(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
H04L 12/28 (2006.01)

(45) 공고일자 2006년11월13일
(11) 등록번호 10-0644011
(24) 등록일자 2006년11월01일

(21) 출원번호 10-2000-0015184
(22) 출원일자 2000년03월24일

(65) 공개번호 10-2000-0076955
(43) 공개일자 2000년12월26일

(30) 우선권주장

09/276,428	1999년03월25일	미국(US)
09/347,042	1999년07월02일	미국(US)
09/346,592	1999년07월02일	미국(US)
09/455,106	1999년12월06일	미국(US)
09/482,213	2000년01월12일	미국(US)

(73) 특허권자
컨버전트 테크놀로지즈 인코퍼레이트
미국, 캘리포니아 95131-1845, 산호세, 트레이드 불바드존 2222

(72) 발명자
마이클지패너스
미합중국,캘리포니아주94542,헤이워드,레오나드라이브24567

알렌알머렐
미합중국,캘리포니아주94539,프레몬트,체닌블랭크드라이브48835

조셉알트마이어
미합중국,아이오와주52327,리버사이드,사우스웨스트540스트리트,3689

제임스에이테일러
미합중국,캘리포니아주94550,리버모어,플로렌스로드1033

로날드엘파크스
미합중국캘리포니아주94526,댄빌,머스탱코트55

앨러스테어테일러
미합중국캘리포니아주95123,산호세,칼레로아베뉴755

셰리제이놀란
미합중국캘리포니아주95132,산호세피나클드라이브3470

제프리에스네스포
미합중국캘리포니아주94566,플레산톤,코트베라크루즈2720

조지더블유해리스주니어
미합중국,캘리포니아주94041,마운틴뷰,뷰스트리트327

리차드에이레규오주니어
미합중국뉴햄프셔03051,허드슨,파인우드로드11

제리파커레인
미합중국,캘리포니아주95136,산호세,토니노드라이브4829

(74) 대리인 이대선

심사관 : 신성길

(54) 저장 도메인 관리 시스템

요약

저장 도메인 관리 시스템이 저장 도메인들을 지원한다. 저장 서버는 복수의 통신 인터페이스를 포함하고 있다. 복수의 통신 인터페이스의 제 1 세트는 모든 종류의 데이터 사용자들에 접속되도록 채용된다. 복수의 통신 인터페이스의 제 2 세트는 저장 도메인 내에서 이용하기 위한 저장 장치들의 풀 내에서 각각의 장치에 접속되도록 채용된다. 서버 내의 데이터 프로세싱 자원들은 인터페이스들 사이에 데이터를 전송하기 위한 복수의 통신 인터페이스에 결합되어 있다. 데이터 프로세싱 자원들은, 복수의 드라이버 모듈들, 및 드라이버 모듈들을 데이터 경로들로 링크하는 구성가능 로직을 구비하고 있다. 각각의 구성된 데이터 경로는 복수의 드라이버 모듈로부터 선택된 드라이버 모듈들의 세트를 포함하고 있는 가상 회로의 역할을 한다. 통신 인터페이스에서 수신된 데이터 저장 트랜잭션은 그 구성된 데이터 경로들 중의 하나로 매핑된다. 디스플레이 및 사용자 입력 장치는 디스플레이 장치 상에 디스플레이되는 이미지를 관리하기 위해 데이터 프로세싱 구조들과 함께 포함되어 있다.

대표도

도 1a

색인어

저장 영역 네트워크, 저장 도메인 관리, 가상 회로, 데이터 저장 트랜잭션

명세서

도면의 간단한 설명

도 1a 는 저장 도메인의 관리를 위한 저장 라우터 또는 저장 디렉터로서 구성된 본 발명에 따른 저장 서버를 구비하는 저장 영역 네트워크를 나타내는 도.

도 1b 는 지능적 저장 영역 네트워크 서버들의 다양한 이용형태를 나타내는 도.

도 2 는 이형적 네트워크에서 저장 도메인 관리를 위한 저장 라우터 또는 저장 디렉터로서 구성된 본 발명에 따른 저장 서버를 구비하는 대체적 구성에 있어서의 저장 영역 네트워크를 나타내는 도.

도 3 은 확장된 저장 영역 또는 저장 영역들을 지원하는 수개의 저장 서버를 포함하고 그들 사이에 직접형 통신 채널을 구비하는 본 발명에 따른 더욱 복잡한 저장 영역 네트워크를 나타내는 도.

도 4 는 본 발명에 따른 저장 도메인 관리를 지원하는 저장 서버의 블록도.

도 5 는 본 발명에 따른 저장 도메인 관리를 지원하는 대체적 저장 서버의 블록도.

도 6 은 지능적 저장 영역 네트워크 서버의 하드웨어 아키텍처의 블록도.

도 7 은 지능적 저장 영역 네트워크 서버의 운영체제 및 지원 프로그램들의 블록도.

도 8 은 본 발명에 따른 시스템에 이용되는 파이버 채널 인터페이스용 하드웨어 드라이버 모듈의 개략도.

도 9 는 본 발명에 따른 하드웨어 드라이버 모듈을 포함하는 고체 상태 저장 시스템의 개략도.

도 10 은 본 발명에 따른 저장 서버의 일 실시예에 장착되는 디스크 드라이브들의 내부 배열을 나타내는 도.

도 11 은 지역적 응답 능력을 가진 본 발명에 따른 타겟 서버 내부 서비스 모듈의 개략도.

도 12 는 디스크 미러를 구현하는 내부 서비스 모듈을 나타내는 도.

도 13 은 파티션 기능을 구현하는 내부 서비스 모듈을 나타내는 도.

도 14 는 캐시 기능을 구현하는 내부 서비스 모듈을 나타내는 도.

도 15 는 본 발명에 따른 가상 회로 구성을 나타내는 도.

도 16 은 본 발명에 따른 영구적 테이블 저장 관리자를 구현하는 내부 서비스 모듈을 나타내는 도.

도 17 은 본 발명에 따른 영구적 저장 하드웨어 드라이버를 개략적으로 나타내는 도.

도 18 은 본 발명에 따른 3 스테이지의 핫 카피 자원을 가진 중간 장치를 구비하는 네트워크의 개략도.

도 19 는 본 발명에 따른 핫 카피 프로세스를 구현하는 드라이버의 일례에 이용되는 데이터 구조를 나타내는 도.

도 20 은 본 발명에 따른 드라이버에 의해 실행되는 핫 카피 프로세스를 나타내는 흐름도.

도 21 은 핫 카피 프로세스 동안에 기록 요구를 처리하는 것을 나타내는 흐름도.

도 22 는 핫 카피 프로세스 동안에 판독 요구를 처리하는 것을 나타내는 흐름도.

<도면의 주요 부분에 대한 부호의 설명>

1200 : 지능적 저장 영역 네트워크(ISAN) 서버

1201, 1202, 1203 : 클라이언트 서버

1210, 1211, 1212 : 클라이언트 인터페이스

1213, 1214 : 저장 인터페이스

1205, 1206, 1207 : 저장 장치

1213 : 통신 채널

1204 : 허브

109 : 통신 링크

108 : 관리 인터페이스

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 대용량 저장 시스템에 관한 것이다. 특히, 본 발명은 지능적 저장 영역 네트워크에서의 저장 트랜잭션의 관리 및 그 네트워크의 구성에 관한 것이다.

소위 대용량 저장 시스템에 있어서, 대량의 데이터의 저장이 통상적으로 실행되고 있다. 대용량 저장 시스템은 전형적으로 데이터 네트워크 상에서 파일 서버에 연결된 저장 장치들을 포함한다. 네트워크에서 사용자들은 데이터에 액세스하기 위해 파일 서버와 통신한다. 파일 서버들은 전형적으로 데이터 채널을 통하여 특정의 저장 장치들과 연결되어 있다. 데이터 채널들은, 통상적으로 저장 트랜잭션을 관리하기 위해 설계된 점대점 통신 프로토콜로써 구현된다.

저장 량이 증가함에 따라, 통신 네트워크 내의 파일 서버의 개수도 증가되고, 저장 영역 네크워크(SAN; storage area network)라는 개념도 생기게 되었다. 저장 영역 네트워크들은 저장 트랜잭션들을 위해 최적화된 통신 네트워크 내에서 다수의 대용량 저장 시스템과 연결되어 있다. 예를들면, 파이버 채널 중재형 루프 네트워크(FC-AL network; fibre channel arbitrated loop network)들은 SAN 들로서 구현되고 있다. SAN들은, SAN 상의 저장 시스템들과 특정의 저장 시스템들의 사용자들 사이에서 다양한 점대점 통신 세션을 지원한다.

파일 서버들과 저장 시스템들의 다른 사용자들은 특정의 저장 매체와 통신하도록 구성되어 있다. 저장 시스템이 확장되거나 그 시스템 내에서 저장 매체가 대체되면, 파일 서버 및 다른 사용자들에 있어서의 재구성이 요구된다. 만약, 소위 데이터 이동 동작에 있어서, 어떤 장치로부터 다른 장치로 데이터를 이동시킬 필요가 있다면, 이동 프로세스 동안에 데이터에의 블록 액세스를 할 필요가 있게 된다. 이동이 완료되면, 사용자 시스템에서의 재구성이 실행되어 데이터는 새로운 장치로부터 이용가능하게 되도록 만들어진다.

발명이 이루고자 하는 기술적 과제

전체적으로, 저장 시스템들 및 네트워크들의 복잡도와 크기가 증가함에 따라, 데이터 및 저장 시스템 자체의 사용자들의 구성의 관리에 문제점이 증가되고 있다. 따라서, 저장 시스템의 관리를 단순화시키는 한편 SAN 아키텍처의 유연성 및 성능에 이점을 가져오는 시스템이 필요하게 된다.

발명의 구성 및 작용

본 발명은 저장 도메인 관리를 위한 시스템 및 방법을 제공한다. 저장 도메인 관리는 중앙집중화되고 안정적인 관리 능력을 가지고 있어, 기존의 저장 영역 네트워크 하드웨어 하부구조(infrastructure)의 상위의 계층이 고도의 성능, 고도의 가용성, 및 이형적(heterogeneous) 환경들에 대한 진보된 저장 관리 기능성(functionality)을 제공하도록 하게 한다. 저장 도메인 관리는, 상속(Legacy) 및 새로운 설비를 통합할 수 있은 로버스트 SAN 조직의 코어, 서버들과 저장 자원들로부터의 오프로드 네크워크와 저장 관리 태스크들, 및 호스트 네트워크 기반형 애플리케이션들을 제공하여, 그들이 모든 저장 영역 네트워크 성분들에 걸쳐 지배력을 가질 수 있게 된다. 저장 영역 네트워크는, 종래기술의 시스템 및 기법을 이용해서는 가용될 수 없는 이형적 저장 영역 네트워크 환경의 생성 및 최적화를 가능케 한다.

본 발명은 저장 도메인들에 따라 저장 네트워크 내의 저장 자원들을 관리하기 위한 시스템을 제공한다. 그 시스템은, 클라이언트들과 저장 시스템들과 저장 네트워크에 통신 매체를 통하여 접속되도록 채용되는, 복수의 통신 인터페이스를 포함하고 있다. 프로세싱 유니트가, 복수의 통신 인터페이스에 결합되어 있고, 하나 이상의 클라이언트 중의 적어도 하나의 클라이언트로 이루어지는 세트를 위한 저장 도메인으로서 하나 이상의 저장 시스템으로부터 저장 위치의 세트를 구성하는 로직을 포함하고 있다. 그 시스템은, 복수의 통신 인터페이스에 걸쳐 다중 프로토콜 지원을 제공하는 요소들, 프로토콜들 내에서 반송되는 트랜잭션 식별자들에 응답하여 저장 도메인 내에서 저장 트랜잭션을 경로지정하는 로직, 저장 도메인들을 구성하기 위한 관리 인터페이스, 복수의 통신 인터페이스들 사이에서 그 시스템을 경로지정하기 위해 통신 인터페이스들을 트레버싱(traversing)하는 저장 트랜잭션을 통상의 포맷으로 및 통상의 포맷 이외의 것으로 번역하는 로직; 저장 트랜잭션들의 데이터 주체를 캐싱하기 위한 자원들, 및 네트워크 내에서 하나의 저장 위치로부터 다른 저장 위치로의 데이터 세트의 이동을 관리하는 로직을 포함하고 있다.

일 실시예에 있어서, 본 발명에 따르는 시스템은 파일 서버와 같은 클라이언트 프로세서들과 클라이언트들을 위한 저장 도메인 내의 저장 자원들로서 이용되는 저장 시스템들 사이에서 중간 장치로서 저장 영역 네트워크 내에 포함되어 있다. 저

장 트랜잭션은 중간 장치에 의해서 수신되고, 중간 장치 내의 구성 로직에 의해 정의되는 저장 도메인의 구성에 따라 관리된다. 중간 장치는 유연적 구성, 리던던시, 페일오버, 데이터 이동, 캐싱, 및 다중 프로토콜의 지원 등을 하게 하는 저장 영역 네트워크 내에서의 관리 사이트를 제공한다. 더나아가, 일 실시예에서의 중간 장치는, 클라이언트의 재구성을 위한 요청이 없이도 클라이언트를 위한 상속 저장 장치를 저장 도메인이 포함하는 것을 허락하여, 상속 시스템들의 에뮬레이션을 제공한다.

저장 도메인들은, 네트워크 내의 클라이언트에게 논리적 저장 범위(extent)를 할당하고, 클라이언트들의 논리적 저장 범위들에게로 네트워크 내의 저장 자원을 매핑함으로써 관리된다. 클라이언트들에의 논리적 저장 범위들의 할당은, 중간 시스템 내에서, 즉 네트워크 내의 저장 자원들의 클라이언트와 논리적으로 독립된(즉, 분리되어 있는) 다른 시스템 내에서, 클라이언트에 할당된 논리적 저장 범위로의 매핑에 의해서 성취된다. 이러한 방식으로, 저장 도메인 관리자를 통하여 액세스 가능한 저장 자원들의 저장 도메인은, 중간 장치로서의 저장 도메인 관리자를 이용하여 관리된다.

본 발명에 따르는 저장 서버는, 프로세싱 유니트, 프로세싱 유니트에 연결된 버스 시스템, 통신 인터페이스, 및 프로세싱 유니트에 연결된 운영 체제를 포함하고 있다. 버스 시스템은 서버 새시(chassis) 상에 또는 슬롯들에 결합되는 통신 채널들에 걸쳐서 위치되어 있는, 데이터 저장부에의 인터페이스를 유지하기 위해 채용되는 슬롯들을 구비하고 있다. 운영 체제는 버스 시스템에서의 전송을 제어하기 위한 로직을 제공한다. 운영 체제는 클라이언트 서버들로부터의 통신 인터페이스에 수신되는 저장 트랜잭션들을 내부 포맷으로 번역하기 위한 로직을 제공한다. 운영 체제는, 트랜잭션의 프로토콜의 범위 내의 저장의 특정 유니트를 위한 통신 인터페이스 상에서의 저장 트랜잭션을, 내부 포맷을 이용하여 그 범위에 대응하는 가상 회로로 매핑하는, 구성 데이터에 따라 내부 포맷을 프로세싱하는 로직을 포함한다. 가상 회로는 이어서, 인터페이스 내의 하나 이상의 드라이버를 통하여 하나 이상의 물리적 데이터 저장부로의 트랜잭션들의 경로지정을 관리한다. 또한, 서버는 물리적 저장 장치들을 에뮬레이션 자원들을 포함하고 있어, 저장 트랜잭션들에 대한 클라이언트 서버의 구성에서의 변경없이 가상 장치들에의 액세스를 위한 표준 저장 트랜잭션 프로토콜들을 클라이언트 서버들이 이용할 수 있게 한다.

본 발명의 다른 태양에 따르면, 저장 라우터가 제공된다. 저장 라우터는, 제 1 통신 인터페이스, 다른 통신 인터페이스, 프로세싱 유니트, 및 버스 시스템을 포함하고 있다. 버스 시스템은 프로세싱 유니트, 제 1 통신 인터페이스, 및 다른 통신 인터페이스와 접속되어 있다. 프로세싱 유니트는 운영 체제를 지원한다. 운영 체제는, 가상 장치 아키텍처 및 에뮬레이션을 이용하여, 구성 데이터에 따라 제 1 통신 인터페이스에서 수신된 저장 트랜잭션들을 획득하고, 다른 통신 인터페이스로 저장 트랜잭션들을 디렉팅(directing)한다.

어떤 실시예에 있어서, 통신 인터페이스는 광섬유 매체에 대한 인터페이스이다. 어떤 실시예에 있어서, 통신 인터페이스는 파이버 채널 중재형 루프와 호환되는 드라이버들을 포함한다. 어떤 실시예에 있어서, 통신 인터페이스는 표준적 "소형 컴퓨터 시스템 인터페이스 버전 3(SCSI-3)" 과 호환되는 드라이버들을 포함한다.

어떤 실시예에 있어서, 프로세싱 유니트는 복수의 프로세싱 유니트들로 이루어진다.

어떤 실시예에 있어서, 버스 시스템은 상호접속된 컴퓨터 버스들로 이루어진다. 어떤 실시예에 있어서, 컴퓨터 버스들은 표준적 "주변장치 상호접속"(PCI) 버스에 호환되는 것이다. 어떤 실시예에 있어서, 통신 인터페이스는 버스 시스템에 결합된다.

어떤 실시예에 있어서, 저장 서버는 비휴발성 저장을 포함하고 있다. 어떤 실시예에 있어서, 비휴발성 저장은 플래쉬 메모리와 같은 집적회로 비휴발성 메모리를 포함한다.

어떤 실시예에 있어서, 저장 서버는 디스크 드라이브에 대한 제어기들을 포함한다. 어떤 실시예에 있어서, 그 제어기들은 디스크 드라이브들의 배열을 지원한다. 어떤 실시예에 있어서, 그 제어기들은 표준적 " 리던던시 배열형 독립적 디스크"(RAID) 프로토콜을 지원한다. 어떤 실시예에 있어서, 디스크 드라이브들은 광섬유 매체에 의해 제어기들과 접속되어 있다. 어떤 실시예에 있어서, 디스크 드라이브들은 광섬유 매체에의 접속을 위한 이중 인터페이스들을 구비하고 있다. 어떤 실시예에 있어서, 각각의 디스크 드라이브는 적어도 2개의 제어기에 접속되어 있다.

어떤 실시예에 있어서, 운영 체제는 통신 인터페이스에 수신된 SCSI-3 명령들과 데이터를 내부 포맷으로 번역하기 위한 로직을 포함한다. 어떤 실시예에 있어서, SCSI-3 명령과 관련된 논리적 유니트 넘버(LUN)는, 저장 서버 내에 데이터 저장부들을 포함하는 가상 장치와 SCSI-3 명령 및 데이터를 연관시키는데 이용된다. 어떤 실시예에 있어서, 개시자 SCSI-3 식별 넘버(ID) 및 LUN은 저장 서버에 연결된 데이터 소스들을 포함하는 가상 장하다 SCSI-3 명령 및 데이터를 연관시키는데 이용된다.

어떤 실시예에 있어서, 운영 체제는 저장 서버들의 성능 및 조건을 모니터하는 로직을 포함한다. 어떤 실시예에 있어서, 장치의 실패를 처리하고 리던던시 성분들에게 제어를 전송하기 위한 로직이 있다.

본 발명은 데이터의 저장 및 관리를 위한 가상 장치들 및 가상 회로들을 지원하는 저장 서버 아키텍처를 제공한다. 본 발명에 따르는 저장 서버는 복수의 통신 아키텍처를 포함하고 있다. 복수의 제 1 세트의 통신 인터페이스는 데이터의 모든 종류의 사용자에 접속되도록 채용된다. 복수개의 제 2 세트의 통신 인터페이스는 저장 장치들의 풀 내의 각각의 장치들에 접속되도록 채용된다. 저장 서버 내의 데이터 프로세싱 인터페이스들은 인터페이스들 사이에서 데이터를 전송하기 위한 복수의 통신 인터페이스들에게 접속되어 있다. 데이터 프로세싱 자원들은 복수의 데이터 모듈 및 데이터 경로들로 드라이버 모듈들을 링크시키는 구성가능 로직을 포함하고, 이들은 바람직한 시스템에 있어서 리던던시를 위해 쌍으로 이루어져 구현된다. 각각의 구성된 데이터 경로는 복수의 드라이버 모듈로부터 선택된 일 세트의 드라이버 모듈을 포함하는 가상 회로로서의 역할을 한다. 통신 인터페이스에서 수신되는 데이터 저장 트랜잭션은, 그 구성된 데이터 경로들 중의 하나로 매핑된다.

본 발명의 다른 태양에 따르면, 복수의 드라이버 모듈은 복수의 통신 인터페이스 내의 통신 인터페이스 상에서 지원되는 프로토콜을 위한 프로토콜 서버를 포함한다. 프로토콜 서버는 인터페이스 상에서 프로토콜에 따라 특정의 저장 범위를 식별하는 타겟 식별자를 인식한다. 특정의 저장 범위에 어드레싱된 트랜잭션들은 서버 내의 그 구성된 특정의 데이터 경로로 매핑된다.

