네트워크 저장장치를 위한 클러스터 파일 시스템 개발

신 범 주*, 김 경 배**, 김 창 수***, 김 명 준****

● 목 차 ●

1. 서 론
2. 관련 연구
3. 시스템 환경
4. 볼륨 관리
5. 파일 시스템
6. 시스템 관리
7. 관련 응용
8. 정리 및 향후 계획

1. 서 론

클러스터(Cluster) 시스템은 독립적으로 동작할 수 있는 컴퓨터들을 네트워크로 연결하여 사용자 및 응용 프로그램에 단일 이미지를 제공하는 시스템을 의미한다[9]. 클러스터 시스템은 가격 대 성능비가 뛰어날 뿐 아니라 고 확장성(High Scalability), 고 가용성(High Availability)을 지원하기 때문에 다양한 컴퓨팅 분야에 활용되고 있으며, 특히 인터넷 컴퓨팅을 위한 기반 구조로 각광을 받고 있다. 그러나 이 같은 클러스터 시스템도 저장장치 측면에서 볼 때 특정 서버에 저장장치가 부착되어 사용되는 DAS(Direct Attached Storage) 구조를 가짐으로써 저장장치의 확장성, 가용성을 저하할 뿐 아니라 공유성을 제공하지 못하기 때문에 특정 서버의 성능 병목 현상을 초래할 수 있다. 이 같은 단점을 해결하기 위한 것이 네트워크 저장장치이다.

네트워크 저장장치는 클러스터 구조가 갖는 장점을 저장장치에 적용한 것이다. 즉 네트워크 저장장치는 고속의 네트워크로 저장장치들을 연결하여 여러 노드들이 공유할 수 있게 함으로써 클러스터 구조가 제공하는 고 확장성, 고 가용성, 고 성능(High Speed)과 같은 장점 뿐 아니라 자료의 공유가 가능케 하기 때문에 특정 서버의 병목 현상을 제거할 수 있게 한다. 이 같은 네트워크 저장장치를 SAN(Storage Area Network)이라 한다. 협의의 SAN은 파이버 채널(Fibre Channel)[1]로 대변되는 저장장치 전용 네트워크을 의미하나, 광의로는 네트워크로 연결된 저장장치 구조를 의미한다. 앞서 기술한 바와 같이 SAN은 DAS 구조의 저장장치가 안고 있는 여러 문제들을 효과적으로 해결할 수 있는 기반을 제공한다.

SAN 하드웨어 환경을 보다 효과적으로 활용할 수 있기 위해서는 사용자 또는 상위 계층의 응용에 하드웨어 투명성(Transparency)을 지원할 수 있는 일관된 기능의 인터페이스를 제공하는 것이 필요하다. SAN 가상화(SAN Virtualization)로 불리는 이러한 기능은 Virtual Memory의 개념과 유사하다. 소프트웨어로 제공되는 SAN 가상화는 시스템에 따라 다양한 기능들이 제공될 수 있으나, 일반적으로 소프트웨어 RAID, Snapshot, Storage on Demand 등

* 한국전자통신연구원 책임연구원
** 한국전자통신연구원 선임연구원
*** 한국전자통신연구원 연구원
**** 한국전자통신연구원 책임연구원

의 기능을 포함한다. 이 같은 SAN 가상화 계층을 기반으로 클러스터 파일 시스템이 제공된다. 클러스터 파일 시스템은 대용량 및 여러 호스트에서 파일을 공유할 수 있는 특성을 가지기 때문에 설계 시에 이 같은 특성들을 고려하여야 효율적인 시스템을 지원할 수 있다. 클러스터 파일 시스템에서 고려되는 사항들은 64비트용 파일 시스템, 파일 공유 시멘틱의 지원 및 이에 따른 효율적인 메타데이타(Metadata) 관리, 빠른 회복 기능, 대량의 파일들을 저장하기 위한 디렉토리 관리 기법 등이다.

상기에 기술한 SAN 가상화 기능 및 클러스터 파일 시스템은 운영체제에서 지원되어야 하는 기능들이다. 그러나 현재 널리 사용되는 대표적 운영체제인 유닉스, 리눅스 및 Windows 등은 서버에 직접 연결되어 있는 디스크들을 효과적으로 사용하고, 관리하기에 편리하도록 설계되었기 때문에 수백 개 이상의 저장 장치들을 네트워크로 연결하여 동시에 여러 호스트가 접속할 수 있는 기반을 제공하는 SAN 환경에 적합하지 않다. 따라서 SAN 가상화 기능 및 클러스터 파일 시스템을 제공하기 위해서는 기존 운영체제를 확장하는 것이 필요하다. 본고는 SAN을 기반으로 Linux 용 클러스터 파일 시스템을 제공하기 위하여 한국전자통신연구원에서 개발 중인 SANtopia 시스템에 대하여 소개한다. SANtopia 개발은 정보통신부 국책 연구개발 과제로 진행되고 있다[24].

본 고의 구성은 다음과 같다. 2장은 기존 연구들에 대해 기술하고 3장에서 SANtopia 시스템이 수행되는 환경을 살펴본다. 4장에서는 볼륨 관리를 기술하며, 5장에서 파일 시스템에 대하여 다룬다. 6장 및 7장에서 각각 시스템 관리 기능과 응용에 대해 언급하고, 끝으로 8장에서 결론 및 향후 연구 계획을 기술한다.

2. 관련 연구

본 연구와 관련된 기존의 많은 연구 결과들이 존재한다. 파일 시스템과 관련된 연구 결과로는 xFS [14], Frangipani[2], GFS[13] 그리고 SANergy[7] 등이 있으며, 볼륨 관리자로는 Frangipani의 Petal[5], 그리고 GFS의 볼륨 관리자인 Pool Driver를 들 수 있다. 본 장에서는 본 연구와 관련이 있는 중요 연구에 대해 기술한다.

