Snapshot 알고리즘을 이용한 On-line 백업시스템 설계

(Designing of On-line Backup System using Snapshot Algorithm and SCSI-Protocol)

김 정 기 *, 이 춘 구*, 박 순 철*
(Jeong-Gi Kim, Chun-Gu Lee, Soon-Cheol Park)
{xabec,johny}@internet.chonbuk.ac.kr, scpark@moak.chonbuk.ac.kr

요 약 디스크의 용량이 커지고 정보화 시대와 더불어서 데이터가 폭발적으로 증가하게 되었다. 또한 현재 이 데이터들에 대한 정보 교환이나 접근은 시간과 공간에 관계없이 항상 이루어져야 하는 상황에 있다. 본 논문에서는 365일 지속적으로 운용을 해야만 하는 서버에서 데이터를 효율적으로 백업하기 위한 시스템 사항 및 설계를 소개한다. 백업은 크게 논리적인 백업과 물리적인 백업으로 구분할 수 있는데 일반적으로 물리적(블록단위) 백업을 성능 면에서 우수하지만 구현상의 어려움이 있다. 우선 지금까지의 백업의 방식이나 방법에 대해서 간략히 언급을 하고 온라인(on-line) 백업을 수행하기 위한 기술적인 사항에 대해서 알아본 다음, 백업을 하기 위한 시스템 설계에 대해서 소개하고자 한다. 본 백업 시스템은 데이터 전송의 효율을 높이기 위해서 SCSI-Protocol을 지원한다.

Abstract In these day, the storage capacity has been growing and the informations increase in geometrically progression in a modern information age. At present the situation is required that the exchanges and access of the informations is available without regard to time and space. In this paper we present a effective backup systems to backup the information of server that continuously has to be accessible in 365's days. There are two backup cases. one is a file-base backup and the other is a block-based backup. In general, a block-based backup has a good performance than a file-based backup but it is difficult to implement a block-based systems. First, we introduce the technique of backup and methods, on-line backup. Next, we present our backup systems based on client-server model and have a snapshot facility. Finally we introduce SCSI-Protocol in order to increase the data transfer rate.

1. 서 론

정보화 시대와 더불어서 정보를 공유하고 접근하는 방식에 있어서 인터넷이라는 새로운 패러다임을 만들었으며 이로 인해서 창출되는 정보의 양은 기하급수적으로 증가하게 되었다. 이에 따라서 정보를 관리하고 유지하는 것이 매우 중요한 문제로 대두되기 시작했으며, 이런 정보들을 효율적으로 관리하고 저장하기 위한 다양한 방법이 새로이 등장하게 되었다. 스토리지 쪽에서는 최근 서버와 스토리지를 분리한 네트워크 연결형 스토리지 스시템이 등장했으며 NAS와 SAN이 대표적인 예이다.

스토리지를 관리하기 위한 소프트웨어에서 가장 중요한 것이 스토리지 속에 들어 있는 데이터들을 항상 가용한 상태로 유지하는 데에 있으며 이런 소프트웨어의 정점에 있는 것이 바로 백업에 관련된

* 전북대학교 정보통신공학과

프로그램이다. 백업의 기본적인 개념은 유사시 시스템의 장애나 오류로 인한 손실에 대비해서 주기적으로 중요 데이터들을 저장하고 장애 발생시 일정 시점으로 복원하는 데에 있다.

본 연구에서는 365일 운용 가능한 정보 시스템에서의 백업을 위한 온라인 백업과 이를 수행하기 위해 필요한 스냅샷(Snapshot)의 개념 및 원리에 대해서 다룰 것이며, 백업의 방식이나 방법 등에서도 다룰 것이다. 그리고 본 시스템의 클라언트/서버 구조 백업 시스템을 소개할 것이며, 마지막으로 데이터 전송을 높이기 위한 방안으로 SCSI-Protocol 사용에 대해서 기술할 예정이다.

2. 관련 연구

본 연구와 관련된 연구는 외국의 몇몇 스토리지 업체들이 이미 스토리지 솔루션으로 발표했고 또한 온라인 백업을 제공하는 전용의 파일 시스템이 고안되기도 하였다. 대부분 백업 시스템들은 온라인 백업을 지원하기 위해 백업을 위한 전용의 파일 시스템들이 대부분을 차지한다. 관련된 연구 결과로써는 Vertias에서 내 놓은 NetBackup이나 Legato의 Networker, UniTree의 Hierarchical Storage Manager, EMC의 SnapView, IBM의 Tivoli Storage System, Network Appliance의 SnapMirror와 SnapRestore, 그리고 Mountainviewdata의 MVD Snap등을 들 수 있겠다. 파일시스템으로는 Petal System & Frangipani System, Plan 9, 로그를 기반한 Log-Structured File systems, 역시 로그를 기반한 Spiralog File Systems, SnapFS, Andrew File System 등 많은 파일 시스템들이 있다.

