SAN을 위한 전역 파일 공유 시스템의 개발

한국전자통신연구원 김경배* · 김영호 · 김창수 · 신범주*

1. 서 론

인터넷 사용의 확산은 업무 환경의 급속한 변화와 함께 저장되어야 할 데이터 양의 폭발적인 증가를 초래하고 있다. 그러나, 하나의 서버에 접속되어 데이터를 저장하고 관리하는 클라이언트 서버 형태의 데이터 관리 시스템이나 파일 서버를 기반으로 하는 네트워크 파일 시스템과 같은 기존의 자료저장 시스템들은 기하급수적으로 늘고 있는 엄청난 양의 데이터를 처리하는 데 한계가 있다[1]. 이러한 문제를 해결하기 위하여 등장한 것이 SAN(Storage Area Network)[2, 3]이다. SAN은 파이버 채널(fibre channel)[4]로 연결되는 고속의 저장장치 전용 네트워크로 서버에 디스크를 직접 연결한 기존의 DAS(Direct Attached Storage)와 달리 저장 장치를 고속의 파이버 채널에 직접 연결시켜 사용할 수 있기 때문에 고성능(high performance), 고 확장성(high scalability), 고가용성(high availability) 및 공유성(sharability)을 제공한다. 이 같은 SAN의 장점은 현재의 자료저장 시스템이 안고 있는 대용량 저장장치 문제들을 효과적으로 해결할 수 있게 한다.

사용자들이 SAN 환경을 보다 효과적으로 활용할 수 있기 위해서는 SAN에 대한 사용자 관점의 뷰(View)를 제공하는 것이 필요하다. SAN Virtualization으로 불리는 이러한 기능은 소프트웨어로 제공되는 SAN 지원 자료 저장 시스템에 따라 다양하게 제공될 수 있다. GFS(Global File System)[1, 6], Veritas[7], SANergy[8], SANux[9] 등은 현재까지 개발된 시스템들이며, 각각 나름의 특성들을 제공하고 있다. 그러나 이들이 가지는 공통적인 특성은 효율적인 대용량 공유 파일 시스템을 지향한다는 것이다.

현재 널리 사용되는 대표적 운영체제인 유닉스, 리눅스 및 Windows는 하나의 서버에 직접 연결되어 있는 몇 개의 소용량 디스크들을 효과적으로 관리하기 위한 위해 개발되었기 때문에 파이버 채널 기술을 이용하여 수백 개 이상의 저장 장치들을 연결하여 동시에 여러 호스트가 접속할 수 있는 SAN 환경에서는 사용할 수 없다. 따라서 SAN virtualization을 제공하기 위해서는 기존 운영체제를 확장하는 것이 필요하다. 앞서 기술하였던 자료저장 시스템에서는 다음과 같은 공통 기능들을 제공하고 있다.

- 64비트용 파일 시스템
- 논리 볼륨(logical volume)의 지원
- 전역 파일 공유 기능 지원
- on-line resizing
- 스냅샷(snapshot)
- 서버에 부담을 주지 않는server-less backup
- 빠른 회복 기능
- 대용량 파일의 저장 및 관리
- 대량의 파일을 저장하기 위한 디렉토리 관리 기법

본 고는 리눅스용 네트워크 연결형 자료저장 시스템 소프트웨어로 SAN 환경에서 대용량 파일에 대한 전역 파일 공유 기능을 지원하기 위해 한국전자통신연구원에서 개발 중인 SANtopia 시스템에 대하여 소개한다.

본 고의 구성은 다음과 같다. 2장에서 SAN환경을 위한 전역 공유 파일 시스템인 SANtopia의 전체 시스템 구조와 각 구성요소에 대하여 설명하고, 3장에서는 SAN에 연결된 물리적인 디스크를 모아서 하나의 논리적인 볼륨을 제공하는 볼륨 관리에

* 정회원

대하여 설명하고, 4장에서는 대용량 파일을 처리하기 위한 SANtopia 파일 시스템에 대하여 설명한다. 5장에서는 SANtopia의 성능 향상을 위한 전역 버퍼관리 기법과 일관성 유지하기 위한 전역 잠금 관리자의 기능과 역할에 대하여 설명하고, 6장에서 본 고의 결론을 맺는다.