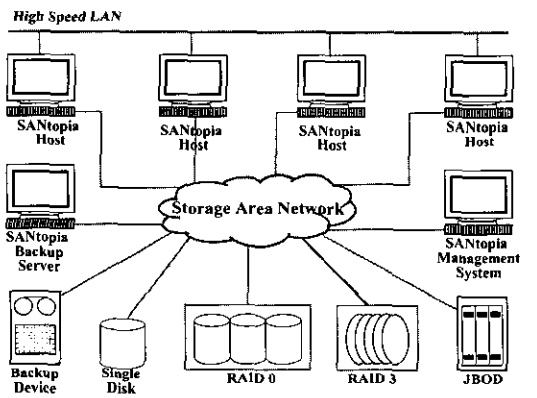
xFS와 Frangipani 파일 시스템은 기존의 서버 기반 분산 파일 시스템을 탈피한 분산 공유 파일 시스템을 제공하기 위한 연구 결과들이다. 이들 연구 결과들의 많은 부분들이 SAN 기반 클러스터 파일 시스템의 개발에 직간접의 영향을 미치고 있다. 그러나 이들은 물리 디스크가 하나의 서버에 연결되어 사용되는 전통적인 DAS 구조 상에서 동작하므로 SAN 환경에 적용하기 위해서는 부가적인 변형이 요구된다.

GFS와 GFS의 하부 볼륨 관리를 지원하는 Pool Driver는 SAN 환경에서 수행되는 시스템이다. Pool Driver는 레이드 레벨로서 striping과 mirroring 만을 제공하며, 계산 방식을 이용하여 볼륨의 논리 주소를 물리 디스크의 주소로 매핑하는 방식을 사용하기 때문에 스냅샷(Snapshot)이나 데이타의 동적 이동과 같은 기능을 지원하지 못하고 있다. GFS는 파일 캐쉬를 지원하지 않기 때문에 파일을 공유할 경우 성능이 저하되며, 전역 잠금을 위하여 저장장치 기반 잠금을 사용하기 때문에 상대적으로 유연성이 떨어지는 단점을 가진다.

SANergy는 기존의 서버 중심 분산 파일 시스템인 NFS 및 CIFS와 결합하여 LAN이 아닌 SAN 상에서 데이터 공유를 지원하며, 이 기종 운영체제에서도 수행될 수 있는 장점을 제공한다. 그러나 중앙 집중된 논리 볼륨 관리자에 의존하는 단점이 있으며, NFS가 가지는 단점을 탈피하지 못하기 때문에 진정한 분산 공유 파일 시스템으로 보기 어렵다.

3. 시스템 환경

SANtopia가 동작하는 시스템 환경은 그림 1과 같다. 그림에 나타낸 바와 같이 SANtopia는 디스크, JBOD, 그리고 RAID 및 백업 장비와 이를 이용하기 위한 호스트들이 화이버 채널로 연결된 SAN 환경에서 각 호스트에서 수행되는 Linux 운영체제에 포함된 시스템 소프트웨어이다. SANtopia는 이 같은 SAN 환경에서 대용량 데이타를 저장할 수 있도록 물리 디스크들을 모아서 저장 풀(storage pool)을 제공하는 논리 볼륨(logical volume)을 지원하고, SAN에 접속된 호스트들 사이에 공유 가능한 대용량 파일 시스템을 지원한다.



(그림 1) SANtopia 시스템 환경

여러 호스트들 사이에 공유 가능한 파일 시스템을 제공하기 위해선 호스트들 사이에 통신이 필수적이다. 호스트들 사이에 통신하기 위하여 사용될 수 있는 경로로 두 가지가 가능하다. 첫째는 통신을 위한 경로로 SAN을 이용하는 것이며, 둘째는 서버들을 연결하는 LAN을 이용하는 것이다. 초기 설계 단계에서 전자의 방법을 사용하는 것을 고려하였으나[17], SAN이 블록 데이타의 전송에 적합하게 만들어져 있을 뿐 아니라 추가적인 프로토콜을 개발하여야 하며, 블록 데이타 전송을 방해할 수 있기 때문에 장점보다는 단점이 많은 것으로 판단되어 제외하였다. 따라서 후자의 방법을 사용한다. 후자의 방법은 TCP/IP를 이용할 수 있다는 장점이 있는 반면, LAN의 속도가 성능 저하의 원인이 될 수 있기 때문에 SANtopia는 Gigabit Ethernet 스위치로 연결된 환경에서 운영되는 것을 가정한다.

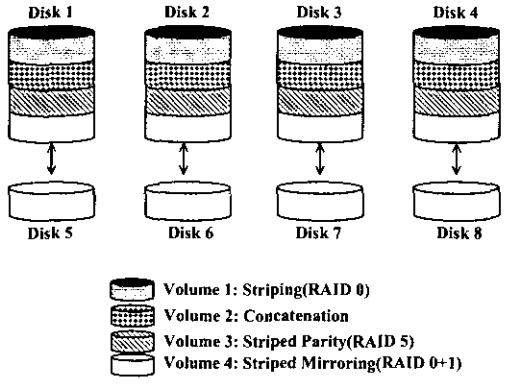
4. 볼륨 관리

볼륨 관리의 기능은 물리적으로 독립적인 여러 개의 저장 공간을 모아 하나의 가상적 기능을 제공하는 논리 저장 공간을 상위 계층에 제공하는 것이다. 즉 SAN 가상화 기능들을 제공하여 물리 볼륨이 제공치 못하는 기능들을 제공한다. 파일 시스템, 데이타베이스 또는 일반적인 데이타 관리 시스템은 이 논리 볼륨 위에 생성되고 볼륨 관리자에 의해 제공되는 서비스를 이용하게 된다. 볼륨 관리자는 운영체제 커널의 가상 디바이스 드라이버로 동작하게 되며, 하나의 논리 볼륨은 하나의 가상 디바이스 파일로 표현된다. 볼륨 관리자는 SAN에 연결된 물리 저장 장치들을 인식하고 시스템 관리자가 정한 방식에 따라 물리 저장장치를 분할하여 논리 볼륨을 구성하고 관리한다. 즉 상위 계층의 모듈이 요청하는 I/O를 논리 볼륨의 특성에 따라 적절한 물리 볼륨 I/O를 실행할 수 있도록 해주는 모듈이다.