이러한 방식으로 구성된 데이터 경로들은 가상 저장 장치들로서의 역할을 한다. 데이터의 사용자는 특정의 저장 장치에 대한 프로토콜에 따라 저장 서버 상에서 통신 인터페이스와 통신한다. 서버 내부에서, 그 프로토콜에 따르는 트랜잭션은 드라이버들의 세트에 의해 구현된 가상 저장 장치로 매핑된다. 특정의 데이터 경로로 수행된 저장 태스크를 셋업하고 변경하는 것, 및 하나의 데이터 경로로부터 다른 데이터 경로로의 저장 범위의 매핑을 셋업하고 변경하는 것은, 저장 서버 내에서 드라이버 모듈들의 세트들을 구성함으로써 성취된다.

본 발명의 다른 태양에 따르면, 복수의 드라이버 모듈은 각각의 통신 인터페이스를 처리하는 하나 이상의 하드웨어 드라이버 모듈, 및 복수의 통신 인터페이스에 독립적으로 데이터 경로 태스크들을 수행하는 하나 이상의 내부 드라이버 모듈을 포함한다. 데이터 경로 태스크는 예로서, 캐시 메모리 관리, 메모리 미러링 관리, 메모리 파티션 관리, 데이터 이동 관리, 및 저장 트랜잭션을 관리하기 위한 그밖의 다른 태스크들을 포함한다. 가상 장치 아키텍처 내의 이러한 타입의 데이터 경로 태스크를 제공함으로써, 이러한 태스크를 관리하는 저장 시스템의 구성은 사용자들에 본질적으로 투명한(transparent) 상태로 된다. 또한, 이러한 태스크를 수행하도록 최적화된 저장 서버에서 가상 장치 능력을 제공하는 것은, 향상된 성능과 더욱 큰 유연성을 가져오게 한다.

본 발명의 다른 태양에 따르면, 복수의 드라이버 모듈은 내부 메시지 포맷에 따라 서버 환경 내에서 데이터를 통신하기 위한 로직을 포함한다. 인입(incoming) 저장 트랜잭션은 내부 메시지로 번역되고, 특정의 트랜잭션을 위한 그 구성된 데이터 경로에 위치된다. 바람직한 실시예에 있어서, 프로토콜 서버는 프로토콜 번역 기능 및 가상 회로의 매핑 기능을 수행한다.

구성가능 로직은, 구성 데이터를 받아들이기 위한 사용자 인터페이스 및 데이터 경로를 이루는 드라이버 모듈들의 각각의 세트의 테이블들 또는 리스트들을 저장하는 메모리를 포함한다. 일 실시예에서의 구성가능 로직은, 그래픽 사용자 인터페이스, 예로서 입력 신호를 받아들이기 위한 터치 스크린을 포함하는 디스플레이를 이용하여 구현된다. 그래픽 사용자 인터페이스는 유연성과 사용 용이성을 가진 구성 툴의 구현을 가능케 한다.

본 발명의 다른 태양에 따르면, 구성가능 로직은 가상 회로에 대한 데이터 경로를 식별하는 테이블의 폼 내에 구성 데이터를 저장하기 위한 메모리를 포함한다. 일 실시예에서의 메모리는 저장 시스템의 리세트 시 및/또는 전원 차단 시에도 유지되는 비휴발성 메모리 내에 그 테이블들을 유지하는 영구적 테이블 저장 프로세스를 이용하여 구현된다. 또한, 구성 로직은 시스템 내의 리던던시 하드웨어상에서 리던던시 드라이버 모듈들을 이용하여 가상 회로들에 대한 데이터 경로들을 구현한다. 그래서, 어떠한 단일의 실패 포인트라도 특정의 저장 트랜잭션을 방해하지 않게 된다.

바람직한 실시예에 있어서, 저장 도메인들 내의 자원들은, 복수의 드라이버 모듈과 데이터 경로들로 드라이버 모듈들을 링크시키는 구성가능 로직을 포함하는 가상 회로를 이용하여 정의되고, 이들은 바람직한 시스템 내에서 리던던시를 위해 쌍으로 구현된다. 각각의 구성된 데이터 경로는 복수의 드라이버 모듈로부터 선택된 일 세트의 드라이버 모듈을 포함하는 가상 회로로서의 역할을 한다. 통신 인터페이스에서 수신되는 데이터 저장 트랜잭션은, 그 구성된 데이터 경로들 중의 하나로 매핑되고, 그럼으로써 저장 도메인 관리자 내에서 관리되고 그 구성된 저장 도메인 내에서 제어된다.

기본적으로, 저장 도메인 관리는, 비즈니스 문제를 어드레스하는 저장 영역 네트워크를 통해서 기대되는 것 전부를 고객이 실현할 수 있게 한다. 저장 도메인 관리 플랫폼은, 지능적 목적-구축형(purpose-built) 플랫폼에게, 저장 시스템들의 이형적 상호동작성을 제공하고, 안정되고 중앙집중화된 관리를 제공하고, 스케일링 가능성 및 고도의 성능을 제공하고, 신뢰성, 가용성, 서비스가능성의 특징들을 제공한다.

본 발명의 다른 태양과 이점은 이하의 첨부된 도면, 상세한 설명, 및 특허청구범위를 참조하며 이해될 수 있다.

(발명의 바람직한 실시예에 대한 상세한 설명)

개관

도 1a 는 저장 도메인 관리를 제공하는 지능적 저장 영역 네트워크(ISAN) 서버(1200)를 포함하는 네트워크를 나타낸다. 저장 영역 네트워크(SAN)는 클라이언트 컴퓨터를 위한 데이터 저장 서비스를 제공하는데 이용될 수도 있다. 저장 영역 네트워크는 파일 서버, 웹 서버 및 최종 사용자 컴퓨터 등과 같은 클라이언트 컴퓨터를 위한 넓은 대역폭과 높은 처리량의 저장을 제공하도록 최적화되어 있다. 본 발명에 따른 저장 서버(1200)는, 바람직한 실시예에 있어서, 온새시(on-chassis) 데이터 저장, 저장 트랜잭션 캐시 서비스, 저장 경로지정, 및 가상 장치 관리를 제공한다.

네트워크 상의 저장 서버(1200)는 클라이언트 서버(1201, 1202, 1203) 각각에 연결되는 클라이언트 인터페이스(1210, 1211, 1212)를 구비하고 있다. 저장 인터페이스(1213, 1214)는 저장 장치(1205, 1206, 1207)에 통신 채널들을 통하여 연결되어 있고, 저장 서버(1200) 내에 어떠한 저장이라도 결합되는 경우에는, 저장 서버(1200) 내에서 관리되는 저장 영역에 대한 물리적 저장을 제공한다. 본 실시예에서, 통신 채널(1213)은 허브(1204)를 통하여 저장 장치(1205, 1206)에 연결되어 있다. 동작에 있어서, 클라이언트 인터페이스는, 저장 영역 식별을 위해 충분한 파라미터(예를들면, 개시자(initiator)의 식별자, LUM 넘버 등의 논리적 범위, 및 타겟 저장 장치의 식별자들 중의 하나 이상을 포함한다)를 반송하고 있는 명령에 의해 클라이언트 서버가 저장 트랜잭션을 요구하는 프로토콜에 따라 동작한다. 저장 서버(1200)는 그 요구된 트랜잭션을 가상 장치로 매핑하고, 가상 장치는 이어서 물리적 저장 장치들로부터 트랜잭션에서 이용하기 위해 물리적 저장을 할당한다. 저장 장치(1200)는 또한 그 요구에서 식별되어 있는 타겟 물리적 장치를 에뮬레이션 자원을 포함한다. 저장 서버(1200)는 지역적 구성 데이터를 이용하는 저장 트랜잭션을 디렉팅하고, 클라이언트 서버를 위한 저장의 관리를 단순화할 수 있다.

높은 처리량을 제공하기 위해, 저장 서버(1200)는 파이버 채널 또는 기가-비트 이더넷과 같은 고속 네트워크 매체에 의해 클라이언트 서버(1201 ~ 1203)와 연결되어 있다. 클라이언트 서버(1201 ~ 1203)는 전형적인 구성으로 네트워크 링크에 의해 최종 사용자 컴퓨터와 연결되어 있다.

도 1a 는 통신 링크(109)를 통하여 서버(1200)에 연결되어 있는 관리 인터페이스(108)를 나타낸다. 국(108) 및 서버(1200) 내에 인터페이스에 의해 서비스되는 통신 링크는, 다양한 실시예에 있어서 예로서 이더넷 네트워크 링크, 직렬 포트에 연결되는 직렬 케이블, 또는 내부 버스 인터페이스를 포함한다.

서버(1201 ~ 1203)와 저장 장치(1205 ~ 1207) 사이의 통신은, 중간 장치로서 파이버 채널 중재형 루프 네트워크에 의해 저장 서버(1200)를 통과하도록 제공된다. FC-AL에 걸쳐 있는 채널들은, 표준적 소형 컴퓨터 시스템 인터페이스 버전 3(SCSI-3)에 따르는 프로토콜을 이용하여 실현될 수 있고, 바람직하게는 파이버 채널 프로토콜(FCP)(예로서, SCSI B X3T10 및 FCP X3.269-199X)로 불리우는 파이버 채널 매체 를 이용하여 실현될 수 있다. 다른 실시예에 있어서는, 인터넷 프로토콜(IP)과 같은 프로토콜이 다양한 프로토콜에서의 저장 트랜잭션을 반송하는 파이버 채널 조직에 대해 이용될 수 있다. 어떤 실시예에서는, 저장 서버(1200)가 데이터 저장 트랜잭션을 위해 다중의 프로토콜을 지원하기도 한다.

도 1b 는 지능적 저장 영역 네트워크(ISAN) 서버의 다양한 이용형태를 나타낸다. 저장 영역 네트워크는 클라이언트 컴퓨터를 위해 데이터 저장 서비스를 제공하는데 이용될 수도 있다. 저장 영역 네트워크는 파일 서버 및 웹 서버와 같은 클라이언트 컴퓨터를 위한 넓은 대역폭과 높은 처리량의 저장을 제공하도록 최적화되어 있다. ISAN 서버는, 데이터 저장 및 검색 이외에 저장 경로지정 및 가상 장치 관리와 같은 추가적인 기능도 제공한다.

도 1b 는 서버(100A ~ 100D), ISAN 서버(100A ~ 100F), 씬 서버(thin server, 104A ~ 104C), 및 저장 배열 부(106)를 포함하고 있다. 서버(100A ~ 100D)는 UNIX 서버, WINDOW^{TM} NT 서버, NetWare^{TM} 서버 또는 어떤 다른 타입의 파일 서버일 수도 있다.

서버(100A ~ 100D)는 네트워크 링크를 통해서 클라이언트 서버와 연결되어 있다. ISAN 서버(102A)는 네트워크 링크를 통해서 서버(100A)와 연결되어 있다. ISAN 서버(102A)는 그 요구된 저장 트랜잭션을 수행함으로써 서버(102A)에 데이터 저장 서비스를 제공한다. ISAN 서버(102A)는 서버(100A)에 의해 일종의 저장 장치와 같이 취급된다. ISAN 서버(102A)는 전형적인 하드디스크 드라이브 또는 하드드라이브 배렬 부보다 더 많은 저장 용량을 유지할 수 있다. ISAN 서버(102A)는 저장 라우터로서 이용되어, ISAN 서버(102A)에 연결된 데이터 저장부들에 대해 지능적 경로지정을 제공하는 역할을 할 수도 있다.

ISAN 서버(102A)는 또한, 전형적인 하드디스크 드라이브 또는 하드드라이브 배열보다 더 넓은 대역폭과 높은 처리량의 프로세싱을 제공한다. ISAN 서버(102A)는 그래서, 멀티미디어 데이터 스트림 및 그밖의 다른 대용량 데이터 스트림에 의해 생성되는 요구 량을 처리할 수 있다.

최고의 처리량을 제공하기 위해, ISAN 서버(1-2A)는 파이버 채널과 같은 고속 네트워크 매체에 의해 서버(100A)에 연결될 수도 있다. 서버(100B ~ 100D)는 네트워크 링크에 의해 클라이언트 컴퓨터에 연결된다. 서버(100B ~ 100D)는 파이버 채널 조직에 의해 저장 영역 네크워크에 연결된다. 저장 영역 네트워크는 ISAN 서버(102B ~ 102D) 및 저장 배열(106)을 포함한다. 서버(100B ~ 100D) 및 ISAN(102B ~ 102D)는 파이버 채널 중재형 루프(FC-AL)를 위한 드라이버를 지원한다.

서버(100B ~ 100D)와 FC-AL에 걸쳐 있는 저장 장치 사이의 통신은 표준적 소형 컴퓨터 시스템 인터페이스 버전 3(SCSI-3)에 따르는 프로토콜을 이용하여 실현될 수 있고, 바람직하게는 파이버 채널 프로토콜(FCP)(예로서, SCSI B X3T10 및 FCP X3.269-199X)로 불리우는 파이버 채널 매체를 이용하여 실현될 수 있다. 다른 실시예에 있어서는, 인터넷 프로토콜(IP)과 같은 프로토콜이 다양한 프로토콜에서의 저장 트랜잭션을 반송하는 파이버 채널 조직(108)에 대해 이용될 수 있다. 어떤 실시예에서는, ISAN 서버(102A)가 다중의 프로토콜을 지원하기도 한다.

씬 서버(thin server)(104A ~ 104C)는 네트워크 링크에 의해 클라이언트에 연결되어 있지만, 데이터 저장을 위해 저장 영역 네트워크를 이용하지는 않는다.

ISAN 서버(102E ~ 102F)는 네트워크 링크에 의해 클라이언트와 직접 연결되어 있다. 그래서, 중간 파일 서버가 없다. ISAN 서버(102E ~ 102F)는 파일 서버, 웹 서버, 및 다른 타입의 프로세싱 등의 기능을 제공하는 애플리케이션 특정 프로세서(ASP)를 포함할 수도 있다.

도 2 는 저장 영역 네트워크의 다른 실시예를 나타낸다. 도 2 에 있어서, 상술한 바와 같이 저장 디렉터 로직과 캐시 메모리를 구비하고 있는 서버(1250)는, 휴렛패커드 서버(1255), Sun 서버(1256), 및 SGI 서버(1257)등의 다양한 플랫폼 상에서 클라이언트 서버와 연결되어 있고, 이들 각각의 서버는 저장 트랜잭션의 관리를 위해 다른 프로토콜을 실행할 수도 있다. 저장 도메인으로 이용되는 물리적 자원을 구성하는 복수의 물리적 저장 장치는 또한, 서버(1250)에 연결되어 있고 지금까지 설명한 가상 장치 아키텍처에 따르는 저장 디렉터에 의해 관리된다. 본 실시예에서의 복수의 물리적 저장 장치는 휴렛패커드 플랫폼(1251) 상에의 저장, Sun 플랫폼(1252) 상에의 저장, 및 EMC 플랫폼(1253) 상에의 저장을 포함한다. 그래서, 저장 디렉터 로직을 포함하고 있는 서버는, 이형적 환경에서 상속 서버 및 저장을 지원할 수 있는 공유 저장 풀(pool)의 생성을 허락한다. 복수의 저장 장치 및 서버들 간의 비호환성은 가상 장치 아키텍처를 이용하여 필요할 때에 마스크되고 모조될(mimicked) 수 있다. 저장 영역 네트워크 환경은 저장 서버 레벨에서 실현될 수 있으며, 호스트, 조직, 및 저장 상호동작성 발송 모두가 저장 서버 레벨에서 관리될 수 있다.

가상 장치 아키텍처를 이용하는 저장 디렉터 로직은, 저장 도메인 구성을 이용하는 저장에 대한 클라이언트 서버 액세스의 구성에 단일의 지능적 좌표점을 제공한다. 새로운 장치를 추가하거나 기존의 장치의 관리를 변경하는데 있어서, 하드웨어를 재구성할 필요가 거의 없거나 전혀 없게 된다. 저장 서버의 구성은, 물리적 저장 내의 데이터 세트의 서버로의 매핑의 자동적 유지관리를 허락함으로써 정확한 구성 정보 및 제어를 제공한다. 물리적 저장의 정확한 매핑을 유지하는 것은 저장 영역 네트워크의 관리를 상당히 단순화시킨다. 또한, 서버에서의 저장 디렉터는, 기존의 장치들 및 새로운 장치들이 온라인 상태에 있는 때에, 기존의 저장 장치로부터 새로운 저장 장치로의 데이터의 활성적 이동을 제공한다. 또한, 저장 객체는 그 크기에 있어서 하나의 배열로 생성될 있는 최대 객체의 크기까지 만으로 더 이상 제한되지 않는다. 복수의 배열이, 클라이언트 서버들 상에서 실행되는 호스트 운영 체제에 무관하게, 단일의 저장 객체 내로 결합될 수 있다. 저장 디렉터는 또한, 비휴발성 캐시 내의 데이터의 스냅 샷을 하는 등의 테스트 및 백업 동작을 할 수 있고, 예로서 클라이언트 서버를 통해 경로지정됨이 없이 디스크로부터 테이프로 데이터를 복사함으로써 데이터 백업을 처리할 수 있다. 더나아가, 지역적 캐시는, 배열이 복구되거나 재구축되는 때에, 리던던시를 잃어버린 배열들로부터 데이터를 이동시키는데에 이용될 수 있고, 리

던던시 저장을 복구하고 데이터의 모든 가용성을 유지하는데에 이용될 수 있다. 하나의 공통 데이터 세트에 액세스하는 복수의 서버를 가진 애플리케이션에 대해서, 가상 장치 아키텍처를 이용하는 간단한 스케일링가능 솔루션을 제공하는 방식으로 저장 서버 내에 로킹 로직이 위치된다.

저장 서버 내의 저장 디렉터 로직은 서버 및 저장부 양쪽으로부터의 캐싱 요구들을 결합하는 동작을 하여, 저장 영역 네트워크에 요구되는 캐시 메모리의 전체 량을 감소시킨다. 시스템은, 내부 메모리로서 유효하게 제공될 수 있는 량보다 더 많은 캐시를 클라이언트 서버 또는 저장 시스템에게 할당할 수도 있다. 또한, 캐시는 시스템을 이용하는 애플리케이션에 대해 정의된 바에 따라 동적으로 또는 정적으로 할당될 수도 있다.

도 3 은 본 발명에 따른 상호접속된 복수의 저장 서버들을 이용하는 저장 영역 네트워크에 대한 더욱 구체적인 실시예를 나타낸다. 저장 서버(1300, 1301, 1302)는, 예로서 파이버 채널, 기가-비트 이더넷, 또는 비동기 전송 모드(ATM) 등의 고속 프로토콜을 이용하는 통신 채널(1350, 1351)에 의해 상호접속된 채로 포함되어 있다. 각각의 서버는, 바람직한 실시예에 있어서, 저장 디렉터 로직 및 비휴발성 캐시를 포함한다. 저장 서버(1300, 1201, 1202)는 본 실시예에 있어서 복수의 클라이언트 서버(1310 ~ 1318)에 연결되어 있다. 클라이언트 서버(1313, 1314)는 허브(1320)를 통해 저장 서버(1301)와 연결되어 있다. 마찬가지로, 클라이언트 서버(1316 ~ 1318)는 허브(1312)에 연결되어 있고, 그 허브는 이어서 저장 서버(1302)에 연결되어 있다. 클라이언트 서버(1310 ~ 1318)는 상술한 바와같은 FCP 등의 저장 채널 프로토콜을 이용하여 저장 서버와 통신한다. 이러한 프로토콜에 따라, 저장 트랜잭션이 요구되고, 그 요구의 개시자의 식별자, 논리적 유니트 넘버(LUM), 및 타겟 저장 장치의 식별자를 반송하고 있다. 이러한 파라미터들은, 저장 도메인 내의 가상 장치로 저장 트랜잭션을 매핑하기 위해 저장 디렉터 로직에 의해 이용된다. 서버들은 또한, 타겟 저장 장치를 에뮬레이션 자원을 포함하여, 클라이언트 서버들이 저장 영역 네트워크 내의 복수의 저장 장치와의 상호동작을 원활하게 하도록 하게 한다.

도 3 에 있어서, 저장 서버(1300 ~ 1302)에 연결되어 있는 복수의 저장 장치(1330 ~ 1339)가 도시되어 있다. 도 3 에 있어서, 저장 장치를 나타내기 위해, 그리고 네트워크가 이형적이고 서버(1301 ~ 1302)에서 가상 장치 인터페이스에 의해 관리되는 매우 다양한 장치를 이용할 수 있음을 나타내기 위해, 다양한 기호가 사용되어 있다. 또한, 통신 채널들도 변경될 수 있다. 그래서, 허브(1340, 1341, 1342)는 저장 장치와 저장 서버 사이에 다양한 통신 프로토콜을 이용할 수 있도록 네트워크 내에 포함되어 있다.

지능적 저장 영역 네트워크 서버

도 4 는 본 발명에 따른 저장 시스템 관리 자원들을 포함하고 있는 바람직한 일 실시예에서의 저장 서버의 블록도이다.