중요한 것은 위에서 언급한 업체의 솔루션이나 파일시스템들이 모두 온라인 백업을 지원한다는 것이고 이를 위해서 스냅샷을 지원한다는 점이다. 스냅샷에 대한 개념이나 원리는 나중에 다시 다룰 것이다 스냅샷을 지원하게 되면 어느 한 시점에서의 그 디스크내의 전체 이미지를 만들 수 있으며, 이 스냅샷을 이용해서 백업 프로그램은 서버를 다운시키지 않고도 백업을 수행하게 된다. 본 연구에서는 스냅샷을 지원하는 클라이언트/서버 방식의 백업 시스템을 리눅스 머신에 설계하려고 한다.

3. 온라인 백업

온라인 백업이란 정보 시스템의 자료들을 사용자들의 계속적인 접근을 허락하면서도 백업을 수행하는 것이다. 온라인 백업은 실제로는 사용자의 접근이 지속적으로 이루어지기 때문에 파일 시스템이 계속 변하지만 백업을 수행을 하는 프로그램의 입장에서는 볼 때는 백업을 수행한 그 시점을 이후로는 어떠한 변환도 없는 것으로 보여야만 한다. 그래야 한 시점(point-in-time)에 대해서 시스템의 데이터의 무결성을 보장할 수 있다.

네트워크 연결형 스토리지 시스템에 등장하기 이전의 기존의 정보시스템들의 구조는 모두 서버에 종속적인 스토리지 시스템이었다. 이로 인해서 중요 데이터들을 백업하기 위해서는 서버의 운영을 중지한 뒤 오프라인(off-line)상에서 백업을 해야 한다는 단점이 있었다. 이에 비해서 네트워크 연결형 스토리지 시스템은 서버의 운영을 중지하지 않은 채 해당 디스크에 들어있는 데이터를 다른 백업 스토리지 라이브러리에 백업할 수 있다. 하지만 이것은 어디까지나 서버의 운영을 중지하지 않고 백업할 해당 디스크에 대해서 따로 백업을 수행할 뿐이지 서버 종속적인 스토리지 시스템 역시 마찬가지이다. 다시 말해서 네트워크 연결형 스토리지 시스템에서의 백업은 사용자의 네크워크에 대한 부하는 완전히 제거했지만(LANFree 백업) 여전히 서버의 자원을 사용하며 또한 해당 디스크를 백업할 때 이 디스크에 들어 있는 정보는 서버 종속적인 스토리지 시스템과 마찬가지로 오프라인 즉, 서비스를 중단한 상태에서 백업을 수행해야 한다. 이렇게 해야만 하는 이유는 어는 한 시점에서의 완전한 전체 백업 이미지를 갖고 있어야하지 그렇지 않으면 복구시 데이터 무결성에 문제가 발생하게 된다. 다시 말해서 백업 도중에 어떠한 디스크의 I/O가 일어나서는 안 된다는 의미이다.

예를 들어서 백업을 10시에 시작했다고 하고 그리고 백업이 걸릴 시간이 1시가 30분이라고 하면 백업이 끝나는 시각은 11시 30분이다. 그리고 A라는 파일이 백업하고자 하는 디스크에 존재한다고 하자.

이때 10시부터 11시30분 동안 사용자들은 계속적으로 시스템 자원을 사용하게 될 것이고 필요에 따라서는 데이터를 수정, 삭제, 생성을 하게 될 것이다. 만약 백업 시스템이 온라인 백업을 위한 기술적인 부분을 지원하지 않는다고 하면 백업을 하는 도중에 A라는 파일이 사용자에 의해서 변경되었고 그리고 구동된 백업 프로그램은 아직 A라는 파일을

아직 백업 하지 않았다고 한다면 백업이 끝나는 시간 11시 30분 이후에 백업된 이미지 속에 들어 있는 A라는 파일은 처음 백업이 시작되는 시간 10시 이전의 내용과 다른 것을 갖고 있게 된다. 마찬가지로 이미 백업을 수행한 부분에서 어떤 데이터 변경. 이루어 졌을 경우 이 변경된 부분은 백업이 이루어지지 않은 채 남아 있게 된다.

이런 이유로 온라인 백업 프로그램은 반드시 어는 한 시점에서 그 디스크(혹은 파일 시스템)에 완전한 이미지를 보장해줘야 한다. 또한 위의 글을 읽으면서 약간은 느꼈겠지만 동시성의 문제가 있다. 즉, 백업을 하고 있는 파일에 대해서 동시에 쓰기가 일어나는 경우 동시성 문제가 발생한다. 백업 도중에 발생하는 디렉토리나 파일이 이동이나 변경을 막기 위해서 시스템은 쓰기나 이동, 삭제 등에 대해서 락(lock)을 걸어야 한다. 하지만 이런 락은 파일 시스템의 성능을 저하시키기 때문에 때에 따라서 오버헤드가 가질 수 있다. 이런 오버헤드는 락을 거는 시간을 최소화함으로써 어는 정도 극복할 수 있다.