4.1 볼륨 구성

볼륨 관리자의 역할 중 하나는 논리 볼륨을 구성하는 것이다. 독립적인 물리적 디스크 장치들을 한 곳으로 모아 저장 공간 풀(storage pool)을 형성하고 이들을 사용자의 요구에 따라 적절한 구성을 갖도록 배치하여 사용자가 원하는 용량 만큼을 원하는 RAID level로 구성해 주는 역할을 수행한다. 논리 볼륨은 디스크 파티션을 최소 구성 단위로 하는 연속적인 주소 공간을 이루는 확장 가능한 디스

크 파티션들의 집합이다. 논리 볼륨의 크기는 시스템 운영 중 언제든 변경 가능하며, 디스크 파티션 단위로 변경이 행해지게 된다.



(그림 2) 디스크를 이용한 볼륨의 구성

논리 볼륨의 구성 시에 고려되어야 하는 요소 중 하나는 익스텐트(extent)의 크기를 결정하는 것이다. 익스텐트는 동일 크기를 갖는 연속적인 공간이다. 또한, 정보의 저장을 위해 할당될 수 있는 디스크 공간의 최소 단위이다. 익스텐트의 크기는 하나의 논리 볼륨에 대해 동일하다. 그러나 서로 다른 논리 볼륨은 서로 다른 크기의 익스텐트를 가질 수 있으며 그 크기는 논리 볼륨 생성시에 결정된다.

논리 볼륨 구성에 대한 정보 및 해당 논리 볼륨을 관리하기 위한 메타데이타는 디스크 파티션의 헤더 영역에 기록된다. 메타데이타에는 매핑 정보, 물리 디스크 영역의 할당 맵 등이 포함된다.

그림 2는 8개의 물리적 디스크 장치로 이루어진 저장 공간 풀에서 가능한 볼륨 구성을 나타낸다. 그림은 볼륨 1이 RAID 레벨 0를 지원하는 반면 볼륨 4는 RAID 레벨 0+1을 지원하도록 구성되었음을 보여준다.

4.2 Software RAID

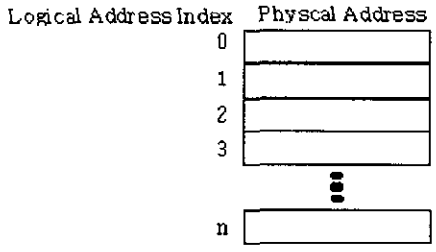
전통적인 디스크 관리 기법에서는 사용자에게 최대의 가용시간, 최적의 성능, 고용량 등의 요구사항에 만족스러운 해법을 제시하지 못했다. 이러한 문제점을 해결하고자 하드웨어 RAID 시스템이 개발되었다. 하드웨어 RAID 시스템은 여러 개의 독립적인 디스크 장치들을 모아서 RAID 컨트롤러의 제어 하에 가용성, 성능, 용량적인 측면에서 사용자의 요구에 부응하도록 다양한 RAID 레벨을 제공한다. RAID 레벨은 저장 장치의 사용 환경이나 사용자의 요구에 따라, 스트라이핑(striping), 미러링(mirroring), 패리티를 갖는 스트라이핑(striping with parity) 등으로 구분된다. 하드웨어 RAID 시스템은 RAID의 관리를 RAID 컨트롤러가 수행하기 때문에 CPU의 활용도를 높일 뿐 아니라 버스 트래픽을 감소시킴으로써 높은 성능을 제공한다. 그러나 하드웨어 RAID 시스템의 비용이 매우 고가이며, 컨트롤러에 오류가 발생할 경우 모든 RAID 시스템을 사용할 수 없게 되는 문제를 안고 있다.

SANtopia에서는 이러한 하드웨어 RAID 기능을 소프트웨어적으로 구현한 소프트웨어 RAID를 지원한다. 소프트웨어 RAID는 하드웨어 컨트롤러 대신 CPU가 RAID의 관리를 수행하게 된다. 이것은 컨트롤러의 오류 발생 시 모든 것이 정지되는 문제를 해결하며, 상대적으로 저렴한 가격의 RAID 시스템을 구성할 수 있게 해준다. SANtopia의 소프트웨어 RAID는 하드웨어 RAID 레벨을 제공할 뿐 아니라 추가로 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 가능하다.

4.3 매핑 (Mapping)

볼륨 관리자는 물리적 저장공간을 다양한 기능들로 가상화된(virtualized) 논리적 저장공간을 제공한다. 이렇게 가상화된 저장공간은 물리적 저장장

치가 제공하는 한계를 극복할 수 있게 할 뿐 아니라 온라인 상에서 동적인 구성 환경을 지원할 수 있게 함으로써 사용자에게 많은 융통성을 부여할 수 있게 한다. 이러한 물리적 저장공간에 대한 가상화는 논리 저장공간과 물리 저장공간과 매핑 과정에 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 SANtopia 맵핑 관리자는 매핑 정보를 담고 있는 매핑 테이블을 유지한다. 매핑 테이블의 구조는 그림 3과 같다.