저장 서버(102)는 사용자에 대해 채용된 통신 인터페이스 및 데이터 처리 기능에 대해 채용된 통신 인터페이스의 세트를 포함하는 접속 옵션(130)과, 저장 장치에 대해 채용된 통신 인터페이스의 세트를 포함하는 저장 옵션(128)을 구비하고 있다. 저장 서버(102)는 하드웨어 인터페이스(126), 운영 체제(124), 블록 저장 인터페이스(118), 관리 인터페이스(120) 및 프로토콜 인터페이스(122)를 구비하고 있다. 접속 옵션(130)은 직렬 접속부(140), 일 실시예에 있어서 구성 관리 루틴을 지원하는 프런트 패널 접속부(142), 원격 관리 국과의 통신을 지원하는 이더넷 접속부(144), 및 네트워크 인터페이스(146)를 포함하고 있다. 저장 옵션(128)은 드라이브 배열(132), 고체 상태 드라이브(SSD)(134), SCSI 인터페이스(136), 및 네트워크 인터페이스(138)를 포함하고 있다. SCSI 인터페이스(136)는 DVD/CD-R(148)과 연결되어 있다. 네트워크 인터페이스(138)는 저장 서버(102G) 및/또는 저장부(150)와 연결되어 있다.

접속 옵션(130)은, 서버 및 클라이언트를 저장 서버(102)에 연결시키는 다양한 방법이다. 직렬 접속부(140)는 네트워크 관리, 원격 관리를 위한 모뎀, 및 무정전 전원 메시지를 지원한다. 프런트 패널 접속부(142)는 저장 서버(102)의 프런트 패널 디스플레이부와의 접속의 관리를 지원한다. 이더넷 접속부(144)는 관리 프로토콜을 위한 이더넷 인터페이스를 지원하며, 데이터 전송의 지원도 가능하다. 네트워크 인터페이스(146)는 서버 상에서 잠정적인 수많은 고속 인터페이스들 중의 하나이다. 어떤 실시예에 있어서는, 네트워크 인터페이스(146)는 파이버 채널 중재형 루프(FC-AL)를 위한 드라이버와의 파이버 채널 인터페이스이다. 네트워크 인터페이스(146)는 또한 파이버 채널 프로토콜(FCP)을 이용하는 파이버 채널 매체에 걸쳐서 SCSI-3 를 위한 드라이버를 포함한다.

하드웨어 인터페이스(126)는 인터페이스 특정의 하드웨어 성분들을 포함하고 있다. 예로서, 네트워크 인터페이스(146)는, 구성, 진단, 성능 모니터링, 및 헬스 및 상태 모니터링을 지원하는 네트워크 인터페이스 특정의 소프트웨어 세트를 구비하고 있다.

운영체제(124), 테이블(116) 및 인터페이스(118 ~ 122)는 저장 서버(102)의 저장 경로지정 기능 및 가상 장치를 지원한다. 저장 서버(102)의 이러한 성분들은, 시스템 내의 드라이버 모듈의 구성된 세트를 이용하여 적절한 저장 옵션들(128) 및 접속 옵션들(130)에게 저장 트랜잭션을 경로지정한다.

운영체제(124)는 고장안전(failsafe) 부에 부가하여 메시지 루틴 및 전송 부를 제공한다. 운영체제(124)의 메시지 루틴 및 전송 부들은, 저장 서버(102)의 성분들 사이에서의 저장 트랜잭션을 포함하는 메시지를 경로지정하는데 이용된다. 이들 메시지는 가상 회로의 성분들 사이의 내부 포맷의 메시지를 포함한다. 이들 메시지는 또한 다른 포맷의 제어 메시지를 포함한다.

블록 저장 인터페이스(118)는 블록 데이터 전송을 지원하는 소프트웨어 모듈을 제공한다. 인터페이스(118)는 스트립트된 데이터 저장, 미러링된 데이터 저장, 파티션된 데이터 저장, 메모리 캐시 저장, 및 RAID 저장을 지원하는 것을 포함한다. 서로 다른 타입으로 지원되는 저장들은, 미러링된 데이터 저장 등과 메모리 캐시와의 다양한 결합을 형성하도록 링크될 수 있다.

프로토콜 인터페이스(122)는 다양한 프로토콜에서의 요구를 변역하고 응답하기 위한 소프트웨어 모듈을 제공한다. 일 세트의 모듈이 이더넷 접속의 계층에 제공되는데, 그것들로는 하드웨어 드라이버, 데이터 링크 드라이버, 인터넷 프로토콜(IP) 드라이버, 전송 제어 프로토콜(TCP) 드라이버, 사용자 데이터그램 프로토콜(UDP), 및 그밖의 다른 드라이버가 있다. 또다른 세트의 모듈이 FCP에 제공된다.

관리 인터페이스(120)는 저장 서버(102)를 관리하는 소프트웨어 모듈을 제공한다. 관리 인터페이스(120)는 테이블(116)에의 액세스를 관리하는 인터페이스를 포함한다. 관리 인터페이스(120)는 또한 시스템의 룰-기반 관리를 위한 인터페이스를 포함하는데, 스케줄링 또는 프로세스 조율(orchestration), 시스템의 모니터링, 통지된 동의(informed consent) 관리, 시스템 프로세스 및 이벤트의 처리 등을 포함한다. 통지된 동의 관리 모듈은 저장 서버(102)를 구성하고 유지하기 위한 룰-기반 관리의 제안을 제공하는 것이 전제로 되어 있다.

<u>저장 트랜잭션의 처리</u>

저장 트랜잭션은 접속 옵션(130)들 중의 하나에 대해서 수신된다. 저장 트랜잭션은 상태 질의 뿐만아니라 판독 요구 및 기록 요구를 포함한다. 이들 요구는 블록 지향형으로 될 수도 있다.

전형적인 판독 저장 트랜잭션은 판독 명령 및 어드레싱 정보로 이루어진다. 기록 저장 트랜잭션은, 전송되는 데이터량에 관한 정보를 포함하고 기록된 데이터에 의해 후속된다는 것을 제외하고는 판독 저장 트랜잭션과 유사하다. 특히, SCSI-3 프로토콜을 이용하는 각각의 장치는 식별자(ID)를 구비하고 있다. 요구를 발송하는 머신은 개시자라고 불리며, 요구에 응답하는 머신은 타겟이라고 불리운다. 본 실시예에서, 서버(100A)가 개시자이고 ID(7)을 구비한다. 본 실시예에서, 저장 서버(102)가 타겟이고 ID(6)를 구비한다. SCSI-3 프로토콜은 2개 이상의 성분으로서, 논리적 유니트 넘버(LUN) 및 어드레스를 제공한다.

LUN은 타겟 ID의 하부 성분을 특정한다. 예를들면, 하드디스크/테이프 디스크의 결합 구성에 있어서, 두 장치는 하나의 ID를 공유하며 다른 LUN을 가진다. 제3의 어드레싱 성분은 데이터가 판독되거나 저장되는 장치에 대한 어드레스이다. 저장 서버(102A)는 단위(per) 개시자 기반으로 가상 LUN을 제공한다. 그래서, 단일의 저장 서버(102A)는 예로서 만개 이상의 가상 LUN을 지원하기도 한다.

저장 서버(102A)는 가상 LUN 에 대응하는 가상 회로로 SCSI-3 저장 트랜잭션 요구를 매핑할 것이다. 가상 회로는 하나 이상의 가상 장치의 시퀀스이다. 가상 장치는 소프트웨어 모듈이나 하드웨어 성분과 같은 하나 이상의 장치로 이루어진다. 예를들면, 2개의 네트워크 인터페이스 장치를 결합하여 하나의 가상 장치가 될 수도 있다. 마찬가지로, 2개의 캐시 장치가 하나의 가상 장치로 결합될 수도 있다. 이러한 설계는, 저장 서버(102)의 저장 트랜잭션 프로세싱 능력을 저하시키지 않고도 성분들을 페일 상태로 되게 할 수 있다.

가상 회로는 저장 트랜잭션을 지원하는 필수적인 가상 장치를 구비한다. 전형적으로, 가상 회로의 제 1 성분은 본 실시예에서 저장 트랜잭션 통신 채널 포맷(FCP)으로부터 내부 포맷으로의 저장 트랜잭션의 변역을 위한 드라이버이다. 이러한 내부 포맷은 지능적 입력 및 출력(I_20) 블록 저장 아키텍처(BSA) 메시지 포맷과 유사할 수도 있다. 내부 포맷은, 바람직한 실시예에 있어서, 저장 매체와 통신 채널 뉴트럴(neutral) 이다.

가상 회로의 중간 가상 장치는 캐싱, 미러링, RAID 등의 추가적인 서비스를 제공한다. 내부 포맷이 저장 매체 뉴트럴이기 때문에, 모든 중간 가상 장치는 내부 포맷으로 동작하도록 설계될 수 있고 그래서 회로 내의 다른 가상 장치와 상호동작한다.

가상 회로 내의 최종의 가상 장치는 전형적으로, 저장을 제어하기 위한 포맷 번역 및 통신 채널 드라이버이다. 예를들면, 드라이브 배열(132)은 가상 장치를 형성하기 위해 그룹화된 리던던시 하드웨어 드라이브 모듈(HDM)에 의해 제어된다. HDM은 BSA를 SCSI 번역에 제공하고, HDM은 드라이브 배열(132)을 이루는 드라이브들에의 인터페이스를 처리한다. 마찬가지로, 가상 회로가 네트워크 인터페이스(138)에 대한 어떤 다른 타입의 저장에의 링크이라면, 저장 장치 통신 채널 프로토콜에의 BSA 번역을 위한 지원을 하는 가상 장치가 될 것이다.

저장 서버는 또한, 운영체제 내의 자원과, 물리적 저장 장치를 에뮬레이팅하는 클라이언트 서버에의 인터페이스에서의 자원을 포함한다. 에뮬레이션은 저장부에 액세스하는 클라이언트 서버에 가상 장치들이 마치 물리적 장치들인 것처럼 나타나 보이도록 한다. 그래서 클라이언트 서버는, 저장 트랜잭션을 위한 SCSI 명령을 이용하는 FCP 등의 표준 프로토콜을 이용하여 통신하도록 구성될 수 있다. SCSI 명령을 이용하는 실시예에 있어서, 이러한 에뮬레이션은 장치 식별자로써 SCSI 프로토콜에 따른 질의 명령 및 개시 서버에 의해 기대되고 있는 또는 그 서버와 호환성이 있는 장치 용량 정보에 응답하도록 연관되어 있다. 또한, SCSI 프로토콜에 있어서의 판독 용량 명령 및 모든 페이지 데이터 명령은, 저장부를 이용하는 서버가 물리적 저장 장치에 대한 표준 구성 정보에 의존할 수 있게 하는 방식으로 에뮬레이션 자원에 의해 처리되고, 한편 저장 서버는 클라이언트 서버와의 인터페이스에서의 물리적 저장 장치를 에뮬레이팅함으로써 그 클라이언트 서버를 스푸프(spoof)하고 실제의 저장 트랜잭션을 가상 장치로 매핑한다. 에뮬레이션 자원은 또한, 요구 내에서 식별되어 있는 특정의 물리적 타겟 장치에 저장 트랜잭션이 연관되어 있도록 함이 없이, 가상 장치들이 개시자, 논리적 유니트 넘버(LUN), 및 타겟 장치 식별자의 결합에 의해 식별되도록 한다.

도 5 는 저장 도메인 관리에 이용하기 위한 저장 관리 시스템(151)으로서 동작하는 도 4 에 도시된 바와같은 서버의 기능적 성분들을 나타내는 블록도이다. 시스템(151)은 저장 관리자 운영 체제(152)를 포함하고 있다. 저장 관리자 운영 체제(152)와 함께, 기능적 성분들은 저장 도메인 경로지정 자원(153)과, 상속 장치 에뮬레이션 자원(154)과, 데이터 이동 자원(155)과, 리던던시, 핫 스왑 및 페일오버 자원들(156)을 포함하고 있다. 저장 관리자 운영 체제는 이들 자원들, 온새시 캐시(157), 관리 인터페이스(158) 및 본 실시예에서의 온새시 저장 배열(159) 사이에서의 통신을 조절한다.

캐시(157)는, 본 발명의 일 실시예에 있어서, 저장 트랜잭션의 안정성 지원을 위해 고체 상태 비휴발성 메모리 배열을 구비한다. 다른 실시예에 있어서는, 캐시(157)는 추가적인 결함 허용을 위해 리던던시 배열을 구비한다.

복수의 통신 인터페이스(160 ~ 165)가 시스템(151)에 제공된다. 본 실시예에 있어서, 인터페이스(160)는 클라이언트와 저장 관리 시스템(151) 사이에서 프로토콜(X)을 수행하도록 채용되고; 인터페이스(161)는 클라이언트와 저장 관리 시스템(151) 사이에서 프로토콜(Y)을 수행하도록 채용되고; 인터페이스(162)는 저장 장치와 저장 관리 시스템(151) 사이에서 프로토콜(Z)을 수행하도록 채용되고; 인터페이스(163)는 저장 장치와 저장 관리 시스템(151) 사이에서 프로토콜(A)을 수행하도록 채용되고; 인터페이스(164)는 저장 장치와 저장 관리 시스템(151) 사이에서 프로토콜(B)을 수행하도록 채용되고; 인터페이스(165)는 저장 관리 시스템(151)과 네트워크 상의 다른 저장 관리 시스템 사이에서 프로토콜(C)을 수행하도록 채용되어 있다.

본 실시예에 있어서, 프로토콜(X ~ Z) 및 프로토콜(A ~ C)은 저장 관리 시스템(151)에 의해 지원된다. 이들 프로토콜은 복수의 서로 다른 프로토콜이거나, 단일의 프로토콜의 변형들이거나, 또는 모두가 그 시스템이 이용되는 특정의 저장 영역 네트워크에서 동일한 것일 수도 있다.

저장 트랜잭션은 각각의 통신 매체로부터 저장 관리 시스템(151)의 내부 자원들로 인터페이스(160 ~ 165)를 트래버싱한다. 바람직한 시스템에 있어서, 저장 트랜잭션은 다양한 인터페이스들 사이에 경로지정을 위해 시스템에 대한 공유 메시징 포맷으로, 이들 인터페이스에 의해 실행되는 프로토콜에 무관하게, 번역된다. 저장 도메인 경로지정 자원(153)은 특정의 클라이언트 자원과 저장 장치를 위해 구성된 가상 회로를 이용하여 저장 도메인 내에 그 트랜잭션을 매핑한다. 상속 에뮬레이션 자원(154)과 데이터 이동 자원(155)은, 네트워크로부터 새로운 장치가 추가되고 제거되는 때에, 저장 도메인이 저장 관리 시스템(151)에서 재구성하도록 하게 한다. 예를들면, 새로운 저장 장치가 네트워크에 추가될 수도 있고, 기존의 저장 장치 내의 데이터 세트가 새로운 저장 장치로 이동될 수도 있으며, 그 데이터 세트를 이용하는 클라이언트들로부터의 저장 트랜잭션이 이동 중에 기존의 저장 장치 상에 유지되어 있는 것처럼 보일 수도 있고, 그 이동은 타겟 에뮬레이션이 제공됨으로써 완료된다. 리던던시, 핫 스왑, 및 페일오버 자원(156)은 폴트 허용도(fault-tolerance)를 보장하며, 높은 처리량의 데이터 저장 네트워크에 대한 저장 관리 시스템(151)의 연속적 동작을 지원한다.

하드웨어 아키텍처의 개관

도 6 은 지능적 저장 영역 네트워크(저장) 서버의 적절한 하드웨어 아키텍처의 블록도이다. 하드웨어 아키텍처는 리던던시를 구현하며, 분산형 소프트웨어 시스템을 지원하여 어떠한 단일의 고장 포인트라도 특정의 저장 트랜잭션에 간섭하지 못하도록 한다.

도 6 은 저장 서버(102A)를 포함하고 있다. 저장 서버는 표준적 성분들과 표준 기반형 장치를 이용하면서 고도의 리던던시를 제공하도록 설계되어 있다. 예를들면, 저장 서버(102A)는 고속 버전의 표준 주변장치 상호접속(PCI) 구현체와 표준적 파이버 채널 중재형 루프(FC-AL) 인터페이스를 이용한다. 다양한 다른 프로토콜과 인터페이스들이 다른 실시예에서 이용될 수도 있다.

저장 서버(102A)는 4개의 별도의 64-bit 66MHz PCI 버스(200A ~ 200D)를 구비하고 있다. PCI 버스의 슬롯에 있는 네트워크 인터페이스들과 저장 장치들의 많은 서로 다른 구성들이 가능하다. 일 실시예에 있어서, PCI 버스들은 2개의 그룹으로 분할되는데, 즉 SSD PCI 버스(200A, 200B)와 인터페이스 PCI 버스(200C, 200D)가 그것이다. 각각의 버스는 상위 버스와 하위 버스로 명명되는 2개의 버스를 가진다. 각 그룹의 상위 버스와 하위 버스는 리던던시 서비스를 제공하도록 구성될 수 있다. 예를들면, 하위 SSD PCI 버스(200B)는 상위 SSD PCI 버스(200A)와 동일한 구성을 가진다.

PCI 버스(200A ~ 200D)는 호스트 브릿지 제어기(HBC) 모듈(202A ~ 202B)에 연결되어 있다. HBC 모듈(202A ~ 202B)은 PCI 버스(200A ~ 200D)를 확장시키고 리던던시 브릿지 경로를 제공한다.

SSD PCI 버스(200A ~ 200B)는 고체 상태 드라이브(SSD) 모듈(204A ~ 204G)을 지원한다. SSD 모듈(204A ~ 204G)은 플래쉬 메모리 저장부와 같은 고체 상태 저장부를 제공한다.

인터페이스 PCI 버스는, 네트워크 인터페이스 제어기(NIC) 모듈(206A ~ 206B), 리던던시 배열형 독립적 디스크(redundant arrays of independent disks; RAID) 제어기(RAC) 모듈(206A ~ 206B), 및 애플리케이션 특정 프로세싱(ASP) 모듈(208A ~ 208D)로부터 HBC 모듈(202A ~ 202B)에의 연결을 제공한다.

저장 서버(102A)를 외부 FC-AL에 연결시킨 것에 부가하여, NIC(206A ~ 206B)는 파이버 채널 허브(FCH) 모듈(214A ~ 214D)에 연결될 수도 있다. 각각의 FCH 모듈(214A ~ 214D)은 NIC 모듈(206A, 206B) 양쪽 모두에 연결되어 있다. 각각의 FCH 모듈(214A ~ 214D)은 10개의 FC-AL 포트로 제공하며, NIC 모듈(206A, 206B)을 통하여 케스케이딩 연결되어 20개 국의 FC-AL 허브를 제공한다.

디스크 드라이브 허브(DDH) 모듈(216A ~ 216D)은, 리던던시 FC-AL 조직을 제공하여 RAC 모듈(212A ~ 212B)에 디스크 드라이브들을 연결한다. 각각의 DDH 모듈(216A ~ 216D) 내의 FC-AL 조직은 2개의 리던던시 루프를 구비하고 있는데, 그들은 RAC 모듈(212A ~ 212B)을 가지고 DDH 모듈과에 부착된 모든 드라이브와 연결되어 있다. RAC 모듈은 DDH 모듈(216A ~ 216D) 사이의 하나의 루프를 관리한다. DDH 모듈(216A ~ 216D) 각각은 디스크 드라이브(218)와 같은 5개의 이중 포트 디스크 드라이브들을 지원한다.

시스템 미드-플레인(system mid-plane; SMP)은 도 6 에는 도시되어 있지 않다. SMP는 수동적 미드-플레인으로서, HBC 모듈(202A, 202B), SSD 모듈(204A ~204H), RAC 모듈(212A, 212B), NIC 모듈(206A, 206B), FCH 모듈(214A ~214D), DDH 모듈(216A ~ 216D), 및 ASP 모듈(208A ~ 208D) 사이에서 도 6 에 도시된 바와같은 상호접속을 제공한다. SMP 는 4개의 주문형 컴팩트한 PCI 버스(200A ~ 200D), RAC-DDH 상호접속, 및 NIC-FCH 상호접속을 가지며, 미드-플레인 신호를 포함하는 여러 가지의 제어 버스를 가진 컴팩트한 PCI 기반형이다. 부가적으로, SMP 는 전원 서브시스템으로부터의 모듈들에게 48V, 12V, 5V, 및 3.3V의 전압에서 전력 분배를 제공하며, 이는 도 6 에는 도시되어 있지 않다.

프런트 패널 디스플레이(FPD)(220)는 저장 서버(102A)에 대한 사용자 인터페이스를 제공한다. FPD 는 디스플레이 장치와 입력 장치를 포함하고 있다. 일 실시예에 있어서, 터치 민감형 액정 디스플레이 장치(LCD)가 입력 기능을 가진 터치 민감형 스크린을 제공하는데에 이용된다. FPD(220)은 HBC 모듈(202A ~ 202B)에 연결되어, 상태 디스플레이, 구성 디스플레이 및 관리, 및 그밖의 관리 기능들을 지원한다.