동시성의 문제뿐만 아니라 요구되어야 할 또 하나의 것은 파일이나 디렉토리의 변경에 대한 디텍션(detection)이다. 이 것은 과거의 파일 시스템과 현재의 파일 시스템과 비교했을 때 변경되는 부분을 발견해서 다음 백업시 반드시 포함시켜야 한다.

위에 언급한 락킹이나 디텍션(detection)은 전체적으로 백업 성능을 어느 정도 저하시키고 백업을 하더라도 어느 한 시점에서 디스크 이미지를 보장해 주지 못한다. 한 시점에서의 디스크 이미지를 보장해 주기 위해서는 반드시 스냅샷을 지원해야만 한다.

4. 백업 테크닉

백업 프로그램을 디자인 할 때 디자인 할 백업 시스템이 가져야할 여러 가지 특성을 고려해서 요구된 환경에 맞는 프로그램을 작성해야만 한다. 이 장에서는 백업 시스템이 일반 적으로 갖는 특성에 대해서 알아보고자 한다.

백업 시스템이 갖는 특성을 나누면 다음과 같이 몇 가지로 구분할 수 있다. 우선 풀(Full) 백업과 증분(Incremental) 백업, 차등(Differential) 백업; 파일 기반의 백업과 물리적 기반의 백업; 온라인 백업 지원; 스냅샷과 COW(copy-on-write) 메커니즘; 다중 백업; 데이터 압축 등의 범주로 나눠 볼 수 있다.

우선 풀 백업은 파일 시스템 전체를 백업 미디어에 저장하는 방법이다. 백업하고자 하는 파일 시스템의 크기에 따라서 소요되는 백업 미디어도 비례해서 증가를 하지만 시스템 장애 발생시 풀 백업으로 전체 파일 시스템을 복구 할 수 있다. 풀 백업은 백업을 하기 위해 소비되는 백업 미디어의 증가뿐만 아니라 파일 시스템 전체를 백업을 하기 때문에 소요되는 시간이 오래 걸리고 속도 또한 많이 걸리게 된다. 이를 보완하고자 나온 것이 증분 백업과 차분 백업이라고 볼 수 있다. 먼저 증분 백업은 이전 백업 이후에 새로이 생성된 파일이나 변경된 파일만을 백업하는 방법이다. 따라서 풀 백업에 비해서 빠르며 백업 용량도 작게 된다. 차분 백업은 증분 백업과 비슷하나 백업하는 대상이 다르다. 증분 백업은 이전 백업 이후 변경되거나 생성된 파일만을 백업하는 데에 비해 차분 백업은 이전 백업 이후에 변경된 파일을 백업하는 것에서는 동일하나 백업을 완료 된 후 백업을 했음 나타내는 속성을 활성화 시키지 않는다. 따라서 현재의 백업 다음인 다음 백업에서도 현재 백업을 했던 파일들이 그대로 모두 다시 백업을 하게 되는 것이다. 따라서 백업에 소요되는 미디어도 증분 백업보다는 많다. 하지만 풀 백업에 들어가는 미어보다는 작다. 차분 백업의 장점은 풀 백업 뒤에 차분 백업을 했다고 가정하면 복구할 때 풀 백업을 먼저 복원하고 여러 번의 차분 백업 중에서 맨 마지막 차분 백업만을 복원하면 전체 파일 시스템이 마지막 백업 시점까지 완전히 복구가 된다. 이에 반해 차분 백업은 풀 백업을 복원하고 매 차분 백업에 대해서 차례대로 복원을 해야만 완전한 파일 시스템으로 복구가 된다. 백업을 정책에 관해서는 본 연구에서는 언급하지 않겠다.

백업은 크게 두 가지 방식으로 백업을 할 수 있다. 하나는 파일 기반의 백업(논리적 백업)이고 나머지 하나는 물리적 기반의 백업이다. 이 기반위에서 풀 백업 및 증분 백업 등이 이루어지는 것이다. 일반적으로 파일 시스템은 물리적인 디스크를 논리적인 개념으로 추상화해서 접근을 하게 된다. 이때 논리적인 단위를 블록(block)이라고 하는데 대체로 4 kbytes(리눅스의 경우)로 되어 있다. 물론 운영체제에서 지원하는 파일 시스템의 종류에 따라서 이 블록의 크기는 변할 수 있다. 파일 기반의 백업은 파일과 디렉토리 구조, 그리고 이들의 속성을 그대로 보존해서 백업 미디어에 저장을 하게 된다. 백업 프로그램은 백업하고자 하는 파일 시스템은 순차적으