2. SANtopia 시스템 구조

2.1 시스템 환경

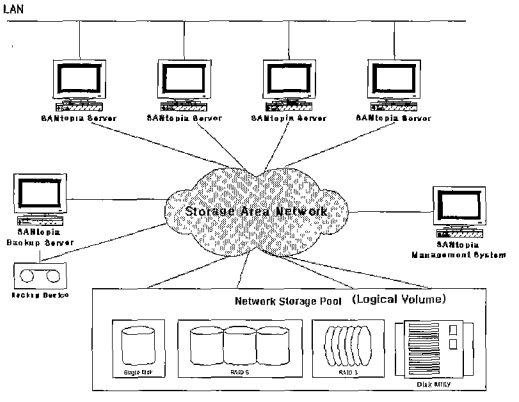


그림 1 SANtopia 시스템 환경

SANtopia 파일 시스템을 위한 시스템 환경은 그림 1과 같다. 디스크, JBOD, 그리고 RAID의 대용량 데이터를 저장하기 위한 디스크들을 모아서 저장 풀(storage pool)을 제공하는 논리 볼륨(logical volume), 기존의 LAN에 접속되어 네트워크를 통하여 사용자들이 요구하는 파일 시스템에 대한 연산을 제공하는 SANtopia 호스트 서버들, 그리고 호스트 서버와 저장 장치를 연결하는 SAN 망으로 구성된다. SANtopia 서버에는 파일 시스템 서버뿐만 아니라 시스템 환경을 유지 관리 하기 위한 시스템 관리 서버(system management server)와 server-less 백업을 지원하는 백업 서버(backup server)가 있다. 논리 볼륨에 저장된 파일에 접근하기 위해서 사용자는 LAN을 통해서 SANtopia 서버에 원하는 서비스를 요구하면 SANtopia 서버는 해당 파일의 위치를 파악하여 논리 볼륨으로부터 읽어들여 처리한다.

2.2 SANtopia 시스템 구조

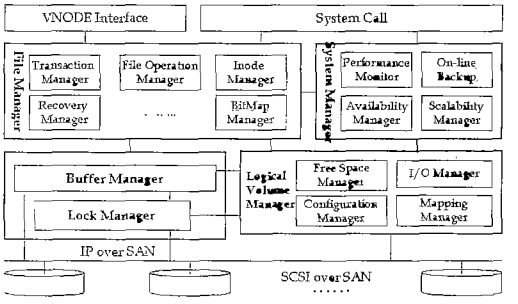


그림 2 SANtopia 시스템 구성도

SANtopia는 각 기능에 따라 볼륨 관리자, 버퍼 관리자, 잠금 관리자, 파일 관리자로 구성된다.

볼륨 관리자는 SAN에 연결된 모든 디스크를 하나의 관리 대상으로 볼 수 있도록 모은 다음 사용자가 정한 방식에 따라 이것을 분할하여 볼륨을 구성하고 상위레벨의 모듈에서 이 볼륨 위에 I/O를 실행할 수 있도록 해주는 모듈이다. 볼륨 관리자는 SAN에 연결되어 있는 독립적인 디스크 장치들을 모아 하나의 커다란 저장공간 풀을 형성하고 사용자의 요구에 따라 적절히 공간을 분할하여 볼륨을 형성하거나 공간을 회수 추가하여 볼륨을 변경 또는 삭제할 수 있도록 해주는 구성관리(configuration manager) 모듈, 구성 관리자에 의해 생성된 볼륨에 대해 상위 모듈에서 요구하는 입출력을 실제로 물리적 저장 공간인 디스크 장치 위에 수행하는 I/O 관리자, 물리적 디스크 장치들의 모임으로 구성된 저장 공간을 논리적인 저장 공간으로 가상화하여 사용자에게 동적인 구성환경을 제공하는 매핑 관리자(mapping manager), 그리고 볼륨 관리자가 관리하는 저장 공간의 사용 유무에 대한 정보를 유지하고 상위 모듈에서 저장 공간을 요구/철회시 이를 반영하도록 해주는 자유공간 관리자(free space manager)로 구성된다.

기존 리눅스 시스템들은 각 호스트들이 독립적으로 운용되는데 비해서 SANtopia의 호스트는 클러스터 형태로 운용된다. 클러스터 환경에서 시스템의 성능향상을 위해서는 버퍼의 통합관리가 필수적이다. 즉, 개별 호스트에 있는 버퍼들을 통합해서 전역 버퍼(Global Buffer)로 관리함으로써 버퍼의 자료 적중률(hit ratio)을 높여야 한다. 각 호스트에 산재해 있는 버퍼에 어떤 내용이 배치되어 있는지 관리하고 이를 바탕으로 호스트 간에 버퍼의 내용

을 주고 받음으로써 전체 클러스터의 디스크 접근 횟수를 줄이고 성능을 향상시키는 역할을 담당한다.

잠금 관리자는 다수의 호스트가 동시에 자료를 사용하기 때문에 자료의 일관성(consistency)을 유지하기 위한 잠금 기능을 지원한다. 잠금 관리자는 호스트 간의 자료에 대한 경쟁이 발생할 때 접근 순서를 결정해주는 역할을 담당함으로써 자료의 일관성을 보장함은 물론 데드락 및 기아(starvation) 상황이 발생하지 않도록 하는 기능을 담당한다.

파일 관리자는 대용량 파일을 위한 메타데이타 기능을 지원하고 있으며, SANtopia용 파일 시스템을 위한 슈퍼블록과 파일 및 디렉토리를 위한 inode 등의 메타데이타를 관리하고 사용자가 요구하는 파일 및 디렉토리 등에 대한 각종 연산기능을 제공한다. 또한 저널링(journaling) 기법[10]을 이용하여 파일 시스템의 고장으로부터 메타데이터를 안전하게 회복할 수 있는 기능을 지원한다.

