {
        // backup_title에 대한 모든 파일의 리스트를 보냄
        // 클라이언트는 이에 대해 선택한 리스트를 보냄
        upload_list(backup_title);
    }
    // 클라이언트에게 파일 업로드
    if( upload(backup_title, filelist) ) {
        return 1;
    } else {
        return 0;
    }
}
```

[그림 6] 복구 알고리즘

[그림 6]은 복구를 수행하는 알고리즘이다. 클라이언트에서 백업이름과 전체복구를 할 것인지 부분복구를 할 것 인지를 결정해 전송한다. 클라이언트에서 전체 복구를 요청할 경우는 최근 백업리스트를 찾아서 클라이언트로 보내준다. 부분 복구일 경우에는 클라이언트가 전체 리스트사이 몇 가지를 선택하면 그 선택 리스트를 전송한다.

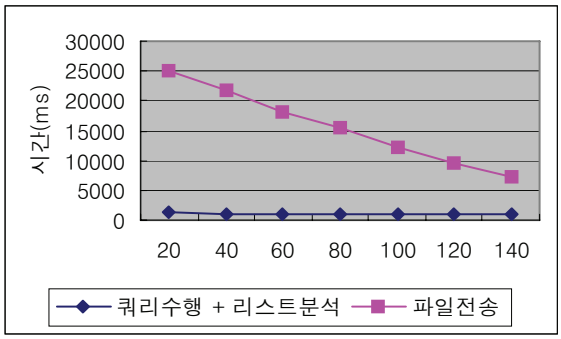
4.3. 실험환경 및 평가

본 논문에서 제안하는 백업 프로그램의 환경은 백업 서버 프로그램의 제작을 위해 gcc, TCP/IP socket, mysql API를 사용하였다. 서버 테스트 환경은 Redhat Fedora Core 4에서 구현하여 테스트 하였다. 서버와의 통신을 위해 클라이언트는 Winsock API를 사용하여 Windows XP에서 구현 및 테스트를 하였다



[그림 7] 파일 개수와 용량에 따른 완료시간

[그림 7]의 그래프는 1M파일이 기본단위인 파일의 집합으로 이루어진 (5, 10, 15, 20, 50, 100, 200)M 바이트의 데이터에 밀리 초 단위의 백업시간을 나타내었다. 파일 개수와 용량이 증가할수록 측정시간은 비례하며 파일 개수가 증가할수록 데이터베이스 수행시간, 기타 작업 시간이 늘어나는 것을 볼 수 있다.



[그림 8] 중복 파일에 따른 완료시간

[그림 8]은 1M파일이 기본단위인 파일 집합 200M 바이트의 데이터에서 (20, 40, 60, 80, 100, 120, 140)M 바이트의 중복 파일을 있는 데이터를 백업한 수행시간이다. 중복 파일이 증가 할수록 수행시간이 상대적으로 줄어듦을 알 수 있다.

5. 결론

본 논문에서는 자동, 주기적인 백업을 통해서 사용자의 정보손실을 줄이고 범용적인 환경에서 사용할 수 있는 소프트웨어를 설계하고 구현하였다. 본 실험을 통해서 백업 수행 시간의 대부분이 파일 전송에 걸리는 시간임을 알 수 있었다. 본 연구에서 개발된 소프트웨어는 개인 사용자나 소규모 사용자 그룹이 간단히 PC를 백·복구서버로 활용하여 중요한 데이터를 안전하게 관리할 수 있으며 파일 전송 속도만 향상시킨다면 더 좋은 성능을 보장받을 수 있다.

참고 문헌

[1] "Backup/Recovery Tutorial," Storage Networking Industry Association, 2001.

[2] A. L. Chervenak, V. Vellanki, Z. Kurmas and V. Gupta. "Protecting File System, A Survey of Backup Techniques," Pro. Joint NASA and IEEE Mass Storage Conference, 1998.

[3] "왜 백업이 필요한가?", http://www.veritas.co.kr/whitepaper.asp

[4] W. Curtis Preston and Gigi Estabrook, "UNIX Backup and Recovery," O'REILLY, 730 pages, 1st Edition November 1999.