로 탐색을 하면서 백업을 수행하게 된다. 파일 기반의 백업은 장점은 파일 시스템하에서 백업을 수행하기 때문에 개개의 파일에 대해서 쉽게 복원이 가능하다. 또한 원하는 파일만을 백업할 수도 있고 반대로 배제하고 싶은 파일을 지정해서 그 파일들만을 배제한 채 백업을 할 수도 있다. 하지만 파일들이 파일 시스템 내에서 연속적으로 할당되어 있다고 불수 없고 대체로 디스크 내에 여러 곳에 분할되어서 저장되어 있기 때문에 파일단위로 백업을 한다는 것은 디스크 내에 여러 곳을 탐색을 해야 하기 때문에 이로 인한 디스크 오버헤드가 증가하게 된다. 또한 하나의 파일에 대해서 아주 조금만의 변화가 있었다고 하더라도 그 파일 전체를 백업을 하게 된다. 물론 지금과 같이 저장 매체의 기술이 발달하고 또한 저장 공간이 늘어난 시점에서 문제의 소지가 적겠지만 1 bytes만의 변화에 대해서 전체 파일을 백업을 해야 한다는 것은 비효율적인 방법으로 보여질 것이다.

이에 반해서 물리적 기반의 백업은 바로 위에서의 문제점을 해결할 수 있다. 즉 변화된 블록에 대해서 백업을 수행함으로서 전체 파일을 백업하지 않아도 된다는 점이다. 물리적 기반의 백업은 파일 시스템을 무시하고 디스크의 블록에 기준을 두고 백업을 수행하게 된다. 다시 말해서 백업을 수행할 때 디스크의 전체 블록에 대해서 파일 시스템의 구조와 상관없이 백업을 수행을 한다. 만약 풀 백업을 수행한다고 하면 디스크의 0번째 블록부터 그 디스크의 마지막 블록까지 순서대로 백업을 하게 된다. 이후에 증분 백업을 하게 되면 변화된 블록들만은 체크해서 백업을 수행하면 된다. 물리적 기반의 백업은 개념은 파일 기반의 백업에 비해서 상당히 좋은 성능을 보여 준다 또한 소요되는 백업 미디어도 변환된 블록만을 백업하기 때문에 상당히 적게 들어간다. 하지만 파일 단위로의 백업을 지원하기 위해서는 백업한 블록과 이 블록이 어떤 파일의 내용을 백업 했는지 정보들이 같이 백업 미디어에 저장 시켜줘야 한다. 또 하나의 문제는 변화된 블록을 알아내는 방법이다. 변화된 블록을 알아내기 위해서는 운영체제 내부에 변환된 블록을 체크할 수 있는 모듈이 설계되어 있던지 아니면 파일 시스템 자체가 이를 지원해야만 한다. 때문에 상당히 특수화된 파일 시스템을 요구하게 되어 범용적이지 못하게 된다. 마지막으로 또 하나의 문제는 버퍼 방식의 디스크 입출력 때문에 발생하는 데이터 무결성이다. 일반적으로 운영체제는 디스크 입출력 효율을 높이는 방식으로 커널내에 버퍼를 두어서 사용하게 되는데 이로 인해서 지연 쓰기가 된다. 즉 버퍼에 있는 내용일 디스크에 쓰여지기 전에는 디스크에 내용은 변화되기 이전의 내용을 포함하고 있는 것이다. 따라서 백업이 이 부분을 고려치 않으면 데이터 무결성에 문제가 생기게 된다.

스냅샷은 일반적으로 블록 단위로 정보를 관리하면 파일 시스템 내에서 변화된 블록에 대한 내용을 보관하게 된다. 또한 위에서 언급한 데이터 무결성 또한 고려해서 스냅샷이 생성 될 대 모든 버퍼를 플러쉬(flush)해 무결성 문제를 처리를 한다. COW는 통상 스냅샷과 함께 사용되어진다. 스냅샷이 만들어지게 되면 이후에 발생하는 파일이나 디렉토리에 대한 변경에 대해서 변경되 내용을 실제로 디스크에 쓰기 전에 변경된 내용이 저장될 수 있는 새로운 블록을 할당한 후 이 블록에 변경된 내용을 저장한다. 따라서 원래의 위치에 있는 블록의 내용은 없어지지 않고 그대로 남아 있게 된다.

다중 백업은 말 그래도 동시에 여러 개의 백업을 수행하는 것을 말한다. 대규모의 네트워크 규모를 갖고 있다면 백업 수행을 하나만을 지원해서는 비효율적일 것이다. 이런 다중의 백업을 수행하다 보면 동시성의 문제가 발생할 수 도 있다. 즉 같은 곳에 대해서 중복된 백업을 수행한다든지 두 백업 프로세스가 같은 곳을 접근하는 하는 문제가 그것 일 것이다. 또한 다양한 파일 시스템들이 존재하기 때문에 백업 프로그램이 이들 파일 시스템에 대한 지원여부도 문제가 될 수 있다.