3. 논리 볼륨

논리 볼륨이란, 물리적으로 독립적인 여러 개의 저장 공간을 모아 하나의 가상적인 저장 공간으로서 제공되는 논리적 저장 공간을 말한다. 파일 시스템, 데이터베이스 또는 일반적인 데이터 관리 시스템은 이 논리 볼륨 위에 생성되고 볼륨 관리자에 의해 제공되는 서비스를 이용하게 된다.

볼륨 관리자는 운영체제 커널 내에서 가상 디바이스 드라이버로서 동작하게 되며, 하나의 논리 볼륨은 하나의 가상 디바이스 파일로서 표현된다. 볼륨 관리자는 SAN에 연결된 모든 디스크를 하나의 관리 대상으로 볼 수 있도록 모은 다음 사용자가 정한 방식에 따라 이것을 분할하여 논리 볼륨을 구성하고 상위레벨의 모듈에서 이 볼륨 위에 I/O를 실행할 수 있도록 해주는 모듈이다.

3.1 Logical Volume Layout

볼륨 관리자는 독립적인 물리적 디스크 장치들을 한 곳으로 모아 저장 공간 풀(storage pool)을 형성하고 이들을 사용자의 요구에 따라 적절한 구성을 갖도록 배치하여 사용자가 원하는 용량만큼을 원하는 RAID level로 구성해주는 역할을 담당한다. 아래 그림 3은 8개의 물리적 디스크 장치로 이루어진 저장 공간 풀에서 가능한 볼륨 구성을 나타낸다.

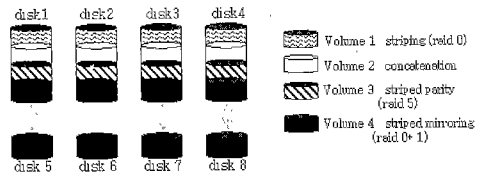


그림 3 디스크를 이용한 볼륨의 구성

논리 볼륨을 구성하는 가장 작은 저장장치 단위로서 볼륨에 할당, 해제되는 저장 장치 단위는 파티션이다. SANtopia에서는 일반적인 방식의 파티션 기법을 사용하여 2개의 파티션을 생성한다. 이러한 파티션을 물리 파티션(physical partition:PP)이라 부른다. 이 두 개의 파티션 중 하나는 private partition이고 나머지 하나는 public partition이다. 사용자에게 보여지는 파티션은 public partition이며, private partition은 사용자에게는 보여지지 않고 시스템 내부에서만 사용하게 된다. 실제 사용자가 인식하는 파티션은 public partition내에 가상적으로 생성하는 논리 파티션(logical partition:LP)이다. 이를 나타내는 그림이 그림 4에 보여지고 있다.

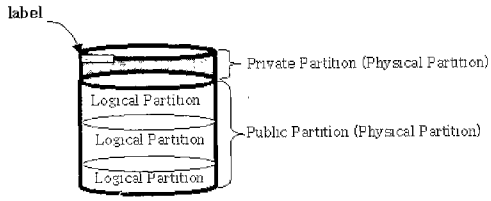


그림 4 디스크 파티션의 구성

Private partition은 해당 디스크 장치내의 논리 파티션(logical partition)에 대한 정보를 유지하고 있다. 볼륨 관리자는 이 정보를 통해 실제 논리 파티션이 어떻게 설정되었는지를 알고 해당 논리 파티션에 대한 동작을 수행하게 된다. 또한 논리 볼륨의 구성 정보도 이 영역에 저장되게 되며, 이 정보는 모든 디스크 장치의 private partition 영역에 중복되어 저장되게 된다. 반면, Public Partition내의 각 논리 파티션 내에는 해당 논리 파티션에 대한 할당 비트맵(allocation bitmap) 정보와 매핑 관리자(Mapping Manager)가 관리하는 매핑 정보가 저장되게 된다.

3.2 Software RAID의 제공

전통적인 디스크 관리 기법에서는 사용자에게 최대의 가용시간, 최적의 성능, 고용량 등의 요구사항에 만족스러운 해법을 제시하지 못했다. 이러한 문제점을 해결하고자 하드웨어 RAID 시스템이 개발되었다. 하드웨어 RAID 시스템은 여러 개의 독립적인 디스크 장치들을 모아서 RAID 컨트롤러의 제어하에 가용성, 성능, 용량적인 측면에서 사용자의 요구에 부응하도록 다양한 RAID 레벨을 제공하였다. RAID 레벨은 저장 장치의 사용 환경이나 사용자의 요구에 따라, 스트라이핑(striping), 미러링(mirroring), 패리티를 갖는 스트라이핑(striping with parity) 등으로 구분된다. 하드웨어 RAID 시스템은 이러한 RAID 레벨의 관리를 RAID 컨트롤러에서 행함으로써 CPU의 활용도를 더욱 높일 수 있고, 디스크 버스상에 하나의 명령만을 내리므로 버스 트래픽을 감소시킨다. 따라서 높은 성능을 제공할 수 있게 된다. 그러나, 하드웨어 RAID 시스템의 비용이 매우 고가이며, 컨트롤러 에러시 모든 RAID 시스템을 사용할 수 없게 되는 문제를 안고 있다.