(그림 3) 논리볼륨의 매핑 테이블

그림 3에서 logical address index는 해당 논리주소를 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, 데이타 저장을 위해 새로운 저장공간 요구 시 매핑 관리자는 해당 RAID 레벨에 부응하는 방식으로 물리 디스크 장치를 선택하게 된다. 선택된 장치내의 자유 공간이 할당되고 그곳에 데이타가 기록되게 된다.

대용량의 SAN 환경에서 백업을 효율적으로 처리하기 위해서는 스냅샷(snapshot) 기능이 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 때 까지 유지된다. 사용자의 요구에 의해 데이타의 변경, 추가 혹은 삭제가 발생할 경우 해당 데이타의 디스크 익스텐트는 새로운 공간을 할당받아 복사된 후, 새로이 할당된 영역에서 변경이 이루어지게 되는 변경 시 복제(copy-on-write) 기법을 사용한다. 즉, 원래 매핑 테이블은 스냅샷 시점의 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 데이타의 백업을 취하게 된다.

4.4 Online Resizing

이미 존재하는 논리 볼륨의 크기를 재조정해야 할 상황이 발생할 경우가 있을 수 있다. 사용 중인 논리 볼륨의 크기를 확장 또는 축소할 수 있는 기능을 online resizing이라 한다. SANtopia의 볼륨 관리자는 논리 파티션을 기본 단위로 하는 online resizing을 지원한다. 특정 볼륨에 논리 파티션의 추가 및 삭제는 해당 볼륨을 구성하는 모든 물리 디스크 장치의 특정 영역에 해당 정보의 갱신을 야기하며, 데이타의 이동을 동반할 수 있다.

볼륨을 축소하는 경우 삭제될 논리 파티션 내에 데이타가 없어야 한다. 만약 데이타가 존재한다면 매핑 관리자가 제공하는 데이타 이동 기능을 사용하여 해당 데이타를 다른 논리 파티션으로 이동시킨 후 삭제할 수 있다. 추가 시는 새로 추가된 파티션을 포함하여 전체 볼륨에 대해 RAID 구성이 재

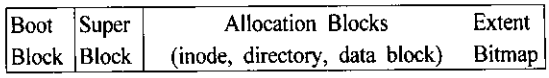
조정되어야 한다. 매핑 변환 함수 등의 사용에 의한 고정 방식의 매핑 기법에서는 이러한 재조정 시 시스템의 일반적인 운영이 제한된다. 즉, 새로운 파티션을 포함하도록 데이타를 재조정하는 기간 동안에 일반적인 데이타 액세스 요구가 있는 경우, 이 데이타가 재조정 작업을 이미 마친 데이타인지 아직 재조정 작업이 이루어지지 않은 데이타인지를 판별할 수 없는 것이다. 그러나, SANtopia에서는 매핑 테이블을 유지함으로써 모든 데이타의 이동 상황을 추적 유지하므로 데이타의 이동 중에도 일반적인 데이타 액세스를 지속할 수 있게 한다.

5. 파일 시스템

SANtopia 파일 시스템은 SAN 환경을 기반으로 하는 클러스터 파일 시스템이며, 64비트 어드레스를 지원하는 시스템으로 익스텐트(1KB~64KB)를 할당의 기본 단위로 사용한다. 본 파일 시스템은 앞 장에서 기술한 논리 볼륨 관리자를 하부 계층으로 사용하며, 클라이언트/서버 모델의 공유 파일 시스템과 구별되는 대등 관계 모델의 공유 파일 시스템이다. 즉 SANtopia 호스트들은 동일한 권한과 기능으로 파일을 공유한다. 본 절에서는 SANtopia 파일 시스템의 구조와 대용량 데이타 처리를 위한 메타데이타 구조에 대하여 설명한다.

5.1 파일 시스템 레이아웃

SANtopia 파일 시스템의 레이아웃은 그림 4와 같다. 파일 시스템은 기본적으로 64비트 주소 체계를 사용하여 0~264 -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 데이타, inode, 디렉토리를 저장하기 위한 할당 블록(allocation blocks)으로 구성되어 있다.

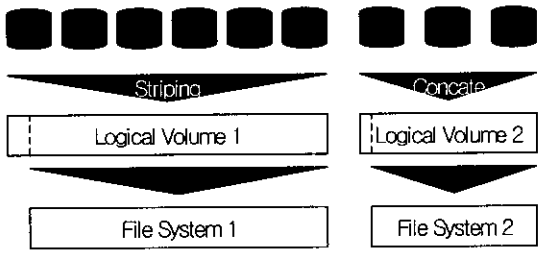


(그림 4) SANtopia 파일 시스템 레이아웃

SANtopia 파일 시스템에서는 기존의 파일 시스템에서 고정적인 할당 방법과 영역 구분으로 인한 자원의 부족 현상을 데이타, inode, 디렉토리를 위한 영역을 구분하지 않고 요구되는 시점에 할당함으로써 해결한다. 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 해결한다.

5.2 파일 시스템의 생성

SANtopia 파일 시스템은 논리 볼륨 관리자에 의해 제공되는 논리 볼륨 위에 생성된다. 파일 시스템의 생성은 기반이 되는 논리 볼륨 또는 디스크 파티션에 파일 시스템 레이아웃에 따라 적절한 초기화를 수행하게 한다. 이러한 초기화는 파일 시스템에 대한 이후의 정상적인 연산을 위해 필요하다. 그림 의 예에서는 6 개의 물리 디스크를 바탕으로 스트라이핑 형식으로 구성된 논리 볼륨 1 위에 파일 시스템 1이 생성되고, 3 개의 물리 디스크를 연결형 형식으로 구성한 논리 볼륨 2위에 파일 시스템 2가 생성될 수 있음을 보여준다.