데이터 압축은 파일 기반의 백업에 있어서 필요한 옵션일 수 있다. 앞에서도 언급했지만, 파일 기반의 백업은 파일 시스템 전체를 백업 미디어에 저장하기 때문에 미디어 소요가 많이 된다. 만약 백업을 수행시 데이터를 압축해서 하게 된다면 어는 정도 백업 미디어를 줄 일 수 있을 것이다.

5. 스냅샷 (Snapshot)

앞에서도 잠깐 언급했지만 스냅샷은 온라인 백업을 위해서는 필수적이라 하겠다. 스냅샷은 현재 사용하고 있는 파일 시스템에 대한 읽기만 가능한 클론이미지라 볼 수 있다. 즉, 스냅샷이 생성되면 그 순간에 사용중인 파일 시스템과 동일한 구조를 갖는 클론 파일 시스템을 생성하는 것이다. 그리고 이후

에 발생하는 연속적인 디스크 연산은 COW 메커니즘을 사용해서 활성화된 파일 시스템의 원본 이미지는 그대로 놔두고 새로운 블록을 할당해서 그 위치에 변경된 내용을 쓰고 활성화된 파일 파일 시스템은 이 블록에 대한 위치 정보를 갖게 한다. 이렇게 함으로써 변경 이전의 블록 정보는 스냅샷 공간이 갖고 있게 되고, 활성화된 파일 시스템은 변경된 블록 정보를 포함한 블록 위치 정보를 갖게 되는 것이다. 스냅샷이 생성된 이후의 모든 데이터 변경은 위에서 언급한 방법으로 작업을 수행하게 된다. 중요한 것은 스냅샷이 블록 단위로 데이터를 연산한다는 점이다. 즉 하나의 파일에 대해서 변경된 블록이 있을 경우 파일 전체를 스냅샷 공간에 복사하는 것이 아니라 변경된 그 블록만을 스냅샷 공간에 복사를 한다. 이런 이유로 실제로 스냅샷이 파일 시스템에서 차지하는 비율은 상대적으로 작다.

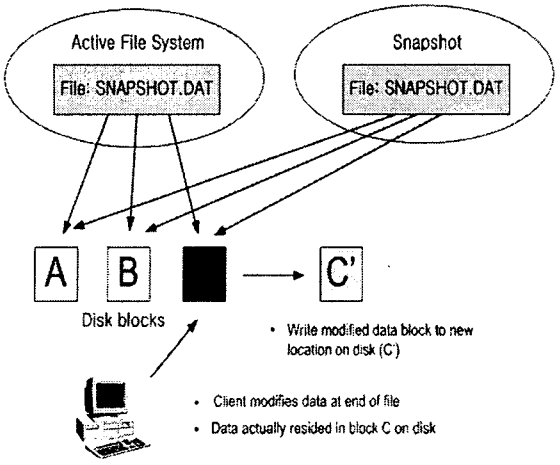

그림 1

디스크에 SNAPSHOT.DAT라는 파일이 있고 이 파일은 디스크에서 세 개의 블록 A, B 그리고 C를 차지하고 있다고하자. 스냅샷이 생성이 되면 SNAPSHOT.DAT라는 파일은 활성화된 파일 시스템뿐만 아니라 스냅샷 공간에도 동등한 파일이 존재하게 된다. 그림 2에서와 같이 클라이언트가 파일 수정을 했고, 수정한 부분이 이 파일의 끝부분 이었다고 가정했을 경우 스냅샷은 COW 메커니즘에 의해서 변경된 블록이 저장 될 수 있는 새로운 블록 C'를 생성한 후 이 블록에 변경되 데이터를 쓴다. 그리고 예전에 갖고 있는 블록 정보는 버리고 새로이 할당 받은 블록에 대한 정보를 활성화된 파일 시스템이 갖는 것이다. 이 과정이 끝난 후의 활성화된 파일 시스템과 스냅샷 공간의 파일이 위치하고 있는 블록 정보는 그림 2에서과 같이 나타내어 진다. 즉, 스냅샷 공간에서의 파일은 변경 전의 이미지를 그래도 간직하고 있는데 반하여 활성화된 파일 시스템에서는 변경된 내용이 적용된 블록 정보를 갖고 있다.