전원 및 팬(fan) 서브시스템은(도 6 에 도시되지 않음)은 리던던시 AC-DA 전원, 리던던시 DC-DC 전력 변환, 정전에 대비한 배터리 백업 및 리던던시 푸시풀 팬 서브시스템을 제공한다. 이들 성분은 저장 영역 네트워크가 배치되는 때에 중요한 고도의 가용성 및 낮은 다운 시간에 대한 특성을 지원한다.

저장 서버(102A)는 저장 영역 네트워크 내의 단일의 네트워크 포트 또는 저장 장치에 부착되어 있는 네트워크로서 나타나는 다른 저장 서버들에 결합될 수도 있다. 이러한 결합은 HBC 모듈(202A, 202B) 각각에 연결되어 있은 FC-AL 확장 포드에 대해서 다운되어 있을 수도 있다. 추가적으로, HBC 모듈(202A, 202B)은 외부대역(out-of-band) 관리를 위해 10/100 이더넷 포트와 RS232C 직렬 포트를 제공한다.

버스 시스템은 저장 서버(102A) 내의 모든 버스들을 포함한다. 본 실시예에 있어서, 버스 시스템은 호스트 브릿지 제어기들에 의해 상호접속되어 있는 4개의 PCI 버스를 포함한다. 버스 시스템은 또한, 추가적인 인터페이스를 제공하는 HBC 모듈에 상호접속된 PCI 버스들을 포함하고 있다. 슬롯들은, 인터페이스를 수용할 수 있는 버스 시스템 상의 모든 위치를 포함한다. 본 실시예에 있어서, HBC 모듈 외부에 있은 4개의 PCI 버스 각각은 4개의 인터페이스를 수용할 수 있다.

인터페이스들은 슬롯들 내에 위치되어 있는 카드들이거나 그밖의 다른 장치들이다. 인터페이스들은 그 인터페이스들에 연결된 데이터 저장부를 위한 드라이버 및 하드웨어를 지원한다.

<u>리던던시 및 페일오버</u>

저장 서버(102a)는 고도의 리던던시를 제공한다. 일 실시예에 있어서, 리던던시 NIC, RAC 및 HBC 모듈들이 있다. SSD 모듈들과 드라이브들은 미러링을 지원한다. 이 드라이브들은 또한 패리티 및 이중 채널 액세스를 지원한다. 각각의 DDH 모듈은 RAC 모듈에의 접속을 위한 완전 리던던시 FC-AL 조직을 포함하고 있다. 페일오버는 HBC 모듈에 의해 처리되고, 그것은 저장 서버 내의 다른 모듈들을 제어한다. 이러한 제어는 다수 계층의 구조로 되어 있다.

HBC 모듈에 대한 제어의 제 1 계층은 전원제어이다. 각각의 모듈은 그 모듈 상의 CMB 제어기에 의해 제어되는 개별적인 전원 인에이블 신호를 구비하고 있다. HBC 모듈이 리던던시 상태에 있지만, 단 하나의 HBC 모듈은 마스터 HBC 모듈로서 역할을 하게 되며, 시스템을 지시하고 제어한다. 다른 HBC는 슬레이브로서 역할을 하게 된다. 하나의 모듈이 하나의 슬롯에 플러그인되는 경우에, 전원 초기에 디스에이블되게 된다. 마스터 HBC 모듈만이 전원을 인에이블할 수 있다. 어떤 모듈이 부적절하게 시작하고 명령에 응답하지 못하면, HBC 모듈은 그 모듈에 대한 전원을 디스에이블할 수 있다.

HBC 모듈에 대한 제어의 제 2 계층은 카드 관리 버스(CMB)이다. 각각의 모듈은, CMB 에 연결되어 있은 Ateml AT90S8515(AVR) 마이크로제어기를 구비한다. HBC 모듈 자체는 마스터 또는 슬레이브로서 역할을 할 수 있은 CMB 에 연결되어 있는 AVR 마이크로제어기를 구비하고 있다. CMB 마이크로제어기는 미드-플레인에 연결됨으로써 모듈 상의 주-프로세서에 공급되는 전력과 무관하게 전원공급된다. CMB 는 마스터 HBC가 카드의 타입을 판독하고, 카드가 존재하는지 여부를 판정하고, 마스크불가능한 인터럽트를 카드에 전송하거나, 카드의 하드 리세트를 수행하도록 하게 한다. 모듈 프로세서 및 마스터 HBC 모듈은 또한 모듈 상에 있는 AVR 마이크로제어기 상의 직렬 포트를 통해 통신을 수행할 수 있다. 이러한 통신 경로는 PCI 실패의 이벤트에 있어서 제어 통신을 위한 백업으로서 이용될 수 있다.

HBC 모듈에 대한 제어의 제 3 계층은 PCI 버스이다. 모듈은 PCI 버스에 대한 제어 프로세스를 이용하여 응답하지 않는다면, CMB를 통해서 질의될 수 있다. 모듈이 여전히 응답하지 않는다면, 마스크불가능 인터럽트가 CMB를 통해서 세트될 수 있다. 모듈이 그래도 여전히 응답하지 않는다면, CMB를 통해서 리세트될 수 있다. 리세트 후에도 여전히 모듈이 응답하지 않는다면, 전원이 차단되고 모듈을 대체하기 위해서 경고가 발송될 수 있다.

<u>HBC 모듈 리던던시</u>

HBC 모듈 리던던시 및 페일오버는 시스템 리던던시를 지원한다. HBC 모듈(202A, 202B) 모두는 한 번에 활성화될 수 있지만, 단지 하나만이 HOST_SEL 신호에 의해 마스터로서 지정된다. 마스터 HBC 모듈은 모든 PCI 버스에 대한 PCI 중재를 제공하고, 다른 모듈들에게 전원 인에이블 모두를 제어하며, CMB 장치 상에서 인식된 마스터이다. 백업 HBC 모듈의 PCI 중재 신호 및 전원 인에이블은 HOST_SEL 신호에 의해 디스에이블된다. CMB는 카드의 슬레이브 CMB 또는 FCB 장치 각각에서 HOST_SEL 신호에 의해 스위칭된다. HOST_SEL 신호는 레지스터에 의해 시스템 미드-플레인(SMP) 상에서 풀다운되어, HBC 모듈(202A)이 디폴트 마스터 HBC로 되게 한다. HBC 모듈(202B)은 자체를 마스터로 만들기 위해서 HOST_SEL 신호를 구동할 수도 있지만, 전형적으로는 HBC 모듈이 존재하지 않는다면 페일오버 동안에 또는 시작 시에만 일어나게 될 것이다.

에러의 발생을 감소시키기 위해, EVC 는 HOST_SEL 신호를 구동하고, 특정 패턴의 2개의 별도의 메모리 위치에 대한 하나의 기록 동작을 요구한다. 이것은 오동작하고 있는 HBC 모듈이 그 자체를 마스터로 만드는 것을 방지한다. HBC 모듈의 전원 인에이블 신호들 모두는 시작 시에 카드 모두에 대한 SMP 전원 인에이블링을 로우(low)로 풀다운하게 된다. HBC 모

듈(202A)은 HBC 모듈(202B)에 대한 전원 인에이블을 제어할 수 있다. 마찬가지로, HBC 모듈(202B)은 HBC 모듈(202A)에 대한 전원 인에이블을 제어할 수 있다. 다시, 에러의 발생을 감소시키기 위해, HBC 모듈의 전원 인에이블 신호를 구동하는 것은, 특정 패턴의 2개의 별도의 메모리 위치에 대한 하나의 기록 동작을 요구한다.

PCI 브릿지들은 이중 호스트들을 지원하지 않는다. PCI 브릿지를 특별히 구성함으로써, HBC 모듈들은 시스템 PCI 버스들 상에서 구성될 수 있게 된다. 양쪽 HBC 모듈 상의 PCI 브릿지들은, 하나의 HBC 모듈에 의해 제어되는 어드레스 공간이 시스템 PCI 버스 모두에게 위치지정된 메모리 공간으로서 다른 HBC 모듈의 PCI 브릿지 상으로 매핑되도록 고려되도록 구성된다. 하나의 HBC 모듈이 다른 HBC 모듈의 PCI 어드레스 공간에 대해 판독 또는 기록을 하려고 한다면, 그 결과는 에러로 된다. 이러한 에러는, 시스템 PCI 버스에 대한 4개의 브릿지가 심각한 에러를 발생시키는 트랜잭션을 승인하기 때문에 발생한다. 그래서, 하나의 HBC 모듈이라도 시스템 버스들에 걸쳐 다른 HBC 모듈에의 액세스를 시도해서는 안된다.

HBC 모듈은 PCI 버스를 통해서 통신할 수 없지만, HBC 모듈은 전용 직렬 포트 및 CMB 의 2개의 별도의 통신 경로를 가지고 있다. 전용 직렬 포트는, 메시지 통과를 허락하여 다른 HBC 에 대한 새니티(sanity) 체크를 제공하도록 하는 일차 통신 경로이다. 이 직렬 포트가 페일상태라면, CMB 가 HBC 모듈이 페일로 된 것을 판정하여 백업으로서 이용될 수 있다.

HBC 모듈 시작 시퀀스

시스템의 전원이 업 상태로 되면 두 HBC 모듈은 EVC에 의해 전원이 업 상태로 되기 때문에, 그들의 전원이 업 상태로 된 때에 존재하는 다른 HBC 모듈이 있는지는 판정할 필요가 있다. 이것은 CMB를 통해서 수행된다. HBC 모듈(202A)이 존재한다면, HBC 모듈(202A)이 마스터로서 디폴트된다. 아무런 HBC 모듈(202B)도 존재하지 않는 것으로서 전원 업 상태를 HBC 모듈(202A)이 판정하면, HBC 모듈(202A) 은 HBC 모듈(202B) 카드 슬롯에 전원을 디스에이블시킬 수 있다. 이것은, 마스터 HBC 모듈의 제어 하에서 제 2 HBC 모듈이 추가되고 전원 업 상태로 되는 것을 허용하게 한다. HBC 모듈(204A)은 HBC 모듈(202B)이 존재하는 것으로 판정하면, 직렬 포트를 통해서 통신을 해야 한다. HBC 모듈(202A)이 존재하지 않는 것으로서 전원 업 상태를 HBC 모듈(202B)이 판정하면, HBC 모듈(202B)은 HOST_SEL 신호를 세트하고 HBC 모듈(202A) 카드 슬롯에 전원을 디스에이블링함으로써 자신을 마스터 HBC 모듈로 만들게 된다. HBC 모듈(202A)이 존재하는 것으로 HBC 모듈(202B)이 판정하면, HBC 모듈(202B)은 직렬 포트를 통해서 통신을 하기 위해서 HBC 0을 대기하게 된다. 일정한 시간 후에도 통신이 이루어지지 않으면, HBC 모듈(202B)은 페일오버 시퀀스를 초기화하게 된다.

HBC 모듈 페일오버 시퀀스

HBC 모듈은 직렬 인터페이스를 통해서 특정한 간격으로 서로 통신해야 한다. 백업 HBC 가 마스터 HBC 와 직렬 통신을 실패하면, CMB를 통해서 마스터 HBC 모듈과 통신하려고 시도하여야 한다. CMB를 통해서 통신이 이루어질 수 있고 두 호스트가 세인(sane) 상태이면, 직렬 통신 링크는 배드(bad) 상태로 된다. 두 카드 모두는 폴트(fault)가 위치하는 곳을 판정하는 진단을 수행해야 한다. 폴트가 백업 HBC 모듈 상에 있거나 또는 분리될 수 없는 상태라면, 경고가 발생되어야 한다. 폴트가 마스터 HBC 모듈 상에 있거나 CMB 통신이 이루어지지 않으면, 백업 HBC 모듈은 마스터 HBC 모듈의 전원을 다운 상태로 하고 자신을 마스터로 만들게 된다.

소프트웨어 아키텍처의 개관

저장 서버는, 고유한 넓은 대역폭, 고도의 처리량, 및 저장 서버의 수요량을 지원하도록 설계된 운영체제에 의해 지원된다. 운영체제는 버스 시스템에 걸쳐 데이터의 전송을 스케줄링하고 제어하며, 시스템을 관리한다. 다수의 서로 다른 운영체제 및 소프트웨어 성분 구조가 가능하지만, 일 실시예에 있어서, 저장 서버를 위해 설계된 고도로 모듈화된 운영 체제가 이용된다.

도 7 은 저장 서버를 위한 지원 프로그램과 운영 시스템의 소프트웨어 모듈들의 블록도이다.

도 7 에는 다음과 같은 운영 시스템 성분들, 즉 하드웨어 인터페이스 모듈(900), Accelerated Technologies 사(미국, 알라바마 모빌 소재)로부터 가용한 Nucleus PLUS^{TM} 실시간 커넬 모듈(902), ISOS 프로토콜 관리 모듈(904), 및 저장 서비스 모듈(906)이 포함되어 있다. 하드웨어 인터페이스 모듈(900)은 저장 서버의 소프트웨어 성분이 저장 서버의 하드웨어 성분과 통신하는 것을 허락한다.

Nucleus PLUS™ 실시간 커넬 모듈(902)은 태스크, 큐, 세마포어(semaphore), 타이머, 및 한계적 섹션 지원 등의 기본적 운영 시스템 기능을 제공하는데 이용되고 있다. Nucleus PLUS™ 실시간 커넬 모듈(902)은 저장 서비스 모듈(906)에 의해 C++ 클래스 내의 함수로서 저장 서버의 소프트웨어 모듈로 익스포트된다.

ISOS 모듈(904)은 저장 서버가 입력 및 출력을 위한 메시징 아키텍처를 지원하는 허락한다. RAID 제어기(RAC) 모듈, 네트워크 인터페이스 제어기(NIC) 모듈, 고체 상태 드라이브(SSD) 모듈, 디스크 드라이브 허브(DDH) 모듈, 파이버 채널 허브(FCH) 모듈, 및 모든 입력/출력 프로세서(IOP) 등의 하드웨어 모듈은 모든 입력/출력 프로세서들(IOP)이다. 마스터 호스트 브릿지 프로세서(HBC) 모듈은 호스트로서의 역할을 한다.

저장 서비스 모듈(906)은 성분들 사이에서 메시지의 신뢰성있는 전송을 지원하는 메시징 클래스를 구현한다. 저장 서비스 모듈(906)은 장치 드라이버 모듈의 동작을 지원하고 가상 장치를 위한 지원을 한다. 장치 드라이버 모듈(DDM)들과 가상 장치(VD)들은 저장 서버 저장 시스템의 빌딩 블록이다. 저장 서비스 모듈(906)은 저장 트랜잭션에 대한 요구를 위한 지원을 제공하면서 구성된다.

어떤 애플리케이션에 있어서는, 저장 서버(102A)와 같은 단일의 저장 서버가, 저장 서버 요구에 대안 응답을 지원하기 위해, 운영 시스템 모듈(900 ~ 906)과 연동하여 동작하는 수백 개의 DDM을 구비하게 된다. 다른 애플리케이션은 수개의 DDM을 다양한 조합으로서 이용한다.

소프트웨어 성분들은 장치 드라이버 모듈(DDM)로 구현되어 있다. 하드웨어 장치에 대한 요구를 일차적으로 서비스하는 DDM은 하드웨어 드라이버 모듈(HDM)로 불리운다. 내부적 중간 프로그램으로 역할을 하는 DDM은 중간 서비스 모듈(ISM)로 불리운다. 예를들면, SSD 모듈을 서비스하는 DDM은 HDM 으로 불리운다. 캐시 서비스, 미러링 서비스, 및 다른 타입의 서비스를 제공하고 하드웨어 장치에 직접 링크되어 있지 않은 DDM은 ISM으로 불릴 수 있다.

단일의 DDM은 단일의 저장 서버 상에서 다수의 인스탠시에이션(instantiation)을 가질 수 있다. 예를들면, 도 7 에 있어서, 성능, 헬스, 및 상태 PHS 모니터들(908A ~ 908D)의 4개의 인스탠시에이션이 있고, 4개의 주요 소프트웨어 서브시스템들, 즉, NIC(910), RAC(920), HBC(930) 및 SSD(940) 각각에 대해 하나씩 주어진다. 각각의 DDM은 자신의 메시지 큐와 고유의 식별자를 가진다. 예를들면, NIC(910) 상의 PHS 모니터(908A)는 장치 ID(DID) (0) 일 수도 있다. 각각의 DDM은 또한 그 DDM 에 의해 처리된 저장 요구들의 클래스를 리스트하고, 운영 시스템 모듈은 그 요구들을 저장 요구들의 클래스에 기초한 DDM 으로 경로지정한다. 요구들은 요구 코드들에 의해 또는 가상 장치 넘버에 의해 경로지정된다.

NIC 소프트웨어 서버 시스템(910)은 3개의 DDM, 즉 프로세서 지원 HDM(912A), 입력/출력 번역 ISM(914A), 및 PHS 모니터(908A)를 포함한다. RAC 소프트웨어 서브시스템(920)은 3개의 DDM, 즉 프로세서 지원 HDM(912B), 입력/출력 번역 ISM(914B), 및 PHS 모니터(908B)를 포함한다. HBC 소프트웨어 서브시스템(930)은 프로세서 지원 HDM(912C), 입력/출력 번역 ISM(914C), 카드 관리 HDM(916), 시스템 모니터(918), 인터넷 프로토콜 DDM(921), 프런트 패널 디스플레이 DDM(922), 애플리케이션 특정 프로세서 지원 DDM(924), 및 PHS 모니터(908C)를 포함한다. SSD 소프트웨어 서브시스템(926)은 고체 상태 드라이브 관리 HDM(926) 및 PHS 모니터(908B)를 포함한다. 프런트 패널 디스플레이(950)는 하이퍼텍스트 마크업 언어(HTML) 클라이언트(928)를 지원한다.

도 8 내지 도 10 은 다양한 하드웨어 드라이버 모듈(HDM)들 나타내고, 도 11 내지 도 14 는 본 발명의 바람직한 아키텍처에 따른 다양한 내부적 중간 서비스 모듈(ISM)들을 나타낸다. 도 15 는 가상 회로로서 동작하는 데이터 경로로 구성되었던 드라이버 모듈들의 세트의 개략도이다.

도 8 은 HDM(524)를 가진 네트워크 인터페이스 카드(520)를 나타낸다. 카드(520)는 파이버 채널 네트워크에 대해 물리적 인터페이스(521)를 구비한다. 네트워크 인터페이스 칩(522)은, 본 실시예에 있어서, Q로직 사(미국 캘리포니아주 코스타 메사 소재)에 의해 제공되는 ISP 2200A 와 같은 Q로직 장치로서, 물리적 인터페이스(521)에 연결되어 있다. 네트워크 인터페이스 칩(522)은 라인(523)에 의해 표시되는 통신을 생성하며, 그것은 HDM(524) 내에서 프로세싱된다. HDM(504)는 시스템 내의 다른 드라이버 모듈에 의해 이용되기 위한 통신을 조건 설정한다. 그래서 라인(523)에 의해 표시되는 통신은 SCSI 포맷을 가진다. 라인(526)에 의해 표시되는 통신은 BSA 포맷과 같은 메시지 포맷을 가진다. 라인(527)에 의해 표시되는 통신은 인터넷 프로토콜(IP) 포맷을 가진다. HDM은 도 8 에 있어서 "Q로직 드라이버"로 표시된 드라이버 클래스의 인스턴스이고, 본 실시예에 있어서 장치 식별자(DID)(401)로 주어진다. 물리적 인터페이스는 NIC#1 으로서 식별되어 있다.

도 9 는 비휴발성 집적회로 메모리 장치들의 배열에 의해 구현되는 저장 장치(720)를 나타낸다. HDM(722)는 이 배열(721)과 연결되어 있고, 배열(721)로부터의 저장 및 검색을 위한 포맷으로 라인(723) 상의 블록 저장 아키텍처 통신을 번역한다. 본 실시예에 있어서, HDM(722)는 장치 식별자(1130)로 주어진다. 물리적 인터페이스는 SSD#4 로서 식별되어 있다.

도 10 은, 도 6 에 도시된 바람직한 실시예에서의 파이버 채널 중재형 루프 아키텍처 내의 저장 서버 새시 상에 장착되어 있는 디스크 드라이버의 배열(820)의 구성을 나타낸다. 도 6 에 또한 도시되어 있은 파이버 채널 디스크 허브 #0(216A), 파이버 채널 디스크 허브 #1(216B), 파이버 채널 디스크 허브 #2(216C), 및 파이버 채널 디스크 허브 #3(216D)은 리던던시 허브 제어 HDM 들(821, 822)에 연결되어 있다.