(그림 5) 논리 볼륨을 이용한 파일 시스템의 생성 예

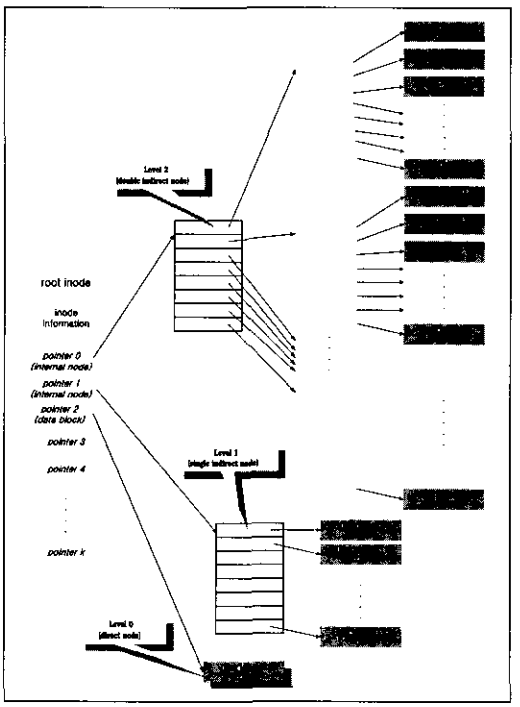
5.3 메타데이타 관리

기존의 파일 시스템들에서 파일을 저장하고 관리하기 위한 inode나 디렉토리 구조는 수십 기가 단위의 소규모 파일 시스템을 위해서 개발이 되었기 때문에 SAN에서 제공되는 대용량 자료저장 시스템에서 효율적으로 동작하기 어렵다. 이러한 문제를 해결하기 위해 SANtopia에서는 대용량 파일 시스템을 위한 새로운 메타데이타 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.

5.3.1 다단계 inode 구조

기존 리눅스의 ext2 파일 시스템에서는 inode에서 데이타 블럭을 위한 포인터를 직접, 간접,이중, 삼중을 사용하는 고정된 방식의 포인터를 사용하므로 인해 데이타 블록이 1KB인 경우 생성할 수 있는 최대의 파일 크기는 약 16GB정도로 제한된다.

SANtopia에서는 익스텐트의 사용으로 인한 메모리의 낭비를 줄이기 위한 방법으로 inode를 초과하기 전까지 inode 내에 데이타를 저장하는 stuffed inode기법을 사용한다. 또한, 대용량 파일을 저장하기 위한 구조로 그림 6과 같이 동적 다단계 inode를 사용한다. GFS에서는 모든 노드의 레벨이 동일하게 유지하는 것에 비해 SANtopia에서는 서로 다른 레벨을 가질 수 있게 한다. 각 레벨이 가득 차게 되면 한 레벨 증가시켜 맨 위쪽에 이전의 레벨의 포인터를 복사한 후 데이타 블록의 크기가 증가하는 경우에 하나씩 데이타 블록을 위한 포인터의 레벨을 증가시키는 기법을 사용한다. 그림 6의 예에서는 inode가 2 레벨의 포인터를 가지고 있으며, n개의 데이타 블록을 위한 포인터를 사용하고 있다. 만약 한 개의 데이타 블록이 증가하여 n+1 번째 데이타 블록이 추가되면 루트 inode의 3 번째 포인터



(그림 6) 동적 다단계 inode 구조

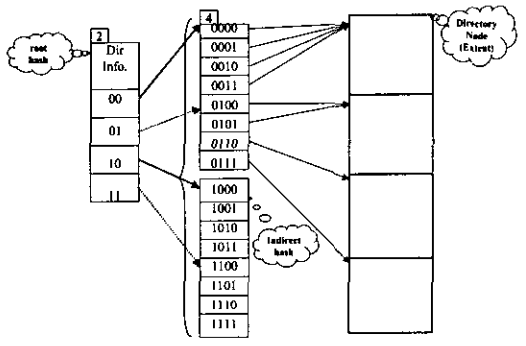
로 연결을 시킨다.

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.

5.3.2 디렉토리 관리

대용량 파일 시스템에서는 대용량 파일 뿐만 아니라 한 디렉토리 내에 수 많은 파일들이 존재할 수 있으므로, 이 같은 상황을 효율적으로 처리하기 위한 디렉토리 구조가 필요하다. 다수의 파일을 처리하기 위한 디렉토리 구조로서 xFS와 같이 B+ 트리를 사용하는 방법[14]과 GFS와 같이 확장 해쉬 기법[13]을 사용하는 방법을 들 수 있다. SANtopia는 디렉토리 관리를 위한 방법으로 확장 해쉬(extensible hash) 기법을 변형한 다단계 동적 확장 해쉬 구조를 사용한다[16]. GFS 등에서 사용하는

기존의 확장 기법과의 차이는 그림 7과 같이 루트 노드가 가득 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 7) 다단계 동적 확장 해쉬 구조

5.4 저널링(Journaling)

기존의 파일 시스템에서는 시스템에 문제가 발생되는 경우 fsck(파일 시스템 체크)를 이용하여 파일 시스템에 발생한 비일관성을 해결한다. 그러나, fsck는 파일 시스템의 크기에 비례하여 많은 시간이 소요되고, 시스템이 오프라인 상태로 수행해야 한다는 문제점으로 인해 SAN과 같은 대용량 클러스터 파일 시스템에서는 사용할 수 없다. 따라서 SANtopia 파일 시스템은 기존의 fsck의 단점을 해결하기 위해 파일 시스템의 고장 시 복구시간을 줄이고, 가용성을 증가시키기 위하여 메타데이타 저널링을 사용한다.

저널링 기법은 파일 시스템에 대한 연산이 발생하는 경우 비정상적인 파일 시스템의 오류에 대비하여 메타데이타에 대한 변경 정보를 로그에 기록하고, 파일 시스템에 오류가 발생한 경우에 로그에 기록된 메타데이타를 이용하여 복구하는 방법이다.