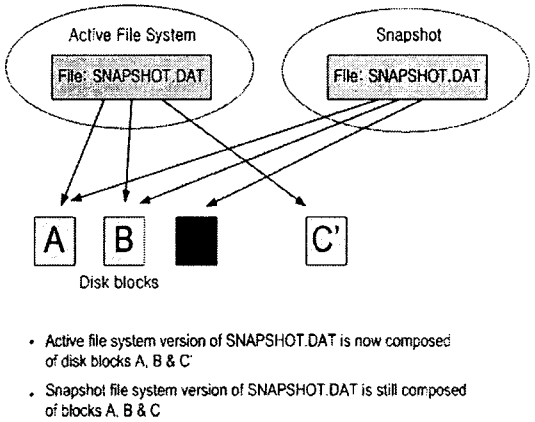

그림 2

스냅샷의 장점은 바로 온라인 상태에서 파일 시스템에 대한 전체 이미지를 갖고 있다는 것이다. 이것이 가능하기 때문에 온라인 백업이 가능한 것이다. 스냅샷의 스케줄링은 관리자가 시스템의 사용 정도를 파악해서 적당한 시간 간격을 만들어서 해야겠지만 일반적으로 하루를 생각했을 경우 4시간마다 스냅샷을 생성하며 이 스냅샷은 24시간동안 유지를 한다. 또한, 매일 자정에 스냅샷을 만들고 이 스냅샷은 2일을 유지하는 정책을 따른다. 이미 언급했지만 이 정책은 어디까지 일반적인 상황에 대한 것이지 모두 다 적용될 수는 없는 것이다. 예를 들어서 디스크의 입출력 찾아 데이터의 변경이 자주 일어난다면 스냅샷은 주기는 좀더 짧아야 할 것이다. 이와 반대로 데이터의 변경이 없는 멀티미디어 서비스를 하는 시스템에서의 스냅샷은 자주 할 필요가 없을 것이다.

스냅샷이 만들어진 후 데이터 변경에 대해서 COW 메커니즘이 적용되는데 COW의 메커니즘을 적용 시킬 때 두 가지 방법이 있을 수있다. 하나는 위에 수행한 방법인 새로이 저장될 내용에 대한 블

록을 할당하고 그 블록에 데이터를 쓴 후 이 블록에 대한 정보를 활성화된 파일 시스템에 갖는 방법이 하나이고 다른 하나는 새로이 저장될 내용을 원래의 블록 위치에 저장을 하는데 덮어 쓰기 전에 원래의 블록에 있던 내용을 새로이 할당 받은 블록에 카피를 한 뒤에 덮어 쓰는 방식이다. 후자의 경우는 스냅샷이 새로이 할당 받은 블록에 정보를 관리하게 된다. 후자의 경우는 원본 내용을 보존하기 위해서 스냅샷 공간으로 데이터 복사가 일어나기 때문에 전자에 비해서 성능이 떨어진다고 볼 수 있다. 아래의 그림에 후자의 경우로 했을 경우 각 파일 시스템에 갖고 있는 블록 정보를 보여 준다.

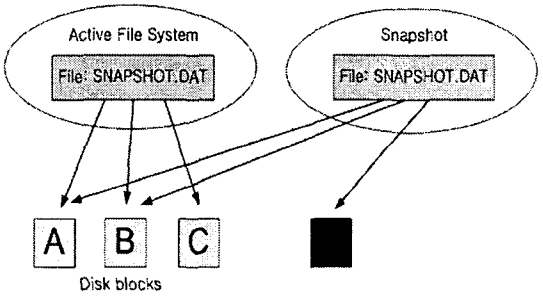

그림 3

스냅샷은 크게 하드웨어적인 스냅샷과 소프트웨어적인 스냅샷으로 나뉠 수 있다. 전자의 방법은 Split-Mirror로 알려져 있는데 이는 Primary 디스크로부터 mirror를 실행 시킨 후 Split-off된 디스크로 백업을 수행한 다음 백업이 완료된 시점에서 mirror된 볼륨과 재동기화를 거쳐서 스냅샷이 이루어지 방식이다. 이 방식의 특징은 데이터에 대한 물리적인 카피를 수행을 한다는 점과 여분의 디스크가 필요하다는 점이다. 데이터의 변경에 대한 캡춰는 볼륨 레벨에서 이루진다. 또한 개별 파일보다는 전체 볼륨을 복구해야 합니다. 하드웨어 스냅샷은 일반적으로 디스크 어레이나 서브시스템에 상주하는 데이터의 물리적 복사본이다.. 이들 데이터 세트는 데이터의 볼륨 레벨 복사본이다. 이 데이터를 백업하기 위한 mirror는 분할되어 있으며 업무 시스템에 영향을 주지 않으면서 백업될 수 있다.

대부분 볼륨 매니져에서 이 기능을 지원하게 된다. 대표적인 예로 GFS에 포함된 LVM(Logical Volume Manager)를 들 수 있겠다.

소프트웨어적인 스냅샷 방법은 위에서 언급한 copy-on-write 방식을 취한다. 하드웨어 스냅샷과는는 데이터가 볼륨 레벨에서 관리가 되기 때문에 파일이라는 개념보다는 거의 물리적인 개념이 강하다 따라서 개별 파일보다는 완전한 볼륨 볼륨세트를 복구 해야만 의미가 있는 것에 반해 소프트웨 스냅샷을 디스크의 논리적 단위이 블록 레벨이나 파일레벨에서 스냅샷을 지원하기 때문에 보다 효율적으로 데이터를 복구 할 수 있다.