HDM(821, 822)는 각각 물리적 파이버 채널 중재형 루프 접속부(823, 824)에 연결되어 있다. HDM(821)는 장치 식별자(1612)로 주어지고, HDM(822)는 장치 식별자(1613)로 주어진다. 접속부(823)는 파이버 채널 인터페이스(825)에 연결되어 있다. 인터페이스(825)는 물리적 인터페이스(840)에 연결되어 있고 HDM(827)에 연결되어 있은 네트워크 인터페이스 칩(826)을 포함하고 있다. ISM(828)은 HDM(827)에 연결되어 있고 내부 통신 경로(829)에 연결되어 있다. ISM(808)은 라인(829) 상의 블록 저장 아키텍처 통신을 HDM(827)을 위한 IOCB 통신으로 번역한다. HDM(827)은 네트워크 인터페이스 칩(826)과 통신하며, 네트워크 인터페이스 칩(826)은 이어서 파이버 채널(823)을 구동한다. ISM(828)은 장치 식별자(1210)로 주어지고, HDM(827)은 장치 식별자(1110)로 주어진다. 물리적 인터페이스(825)는 RAC#0 로 표시되어 있다.

파이버 채널 접속부(824)는 인터페이스(830)와 연결되어 있다. 인터페이스(830)는 인터페이스(825)와 유사한 구성을 가진다. 그래서, 인터페이스(830)는 네트워크 인터페이스 칩(822)에 의해 구동되는 물리적 파이버 채널 인터페이스(831)를 포함한다. 네트워크 인터페이스 칩(832)은 라인(833)의해 표시되는 채널 상에서 HDM(834)와 통신한다. HDM(834)은 채널(816)을 통해서 ISM(835)와 통신한다. ISM(835)는 채널(837)상에서 BSA 포맷 메시지에 대한 인터페이스를 관리한다. 본 실시예에 있어서, ISM(835)는 장치 식별자(1211)로 주어진다. HDM(834)는 장치 식별자(1111)로 주어진다. 인터페이스(830)는 RAC#1 로 표시되어 있다.

도 11 내지 도 14 는 데이터 경로로 구성될 수 있는 본 발명에 따른 다양한 ISM 실시예를 나타낸다.

도 11 은 본 발명에 따른 프로토콜 서버 모듈의 일 실시예인 SCSI 타겟 서버(550)를 나타낸다. 유사한 프로토콜 서버 모듈들이 본 발명의 저장 서버를 통해서 관리되는 데이터의 사용자에 의해 구현된 네트워크 프로토콜이나 어떠한 특정의 저장 채널에 대해서도 구현될 수 있다. 타겟 서버(550)는, 도 8 의 HDM과 같이 사용자와 접속되도록 채용된 통신 인터페이스와 연결되어 있는 어떤 HDM 으로부터의 인입 메시지를 수신하는 메시지 인터페이스를 구비한다. 본 실시예에 있어서, 인터페이스(551) 상의 메시지는 SCSI 포맷을 가진다. 다른 실시예에 있어서는 메시지는 이미 BSA 아키텍처나 또는 제공되고 있는 통신 인터페이스 상의 프로토콜에 적절한 어떤 다른 아키텍처를 가질 수도 있다. 서버(550)는 SCSI-BSA 번역부(532)에게 또는 지역적 응답 함수(554)에게 인입 메시지를 번역시키는 스위치 함수(552)를 포함한다. 전형적으로, 메시지는 번역부(553)에 의해 인출(outgoing) 메시지로서 라인(555)상에서 전송된다. 라인(555) 상의 인입 메시지는, 라인(551) 상에서 이용되는 SCSI 포맷을 인입 BSA 메시지를 번역하는 번역부(556)에게 공급된다.

많은 예에 있어서, SCSI 타겟 장치는 메시지를 더 경로지정함이 없이 SCSI 메시지에 대해 지역적 응답 서비스(554)를 이용하여 응답할 수 있다. 저장부 자체로부터의 판독 또는 기록과 관련되지 않은 다수의 상태 메시지가 지역적 응답 서비스(554)에 의해 처리된다.

본 실시예에 있어서, 타겟 서버(550)는 클래스 SCSI 타겟 서버의 인스턴스이고, 장치 식별자(500)로 주어진다. SCSI 타겟 서버(550)와 같은 프로토콜 서버의 하나의 기능은, 연관된 인터페이스 상에서 저장 트랜잭션의 주체가 되는 저장 범위를 식별하는 것이다. 저장 범위는 저장 서버 내의 구성가능 로직을 이용하여 가상 회로로 매핑되는데, 이에 대해서는 이하에서 상술한다

도 12 는 미러 관리 데이터 경로 태스크를 수행하는 ISM(650)을 나타낸다. ISM(650)은 장치 상의 내부 통신 채널들에 접속되어 있는 인터페이스(651)를 포함하고 있다. 로직 프로세스(652)는 인입 통신 및 데이터를 수신하고 미러링 기능을 관리한다. 로직(652)은 일차 드라이브(653), 이차 드라이브(654), 삼차 드라이브(655) 및 대기 드라이브(656)를 포함하는 복수의 드라이브 인터페이스와 통신한다. 3-웨이 미러링이 도 12 에 도시되어 있지만, 어떠한 개수의 미러 경로라도 가상 회로를 이용하여 "n-웨이" 미러에 대해 구현될 수 있다. "드라이브 인터페이스"라는 용어를 사용하지만, 다른 타입의 저장

장치도 미러링 기능에서 이용될 수 있다. 드라이브 인터페이스(653 ~ 656)는, 내부 통신 채널을 이용하여 미러링 기능에서 이용되는 타겟 저장 장치와 연관된 HDM과 또는 특정의 가상 회로에 적절한 다른 ISM 모듈과 통신한다. 본 실시예에 있어서, 미러 ISM(650)는 클래스 "미러"의 일례로서 구현되며, 장치 식별자(10200)로서 주어진다.

도 13 은 파티션 ISM(750)을 나타낸다. 파티션 ISM(750)은 다른 드라이버 모듈로부터의 내부 통신을 수신하는 인터페이스(751)와, 또한 다른 드라이버 모듈로부터의 내부 통신을 수신하는 인터페이스(752)를 포함하고 있다. ISM(750)은 파티션 로직 프로세스(753), 베이스 어드레스(754)와 리미트 어드레스(755)를 저장하기 위한 데이터 구조, 및 드라이브 인터페이스(756)를 포함하고 있다. 파티션 로직 프로세스(753)는 다양한 저장 관리 기술에 유용한 논리적 파티션 기능을 이용하여 드라이브 인터페이스(756)에 의해 식별된 주체 저장 장치를 구성하여, 물리적 장치가 가상 회로 내에서 하나 이상의 논리적 장치로서 보여지도록 하게 한다. 본 실시예에 있어서, 파티션 ISM(750)는 클래스 "파티션"의 인스턴스이고, 장치 식별자(10400)로 주어진다.

도 14 는 캐시 ISM(850)을 나타낸다. 캐시 ISM(850)은 저장 서버 상의 내부 메시지 패싱 구조에의 인터페이스(851)와 통신하는 로직 프로세스(853)를 포함하고 있다. 캐시 ISM(850) 내의 데이터 구조는 지역적 캐시 메모리 할당(854), 캐시(854) 내에 저장된 데이터를 식별하는 캐시 테이블(855), 및 드라이버 인터페이스(856)를 포함하고 있다. 드라이브 인터페이스는 캐시에 의해 제공되고 있는 특정의 가상 회로와 연관된 HDM 과 채널(857) 상에서 통신하고 있다. 일 실시예에서의 캐시 메모리(854)는 저장 서버 내에서 지역적으로 관리된다. 다른 실시예에 있어서, 캐시는, 도 9 와 관련하여 설명된 것과 같은 아키텍처를 가진 고체 상태 메모리 모듈 등의 고속 비휴발성 메모리 내에 저장될 수도 있다. 바람직한 실시예에 있어서, 캐시 ISM(850)는 클래스 "캐시" 의 인스턴스이고, 장치 식별자(10300)로 주어진다.

도 15 는 본 발명에 따른 드라이버 모듈을 포함하고 데이터 경로에 의해 구현된 리던던시 가상 회로들에 대한 이해를 돕기 위한 도면이다. 가상 회로는 데이터의 사용자와 통신하기 위한 외부 인터페이스, 사용자와의 통신을 드라이버 모듈의 통신 포맷으로 번역하기 위한 프로토콜 번역자, 및 저장 장치에의 통신 인터페이스를 포함하고 있는 저장 객체를 포함하고 있다. 데이터 경로 태스크를 수행하는 저장 연산자는 번역자와 저장 객체 사이에 존재할 수 있다. 캐시, 미러, 파티션 등의 저장 연산자로서 역할을 하는 드라이버 모듈들에 대한 최적의 오더링은, 저장 서버에 의해 제공되는 구성가능 로직을 이용하여 시스템 설계자에 의해 수행된다.

도 15 에 도시되어 있는 실시예에 있어서, 외부 인터페이스는 NIC #0 에 의해 제공되고, 그에 대응하는 HDM은 블록(1010)으로 표시된다. 프로토콜 번역자는 SCSI 타겟 서버 ISM(1011)에 의해 제공된다. 캐시 기능은 ISM(1012)에 의해 제공된다. 미러 기능은 ISM(1013)에 의해 제공된다. 저장 객체는 미러기능(1013)으로서 액세스되고, 블록(1014)으로 표시된 파이버 채널 기본적 데이지 체인 인터페이스와 그에 대응하는 HDM 으로부터 선택된 물리적 저장 인터페이스 또는 외부 LUN 인터페이스, 블록(1015)과 리던던시 블록(1016)으로 표시된 ISM/HDM 쌍을 통하여 액세스되는 파이버 채널 중재형 루프 내의 디스크 드라이브, 블록(1017)에 의해 표시된 고체 상태 저장 장치와 그에 대응하는 HDM, 및 블록(1018)에 의해 표시된 외부 디스크 드라이브와 그에 대응하는 ISM/HDM 쌍을 포함하고 있다. 디스크들((01),(02),(03),(04))에 대한 파이버 채널 인터페이스 상에서 별도의 HDM 모듈이, 파이버 채널 중재형 루프에 걸쳐 인터페이스(1015, 1016)와의 통신을 관리한다.

본 실시예에 있어서, 미러 모듈(1013)은 미러 기능에 대한 제 1 드라이브, 제 2 드라이브, 및 대기 드라이브로서 각각의 디스크들((01),(02),(03))에 액세스한다. 도 12 에 도시된 미러 모듈은 제 3 드라이브 인터페이스를 포함하고 있지만, 제 3 드라이브 인터페이스는 본 실시예의 시스템에서는 이용되지 않는다.

또한 도 12 에는 도시되어 있지 않은 파티션 ISM 모듈(1020, 1021)은, 도시된 가상 회로의 데이터 경로와 접속되어 있지 않다. 이러한 블록들은, 가상 회로 구조를 이용하여 파티셔닝과 같은 새로운 모듈이 저장 서버를 간단하게 구성하기 위해서 경로에 부가될 수도 있다는 것을 나타내기 위해 도시되어 있다.

리던던시 데이터 경로는, 블록(1025)으로 표시된 인터페이스 NIC #1 과 그에 대응하는 HDM, 블록(1026)으로 표시된 SCSI 타겟 서버 ISM, 블록(1027)으로 표시된 캐시 ISM, 및 블록(1028)으로 표시된 미러 ISM을 이용하여 구현된다. 데이터 저장 장치 내의 리던던시는 미러 기능을 이용하여 이루어진다. 리던던시 드라이버 모듈은, 바람직한 실시예에 있어서는, 저장 서버 내에서 별도의 IOP 상에 배치된다.

도 15 에 도시된 바와 같이, 각각의 드라이버 모듈은 도 15 내에서 괄호들 내에 표시된 고유의 드라이버 식별자를 포함한다. 고유의 장치 식별자는 저장 서버에 의해 관리되는 구성 데이터 베이스 내의 테이블에 기초한 구성 로직을 지원하는데에 이용되고, 저장 서버 내의 지역적 구성가능 로직에 의해 제어된다.

바람직한 시스템에 있어서, 구성가능 테이블은 도 16 및 도 17 에 도시된 바와같은 영구적 테이블 드라이버에 의해 관리된다. 도 4 를 다시 참조하면, 저장 서버(102)는 테이블(116) 과 같은 테이블 내에 관리 및 경로지정 정보를 저장한다. 테이블(116)은 관리 인터페이스(120)를 통해서 액세스될 수 있다. 테이블(116)은 전형적으로 비휴발성 메모리와 같은 영구적 메모리 내에 저장된다. 테이블(116)은 고장안전 지원을 제공하기 위해 리던던시 방식으로 유지될 수 있다.

도 16 은 드라이버 모듈 구조의 기본적 아키텍처를 따르는 클래스 "영구적 테이블"의 인스턴스로서 구현되는 영구적 테이블 모듈(1400)을 나타낸다. 영구적 테이블 모듈(1400)은 테이블 액세스 논리적 프로세서(1401)와, 테이블 데이터 액세스 관리자(1402), 영구적 이미지 관리자(1403), 및 영구적 테이블 인스턴스 동기 모듈(1404)을 포함하는 다양한 지원 기능을 포함하고 있다. 테이블 데이터 액세스 관리자(1402)는, 본 실시예에 있어서, 테이블 클래스 관리자(1405)와 결합되어 있다. 테이블 클래스 관리자는 파이버 채널 포트 ID 테이블(1406), LUN 익스포트 테이블(1407), 구성 템플릿 테이블(1408), DDM 롤(roll) 호출 테이블(1409), 가상 장치 테이블(1410), 저장 롤 호출 테이블(1411), 파이버 채널 디스크 롤 호출 테이블(1412), 외부 LUN 테이블(1413), 및 고체 상태 저장 테이블(1414)을 포함하는 복수의 구성 테이블을 관리한다. 영구적 테이블 모듈(1400)에 의해 관리되는 테이블 세트의 특정의 구성은, 특정의 구현체에 적합하도록 변경될 수 있고 장치들의 일정한 클래스에 대해 최적화될 수 있다.

영구적 이미지 관리자(1403)와 테이블 인스턴트 동기 관리자(1404)는 도 11 에 도시된 바와 같이 영구적 데이터 저장 드라이버(1420) 및 제 2 영구적 저장 드라이버와 통신한다. 영구적 데이터 저장 드라이버(1420)는, 클래스 "영구적 저장" 의 인스턴트이며 상술한 드라이버 모듈의 모델을 따르는 장치 식별자로 주어지는 HDM 으로서 구현된다. 바람직한 시스템에 있어서, 영구적 데이터 저장 HDM(1420)은 저장 서버 내의 고체 상태 저장 장치와 통신하고, 가상 회로에서 이용되는 데이터에의 고속 액세스를 제공한다.

영구적 데이터 저장은 시스템에 대한 매우 다양한 구성 정보를 유지한다. DDM 롤 호출 테이블(1409)은 장치 드라이버 모듈들의 모든 인스턴스의 리스트 및 그들의 고유한 장치 ID를 포함하고 있다. 저장 롤 호출 테이블(1411)은 저장 서버에 의해 검출된 활성화된 저장 장치의 모든 리스트를 포함하고 있다. 롤 호출 테이블들은 가상 회로를 생성하기 위해서 가상 장치 테이블(1410) 및 구성 툴에 의해 이용될 수 있다. LUN 익스포트 테이블(1407)은, 저장 채널 트랜잭션 내의 식별된 저장 범위들을 가상 회로로 매핑시키는 기법을 제공한다. 외부 LUN 테이블(14313)은, 저장 서버 상에서 외부 저장 인터페이스를 통하여 접속된 다른 저장 서버들 내에 유지되어 있는 저장의 논리적 유니트 식별한다.

2개의 일차 테이블이, 클라이언트로의 저장의 익스포트하는 것과, 저장 서버(102A)의 저장 경로지정 기능을 지원한다. 이들 테이블은 익스포트 테이블(1407)과 가상 장치 구성 테이블(1410)이다.

익스포트 테이블(1407)

익스포트 테이블(1407)은, 저장 트랜잭션과 함께 수신된 어드레싱 정보를 가상 회로 또는 저장 옵션으로 맵핑한다. 파이버 채널 인터페이스에 대한 SCSI-3 의 경우에 있어서, 이용되는 어드레싱 정보는 개시자 ID, 타겟 LUN, 및 타겟 어드레스이다.

많은 LUN 이 모든 개시자들 또는 클라이언트들에 걸쳐 공유될 수 있기 때문에 그리고 서로 다른 가상 회로를 선택하기 위해서라기보다는 오히려 가상 회로 내에서의 어드레싱을 위해서 대부분의 LUN 이 타겟 어드레스(예로서, 저장 장치 상의 오프셋)를 이용하게 될 것이기 때문에, 각각의 요구를 해결하기 위해 이러한 정보 모두를 이용하는 것이 필수적인 것은 아니다. 그래서, 전형적인 실시예에 있어서, 익스포트 테이블(1407)이 테이블 1 에 도시된 바와 같이 구성된다.

테이블 1

프로토콜	프로토콜 특정 어드레싱 (LUN)	개시자가 특정되었는가? 만약 예 이면, ID	회로 내의 제 1 가상 장치	일차 접속 소유자
SCSI	0	아니오	11	NIC 0
SCSI	1	예 , ID=6	30	NIC 0
SCSI	1	예 , ID=5	60	NIC 1
SCSI	2	아니오	12	NIC 0
TCP/IP	포트 2000	아니오	70	NIC 0

익스포트 테이블(1407)은, 가상 회로의 현재 상태, 가상 회로의 용량, 및 그밖의 정보 등이 다른 컬럼들을 포함할 수도 있다. 일 실시예에 있어서, 익스포트 테이블(1407)은 익스포트 테이블의 컬럼 내에 전체 가상 회로를 리스트한다.

테이블 1 은, 프로토콜 특정의 어드레싱 정보가 적절한 가상 회로에 대해 요구를 경로지정하는데에 이용될 수 있다는 것을 보여주고 있다. 그래서, 저장의 타겟 범위의 식별자로서 포트(2000)를 이용하는 TCP 세션만이, 식별자(70)를 가진 가상 회로로서 시작하여 가상 회로에 경로지정될 수 있다.

테이블 1 은, 프로토콜을 에 대한 단일의 LUN 이 저장 트랜잭션의 개시자에 의존하는 서로 다른 장치들에 접속될 수 있다는 것을 보여주고 있다. 본 실시예에 있어서, LUN 1 은 개시자 ID 에 기초하여 서로 다른 가상 회로로 매핑되어 있다. 또한 가상 회로들은 월드 와이드 네임(WWN) 과 같은 다른 타입의 식별자들에 기초하여 매핑될 수도 있다.

```
#define EXPORT_TABLE   "Export_Table"

struct ExportTable Entry {

        rowID           ridThisRow;         //본 테이블의 rowID
        U32             version;            //익스포트 테이블 레코드의 버전
        U32             size;               //익스포트 테이블 레코드의 크기

        CTProtocolType  ProtocolType;       // FCP, IP, 그밖의 다른 것
        U32             CircuitNumber;      // LUN 또는 그밖의 다른 것
        VDN             vdNext;             //경로에서의 제 1 가상 장치 넘버

        VDN             vdLegacyBsa;        //상속 BSA의 가상 장치 넘버

        VDN             vdLegacyScsi;       //상속 SCSI의 가상 장치 넘버

        U32             ExportedLUN;        // 익스포트된 LUN 넘버
        U32             InitiatorId;        //호스트 ID
        U32             TargetId            // 타겟 ID
        U32             FCInstance;         // FC 루프 넘버
        String32        SerialNumber;       //시리얼 넘버에 대해 스트링 배열을 이용함
        long long       Capacity;           //본 가상 장치의 능력
        U32             FailState;
        U32             PrimaryFCTargetOwner;
        U32             SecondaryFCTargetOwner;
        CTReadyState    ReadyState;         // 현재의 상태
        CTReadyState    DesiredReadyState;  // 소망하는 대기 상태
        String16        WWNName;            //월드 와이드 네임(64 또는 128 비트
                                            IEEE 레지스트됨)
        String32        Name;               //가장 장치 네임

#endif
```

가상 장치 구성 테이블

가상 장치 구성 테이블은, 가상 장치를 지원하는 장치 드라이버와 함께 가상 장치들에 접속된다. 가장 장치들은 리던던시 설계를 지원하도록 설계되어 있다. 그래서 가장 장치 구성을 위한 테이블은 가장 장치 넘버를 장치 모듈로 매핑한다. 일 실시예에 있어서, 테이블 2 와 같은 테이블은 가상 장치들을 그 지원하는 장치 드라이버들로 매핑하는데에 이용된다. 도 15 는 가상 장치(12)를 시작으로 하여 테이블 2 에 의해 구현된 가상 장치들을 나타내고 있다.

테이블 2

가상 장치	일차	대체	파라미터	상태	클래스
1	4000	4001	N/A	일차	영구적 테이브
10	1210	1211	SO(00)	대체	FC 디스크
11	500	501	VD(10)	일차	SCSI 타겟
12	500	501	VD(13)	일차	SCSI 타겟
13	10300	10301	VD(14)	일차	캐시
14	10200	10201	VD(15,16, null,17)	일차	미러
15	1210	1211	SO(02)	일차	FC 디스크
16	1210	1211	SO(03)	일차	FC 디스크
17	1210	1211	SO(04)	일차	FC 디스크

테이블 2 에 도시된 바와 같이. 각각의 가상 장치에 대해서, 정보가 가상 장치를 지원하는 일차 드라이버 모듈 및 대체 드라이버 모듈에 대해 제공된다. 예를들면, 테이블 2 내의 제 2 엔트리에서, 파이버 채널 디스크 드라이브는 가상 장치(VD)(10)로 매핑되어 있다.