SANtopia에서는 이러한 하드웨어 RAID 기능을 소프트웨어적으로 구현한 소프트웨어 RAID를 지원한다. 소프트웨어 RAID는 컨트롤러 대신 CPU가 RAID 레벨 관리를 수행하게 된다. 이것은 컨트롤러의 에러시 모든 것이 정지되는 문제를 해결하며, 매우 저렴한 가격의 일반 디스크 장치를 여러 개 모아 RAID 시스템을 구성할 수 있게 해준다. 소프트웨어 RAID 레벨은 하드웨어 RAID 레벨을 포함하며, 또한 일렬로 디스크 장치를 연결한 형태의 concatenation도 지원한다. 각 RAID 레벨의 구현은 다음에서 설명할 매핑 관리자에 의해 행해지게 된다. 또한, 하드웨어 RAID의 장점을 그대로 취하면서도 그 단점을 해결하기 위해서 하드웨어 RAID 상에 소프트웨어 RAID를 적용하는 방법도 적용이 가능하다.

3.3 매핑(Mapping)

SANtopia에서는 물리적 저장공간을 가상화 한(virtualized) 논리적 저장공간을 제공한다. 이렇게 가상화된 저장공간은 온라인 상에서의 동적인 구성환경을 제공하기에 적합하고 사용자에게 많은 융통성을 부여할 수 있다. 이러한 물리적 저장공간의 가상화는 매핑에 의해 이루어지게 된다.

물리적 저장공간의 가상화는 논리 주소를 해당 물리 주소로의 매핑에 있어서의 가변성을 수용하는 것이다. 즉, 이러한 매핑이 계산에 의해서 이루어지는 형식처럼 고정되어 있는 것이 아니고 어떠한 상황이 발생하여 매핑 관계가 변해야 할 때, 이를 수용할 수 있는 형태이어야 하는 것이다. 이를 위해 매핑 관리자는 매핑 정보를 담고 있는 매핑 테이블을 유지한다. 매핑 테이블의 구조는 다음과 같다.

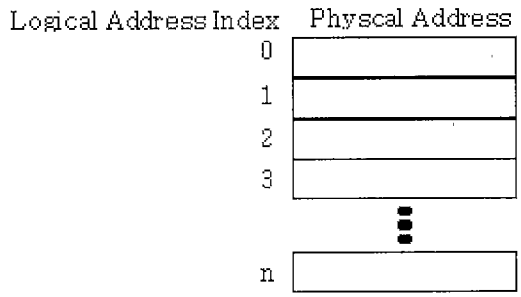


그림 5 논리볼륨의 매핑 테이블

그림 5에서 logical address index는 해당 논리 주소를 익스텐트 크기로 나눈 몫에 해당한다. 매핑 관리자는 이들 정보를 논리 볼륨을 구성하고 있는 각각의 논리 파티션의 헤더 영역에 분산 저장한다. 또한, 디스크 에러시 매핑 정보의 유실을 방지하기 위해 매핑 테이블 정보를 바로 다음 논리 파티션에 이중화 시키는 방법을 사용한다.

하나의 논리 볼륨에 대해 하나의 RAID 레벨이 정해지게 되며, 데이터 저장을 위해 새로운 저장공간을 요구 시 매핑 관리자는 해당 RAID 레벨에 부응하는 방식으로 물리 디스크 장치를 선택하게 된다. 선택된 장치내의 자유 공간이 할당되고 그곳에 데이터가 기록되게 된다.

대용량의 SAN 환경에서 백업을 효율적으로 처리하기 위해서는 스냅샷기능이 특별히 요구된다. 스냅샷 기능은 매핑 관리자에 의해 제공된다. 매핑 관리자는 스냅샷 요구 시, 현재의 매핑 테이블을 그대로 복제한다. 복제된 매핑 테이블은 백업이 완료될 때 까지 또는 다음 번 스냅샷이 요구될 때까지 유지된다. 사용자의 요구에 의해 데이터의 변경, 추가 혹은 삭제가 발생할 경우 해당 데이터의 디스

크 익스텐트는 새로운 공간을 할당 받아 복사된 후, 새로이 할당된 영역에서 변경이 이루어지게 되는 변경 시 복제(copy-on-write) 기법을 사용한다. 즉, 원래 매핑 테이블은 스냅샷시점의 매핑 정보를 그대로 유지한 채 관리되고 새로이 복제된 매핑 테이블이 사용자의 요구에 의해 변경된 매핑 정보를 유지하는 것이다. 이 상황에서 백업이 발생하면 백업은 원래의 매핑 정보를 이용하여 수행되게 됨으로써 스냅샷 시점과 일치하는 데이터의 백업을 취하게 된다.

3.4 Online Resizing

사용자가 이미 존재하는 논리 볼륨의 크기를 온라인 상에서 재조정하기를 원하는 경우가 있을 수 있다. 이때 볼륨 관리자는 논리 볼륨의 크기를 확장/축소할 수 있어야 한다. SANtopia에서 온라인 resizing을 위한 기본 단위는 논리 파티션이다. 볼륨에 논리 파티션의 추가/삭제는 해당 볼륨을 구성하는 데 참여하고 있는 모든 디스크 장치의 private partition 영역에 해당 정보를 갱신한다.