기존 파일 시스템서 스냅샷을 지원하기 위해서는 기존 파일 시스템 위의 계층에서 스냅샷을 처리할 수 있는 부분이 추가가 되어져야할 것이다. 물론 처음부터 스냅샷을 지원하는 전용의 파일 시스템을 작성할 수도 있다. LFS나 Andrew 파일 시스템이 그 예라고 할 수 있다. 본 연구에서는 리눅스 파일 시스템에서 스냅샷을 지원할 수 있도록 가상 파일 시스템과 ext2 및 기타 파일 시스템에 사이에 스냅샷 층을 두어서 온라인 백업이 가능한 시스템을 설계하는 것을 소개할 것이다.

6. 백업 시스템

앞에서 우리는 백업 시스템이 가져야할 여러 가지 기술적인 측면이나 백업 방식에 대해서 언급을 했다. 이 외에도 백업 시스템에 가져야할 요구 사항들은 많이 있다. 최근의 네트워크 연결형 스토리지 스템에서는 서버와 스토리지들이 분리된 형태로 되어 있기 때문에 이런 환경을 지원할 수 있는 형태로 백업 시스템에 설계가 되어져야 할 것이다. 특히 SAN 환경에서의 백업은 LANFree 백업이나 Serverless 백업등을 지원하는 백업 시스템을 설계해야 할 것이다.

본 연구에서의 백업 시스템은 리눅스상에서 운용할 것이며 원격 백업을 지원한다. 또한 온라인 백업을 수행하기 위해서 스냅샷을 지원하는 부분을 리눅스의 가상 파일 시스템 일반 파일 시스템 사이에 두어서 모든 디스크 입출력을 제어를 할 것이다. 즉 VFS에서 지원하는 대부분의 주요 시스템 콜이 opne, read, write, unlink 등을 필터링을 통해 연산을 수행하게 된다. 근본적으로 하나의 새로운 파일

이 생성되거나 변경된 데이터가 쓰여질 경우 현재 스냅샷이 설정되 상황이었다면 실제의 작업을 수행하기 전에 스냅샷을 위한 copy-on-write를 수행하는 다음에 처음의 요청 작업을 수행하는 것이다. 본 연구의 백업 시스템에서 Snapshot을 지원하기 위한 핵심 모듈 구조는 다음과 같다.

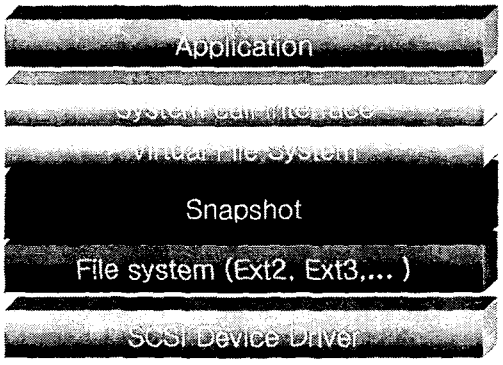

그림 4

그림 4에서 보듯이 Snapshot 모듈이 VFS와 실제 파일 시스템 사이에 존재하게 되어 필요한 시스템 콜을 필터링하게 된다.

본 백업 시스템은 클라이언트/서버 모델을 따르며 백업 서버는 데몬 형태로 정보 시스템의 서버의 메모리에 상주하게 된다. 그리고 클라이언트의 요구에 적절히 대응을 해주며 또한 백업이나 리스토어 요구시 해당 작업을 수행한다. 백업 시스템에서의 데이터 흐름은 디스크의 데이터가 서버 데몬을 경유해서 백업 미디어로 이동하게 하게 된다. 데이터 이동에 있어서 전송 속도를 높이기 위해서 SCSI 프로토콜을 사용했다. 서버쪽의 전체적인 모습은 그림 5와 같다.

네트워크 모듈은 클라이언트와의 통신을 위한 부분이며 클라이언트와의 모든 통신은 이 모듈을 통해서 이루어지게 된다. 클라이언트와의 통신은 정해지 메시지 규칙에 의해서 필요한 작업 요구 및 작업 명령 등을 보낼 수 있다. 백업과 리스토어 모듈은 리눅스 커널에 모듈로 적재되어 있는 snapshot 모듈을 이용해서 온라인 백업을 수행하고 필요시 복구를 하게 된다. 스케쥴 모듈은 백업 서버가 주기적으로 백업을 해야 할 예약 작업을 담당하는 부분이며 로그 모듈은 프로그램이 수행하면서 생성된 모든 로그들을 관리하는 부분이다.