SANtopia에서는 저널링 기법을 사용하기 위해 파일 시스템의 메타데이타를 변경하는 모든 연산을 트랜잭션으로 정의하고, 각 트랜잭션에서 변경된 내용을 먼저 디스크에 있는 로그에 기록한 후 해당 트랜잭션을 완료 시키는 기법을 사용한다. 이러한 기법을 사용하기 위해 SANtopia 저널링 모듈은 파일 시스템에 대한 연산을 관리하기 위한 트랜잭션 관리자, 메타데이타에 대한 로깅과 로그 관리를 담당하는 로그 관리자, 그리고 시스템이나 연산에 대한 이상이 발생한 경우 회복기능을 수행하는 회복 관리자로 구성된다.

SANtopia 파일 시스템에 저장된 파일에 대한 연산이 요구되면 파일 관리자는 파일에 대한 연산을 수행하기 전에 트랜잭션 관리자에게 해당 트랜잭션의 수행 정보를 알린다. 트랜잭션 관리자는 통보된 정보를 이용하여 수행 중인 트랜잭션의 정보 리스트를 유지하고 이를 관리한다. 트랜잭션의 연산 중에서 메타데이타에 대한 변경 연산이 발생하는 경우에 로그 관리자를 통해서 변경된 메타데이타에 대한 내용을 먼저 로그와 트랜잭션 관리 정보에 기록한 후에 트랜잭션을 수행한다. 만약 파일 연산에 대한 트랜잭션이 실패하거나 파일 시스템에 이상이 발생하면, 로그 디스크에 저장되어 있는 메타데이타에 대한 로그를 이용하여 실패한 트랜잭션이 변경한 메타데이타에 대한 내용을 실행 이전의 상태로 복구하는 회복 작업을 수행하여 파일 시스템에 대한 일관성을 유지한다.

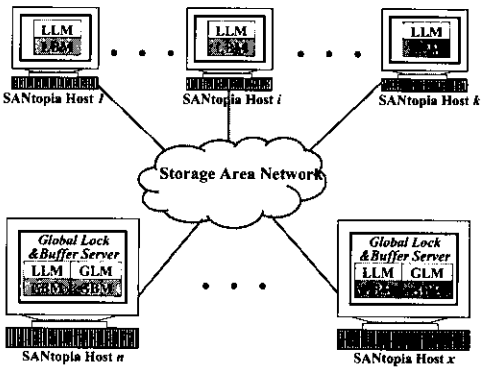
5.5 파일 캐쉬

파일 시스템의 성능을 향상 시키기 위하여 SANtopia에서는 상호 협력 캐쉬(cooperative cache) 기법을 사용한다. 즉, 요청된 파일의 내용이 자신의 버퍼에 존재하지 않는 경우 디스크로부터 읽는 것이 아니라 다른 호스트의 버퍼에 존재하는 가를 먼저 검색하여 존재하는 경우에는 디스크에서 읽지 않고, 해당 호스트로부터 전송을 받는 방법을 사용한다.

이 기법을 사용하기 위해서 각 호스트의 파일 캐쉬에 대한 정보를 취합한 전역 버퍼 리스트(global

buffer list)를 관리한다. 전역 버퍼는 특정 호스트에 의해 관리되며, 부하를 분할하기 위하여 여러 호스트에 나누어 관리한다. 그러나 로컬 버퍼가 변경될 때 마다 전역 버퍼 리스트를 갱신하는 경우 호스트 간의 통신을 빈번하게 발생시키기 때문에 오히려 시스템의 성능을 저하시킬 수 있다. SANtopia에서는 이러한 문제를 해결하기 위해서 버퍼 관리자와 잠금 관리자를 통합한다. 분산 공유 파일 시스템에서 모든 파일에 대한 연산을 처리하기 위해서는 먼저 파일에 대한 잠금을 얻어야 한다. 버퍼 관리자와 잠금 관리자가 항상 동일 호스트에 위치하기 때문에 잠금을 요청하는 시점에 호스트 자신이 유지하고 있는 지역 버퍼 리스트를 전송하여 전역 버퍼 관리자가 전역 버퍼 리스트를 최신의 정보로 갱신할 수 있게 한다. 이 방법은 지역 버퍼가 변경될 때마다 통신이 발생하는 것을 방지할 수 있게 한다.

SANtopia에서는 전역 잠금 관리를 위하여 콜백 잠금(Callback Locking) 기법을 사용한다. 잠금의 할당이 특정 서버에 의해 관리되며, 특정 호스트가 잠금을 획득한 후 사용이 끝 나더라도 반환하지 않으며, 다른 호스트가 동일 잠금을 요청할 때 까지 계속 사용할 수 있게 한다. 해당 잠금에 대한 새로운 요청이 도달하였을 때 잠금 서버는 현재 잠금을 가진 호스트에 잠금에 대한 콜백을 요청하며, 사용이 완료된 경우 한해 새로운 호스트에 잠금을 할당한다. 이 방법은 두 가지 특징을 갖는다. 첫째는 잠금이 특정 호스트에 의해 무제한 사용될 수 있기 때문에 Starvation이 발생할 수 있다. 둘째는 순수 서버 방식에 비해 잠금의 획득과 해제에 소요되는 통신 횟수를 줄일 가능성이 높으나, 서버 기반 방식이 가지는 오류 취약성을 가진다. 첫 번째 문제를 해결하기 위하여 SANtopia 잠금은 예약 시멘틱을 제공한다. 예약이 될 경우 잠금을 획득한 호스트는 잠금을 지속적으로 사용할 수 없다. 두 번째 문제를 해결하기 위하여 잠금 서버를 중복(Replication)시키는 것도 가능한 방법이 될 수 있다. 그러나 SANtopia 1차 버전에서는 잠금 관리자의 오류에 대해 고려하지 않는다. 오류 발생시에 잠금 관리자를 복구할 수 있는 방법은 향후에 고려할 것이다.