삭제 시는 해당 논리 파티션 내에 데이터가 없어야 한다. 만약 데이터가 존재한다면 매핑 관리자가 제공하는 data move 기능을 사용하여 해당 데이터를 다른 논리 파티션으로 이동시킨 후 삭제할 수 있다. 추가 시는 새로 추가된 파티션을 포함하여 전체 볼륨에 대해 RAID 구성이 재조정되어야 한다. 매핑 변환 함수 등의 사용에 의한 고정 방식의 매핑 기법에서는 이러한 재 조정 시 시스템의 일반적인 운영이 제한된다. 즉, 새로운 파티션을 포함하도록 데이터를 재조정하는 기간 동안에 일반적인 데이터 액세스 요구가 있는 경우 이 데이터가 재조정작업을 이미 마친 데이터인지 아직 재조정작업이 이루어지지 않은 데이터인지를 판별할 수 없는 것이다. 그러나, SANtopia에서는 매핑 테이블을 유지함으로써 모든 데이터의 이동 상황을 추적 유지하므로 데이터의 이동 중에도 일반적인 데이터 액세스에 대해 운영을 계속할 수 있게 된다.

4. 대용량 파일 시스템

SANtopia 파일 시스템은 64비트 어드레스를 지원하기 위한 파일 시스템으로 익스텐트(1KB~64KB)를 할당의 기본 단위로 사용하는 파일 시스템이다. 본 절에서는 파일 시스템의 구조와 대용량 데이타의 처리를 위한 메타데이타 구조에 대하여 설명한다.

4.1 파일 시스템 레이아웃

SANtopia 파일 시스템의 레이아웃은 그림 6과 같다. 파일 시스템은 기본적으로 64비트 주소 체계를 사용하여 0~2^{64}-1 까지의 주소 번지를 사용한다. 파일 시스템은 파일 시스템에 대한 정보를 저장하는 슈퍼블록, 각 익스텐트의 사용여부를 표시하여 익스텐트의 할당과 회수를 관리하기 위한 비트맵 블록, 그리고 데이터, inode, 디렉토리를 저장하기 위한 할당 블록(allocation blocks)으로 구성되어 있다.

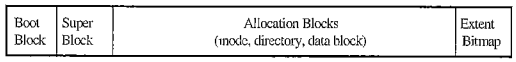


그림 6 SANtopia 파일 시스템 레이아웃

SANtopia 파일 시스템에서는 기존의 파일 시스템에서 고정적인 할당 방법과 영역 구분으로 인한 자원의 부족 현상을 데이터, inode, 디렉토리를 위한 영역을 구분하지 않고 요구되는 시점에 할당함으로써 해결한 것이다. 예를 들면 ext2와 같은 리눅스 파일 시스템에서는 4KB당 1개의 inode를 할당하므로 수많은 4KB이하의 파일이 존재하는 경우 inode의 자원이 부족하여 파일을 생성하지 못하는 문제가 발생한다. 그러나 SANtopia 파일 시스템은 각 자원을 할당 블록에서 할당 받아 사용하기 때문에 이러한 문제를 해결하였다.

4.2 파일 시스템의 생성

SANtopia 파일 시스템은 논리 볼륨관리자에 의해 제공되는 논리 볼륨을 이용하여 생성된다. 앞 절에서 설명한 바와 같이 각각의 디스크들을 묶어서 형성된 볼륨그룹을 이용하여 하나의 논리그룹을 생성하고 생성된 논리 그룹을 이용하여 그림 7과 같이 하나의 SANtopia 파일 시스템을 생성한다. 그림 7의 예에서는 두 개의 3개의 볼륨 그룹을 이용하여 2개의 논리 볼륨을 생성하였고, 각 논리 볼륨에 SANtopia 파일 시스템을 생성할 수 있음을 보여 준다.

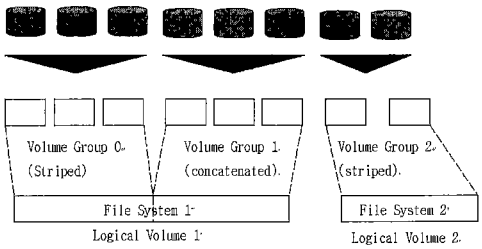


그림 7 논리 그룹을 이용한 SANtopia 파일시스템의 생성 예

4.3 대용량 파일 시스템을 위한 메타 데이터 구조

기존의 파일 시스템들에서 파일을 저장하고 관리하기 위한 inode나 디렉토리 구조는 수십 기가 단위의 소규모 파일 시스템을 위해서 개발이 되어 SAN에서 제공되는 대용량 자료저장 시스템에 적용하기 어렵다.

이러한 문제를 해결하기 위해 SANtopia에서는 대용량 파일 시스템을 위한 새로운 메타 데이터 구조로 대용량 파일을 저장하기 위한 동적 다단계 inode(dynamic multi-level inode)구조를 사용하고, 다수의 파일이 존재 가능한 디렉토리 관리 구조인 다단계 동적 확장 해쉬(multi-level dynamic extensible hash)를 사용한다.