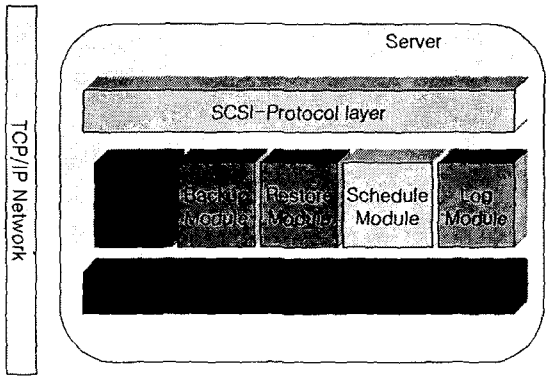

그림 5

7 SCSI-Protocol

SCSI-Protocol은 SCSI 디스크에서 사용되는 프로토콜이다. 현재 리눅스 커널에는 다양한 SCSI 호스트 어탭터에 대한 드라이버가 탑재 되어 있다. SCSI 프로토콜은 효율적인 peer-to-peer 방식의 디스크의 I/O을 제공하기 위해서 설계되었다. EIDE가 갖고 있는 장치의 개수의 제한을 어는 정도 해결 했으며 또한 빠른 데이터 전송 속도를 보여준다. 현재 SCSI-2까지 나온 상태이며 SCSI-3는 드래프트 상태이다. SCSI-3 프로토콜이 지원하는 디스크 장비가 지원된다면 백업에서의 데이터 이동은 데몬 서버를 경유하지 않고 바로 디스크에서 디스크 혹은 디스크에서 백업 미디어로 이동할 수 있는 Severless 백업을 구현할 수 있다. 물론 백업 프로그램이 SCSI-3 프로토콜을 지원해서 작성을 했을 경우에만 해당 되는 얘기이다. 현재 네트워크 연결현 스토리지 시스템에서 Serverless 백업은 디스크 장비에서 지원하는 것이 아닌 스위치(Switch)나 허브(Hub)에서 중계를 하는 방식으로 Serverless 백업을 구현하고 있다.

8 결론

지금까지 우리는 온라인 백업을 하기 위해서 필요한 여러 가지 백업 방식이나 필요한 기술에 대해서

알아보았다. 본 연구에서는 아직 제안한 시스템에 대한 구현을 완전히 하지 못한 상태이다. 중요한 것은 온라인 백업을 수행하기 위해서는 반드시 스냅샷이라는 기술이 필요하다. 온라인 백업은 인터넷 컴퓨팅 시대에 반드시 필요한 기능이다. 이 것은 서버가 24*7시간 동안 항상 가용한 상태로 유지 되어야만 하기 때문이다. 그리고 백업은 중요한 데이터의 보관이라는 측면에서 매우 중요한 사항이다. 실제로 스토리지 솔루션에서 가장 비중을 많이 차지하는 부분이 바로 이 백업 관련 솔루션이다. 지금까지의 생성된 데이터의 양도 많지만 앞으로 몇 년 안에 생겨날 데이터는 지금까지 생성된 모든 데이터보다 몇 배 많이 생성이 된다고 한다. 이런 데이터를 보관하고 유지하는 문제는 보다 효율적이고 확장성이 유연한 백업 프로그램이 반드시 필요하며 또한 스냅샷과 같이 핵심 백업 기술 또한 연구되고 개발 되어져야 할 것이다.

참 고 문 헌

[1] Ann L. Chervenak, Vivekenand Vellanki, and Zacha교 Kurmas. Protecting file systems: A survey of backup techniques. In *Proceedings of the Joint NASA and IEEE Mass Storage Conference*, March 1998

[2] Norman C. Hutchinson Logical vs. Physical File System Backup. In *USENIX Third Symposium on Operatiog systems Design and Implementation* (OSDI '99)

[3] R. J. Green, A. C. Baird, and J. C. Davies. Designing a fast, on-line Backup System for a log-structured file system.. *Digital Technical Journal of Digital Equipment Corporation*, 8(2):32-45, October 1996

[4] Network Appliance SnapMirror and Snap Restore: Advances in Snapshot Technology http://www.netapp.com

[5] J. Johnson and W. Laing, "Overview of the Spiralog file system", *Digital Technical Journal of Digital Equipment Corporation*, vol 8, no.2 (1996, this issue): 5-14

[6] Veritas white paper Technical Overview: Storage Checkpoints for Block-Level Incremental Backup and Storage Rollback http://www.veritas.com

[7] 신범주, 김경배, 김창수, 김명준 "네트워크 저장장치를 위한 클러스터 파일 시스템 개발" 한국정보처리학회지 제8권 제4호, 한국정보처리학회, pp.29~41, 2001. 7.

[8] E. K. Lee and C. A. Thekkath. Petal: Distributed Virtual Disks. In *Seventh International conference on Architectural Support for Programming Languages and Operating systems(ASPLOS-VII)*, October 1996