가상 장치는, 그 가상 장치를 지원하기 위한 하나 이상의 소프트웨어 모듈 또는 하드웨어 모듈을 구비하고 있다. 파라미터 컬럼은 초기화 정보를 제공하는데 이용된다. VD(10) 의 경우에, 파라미터는 저장 옵션 0을 의미하는 SO(00)이다. 각각의 장치 드라이버 모듈 클래스는 클래스 특정의 파라미터를 가진다. 저장 옵션 드라이브들은 특정의 저장 유니트를 지정하는 파라미터들을 이용한다. 미러 드라이버 및 캐시 드라이버와 같은 중간적 드라이버 클래스는, 가상 회로 내에서 후속 가상 장치를 지정하는 파라미터를 이용한다. 이러한 포맷은 단일의 장치 드라이버가 파라미터 세팅에 기초하여 다수의 장치들을 지원하는 것을 가능케 한다. 주목할 것은, 테이블 2 에 있어서, 장치 드라이버(1210)가 가상 장치(10, 15, 16, 및 17)에 의해 이용되고 있지만, 각각은 서로 다른 파라미터를 그 드라이버에 지정하고 있다.

상태 컬럼은, 가상 장치를 지원하는 소프트웨어 모듈 또는 하드웨어 모듈의 상태를 을 나타낸다. 예를들면, 테이블 2 의 제 1 엔트리에서, 상태는 "일차" 이고, 이것은 일차 장치 드라이버(여기서는 4000)가 이용되고 있음을 의미한다. 테이블 2 내의 제 2 엔트리에는, 2 상태가 "대체" 이고, 그것은 일차 장치 드라이버가 실패 또는 적절히 응답하고 있지 않음을 의미한다. 이 경우에, 테이블 2 의 제 2 엔트리의 대체 드라이버(1211)가 이용된다. 장치가 하나 이상의 대체 드라이버를 가진다면, 상태 컬럼은 이용되고 있는 드라이버를 나타내게 될 것이다.

예

예를들어, SCSI 프로토콜을 이용하고 어드레싱 정보 내에 LUN 2 를 지정하여, 저장 서버에게 접속 옵션들(130) 중의 하나에 들어오는 저장 트랜잭션을 고려한다. 저장 서버(102A)는 본 실시예 대한 테이블 1 및 2 에 나타낸 바와 같이 구성된다고 가정한다.

네트워크 인터페이스(146)(이를 통하여 저장 트랜잭션이 수신되고 있음)와 같은 접속 옵션은, 하드웨어 장치 드라이버에 결합된다. 하드웨어 장치 드라이버는, 프로토콜에 무관하게 저장 트랜잭션을 수신하며 그 프로토콜을 처리하기 위한 적절한 가상 장치로 그 저장 트랜잭션을 인출한다.

예를들면, SCSI 저장 트랜잭션은 SCSI 타겟 클래스 내의 장치 드라이버로 보내진다. 마찬가지로, IP 저장 트랜잭션은 IP 타겟 클래스 내의 장치 드라이버로 보내진다. 여기서, 저장 트랜잭션은 SCSI 통신 프로토콜을 이용하여 형성되며, 그래서 SCSI 타겟 장치 드라이버(DID(500))로 경로지정된다.

SCSI 타겟 장치 드라이버는 요구를 세부적으로 분석한다. 이러한 분석의 제 1 부분은, 요구를 어느 가상 회로로 매핑할 것인가를 판정하는 것이다. 이러한 판정은 익스포트 테이블 내의 정보를 이용하여 이루어진다. 본 실시예에 있어서, 테이블 1 은, LUN 2를 지정하는 SCSI 프로토콜을 이용하는 요구가 가상 장치 12 와 함께 시작하는 가상 회로로 경로지정되어야 하는 것을 나타내고 있다. 일 실시예에 있어서, 모든 SCSI 타겟 요구는 단일의 인터페이스에 대해서 동일한 SCSI 타겟 드라이버로 경로지정된다. 본 실시예에 있어서, 타겟 VD(12)에 대한 파라미터 정보는, SCSI 타겟에 대한 제 2 가상 장치에 메시지를 경로지정하는 것보다 오히려 SCSI 타겟 장치의 동태를 제어하는데에 이용된다.

SCSI 타겟 장치(여기서는 드라이버 넘버 500)는 SCSI 메시지를 내부 포맷으로 번역한다. 이러한 포맷은 I_2O 블록 저장 아키텍처(BSA) 포맷에 기초한다. 이 포맷은 장치 및 프로토콜 뉴트럴이며, 중간 장치 드라이버에 의해 이용될 수 있다. 요구가 내부 포맷 내에 있게 되면, 그것은 파라미터 필드(여기서는 파라미터가 VD(13)임)에 의해 표시되는 가상 회로 내의 후속 가상 장치(여기서는 가장 장치(13)임)로 보내진다.

메시지는 VD(13)으로 경로지정되고, VD(13)은 리던던시 캐싱 드라이버(여기서는 10300 및 10301 로 넘버가 주어진 드라이버)를 제공한다. 캐싱 드라이버는 저장 트랜잭션을 캐시하기 위해 메모리를 이용한다. 드라이버에 의해 이용되는 캐싱 알고리즘에 기초하여, 드라이버는 저장 트랜잭션을 적절한 간격으로 가상 회로 내의 후속 가상 장치로 경로지정하게 된다. 여기서, 후속 장치는 파라미터 VD(14)로 지시되고, 가상 장치(14)가 된다.

내부 포맷 내에서, 메시지는 VD(14)로 경로지정된다. 가상 장치(14)는 리던던시 미러링 드라이버들을 포함하고 있다. 이 경우에, 드라이버(12000, 1201)가 이용된다. 미러링 드라이버는 다수의 볼륨들에 저장의 미러링된 이미지를 유지하기 위한 미러링 알고리즘을 구현한다. 미러링 드라이버는 대기 저장부뿐만아니라 일차 저장부, 이차 저장부, 및 삼차 저장부를

지원한다. 다른 미러링 드라이버는 다른 알고리즘을 지원할 수도 있다. 미러링 드라이버는 또한 기존의 저장부와 동기성을 확고하게 가져오는 새로운 저장부의 결합을 지원한다. 드라이버에 의해 이용되고 있는 미러링과 그 미러링된 저장부의 상태에 기초하여, 드라이버는 가상 회로 내의 적절한 가상 장치에 저장 트랜잭션을 경로지정하게 될 것이다. 일차 저장부와 대체 정장부들이 모두 기능을 하고 있다고 가정하면, 미러 드라이버는 파라미터 VD(15, 16, null, 17)에만 따르는 일차 저장부 및 이차 저장부(즉, 가상 장치(15, 16))에게 이 요구를 경로지정하게 될 것이다. 파라미터 리스트 중의 "null" 은, 어떠한 삼차 드라이브도 이 가상 장치에 대해서 현재 이용되고 있지 않다는 것을 나타낸다.

미러링 드라이버는 2개의 장치에 직렬 또는 병렬로 저장 트랜잭션 메시지를 경로지정할 수도 있다. 비록 본 실시예가 이차 저장부, 즉 가상 장치(16)로 확장될 수도 있지만, 본 실시예에 있어서는 가상 장치(15)로의 메시징이 고려될 것이다. 가상 장치(15)는 파이버 채널 드라이브를 제어하기 위한 리던던시 드라이버를 포함한다. 드라이버는 내부 포맷을 드라이브들에 의해 이용되는 포맷으로 번역한다(예로서, BSA 로부터 SCSI로의 번역). 드라이버들은 또한, 어드레싱 정보를 드라이브에 제공한다. 여기서 파라미터 SO(02)가 저장 옵션(여기서는 파이버 채널 드라이브 넘버 2)을 선택하는데에 이용된다.

따라서, 저장 플랫폼 내에서, 하드웨어 기능들(디스크 또는 플래시 저장 등) 및 소프트웨어 기능들(RAID 스트립들 또는 미러들 등)은 통상적으로 장치라고 불리우는 소프트웨어 드라이버들을 통해서 모두 액세스된다.

이들 장치는 쌍을 이루며(이 쌍의 구성요소 각각은 리던던시를 위한 별도의 보드를 실행하는 것이 바람직함), 가상 장치라고 불리운다. 이들 장치는 체인을 이루어 다양한 구성을 만든다. 예로서, 미러 장치는 2개 또는 3개의 디스크 장치가 체인을 이룬 것으로 될 수도 있다. 이러한 타입의 구성을 통해서, 가상 장치 체인들이 구축된다. 이들 가상 장치 체인은 또 다른 구성에서 이용될 수 있는 어떤 BSA 타입의 장치로 구성되고 있는 한은, 추가될 수 있다.

가상 장치 체인들은 FCP/SCSI 타겟 서버 장치에 접속되고, "익스포트"를 위한 FCP 타겟 드라이버의 LUN 익스포트 테이블로 매핑된다 (즉, 외부 세계로부터 FCP 프로토콜을 통해 액세스 가능하게 됨). 이때, 그 헤드에 SCSI 타겟 서버 장치를 가진 가상 장치 체인은 가상 회로라고 불리운다.

가상 회로를 생성하는 역할을 하는 가상 회로 관리자 소프트웨어는 SCSI 타겟 서버 "헤드"를 가상 장치 체인 사에 두며, 그 다음에 FCP 타겟의 익스포트 테이블을 갱신함으로써 가상 회로를 엑스포트한다. 이 소프트웨어는 또한 삭제, 정지(quiesce), 및 페일오버 동작을 지원한다.

가상 회로 관리자 소프트웨어는 또한 각각의 가상 회로 내의 모든 가상 장치들을 단일의 위치 내로 리스트하는 가상 회로 테이블(VCT)을 유지하는 역할을 한다. 이러한 정보는, 페일오버, 핫 스왑 및 차단(shutdown) 등의 많은 동작을 구현하는데에 필요하다.

초기화되는 경우에, 가상 회로 관리자 소프트웨어는 영구적 테이블 저장부 내에 VCT 자체를 정의한다. 가상 회로 관리자 소프트웨어는 또한 VCT 에 대한 삽입, 삭제, 및 어떠한 변경들이라도 청취한다.

새로운 가상 회로를 생성하기 위하여, SCSI 타겟 서버를 초기화하고 새로운 LUN을 매핑하여 익스포트하는데 필요한 정보가, VCT 내의 레코드로 위치되어야 한다. 가상 회로 관리자 소프트웨어는 또한 VCT 로의 삽입을 청취하고, 청취 응답을 수신하는 중에 다음과 같은 동작을 수행한다.

1. 새롭게 삽입된 레코드 내에 정보를 유효화되도록 시도한다. 만약 그 레코드가 무효 정보를 포함하면, 그 상태 필드는 에러를 지시하고, 그 다음의 동작을 취한다.

2. 새롭게 삽입된 레코드에 의해 지정된 가상 회로의 LUN 을 위한 새로운 SCSI 타겟 서버 장치를 생성한다.

3. 새로운 레코드 내의 상태를 "인스탠시에이션"으로 설정한다.

4. 가상 회로에 할당된 저장은 저장 롤 호출 테이블 내에서 이용되도록 플래그될 것이다.

5. 익스포트 테이블은 새로운 SCSI 타겟 서버에 LUN을 인출시키게 될 것이다.

가상 회로 내의 레코드가 삭제되는 경우에, 가상 회로 관리자는 다음과 같은 동작을 수행한다.

1. 가상 회로가 이미 정지되지 않았다면, 그 가상 회로를 정지하고, 그것을 정지된 상태로 마크한다.

2. 가상 회로의 인출 데이터를 익스포트 테이블로부터 제거한다.

3. 가상 회로 레코드로부터의 롤 호출 레코드를 이용하지 않은 상태로 마크한다.

4. 가상 회로와 연관되어 있는 SCSI 타겟 서버를 비-인스탠시에이션시킨다(de-instantiate).

가상 회로 관리자는 또한 VCT 내의 "익스포트된" 필드에 대한 변형을 청취한다. 만약 VCT 내의 어떠한 레코드에서 "익스포트된" 필드가 "참" 으로 설정되면, 가상 회로 관리자는 다음과 같은 동작을 수행한다.

1. 필요한 변경을 함으로써 가상 회로를 FCP 타겟의 익스포트 테이블로 익스포트시킨다.

2. 만약 익스포트 동작 중에 어떠한 에러라도 발생하면, VC 레코드 내의 상태 필드는 설정될 것이고, "익스포트된" 필드는 정확한 상태로 남게 될 것이다. 만약 가상 회로가 익스포트되지 않았다면, 그 익스포트된 플래그는 "거짓"으로 설정될 것이다.

가상 회로 관리자는 또한 가상 회로 테이블 내의 "정지된" 필드에 대한 변형을 청취한다. 만약 VCT 내의 어떠한 레코드에서 "정지된" 필드가 "참"으로 설정되면, 가상 회로 관리자는 다음과 같은 동작을 수행한다.

1. VC 가 현재 익스포트되고 있다면, 익스포트되지 않을 것이며, 그 "익스포트된" 플래그는 거짓으로 설정될 것이다.

2. 가상 회로 내의 모든 가상 장치들은 정지 메시지를 받을 것이다.

3. 만약 정지 동작 중에 어떠한 에러라도 발생하면, VC 레코드 내의 상태 필드는 설정될 것이고, "정지된" 필드는 정확한 상태로 남게 될 것이고, 즉 만약 가상 회로가 정지되지 않았다면, 그 정지된 플래그는 "거짓"으로 설정될 것이다.

사용자 인터페이스

사용자 인터페이스는 본 발명에 따른 저장 서버를 구성하는데의 이용과 디스플레이를 위한 데이터 프로세싱 구조에 의해 생성될 수 있다. 이미지는 로고를 디스플레이하기 위한 필드, 서버의 새시에 관련되는 기본 정보를 디스플레이하기 위한 필드, 및 아이콘들의 세트를 가진 윈도우를 포함하고 있고, 이것은 선택되는 경우에 관리 애플리케이션을 개시한다. 하드웨어 및 소프트웨어를 관리하는데 제공되는 루틴들, 사용자 액세스를 조정하기 위한 루틴들, 및 서버 내의 장시간 실행 프로세스를 모니터하는 루틴들은 버튼에 의해 개시된다. 본 발명에 따르면, 서버에 부착된 호스트를 정의하는 기능, 익스포트된 LUN을 관리되는 자원들로 매핑하기 위한 기능, 및 관리되는 저장을 구성하기 위한 기능은 버튼에 의해 개시된다.

윈도우는 또한, 사용자 네임을 기입하는 필드와 패스워드를 기입하는 필드를 포함하고 있은 사용자 로그온 대화상자를 포함한다.

호스트 관리자

사용자는 호스트 관리자를 버튼을 이용하여 개시시킨다. 이러한 섹션은 저장 서버에 호스트(서버)를 정의하기 위한 자바 기반형 사용자 인터페이스(UI)를 설명한다. 관리 소프트웨어는 윈도우를 열고, 그 윈도우에는 구성 및 이용에 가용하게 된 각각의 호스트에 대해서 호스트 네임, 포트 넘버, 개시자 ID, 및 디스크립션을 수개의 컬럼으로 포함하는 테이블이 나타난다. 다른 필드들은, 네트워크 인터페이스 카드 식별자 및 고유의 호스트 식별자를 다른 컬럼에 포함한다. 바람직한 실시예에 있어서 고유의 호스트 식별자는, 파이버 채널 호스트에 대한 월드 와이드 넘버 값이다.

호스트 관리자는, LUN을 정의하는 프로세스를 용이하게 하기 위하여 NIC 포트 및 개시자 ID에 대해 네임과 디스크립션을 사용자가 할당할 수 있게 하는 저장 서버의 자바 기반형 관리 애플리케이션의 하위 성분이다. 일반적 기능은 마우스 팝업, 툴바 버튼, 및 활성화 메뉴를 통해서 가용되는데, 예로서 새로운 호스트 추가 버튼, 호스트 변경 버튼, 또는 호스트 삭제 버튼을 이용하여 기존의 호스트를 액세스하거나 새로운 호스트를 정의하게 된다.

사용자 인터페이스는 호스트 정보를 디스플레이하기 이해 메뉴 및 테이블 또는 다른 그래픽 구성을 이루어진다. 사용자가 호스트 관리자의 패널로 들어가는 때에, 그 테이블은 모든 기존의 호스트들에 상주하게 된다. 사용자는 테이블 내의 하나의 로우(row)를 선택한다. 각각의 로우는 하나의 호스트에 대한 정보를 포함하고 있다. 사용자는 그 호스트를 변형하거나

삭제하는 선택을 할 수도 있다. 호스트에 대한 '변형'이 선택되면, 사용자가 호스트 네임 및/또는 디스트립션을 변경할 수 있게 하는 대화상자가 나타나게 된다. 사용자는 맞음 버튼 또는 취소 버튼을 누르게 된다. 맞음 버튼을 누르면, 테이블의 변경이 이루어지고 그것은 서버로 전송된다. '삭제'가 선택되면, 대화상자에는 호스트가 삭제될 수도 있다는 내용의 라벨과 맞음 버튼 및 취소 버튼이 나타나게 된다. '추가'가 선택되면, 사용자가 어떤 호스트에 대안 모든 정보가 추가할 수 있게 된다는 대화상자가 나타나게 된다. '맞음'을 선택하면, 그 새로운 호스트에 대한 테이블에 새로운 로우가 추가될 것이면 추가가 서버에 의해 수행될 것이라는 대화상자가 나타나게 된다. 컬럼 라벨을 클릭하는 것에 의해 컬럼이 소팅될 것이다.

저장 매핑

사용자는 저장 요소를 디스플레이하기 위한 계층적 트리 디스플레이 구축을 보여주는 윈도우를 포함하는 이미지를 디스플레이하는 저장 관리자 루틴을 개시시킬 수 있다.

저장 요소는 트리 구조(예로서 미러에서 스트립들로 그리로 스트립들에서 디스크들로의 트리 구조)를 이용하여 정의된다. 이것은 사용자가 생각하고 있은 저장 방식과 일치하는 조직화된 방식으로 그들의 저장을 구축할 수 있게 한다.

저장 요소들의 대표적인 타입은 다음과 같다.

- 미러

- 스트립

- 외부 LUN

- 내부 디스크

- SSD

- 저장 콜렉션

- 저장 파티션

이러한 저장 요소들을 하나의 트리 내로 구축함으로써(예로서 마이크로소프트 익스플로러 류의 트리 디스플레이를 이용함), 사용자는 가상 회로 내에서의 이용을 위해 저장을 사전-구성할 수 있게 된다. 각각의 저장 요소는 파티션될 수 있고, 이들 파티션들은 서로 다른 방식으로 이용될 수도 있다. 예를들면, 스트립 세트가, 일종의 LUN으로 익스포트되는 파티션과 미러(미러 자체도 파티션될 수 있음)의 멤버로서 이용될 수 있는 다른 파티션으로 파티션될 수도 있다.

저장 요소가 파티션되면, 파티션들은 저장 콜렉션 내에 유지되며, 이는 파티션된 저장 요소의 자손으로 된다. 파티션되지 않은 저장 요소에는 이러한 파티션 콜렉션이 존재하지 않게 된다. 각각의 파티션은 어떤 타입의 저장인가에 의해 식별되게 되며, 이는 미러 파티션, 디스크 파티션 등의 파티션으로 된다. 주어진 저장 요소의 파티션들은, 그 저장 요소의 모든 파티션들이 가용한 상태로 될 때(즉, 전체 저장 요소가 사용되지 않는 때)까지는, 단일의 파티션으로 병합될 수 없다. 이를 하기 위해서, 사용자는 단지 사용되지 않는 파티션들만을 갖는 파티션된 저장 요소를 선택하게 되며, "파티션하지 않음" 버튼을 누르게 된다.

전용의 스페어들이 있다면, 이들 스페어가 전용되고 있는 저장 요소의 자손으로 될 저장 콜렉션 내에 이들 스페어가 유지될 것이다.

그래서, 각각의 저장 요소는, 파티션 콜렉션, 스페어 콜렉션, 및 부모 저장 요소를 구성하는 실제 저장 요소들을 자손으로서 잠정적으로 가지게 된다.

저장 관리자는, 어떤 의미에서는, 서버 상에의 모든 접속된 저장을 리스트 하는 저장 롤 호출 테이블로의 뷰(view) 이다. 각각의 가용한 저장 요소는 저장 트리의 헤드로서 보여지게 될 것이다. 예를들면, 미러가 가용 상태로 보여지만, 그 미러의 브랜치들을 나타내는 스트립들과 디스크들은 그들이 미러에 속하기 때문에 가용하지 않은 상태로 보여지게 된다. 언젠가

재이용되게 하기 위해서, 그들은 그 미러로부터(그래서 그 미러로부터 하강하는 저장 트리로부터) 제거될 필요가 있다. 일 실시예에 있어서, 이러한 것은 윈도우 NT 파일 익스플로러 프로그램 내에서 일 디렉토리로부터 다른 디렉토리로 이용되는 것과 같은 방식으로 드래그 앤 드롭에 의해 수행되게 된다.