4.3.1 대용량 파일을 위한 다단계 inode 구조

기존의 리눅스의 ext2 파일 시스템에서는 inode에서 데이터 블럭을 위한 포인터를 직접, 간접,이중, 삼중을 사용하는 고정된 방식의 포인터를 사용하므로 인해 데이터 블록이 1KB인 경우 생성할 수 있는 최대의 파일 크기는 약 2GB정도로 제한된다.

SANtopia에서는 익스텐트의 사용으로 인한 메모리의 낭비를 줄이기 위한 방법으로 inode를 초과하기 전까지 inode 내에 데이터를 저장하는 stuffed inode기법을 사용한다. 또한, 대용량 파일을 저장하기 위한 구조로 그림 8과 같이 동적 다단계 inode를 사용한다. 기존의 GFS가 모든 노드의 레벨이 동일함에 비해 SANtopia에서는 서로 다른 레벨을 가질 수 있으며, 각 레벨이 가득 차게 되면 한 레벨을 증가시켜 맨 위쪽에 이전의 레벨의 포인터를 복사한 후에 데이터 블록의 크기가 증가하는 경우에 하나씩 데이터 블록을 위한 포인터의 레벨을 증가시키는 기법을 사용한다. 그림 8의 예에서는 inode가 2레벨의 포인터를 가지고 있으며, n개의 데이터 블록을 위한 포인터를 사용하고 있다. 만약 한 개의 데이터 블록이 증가하여 n+1째 데이터 블록이 추가되면 루트 inode의 3번째 포인터로 연결을 시킨다.

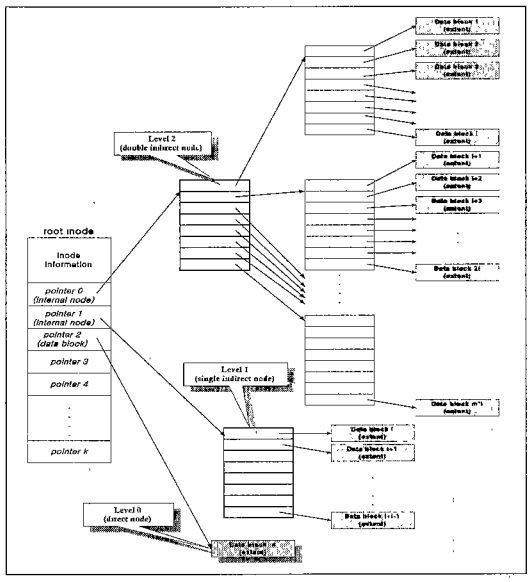


그림 8 동적 다단계 inode 구조

SANtopia에서 사용하는 동적 다단계 inode 구조는 대용량 파일을 효과적으로 관리하면서 GFS에서 레벨의 분기 시에 노드의 생성으로 인해 발생되는 성능의 감소와 빈 노드로 인한 메모리의 낭비를 줄이도록 하였다.

4.3.2 디렉토리 관리 기법

대용량 파일 시스템에서는 대용량 파일뿐만 아니라 한 디렉토리 내에 수 많은 파일들이 존재하므로 디렉토리 내에서 다수의 파일을 처리하기 위한 디렉토리 구조가 필요하다. 다수의 파일을 처리하기 위한 디렉토리 구조로서 xFS와 같이 B+ 트리를 사용하는 방법[11]과 GFS와 같이 확장 해쉬기법[1, 12]을 사용하는 방법을 들 수 있다.

SANtopia에서 사용하는 디렉토리 관리 기법은 확장 해쉬(extensible hash)기법을 변형한 방법을 사용한다. SANtopia에서는 다단계 동적 확장 해쉬 구조를 사용한다. GFS 등에서 사용하는 기존의 확장 기법과의 차이는 그림 9와 같이 루트 노드가 찬

경우에 해쉬 단계를 확장하여 사용한다. 이 기법을 사용하면 대량의 파일이 존재하는 환경에서 디렉토리 엔트리에 접근할 때 읽어야 하는 블록의 수를 감소시킬 수 있다.

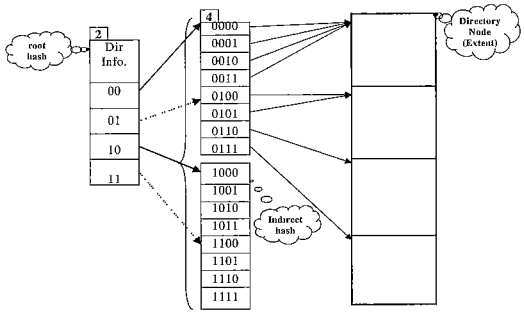


그림 9 다단계 동적 확장 해쉬 구조

5. 버퍼 및 잠금 관리

파일 시스템의 성능을 향상 시키기 위한 기법으로 SANtopia에서는 버퍼를 사용한다. SANtopia 버퍼 기법의 특징은 전역 버퍼(global buffer) 기능을 제공하는 것이다. 즉, SANtopia 호스트에서는 요청된 파일이 자신의 버퍼에 존재하지 않는 경우 디스크로부터 읽는 것이 아니라 다른 호스트의 버퍼에 존재하는 가를 먼저 검색하여 존재하는 경우에는 디스크에서 읽지않고, 해당 서버로부터 전송을 받는 방법을 사용하여 성능을 개선시킨다.