(그림 8) 버퍼 관리자와 잠금 관리자의 통합

그림 8과 같이 SANtopia에는 전체 파일 시스템에 대한 정보를 관리하는 전역 호스트(global host)가 있다. 각 호스트는 전역 호스트의 역할을 담당할 수 있으며, 전역 호스트는 시스템 관리자에 의해 결정된다. 전역 호스트들은 잠금 서버와 버퍼 서버의 역할을 동시에 담당한다. 또한, 분산 서버 구조이기 때문에 모든 호스트에 LLM 및 LBM 모듈이 탑재되며 전역 호스트로 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. 전역 버퍼 관리에서는 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 구체적으

로 관리하는데 비해서 전역 잠금 리스트에서는 각각의 잠금을 어떤 호스트가 획득하고 있는 지만 관리하고 실제로 어떤 프로세스가 잠금을 사용 중인가는 관여하지 않기 때문에 잠금 관련 메시지의 횟수를 줄일 수 있다.

6. 시스템 관리

클러스터 파일 시스템이 수행되는 환경은 많은 구성 요소로 이루어진다. 디스크, 백업 장치, SAN 스위치 그리고 네트워크로 연결된 호스트들이 서로 다른 지역에 위치하며, 복잡하게 얽혀 있게 된다. 이 같은 환경을 효율적으로 관리하기 위해서는 중앙집중의 강력한 관리 기능이 요구된다.[21] 그러나 시스템 관리를 위해 제공되는 기능들은 매우 다양하며, SAN 환경을 구성하는 요소들에 따라 관리되는 기능들이 달라 질 수 있기 때문에 모든 관리 기능을 지원하는 것은 쉬운 일이 아니다.

시스템 관리를 위하여 SANtopia에서 지원하는 기능은 크게 세 가지이다. 첫째는 성능 모니터링이며, 둘째가 오류 탐지 기능이며, 마지막으로 GUI 기반 구성 관리 기능이다. 성능 모니터링에서는 파일 관리자, 볼륨 관리자의 성능 및 I/O의 빈도 수가 주기적으로 기록된다. 오류 탐지 기능은 특정 이벤트를 모니터링하고 특이한 상황에 대한 경고를 발생한다. 구성 관리자는 시스템 전체 구성을 GUI 베이스로 초기 시스템을 구성하고 성능 모니터링 및 오류 탐지 기능의 결과에 따라 시스템 구성을 변경할 수 있게 한다. 또한 파일 단위 및 볼륨 단위의 백업을 위한 응용도 함께 지원된다.

7. 관련 응용

SAN환경에서는 최근 폭발적으로 증가하고 있는 이미지, 동영상, 캐드 데이타와 같은 고차원 특징을 갖는 데이타들이 대량으로 저장되어 관리 될 것이다. SAN 환경에 저장된 이러한 고차원 특징을 갖는 데이타들을 효율적으로 검색할 수 있는 적절한 고차원 색인구조가 필요하다. SAN 환경의 특징은 저장 장치를 공유하는 형태(Shared Disk)의 병렬 환경이라 볼 수 있다. 병렬성을 효과적으로 이용하는 고차원 색인 구조는 높은 성능을 발휘한다. SANtopia 환경을 호스트의 병렬성과 I/O의 병렬성을 최대화하는 방법으로 모델화하여 검색 성능을 향상시킬 수 있는 기법을 개발하여 제공한다.[4][23]

또한 현재 단일 서버에서 동작하는 공간 데이타베이스 관리 시스템을 SANtopia 볼륨 관리자를 기반으로 동작할 수 있도록 이식함으로써 SAN 환경에 최적화된 공간 데이타베이스 관리 시스템을 제공할 계획이다. 이 같은 공간 데이타베이스 관리 시스템이 제공될 경우 지리정보시스템과 같은 실질적이고 다양한 응용들이 SANtopia 환경에서 수행될 수 있을 것이다.

8. 정리 및 향후 계획

본 고에서는 한국전자통신연구원에서 개발 중인 SAN 기반 클러스터 파일 시스템 SANtopia를 소개하였다. SANtopia는 SAN 가상화를 지원하는 익스탠트 기반의 64비트 파일 시스템으로 전역 파일 공유 기능을 지원하며, 저장장치 클러스터 환경에서 고성능, 고 가용성, 그리고 확장성을 지향하는 전역 공유 파일 시스템이다.

현재 볼륨 관리자 1차 프로토타입 구현과 시험을 완료하였으며, 볼륨 관리자의 2차 프로토타입, 파일 관리자 및 시스템 관리 모듈을 올해 10월 완료를 목표로 구현하고 있다. 또한 본 시스템 개발 3 차년이 되는 2002년에는 시스템 관리 모듈의 통합, 시스템 안정화 및 성능 최적화를 위한 작업 그리고 공간 데이타베이스 관리 시스템의 이식 및 최적화 작업들이 병행하여 진행될 것이다. 또한 오류

감내성 지원 또는 오류 복구 기능을 지원하기 위하여 전역 잠금 모듈을 확장하는 작업이 진행될 것이다.

현재 진행되고 있는 개발 계획 외의 향후 계획으로는 호스트들 사이의 통신을 위하여 VIA(Virtual Interface Architecture)를 적용하는 것이다. 현재 호스트들 사이의 통신에 사용되는 프로토콜인 TCP/IP를 VIA로 대체함으로써 성능을 향상시키는 것을 계획하고 있다. 또한 iSCSI와 같이 SAN을 위한 통신 프로토콜의 변경이 SANtopia 시스템에 미칠 영향과 이에 따른 시스템 모듈의 변경 등에 대한 고려도 향후 계획의 하나이다.