모든 저장(이용된 것 및 이용되지 않은 것)의 트리가 본 실시예에 있어서 디스플레이의 좌반부에 도시되어 있고, 각각의 저장 요소는 그 타입을 나타내는 아이콘과 네임 또는 ID를 나타내는 어떤 것을 구비한다.

윈도우의 우반부에서 또는 다른 종래의 위치에서 트리 아래에는, 가용한(이용되지 않은) 저장의 리스트가 도시되어 있다. 이것은 다른 저장 요소 도는 가장 회로에 의해 이용되지 않는 모든 저장의 리스트이다. 명백히 이용되지 않은 대부분의 저장은 일반적 스페어 풀 내에 놓여지는 것이 기대된다. 이러한 가용한(이용하지 않은) 저장 리스트는 편의상 대부분 이용될 것으로 기대되어, 사용자가 새로운 저장 트리들을 그로부터 구축하게 되는 사용되지 않은 저장 요소들을 용이하게 발견할 수 있게 한다. 예를들면, 고체 상태 저장 장치(SSD) 파티션은 스트립 세트(RAID 0)에 의해 미러링되고 있다면, 파티션과 스트립 세트는 모두, 미러 내로 놓일 때까지, 그 가용한 리스트 내에서 가시 상태로 될 것이다. 미러가 2 개의 멤버로부터 생성되게 되면, 미러가 가용 회로 내로 채용될 때까지는, 그 가용한 리스트 내에서 보이는 것으로 될 것이다. 우측에는, 사용자가 마우스로 클릭하여 선택하는 트리 내의 어떤 저장 요소들인지간에 그 저장 요소들과 연관된 정보와 파라미터들이 있게 된다. 가용한 리스트 내에서 가시 상태에 있은 저장 요소가 선택되는 경우에, 그것은 그 가용한 리스트 및 저장 트리 모두에서 선택될 것이다.

변경 기능 뿐만아니라 추가 기능 및 삭제 기능은 엔트리들을 생성하거나 제거하기 위하여 제공되어, 사용자 인터페이스에 의해 제공되는 툴을 이용하여 사용자는 "소유자", " 최근에 서비스됨" 또는 디스크립션" 등의 것을 변경할 수 있다. 사용자는 그들이 추가하고 있은 것(미러, 스트립, 디스크 등)을 지정하게 될 것이고, 적절한 세트의 제어가 그들에게 주어지게 될 것이다.

내부 디스크 및 외부 LUN 에 대해, 사용자는 네임, 크기, 추정적 제작자 등의 것을 지정하게 될 것이다. 내부 디스크를 지정하는 것은, 디스크가 하드웨어의 일부이고 그래서 자동적으로 삭제될 수도 있기 때문에, 특별한 경우에 해당한다, 사용자가 디스크를 부가하는 경우에만, 그들이 추후에 부착될 어떤 하드웨어에 대해 위치 홀더를 놓여지게 한다. 이것은 SSD 보드들에 대해서 수행된다.

RAID 배열들에 대해서 일어나는 것은, 그들이 주어진 RAID 레벨에 대한 배열을 생성하기를 원한다는 것을 사용자가 지정한다는 것이고, 그 다음에 그 배열의 멤버들이 될 저장 요소들을 사용자가 지정할 수 있게 된다는 것이다. 이러한 지정은 가용한 저장 요소들의 리스트 내에서 엔트리들을 선택함으로써 수행될 가능성이 있고, 배열의 용량은 멤버들의 용량에 의해 결정될 것이다. 그 다음에, 그 배열의 멤버들로서 이용되는 저장 요소들은 가용하지 않은 것으로 태그될 것이며(그들이 배열의 일부이므로), 그 배열 자체는 가용한 저장의 리스트에 추가될 것이다. 각각의 RAID 배열은 또한, 멤버들 중의 하나가 실패하는 경우에, 그 배열에 할당되는 전용 스페어들을 구비하고 있다.

저장 요소들은 또한 파티션될 수 있고, 이것은 파티션될 저장 요소들을 선택하고 사용자가 원하는 크기의 천크를 지정함으로써 수행된다. 만약 그 요소가 이전에 파티션되지 않았다면, 2개의 파티션이 생성되는 결과로 되며, 사용자가 요청하는 파티션과 저장의 나머지 부분(즉, 사용되지 않은 부분)이 되는 파티션이 그것들이다. 추가적인 파티션들은 그들이 생성될 때에 그 이용되지 않은 부분으로 나온다.

각각의 저장 요소에 대한 상세한 디스플레이는 우리가 가용하게 가지는 것만큼 많은 정보를 보여줄 것이다. 바람직한 시스템에서 보여지는 것들 중의 하나는 특정의 저장 요소의 파티션들이 크기와 위치와 같은 것이다.

LUN 매핑

사용자 인터페이스의 어떤 버튼을 이용하면, LUN 맵이 구축된다. LUN(Logical Unit Loop) 맵은 필연적으로 LUN 들과 그에 대응하는 데이터의 리스트이다. 이들은 네임들과 디스크립션들로서 디스플레이될 것이다. 어떤 주어진 LUN 과 연관되어 있는 VC(Virtual Circuit)는 이러한 디스플레이 상에 나타내어 진다. 이것은, 사용자가 그 LUN 맵과 요구 세부사항으로부터 하나의 엔트리를 선택하는 경우에 가시 상태로 될 것이다.

LUN 맵은 네임, 디스크립션, 또는 그밖의 필드들에 의해 기존의 리스트를 나타낼 것이다. 필드에는 다음과 같은 것이 포함된다.

- 네임

- 디스크립션

- 익스포트된 상태

- 호스트

- 저장 요소(들)

LUN 맵은 다음과 같은 것을 허락한다.

- 다양한 필드들을 기초하는 소팅

- 필드들에 기초한 필터링. 이것은 하나 이상의 LUN 이 일 시점에서 동작하는 경우에만 필요하게 된다(예로서, 인에블/디스에이블).

- 삭제 또는 에디팅/뷰잉을 위한 LUN 의 선택

- 새로운 LUN을 정의하고 추가함

- 기존 LUN 들의 임포팅(하드웨어드 시작 상에 있은 "Learn Mode"를 통하여 이루어짐)

- LUM 상에 멤버를 추가하고, 핫 카피 미러 프로세스를 시작함

- LUN을 익스포팅하고 익스포팅하지 않음. 이것은 호스트로부터의 데이터의 흐름을 기본적으로 시작 및 중지시키게 될 것이다.

가상 회로들은, 저장 트리로서 또는 호스트에 접속된 대화상자와 같은 다른 그래픽 구현체로서 사용자에게 정의되며, 이것은 버튼을 이용하여 개시된다. 대화상자는 LUN 네임의 엔트리에 대한 필드, 디스크립션의 엔트리에 대한 필드, 타겟 ID의 엔트리에 대한 필드, 및 익스포트된 LUN 에 대한 정보의 엔트리에 대한 필드를 포함하고 있다. 팝업 메뉴는 가용한 리스트에 대한 호스트 버튼 및 가용한 저장 요소들의 리스트에 대한 저장 버튼을 이용하여 개시된다. 캐시 선택 버튼은 체크 상자로서 구현된다.

저장 트리는 실제적으로는 저장 요소들(예로서, 몇 개의 스트립 세트들로 이루어진 미러이고, 이 미러는 다시 몇 개의 디스크로 이루어짐)의 트리이다. 호스트는 실제적으로는 NIC 상에 특정의 포트에 접속되어 있은 특정의 개시자 ID를 가진 서버이다. 이것은 가용한 저장의 어떤 량을 나타내는 미리-정의된 호스트 및 미리-정의된 저장에서의 선택을 통하여 사용자에 의해서 정의될 것이다.

캐시의 이용은 체크 상자를 이용하여 "온" 또는 "오프"로 제한된다. 대체적인 시스템들은 캐시 크기 또는 캐시 알고리즘의 지정을 위한 툴을 제공한다. 캐시 이용은 가상 회로를 따라 데이터의 흐름을 인터럽트 하지 않고도 플라이(fly) 상에서 턴온 또는 턴오프될 수 있다. LUN 이 생성된 때에, 디폴트 값은 "온"이 된다.

LUN 맵의 일 실시예는 가상 회로들을 생성하는데에 필수적인 기능을 가질 것이다. 이것은 2개의 컬럼(호스트에 대해서 하나의 컬럼과 저장에 대해서 하나의 컬럼)을 가진 복수 컬럼 테이블로서 이루어질 것이다. LUN 의 생성는 그것을 자동적으로 익스포트하고, 가용한 기능들에는 "추가', "변경" 및 "삭제"가 포함된다.

LUN 맵 디스플레이는, 일반적으로 기존의 LUN 에 대해서 수행되기 때문에, 핫 카피 미러들이 정의되는 위치이다. 이 프로세스는 LUN을 선택하고, 다음에 저장 트리를 선택하여, 미러의 부가 또는 기존의 미러의 확장을 통해서 기존의 저장 트리에 부가시키는 것이다.

<u>데이터 이동 지원</u>

도 18 은 통신 링크(14)에 걸쳐 있은 제 1 저장 장치(11)와 통신 링크(15)에 걸쳐 있은 제 2 저장 장치(12)에 연결된 저장 서버(10)를 가긴 저장 네트워크 내에 데이터 흐름의 3 단계를 도시하는 개략도이다. 중간 장치(10)는, 통신 링크(13)를 통하여 클라이언트 프로세서에 연결되어 있고, 그것에 의해서 논리적 어드레스(LUN A)에서 데이터를 엑세스하기 위한 요구를 수신한다.

저장 서버(10)는, 버퍼로서 이용되는 비휴발성 캐시 메모리와 같은 메모리와, 링크(13) 상에서 수신된 데이터 액세스 요구들을 링크들(14, 15)에 걸쳐서 액세스 가능한 저장 장치들로 전송하는 전송 자원들을 포함하고 있다. 또한, 저장 서버는 본 발명에 따른 핫 카피 프로세스들을 관리하는 로직 엔진을 포함하고 있다. 이러한 프로세스는 도 18 에 도시된 3개의 스테이지를 고려함으로써 이해될 수 있다.

스테이지 1 에 있어서, 저장 서버(10)는, 전송의 데이터 세트 주체를 식별하고 링크(13)에의 인터페이스 상에서 수신된 모든 데이터 액세스 요구들을, 요구의 데이터 세트 주체를 저장하고 있는 장치(11)에 접속하기 위한 링크(14)로 맵핑한다. 저장 서버는 핫 카피 프로세스를 개시하고 타겟 장치(본 실시예에서는 장치(12))를 식별하는 제어 신호를 수신한다. 이 단계는, 저장 서버(10)를 통하여 제 1 장치(11)로부터 제 2 장치(12)로 백그라운드 프로세스로서 데이터 세트가 전송되는 동안에, 단계 2 로부터 개시한다. 파라미터들은 저장 서버(10) 상에 유지되는데, 이들은 데이터 세트의 전송의 진행을 나타내고, 클라이언트 프로세서로부터의 요구들에 관하여 백그라운드 핫 카피 프로세스의 상대적 우선 순위를 나타낸다. 핫 카피 프로세스 동안에, 데이터 액세스 요구들은 핫 카피의 진행 및 요구의 타입에 무관하게 제 1 장치(11) 및 제 2 장치(12)로 매핑된다. 또한, 저장 서버는 핫 카피 프로세스에 우선순위를 할당하기 위한 자원들을 포함하고 있다. 핫 카피 프로세스의 우선 순위가 낮으면, 클라이언트 프로세서는 데이터 액세스 요구들의 수행에 있어서 상당한 지연을 격게 되지 않는다. 핫 카피 프로세스의 우선 순위가 높으면, 클라이언트 프로세서는 데이터 액세스 요구들의 수행에 있어서 어느 정도의 지연을 격게 되지만, 핫 카피 프로세스는 매우 신속하게 완료될 것이다.

데이터 세트의 전송이 완료됨과 동시에, 스테이지 3 에 도달한다. 단계 3에서, 데이터 세트에 어드레싱된 클라이언트 프로세서로부터의 데이터 액세스 요구들은 통신 링크(15)에 걸쳐 있는 제 2 장치(12)에게 경로지정된다. 저장 장치(11)는 네트워크로부터 제거되거나, 또는 다른 목적으로 이용될 수도 있다.

바람직한 실시예에 있어서, 저장 서버(10)는 상술한 바와같은 저장 도메인 관리자를 구비하고 있다.

저장 장치(11, 12)는 단일 저장 유니트 내에 독립적인 장치 또는 논리적 파티션들을 포함할 수도 있다. 이 경우에, 핫 카피 프로세스는 저장 유니트 내에서 어떤 어드레스로부터 다른 어드레스로 데이터가 이동되는 결과를 준다.

도 19 내지 도 22 는 상술한 지능적 네트워크 서버에서 실행을 위한 핫 카피 프로세스의 소프트웨어 구현에 대한 다양한 태양을 나타내고 있다. 핫 카피 프로세스에 이용되는 다른 저장 서버들에 있어서, 이러한 구현에 대한 변형들이 특정의 시스템에 적응되도록 이루어지게 된다. 가상 회로, 영구적 테이블 저장, 및 사용자 인터페이스 구조의 성분들의 상세한 내역은 첨부된 도면들을 참조하여 설명된다.

도 19 는 핫 카피 프로세스에서 이용되는 기본적 데이터 구조들을 도시하고 있다. 제 1 구조(350)는 유틸리티 요구 구조라고 불리운다. 제 2 구조(351)는 유틸리티 구조라고 불리운다. 제 3 구조(352)는 멤버 구조라고 불리운다. 멤버 구조(352)는 특정의 가상 회로 및 그의 상태를 식별하도록 설정된다. 멤버 구조(352)는 가상 회로 식별자(VD ID), 그 가상 회로에 의해 현재 처리되고 있는 데이터의 블록에 대한 블록 넘버를 유지하고 있는 논리적 블록 어드레스(LBA), 그 가상 회로에 대해 큐되어 있던 요구들의 카운트, 및 상태 파라미터 등의 파라미터들을 포함하고 있다.

유틸리티 구조(351)는 실행되고 있은 현재의 유틸리티(이경우에는 핫 카피 유틸리티)에 관련된 파라미터들을 유지하고 있다. 그것은 소스 데이터 세트의 식별자(소스 ID), 핫 카피 프로세스에 대한 목적지 저장 장치 또는 장치들의 식별자(목적지 ID), 유틸리티와 연동하여 실행될 요구들의 큐, 처리되고 있는 현재의 블록을 지시하는 파라미터들, 및 그 크기 등의 파라미터들을 저장하고 있다.

유틸리티 요구 구조(350)는 프로세스와 관련된 다양한 파라미터들을 포함하고, 핫 카피 프로세스에 대한 요구를 반송하고 있다. 이것은 예를들면, 요구의 상태를 지시하는 파라미터(상태), 그 요구를 지원하는 다양한 플래그들, 대응하는 유틸리티 구조에 대한 포인터, 클라이언트 프로세서들로부터의 입력/출력 요구들을 관한 요구의 우선순위를 지시하는 파라미터, 소스 내의 데이터 세트를 식별하는 소스 마스크, 및 핫 카피 프로세스가 데이터 세트를 카피하여 보내게 될 목적지 장치 내의

위치를 식별하는 목적지 마스크 등의 파라미터들을 포함하고 있다. 일 실시예에 있어서, 단일의 핫 카피 요구에 대한 복수의 목적지 마스크가 있다. 도 19 에 도시된 바와 같이, 논리적 블록 어드레스(LBA)는 처리되고 있은 데이터 세트 내의 현재의 블록의 데이터에 대해서 유틸리티 요구 구조 내에 유지되며, 그것을 또한 멤버 구조에 유지된다.

핫 카피 프로세스를 개시하기 위해, 사용자 입력이 수용되어, 유틸리티 요구 구조의 생성이 일어난다. 저장 서버 내의 영구적 테이블 저장은 그 구조와 함께 갱신되고, 그 데이터 세트와 관련된 소스 장치, 목적지 장치, 및 가상 회로들의 상태가 체크되고, 드라이버들은 핫 카피 프로세스를 개시하도록 셋업되며, 상태 파라미터들은 다양한 데이터 구조들로 세트된다. 핫 카피 프로세스의 진행은, 실패의 경우에 영구적 테이블 저장 내에 유지된다. 이 경우에, 핫 카피 프로세스는 서버 내의 다른 자원들을 이용하고, 영구적 테이블 저장에 저장되어 있은 상태 정보 및 데이터 구조들의 카피를 이용하여 재시작될 수도 있다. RAID 모니터들과 같은 시스템 내의 다른 드라이버들은 핫 카피 프로세스를 통지받는다. 그 요구는 멤버 구조에 큐된다.

셋업이 완료되면, 핫 카피 프로세스를 지원하는 입력 및 출력 프로세스들이 개시된다. 핫 카피 프로세스를 지원하는 입력 프로세스 및 출력 프로세스의 상대적 우선 순위는, 클라이언트 프로세서가 동일한 데이터 세트에 대한 입력 요구 및 출력 요구를 실행하고 있다는 조건하에서, 핫 카피 프로세스에 대한 진행 속도를 판정한다. 바람직한 시스템에 있어서, 클라이언트 프로세서로부터의 입력 요구 및 출력 요구가 우선 실행된다. 핫 카피 프로세스가 지원하는 블록 전송이 실행되고 있은 경우에, 클라이언트 프로세서로부터 입력 요구 또는 출력 요구가 수신되면, 블록 전송은 오토믹 동작으로서 완료되고, 그 다음에 클라이언트 프로세서 요구가 서비스된다. 대체적인 시스템에 있어서는, 프로세스들의 우선 순위를 관리하기 위해 다른 기법들이 이용된다.

핫 카피를 실행하기 위한 기본적 프로세스가 도 20에 도시되어 있다. 그 프로세스는 멤버 구조에 대한 큐의 상부에 도달한 핫 카피 요구로부터 시작한다(단계 360). 그 프로세스는 블록 전송을 지원하는 저장 서버 내에 버퍼를 할당한다(단계 361). 그 버퍼로 데이터 세트 내의 제 1 블록의 카피를 이동시키기 위해, 메시지가 발송된다(단계 362). 현재의 블록은 핫 카피 프로세스에 대한 우선 순위 세트에 따라 버퍼로 이동된다(단계 363). 블록의 이동은, 저장 서버 내에서 다수의 프로세스에 의해 액세스를 제어하는 적절한 메모리 로크 트랜잭션들을 이용하여 이루어진다. 다음에, 버퍼로부터 목적지 또는 목적지들로 블록의 카피를 이동시키기 위해, 메시지가 발송된다(단계 364). 그 블록은 핫 카피 프로세스에 대한 우선 순위 세트에 따라 목적지 또는 목적지들로 이동된다(단계 365). 블록이 이동되면, 그 프로세스를 지원하는 영구적 테이블 저장 및 지역적 데이터 구조들이, 핫 카피의 진행을 지시하는 상태 정보에 의해서 갱신된다(단계 366). 그 프로세스는 데이터 세트 내의 최근 블록이 카피되었는지를 판정한다(단계 367). 만약 그렇지 않다면, 후속하는 블록의 카피를 그 버퍼로 이동시키기 위해, 메시지가 발송된다(단계 368). 프로세스는 단계 363 으로 귀환되어, 목적지 또는 목적지들로 데이터 세트 내의 블록들을 이동시키는 것을 계속한다. 만약 단계 367에서 데이터 세트 내의 최근 블록이 목적지 또는 목적지들로 성공적으로 이동되었다면, 그 프로세스는 종료한다(단계 369)

본 발명의 일 실시예에 따르면, 다수의 목적지와 관련된 핫 카피 프로세스에 대해서, 이용되고 있는 목적지들의 그룹의 멤버 또는 멤버들이 그 프로세스 도중에 실패하는 것이 가능하다. 이경우에, 그 프로세스는, 계속되는 프로세스의 지원에 의해 적절한 테이블들을 갱신하여 동작을 계속하는 목적지 또는 목적지들과 함께 계속할 수 있다.

그래서, 핫 카피 특징은, 대체 드라이브에 대해 아직 다운되지 않은 개개의 멤버로부터의 데이터 세트를 카피하는데에 이용된다. 데이터 세트는 저장 장치의 전체 내용 또는 저장 장치의 내용의 일부를 포함할 수도 있다. 핫 카피 특징은, 적절한 상태 및 파라미터 관리를 통해서 어떤 레벨의 RAID 배열들에서도 이용될 수 있다.