또한, 이 기법을 사용하기 위해서는 버퍼의 최신성을 유지하는 것이 가장 중요한 문제이다. 즉, 버퍼의 최신성을 유지하기 위해서는 빈번한 서버간의 통신이 발생하게 되어 시스템의 성능이 오히려 저하될 수 있다. SANtopia에서는 이러한 문제를 해결하기 위해서 버퍼 관리자와 잠금 관리자를 통합하였다. 분산 공유 파일 시스템에서 모든 파일에 대한 연산을 처리하기 위해서는 먼저 파일에 대한 잠금을 얻어야 한다. 파일에 대한 잠금을 얻기 위해서는 항상 전역 잠금 관리자에게 요청을 하여 처리함으로 SANtopia에서는 잠금을 요청하는 시점에 호스트 자신이 유지하고 있는 지역 버퍼 리스트를 전송하여 전역 버퍼 관리자가 전역 버퍼 리스트를 최신의 정보로 갱신할 수 있게 한다.

그림 10과 같이 SANtopia에는 호스트 자신의 정

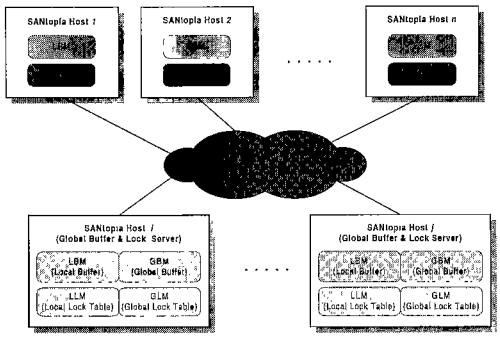


그림 10 버퍼 관리자와 잠금 관리자의 통합

보만을 관리하는 지역 호스트(local host)와 자신의 정보뿐만 아니라 전체 파일 시스템에 대한 정보를 관리하는 전역 호스트(global host)가 있다. 그림에서 나타나는 것처럼 전역 서버들은 잠금 서버와 버퍼 서버의 역할을 동시에 담당한다. 또한, 분산 서버 구조이기 때문에 모든 호스트에 LLM 및 LBM 모듈이 탑재되며 서버로 선정된 호스트에는 GLM과 GBM 모듈이 설치된다. GLM 모듈은 자신이 담당하고 있는 파일에 대한 전역 잠금 리스트를 관리하며 LLM은 해당 호스트가 현재 가지고 있는 잠금에 대한 개별 잠금 리스트를 관리한다. 버퍼 관리 기법에서는 관리 오버헤드를 줄이기 위해서 개별 버퍼 리스트의 내용과 전역 버퍼 리스트의 내용에 차이를 두는 방식을 택했지만 잠금 관리에서는 개별 잠금 리스트와 전역 잠금 리스트 간에 차이가 생기면 파일의 일관성이 깨지는 문제가 발생하기 때문에 개별 잠금 리스트와 전역 잠금 리스트는 항상 일치해야 한다. 단, 개별 잠금 리스트에서는 각 잠금을 어느 프로세스가 사용 중인가를 구체적으로 관리하는데 비해서 전역 잠금 리스트에서는 각각의 잠금을 어떤 호스트가 획득하고 있는 지만 관리하고 실제로 어떤 프로세스가 잠금을 사용중인가는 관여하지 않기 때문에 잠금 관련 메시지의 횟수를 줄일 수 있다.

전역 버퍼의 관리에 있어서 고려되어야 할 또 다른 문제는 버퍼 관리를 위하여 사용되는 메시지 전달 루트이다. 두 가지 방법이 고려될 수 있다. 첫째는 서버 및 클라이언트들이 연결된 네트워크를 이용하는 방법이며, 둘째는 SAN을 이용하여 메시지를 전달하는 방법이다. 전자는 기존 TCP/IP를 이용할 수 있다는 장점이 있는 반면, 성능 저하의 단

점이 있다. 반면 후자는 SAN의 고속성을 이용할 수 있지만 메시지 전달을 위한 추가적인 프로토콜을 개발해야 하는 단점이 있다. SANtopia에서는 성능 향상을 위하여 후자의 방법을 사용하여 전역 버퍼를 관리한다.

6. 결 론

SAN은 네트워크를 통해 컴퓨터 시스템과 저장 장치가 상호 밀접히 연결된 새로운 개념의 자료 저장 시스템으로 서버와 스토리지 간에 대용량 데이타를 고속으로 전송할 수 있어 향후 e-Business나 데이터웨어하우징과 같이 대용량 데이타의 처리나 백업이 요구되는 분야에 폭 넓게 사용될 것이다.

SAN환경에서 대용량 파일 시스템을 지원하기 위해서 개발중인 SANtopia는 익스텐트 기반의 64비트 파일 시스템으로 전역적인 파일 공유 기능을 지원하며, 저장장치의 클러스터 환경에서 고성능, 고 확장성, 공유성, 그리고 고 가용성을 제공하는 전역 공유 파일 시스템이다.