참고 문헌

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

[2] C. A. Thekkath, T. Mann and E. K. Lee, "Frangipani: A scalable Distributed File System," DEC, 1996.

[3] C. S. Kim, G. B. Kim and B. J. Shin, "Volume Management in SAN Environment," IEEE ICPADS2001, June 2001.

[4] C. S. Park, B. J. Shin, S. I. Song and J. S. Yoo, "Parallel High-Dimensional Index Structure on the SAN," The 3rd ICACT, Feb. 2001.

[5] E. K. Lee and C. A. Thekkath, "Petal: Distributed Virtual Disks," 13th Symp. On Operating System Principles, Oct. 1996.

[6] G. B. Kim, C. S. Kim, and B. J. Shin, "A-64 bit, Scalable File System for Storage Area Networks," 5th WSES, Greece, 2001.7.

[7] IBM, "A Practical Guide to Tivoli SANergy," IBM Readbook, June 2001.

[8] K. 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.

[9] M. Baker and R. Buyya, "Cluster Computing at a Glance," High Performance Cluster Computing Vol 1, Pretice Hall, 1999.

[10] M. T. O'Keefe, "Standard File Systems and Fibre Channel," 6th Conference on Mass Storage System and Technologies in cooperation with the Fifteenth IEEE Symposium on Mass Storage Systems, pp.1-16, Colleage Park, Maryland, Mar. 1998

[11] M. J. Folk et al., "File Structures," Addison-Wesley, Mar. 1998.

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

[13] S. R. Soltis, "The Design and Implementation of a Distributed File System based on Shared Network Storage," Ph.D. Thesis, University of Minnesota, 1997.

[14] T. E. Anderson et al, "Serverless Network File Systems," 15th ACM Symp. On Operating Systems Principles, Dec. 1995.

[15] U. Vahalia, "Unix Internals: The New Frontiers," Prentice-Hall, 1996.

[16] Y. K. Lee, S. W. Kim, G. B. Kim and B. J. Shin, "Metadata Management of the SANtopia File System," IEEE ICPADS2001, June 2001.

[17] 김경배, 김영호, 김창수, 신범주, "SAN을 위한 전역 파일 시스템의 개발," 정보과학회지 제19권 3호, 2001.3.

[18] 김대호 ,김은영, 정병수, 박선영, 김창수, "공유 파일 시스템을 위한 광역 버퍼관리자의 설

계", 제 1회 한국정보처리학회 자료저장시스템연구회 워크샵, pp.154~159

[19] 김신우, 이용규, 김경배, "SANtopia의 메타데이타 및 디렉토리 구조",제1회 한국정보처리학회 자료저장시스템연구회워크샵, pp.136~141. 2001.6.

[20] 김창수, 김경배, 신범주, "SAN 환경에서 동작하는 리눅스 클러스터 볼륨 관리자" 제 11회 통신정보합동학술대회, 논문집 2권 중 제1권 pp.184~187, 2001.04.

[21] 민병준, "System Management," 제1회 자료저장시스템 워크샵 발표자료집, 2000.12

[22] 이용규, 김신우, 손덕주, "SAN 환경 공유 디스크 파일 시스템의 메타데이타 관리," 정보과학회지 제19권 3호, 2001.3.

[23] 유재수, "SAN 환경에 적합한 고차원 색인 구조의 동시성 제어 기법에 관한 연구", 한국전자통신연구원 최종연구보고서, 2000.11.

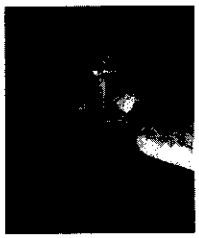
[24] 신범주, "네트워크 연결형 자료저장 시스템 소프트웨어 개발 수행계획서," 한국전자통신연구원 2001-S-015, 2001.1.

[25] 신범주, "네트워크 연결형 자료저장 시스템 연구 동향 및 발전 방향," 제1회 한국정보처리학회 자료저장시스템연구회 워크샵 튜토리얼, 2001.6.

저자약력

신 범 주

1983년 경북대학교 전자공학과(학사)
1991년 경북대학교 컴퓨터공학과(석사)
1998년 경북대학교 컴퓨터공학과(박사)
1987-현재 한국전자통신연구원(책임연구원, 자료저장 시스템S/W연구팀장)
관심분야 : 분산시스템, 이동컴퓨팅 및 클러스터 파일 시스템 등.
e-Mail : bjshin@etri.re.kr



김 경 배

1992년 인하대학교 전자계산학과(학사)
1994년 인하대학교 전자계산학과(석사)
2000년 인하대학교 전자계산학과(박사)
2000년-현재 한국전자통신연구원(인터넷서비스연구부 선임연구원)
관심분야 : 자료저장 시스템, SAN, 이동컴퓨팅, 실시간 데이터베이스시스템
e-mail : gbkim@etri.re.kr

김 창 수

1993년 광운대학교 전자계산학과(학사)
1995년 서강대학교 전자계산학과(석사)
1995년- LG소프트(현재 LGEDS)
1999년-현재 한국전자통신연구원(인터넷서비스연구부 연구원)
관심분야 : 데이터베이스 시스템, 자료저장시스템, 클러스터 시스템
e-mail : cskim7@etri.re.kr

김 명 준

1978년 서울대학교 계산통계학과(학사)
1980년 한국과학기술원 전산학과(석사)
1986년 프랑스 Nancy 1대학 전산학과(박사)
1986년-현재 한국전자통신연구원(책임연구원 인터넷서비스연구부장)
관심분야 : 데이터베이스, 분산시스템, 실시간 DB 및 소프트웨어공학 등
e-mail : joonkim@etri.re.kr