현재 SANtopia 시스템은 디스크들을 묶어서 하나의 대용량 저장장치 풀인 논리볼륨을 지원하기 위한 논리 볼륨 관리자가 구현되어 테스트 중이며, 대용량 파일의 처리와 공유기능을 지원하기 위한 파일 관리자 부분에 대한 개발이 진행 중이다.

참고문헌

[1] Kenneth W. et al., A 64-bit, Shared Disk File System for Linux, In The 7th NASA Goddard Conference on Mass Storage System and Technologies in cooperation with the 16th IEEE Symposium on Mass Storage Systems, pp.22-41, San Diego, USA, March 1999.

[2] Randy H. Katz, High-Performance Network and Channel Based Storage, Proceedings of IEEE, Vol.80, No.8, pp.1238-1261, 1992.

[3] Matthew T. OKeefe, Standard file systems and fibre channel, In The Sixth Goddard Conference on Mass Storage System and Technologies in cooperation with the Fifteen IEEE Symposium on Mass Storage Systems, pp.1-16, Colleage Park, Maryland, March 1998.

[4] Alan F. Benner. Fibre Channel: Gigabit Communications and I/O for Computer Network. McGraw-Hill, 1996.

[5] Uresh Vahalia, Unix Internals: The New Frontiers, Prentice-Hall, 1996.

[6] David Teigland. The Pool Driver: A Volume Driver for SANs. http://www.sistina.com.

[7] http://www.veritas.com

[8] http://www.tivoli.com/products/index/sanergy_file_sharing/

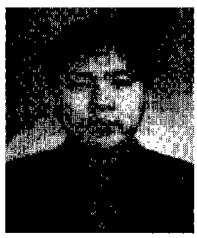
[9] http://www.snaux.com

[10] IBM Co., Journal File System for Linux, http://oss.software.ibm.com/developerworks/opensource/jfs/index.html

[11] Douglas Comer, The Ubiquitous B-Tree, Computing Survey, 11(2), pp.121-137, 1979.

[12] Michael J. Folk et al., File Structures, Addison-Wesley, March 1998.

김 경 배

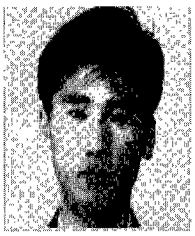


1992 인하대학교 전자계산학과(학사)
1994 인하대학교 전자계산학과(석사)
2000 인하대학교 전자계산학과(박사)
2000~현재 한국전자통신연구원 인터넷서비스연구부 선임연구원
관심분야:자료저장시스템, SAN, 이동컴퓨팅, 실시간데이터베이스 시스템
E-mail:gbkim@etri.re.kr

김 영 호

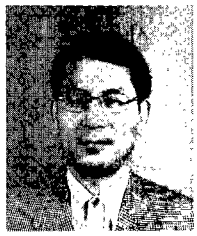
1999 충북대학교 정보통신공학과(학사)
2001 충북대학교 정보통신공학과(석사)
2001~현재 한국전자통신연구원 인터넷서비스연구부 연구원
관심분야:자료저장시스템, 클러스터링시스템, 내용기반 이미지 데이타베이스
E-mail:kyh05@etri.re.kr

김 창 수



1993 광운대학교 전자계산학과(학사)
1995 서강대학교 전자계산학과(석사)
1995~LG소프트(현재 LGEDS)
1999~현재 ETRI 컴퓨터 소프트웨어기술연구소 인터넷서비스연구부 연구원
관심분야:데이터베이스 시스템, 자료저장시스템, 클러스터 시스템
E-mail:cskim7@etri.re.kr

신 범 주



1983 경북대 전자공학과(학사)
1991 경북대 컴퓨터공학과(석사)
1998 경북대 컴퓨터공학과(박사)
1987~현재 한국전자통신연구원 근무(책임연구원, 자료저장시스템 S/W연구팀장)
관심분야:분산객체시스템, 이동 컴퓨팅 및 네트웍 저장 시스템 등
E-mail:bjshin@etri.re.kr

• JCCI 2001 학술대회 •

- 일 자 : 2001년 4월 25~27일
- 장 소 : 무주리조트
- 주 최 : 정보통신연구회
- 문 의 처 : 서울대학교 전기공학부 노종선 교수
  Tel. 02-880-1773
  E-mail : cc1.cnu.ac.kr/jcci/2001

• 제28회 임시총회 및 춘계학술발표회 •

- 일 자 : 2001년 4월 27일(금) ~ 28일(토)
- 장 소 : 경희대학교(수원캠퍼스)
- 논문모집 및 발표일정
  1) 심사결과 통보 : 2001년 3월 19일(월)
  2) 수정논문 접수마감 : 2001년 3월 31일(토)
  3) 논문발표 : 2001년 4월 27일(금), 4월 28일(토)
- 문 의 처 : 한국정보과학회 사무국
  Tel. 02-588-9246/7, 4001/2
  http://www.kiss.or.kr, E-mail:kiss@kiss.or.kr