핫 카피 파라미터는 프로세스의 우선 순위, 소스 멤버 드라이버, 및 목적지 식별자를 포함하고 있다. 핫 카피 요구는 소스 멤버 식별자, 목적지 멤버 식별자, 카피 블록 크기, 및 카피 빈도 또는 우선순위를 포함한다. 핫 카피들은 우선 수위에 따라 일회에 일 블록 크기로 수행된다. 현재의 블록 위치는 상술한 바와 같이 배열 구성 데이터 내에 유지된다. 핫 카피 프로세스는 통상적인 입력 프로세스 및 출력 프로세스와 동시에 수행된다. 핫 카피되고 있는 드라이브에 대한 기록은 양쪽 드라이브에 기록된다. 이러한 방식에 있어서, 만약 핫 카피가 실패되면 본래의 소스 멤버는 여전히 유효하게 된다. 핫 카피가 완료되면, 본래의 소스 멤버는 시스템 관리자 프로그램들에 의해서 그 배열로부터 제거되고 이용될 수 없는 것으로서 지정된다. 마찬가지로, 일 실시예에 있어서, 데이터 세트를 지원하는 가상 장치는 새로운 목적지로 포인트하기 위해 갱신된다.

도 21 및 도 22 는, 핫 카피 프로세스가 실행되고 있은 동안에, 클라이언트 프로세서들에 의해 발송된 데이터 액세스 요구들을 관리하는 저장 서버 내에서 실행되는 프로세스를 나타낸다. 데이터 액세스 요구들은, 판독 요구들, 기록 요구들, 그와 동등한 변형들을 포함하는 복수의 타입 중의 하나를 가진다. 다른 요구들은, 데이터 채널의 관리를 지원하는 요구 등을 포함하고 있다. 도 21 에 있어서, 기록 요구를 처리하는 프로세스가 도시되어 있다.

기록 요구가 큐의 상부에 도달하면, 그 프로세스가 시작된다(단계 380). 그 프로세스는, 그 기록 요구가 현재의 핫 카피 프로세스의 데이터 세트 주체 내에 위치를 식별하는 지를 판정한다(단계 381). 만약 핫 카피되고 있은 데이터 세트 내에 있다면, 그 프로세스는, 기록 요구가 지시된 블록이 이미 목적지로 카피되어 있는지를 판정한다(단계 382). 만약 그것이 카피되었다면, 데이터 세트가 본래 유지되어 있었던 저장 장치에 대해 그리고 목적지 저장 장치 또는 장치들에 대해 기록을 하라는 메시지가 발송된다(단계 383). 다음, 입력 요구 및 출력 요구에 대한 우선 순위에 따라 데이터가 이동되고(단계 384), 그 프로세스는 종료된다(단계 385)

만약 단계 381에서 그 요구가 데이터 세트 내에 있지 않다면, 데이터 세트의 소스에 기록을 실행하라는 메시지가 발송된다(단계 386), 이때에, 프로세스 흐름은 단계 384 로 간다. 마찬가지로, 만약 단계 382에서 기록의 위치 주체가 이미 카피되지 않았다면, 소소 장치로 기록을 실행하라는 메시지가 발송된다(386).

도 22 는 핫 카피 동안에 발생하는 판독 요구의 처리를 나타낸다. 그 프로세스는, 판독 요구가 가상 장치에 대한 큐의 상부에 도달하는 때에 시작한다(단계 390). 그 프로세스는 먼저, 그 판독이 핫 카피의 데이터 세트 주체 내에 있는지를 판정한다(단계 391). 만약 판독이 데이터 세트 내에 있다면, 그 프로세스는, 그 기록이 목적지 또는 목적지들로 이미 카피된 블록 있는지를 판정한다(단계 392). 만약 그 기록이 목적지로 이미 카피된 블록 내에서 발견되면, 새로운 위치로부터 데이터를 판독하라는 메시지를 발송한다(단계 393). 대체적인 시스템에 있어서, 그 시스템 내의 데이터 트래픽의 관리에 영향을 주는 신뢰도, 속도, 및 그 밖의 인자들에 의존하여, 소스 장치로부터 또는 소스 장치 및 목적지 장치 모두로부터 판독이 실행될 수도 있다. 단계 393 후에, 데이터는, 클라이언트 프로세서의 데이터 액세스 요구에 대한 우선 순위 따라 요구자에게 반환된다(단계 394). 다음에, 그 프로세스는 종료한다(단계 395).

만약 단계 391에서, 판독 요구가 핫 카피의 데이터 세트 주체 내에 있지 않다라고 판정되면, 소스 장치를 판독하라는 메시지가 발송된다(단계 396). 마찬가지로 만약 단계 392에서, 판독 요구가 목적지로 이미 카피된 블록을 어드레싱하고 있다라고 판정되면, 소스 장치로부터 데이터를 판독하라는 메시지가 발송된다(단계 396). 단계 394 후에, 그 프로세스는 단계 394로 복귀한다.

특정의 블록이 저장 서버 버퍼를 통해 이동하는 프로세스에 있는 때에 그 블록 내의 데이터에 관하여 판독 요구 또는 기록 요구가 발생하는 경우에는, 데이터 로크 알고리즘이 그 요구들의 처리를 관리하기 위해 이용된다. 그래서, 예를들면, 만약 판독 요구 또는 기록 요구가 수신되는 때에 논리적 블록이 핫 카피 프로세서의 지원에 의해 로크되면, 클라이언트 프로세스는 데이터가 로크되었기 때문에 판독 요구 또는 기록 요구가 재이용되었다라는 통지를 수신할 것이다. 클라이언트 프로세서에 대한 더 높은 우선순위를 지원하는 대체적인 실시예에 있어서, 핫 카피의 지원에 의해 버퍼 내에 유지된 블록이 삭제되는 때에 판독 요구 또는 기록 요구는 계속되도록 허락될 수도 있고, 핫 카피의 상태는 블록이 이동되지 않은 것을 지시하도록 재설정된다. 다양한 다른 데이터 로크 알고리즘은 특정의 구현체에 대해 필요에 따라 이용될 수도 있다.

타겟 에뮬레이션

도 1a , 도 2 및 도 3 에 도시된 구성에 있어서, 저장 서버는 데이터의 사용자들 사이의 중간 장치 및 데이터를 저장하는 저장 도메인 내의 저장 장치들의 역할을 한다. 본 실시예에 있어서, 상속 저장 장치들, 즉 서버가 중간 장치로서 삽입되기 전의 위치의 장치들을 지원하기 위해, 서버에는 상속 저장 장치를 에뮬레이팅하기 위한 자원들이 제공된다. 이러한 방식으로, 서버가 상속 자치와 데이터의 사용자 사이에 삽입되는 경우에, 서버는 사용자와 상속 장치 사이에 이용되고 있은 저장 채널 프로토콜에 따라 상속 장치의 논리적 어드레스를 가상적으로 확득한다. 그 다음에, 저장 서버는, 상속 장치에 어드레싱되고 저장 서버가 수신하는 프로토콜에 따라 모든 요구들에 응답하는 역할을 한다. 더나아가, 저장 서버는 상속 장치로부터 필요한 만큼의 구성 정보를 검색하고 정보를 지역적 메모리에 저장하여, 사용자가 상속 장치 내에서 기대하는 것으로 구성되어 있은 상태 및 구성 정보가 서버 내의 지역적 자원들을 이용하여 제공된다. 이것은 서버와 상속 장치 사이에서의 통신을 절약하고, 서버가 저장 채널 프로토콜에 따라 상속 장치의 동작으로 스푸프할 수 있도록 하여, 사용자의 재구성이 필요없어 지거나, 저장 네트워크에의 서버의 추가를 더욱 단순화시킨다.

발명의 효과

결론

저장 영역 네트워킹(SAN)은 새로운 저장 센트릭-컴퓨팅 아키텍처이다. 파이버 채널 기반 저장 서브시스템 및 네트워크 성분들의 가용도가 대부분 주어지는, SAN은 고속의 데이터 액세스 및 이동, 더욱 유연성있는 물리적 구성, 저장 용량의 이용성의 향상, 중앙집중화된 저장 관리, 온라인 저장 장원의 배치 및 재구성, 및 이형적 환경에 대한 지원 등을 가능케 한다.

종래의 "직접 부착 저장" 에 있어서, 저장 자원들은 단일 서버에 대해서만 고속의 직접 물리적 경로를 가지고 있었다. 다른 모든 서버들은 LAN을 통하여 간접적으로만 저장 자원에 더 늦은 속도의 액세스하게 되었다. 저장 영역 네트워크는 "네트워킹된" 토폴로지로서 서버 각각으로부터 저장 자원 각각으로 직접 고속 액세스 경로를 제공함으로써(파이버 채널을 통하여) 그것을 변경시킨다. 네트워크 아키텍처의 도입은 또한, 특정의 서버로부터 저장 자원들을 탈결합(decoupling)시키고 그들이 서버측 자원들에 대해 최소한도의 충격을 주도록 구성되고 관리되도록 잠정적으로 허용함으로써, 저장 구성 유연성을 상당히 향상시킨다.

SAN이 그 유연성을 어드레싱하기 위한 적합한 토폴로지 및 현재의 환경에서 데이터 액세스 요구사항을 제공하면서, SAN 토폴로지 자체는 비즈니스 이슈들을 적절하게 어드레싱하지 않는다. SAN 을 통하여 서버들과 저장 자원 사이에의 물리적 접속을 단순히 제공하면서, 스위치들, 허브들, 또는 라우터 등의 조직 성분들은 SAN을 통해서 기대되는 것 모두를 이루는데에 충분하지 않지만, SAN 조직은 필요한 안정성있고 중앙집중화된 저장 관리 능력을 계층화하는 하드웨어 하부구조(infrastructure)를 제공한다. 함께 추진되는 이러한 두 발전은, 새로운 환경에 있어서의 비즈니스 목적을 실현하는데에 제공되어야 하는 필수적 데이터에 유연성 및 산재적 액세스를 제공할 수 있다.

SAN 하드웨어 하부구조의 상위에 있은 계층에 필요한 관리 능력은 저장 도메인 관리이다. 최적의 저장 유연성 및 고성능 액세스를 성취하기 위해, 저장 도메인 관리는 서버들이나 저장 장치들 내에서보다는 SAN 자체 내에서 가장 효과적으로 위치된다. 서버 기반형 자원 및 저장 기반형 자원 방식들은, 그들이 서버측과 저장측 모두에 대해 이형성(heterogeneity)을 적절하게 지원하지 않기 때문에, 차선적으로 최적이라 할 수 있다.

저장 도메인 관리는 중앙집중화되고 관리 능력을 확보하게 하여, 기존의 SAN 하드웨어 하부구조의 상위의 계층이 고도의 성능, 고도의 가용성, 및 이형적 환경들에 대한 진보된 저장 관리 기능성을 제공하도록 하게 한다. 저장 도메인 관리의 목적은, 상속 및 새로운 설비, 서버들과 저장 자원들로부터의 오프랜드 SAN과 저장 관리 태스크들, 및 모든 SAN 성분들에 걸쳐 지배력을 가질 수 있은 호스트 SAN 기반형 애플리케이션들을 통합할 수 있은 로버스트 SAN 조직의 코어를 형성하는 것이다. SAN은 저장 도메인 관리를 이용하지 않고 구축될 수 있지만, 최적화된 이형적 SAN 환경을 생성하고 관리하는 것은 중대한 관리 능력을 요구한다.

저장 도메인 관리의 기초는 다음과 같다.

- 이형적 상호동작가능성

- 안정화되고 중앙집중화된 관리

- 스케일링 가능성 및 고도의 성능

- 엔터프라이즈-클래스 신뢰성, 가용성, 및 서비스 가능성

- 지능적 목적-구축형 플랫폼

저장 도메인 관리에 대한 개시는, 비즈니스 문제를 어드레스하는 SAN을 통해서 기대할 수 있는 것 전부를 고객이 실현할 수 있게 한다.

모든 서버 및 저장을 통합함으로써, 오늘날의 새로운 비즈니스 환경에서 흔히 있는 기업흡수합병(M&A) 뿐만아니라 이형성은 기업 환경에서 필수적 요소이다. 단일의 공급자 생산 라인에 대한 SAN 기능성을 제공하는 제품 세트는, 고객이 SAN을 통해서 기대할 수 있는 것 전부를 성취하는데에 충분하지 않다. 고객은 그들이 추가하는 상속 설비 내에 투자를 유지하는 능력을 필요로 하고 새로운 서버 및 저장 산물들에서 이점을 가지며, 그래서 저장 도메인 관리자는 파이버 채널 과 SCSI 부착을 최소로 지원해야 한다. 저장 도메인 관리자가 그들이 도입될 때보다 더 새로운 기법을 수용하기 위해 과도한 시간을 필요로 하게 되기 때문에, 플랫폼은 더 강화된 다중 프로토콜 접속성에 잘 정의된 성장 경로를 제공할 수 있다.

SAN은, 특히, 백업/복구 및 심각한 피해복구의 분야에서 전통적인 "직접 부착" 저장 아키텍처에 관련된 저장 관리 태스크들을 최소화는 것을 중심적으로 관리할 수 있는 대량의 가상화된 저장 풀을 생성한다. SAN은 모든 서버로부터 모든 저장으로 물리적 액세스 경로를 효율적으로 제공하지만 모든 저장이 모든 서버에 논리적으로 액세스가능하지는 않기 때문에, 로버스트한 방식의 보안이 역점 사항이 될 것이다. SAN 조직 공급자들은 "구역(zone)"의 논리적 정의를 통해서 이것을 하며, 각각의 서버는 그 구역 내에 있도록 정의된 데이터를 액세스할 수만 있다. 명백히, 안정된 구역들, 즉 저장 "도메인" 을

정의하는 능력은 저장 도메인 관리자의 일면이다. 포트 레벨에서보다는 LUN 레벨에서 하나의 구역 내로의 포함을 정의하는 것과 같은 도메인의 향상된 구역화의 정의는, 저장 재원(asset) 이용 과도 시간을 향상시키는 상당한 추가적 유용성을 제공한다.

저장 도메인 관리자는, 공급자에 상관없이, 모든 부착된 서버들 및 저장에 걸쳐, 단일의 관리 인터페이스로부터 지배력을 가질 수 있는 중앙집중화된 저장 관리 능력들의 포괄적 세트를 제공한다. 중앙 위치로부터, 시스템 주관자는 이형적 저장 자원들 사이에 데이터의 이동 또는 미러링을 제어할 수도 있고, 서로 다른 이형적 저장 자원 과도 시간에 걸쳐 이들 능력을 동적으로 지배력을 가질 수 있다. 이것은 상당한 비용을 절감하게 하고 시스템의 복잡도를 단순화시키는 결과를 준다. 스케일이 가능하고 지능적인 플랫폼으로서, 저장 도메인 관리자는 완전한 중앙 위치에 상주하여, 모든 부착된 서버 및 저장 자원들에 걸쳐 지배력을 가질 수 있는 저장 관리 기능들을 주재한다.

새로운 비즈니스 환경에서 유발되는 저장 성장 속도에 주어지는, 특정의 SAN 환경은 그 라이프 타임 동안에 저장 용량의 크기를 2차수 정도로(수백배 정도로) 증가시키는 것을 용이하게 할 수도 있다. SAN에서 지능성의 중점으로서, 저장 도메인 관리자는 부하관련 성능 저하없이 상당한 증가량을 수용할 수 있다. 넓은 성능 범위에 대한 유연하고 저렴한 스케일링 가능성을 확보하도록 구성이 성장함에 따라, 지능성이 추가되어야 한다.

지능적 플랫폼 내에 상당한 량의 데이터를 캐시할 수 있는 능력은, SAN 구성을 최적화시켜 애플리케이션 특정 환경들에서의 성능 향상을 성취한다. 예를들면, 파일 시스템 저널들 및 데이터 베이스 인덱스들 또는 로그들과 같은 "핫 스포트들"이 저장 도메인 관리자 자체에 고속 저장으로 캐시될 수 있다면, 이것은 저장 도메인 관리자 없이 구축된 종래의 SAN 구성들에 비해 메시지 경로의 대기 시간(latency)을 상당히 최소화하게 된다. 상당한 량의 온보드 저장이 주어지는, 전체 데이터 베이스들 및 파일 시스템들은 효과적으로 캐시되어 큰 성능 향상을 성취할 수 있다. 온보드 저장 능력은 또한 이동 및 다른 데이터 이동 태스크들 동안에 데이터를 스테이지하는데에 있어서 중요하게 된다.

상술한 바와 같이, SAN 으로의 이동에 대한 주요 이유들 중의 하나는 전체적인 데이터 액세스 가능성을 향상시키는 것이다. 단일의 실패 포인트가 새로운 저장 아키텍처로의 이동의 결과로서 도입되면, 많은 잠재적인 이점이 실현될 수 없다. 이러한 이유로, 데이터 자체 뿐만아니라 그 데이터에 대한 액세스 경로가 모든 시간에 가용해야 한다. 실패에 기인하는 다운 시간을 최소화하는 것은, 자동적 I/O 경로 페일오버, 논리적 핫 스페어링 및 플러그 가능, 핫 스왑 가능한 성분들과 같은 능력들과 상대적인 내부 성분들의 이용을 통해 어드레스되어야 한다. 다운 시간은, 온라인 펌웨어 갱신, 동적 하드웨어 및 소프트웨어 재구성 및 고성능 백그라운드 데이터 이동과 같은 온라인 관리 능력들을 통해 더욱 최소화되어야 한다.

최고 레벨의 성능을 확보하기 위해, 바람직한 저장 도메인 관리자는, 그것에 요청되는 저장-관계형 태스크들에 대해 특별히 최적화된 지능적 및 목적-구축형 플랫폼이다. 이 플랫폼은 상당한 지역적 프로세싱 능력을 지원하여, 데이터 이동 및 저장 관리 애플리케이션 실행에 필요한 지역적 고속 저장에 의해 지지되는 넓은 범위의 저장 관리 태스크들을 수행한다.

지능적 저장 서버로서 이용되고 있는 범용적 플랫폼과 비교하여, 목적-구축형 플랫폼은, 더욱 신속하고 더욱 결정론적인 응답 시간을 가지며, 더 효율적인 I/O 경로 코드에 대한 실시간 운영 시스템에 제공하여, 애플리케이션 엔진보다 데이터 이동자 엔진으로서 최적화된 운영체제 커넬 및 메시지 대기 시간들을 최소화한다. 통합적 경로 페일오버, 온라인 관리 및 동적 재구성과 같은 고도의 가용성 특성들이 코어 운영 시스템에 의해 지원된다. 이형적 SAN 환경들을 지원하는 최적의 위치에서 지능성을 제공함으로써, 저장 도메인 관리자는 다음과 같은 비즈니즈 이점을 최종 사용자에게 제공한다.

- 향상된 저장 재원 할당 및 이용

- 저렴하게 수용되는 동적 고성장 저장 환경에 대한 유용성

- 온라인 관리 및 구성들을 통한 더 높은 가용성

- 저장 운용의 전체 $/GB 를 저감하는 더욱 효율적인 관리

- 이형적 서버들 및 저장을 하나의 통합된 SAN 환경으로 통합하는 능력

- 저장 관리를 추가함으로써 그리고 모든 저장 자원들에 걸쳐 동적으로 지배력을 가질 수 있는 특성들을 캐시함으로써 JBOD 저장의 값을 증가시키는 것

저장 도메인 관리의 개시를 통해서 동시적으로 배치되는 로버스트 SAN 하드웨어 하부구조는, 고도로 가용한 데이터에의 안정하고 고속의 액세스를 여전히 제공하면서도, 예측불가능하게 급변하는 환경에 적응할 수 있은 유연성을 제공한다. 결과적으로 이러나 중앙집중화된 저장 관리의 패러다임은, 기업의 경쟁력에 이점을 주는 데이터의 성장을 관리하는데에 더욱 효율적이고 저렴한 방법이다.

본 발명의 다양한 실시예에 대한 상술한 설명은 예시와 설명을 위해서 개시된 것이다. 이러한 설명은 본 발명을 상술된 구체화된 태양으로 제한하는 것을 의도하지 않는다. 다양한 변형예 및 균등한 구성들이 가능하다는 것은 당해기술분야의 전문가에게 자명할 것이다.

(57) 청구의 범위

청구항 1.

저장 네트워크가 하나 이상의 클라이언트와 하나 이상의 저장 시스템을 구비하고, 상기 하나 이상의 클라이언트가 저장 트랜잭션에 의해 서비스되는 클라이언트를 식별하는데 충분한 정보를 반송하는 각각의 저장 채널 프로토콜을 실행하고 있는 상기 저장 네트워크 내의 저장 도메인을 관리하는 장치에 있어서,

상기 하나 이상의 클라이언트 및 상기 하나 이상의 저장 시스템의 각각에 통신 매체를 통하여 연결되도록 채용되고, 다양한 통신 프로토콜들에 따라 동작하는 복수의 통신 인터페이스;

상기 복수의 통신 인터페이스에 연결되고, 상기 하나 이상의 클라이언트 중의 적어도 하나로 이루어지는 세트를 위한 저장 도메인으로서 상기 하나 이상의 저장 시스템으로부터 저장 위치들의 세트를 구성하는 로직을 포함하는 프로세싱 유니트;

상기 통신 인터페이스를 트레버싱하는 저장 트랜잭션을 통상의 포맷으로 및 통상의 포맷 이외의 것으로 번역하는 로직;

비휴발성 캐시 메모리를 포함하고, 상기 저장 도메인 내에 있는 상기 통신 인터페이스들에게로 통상적 포맷의 저장 트랜잭션을 경로지정하는 리던던시 자원들; 및

상기 프로세싱 유니트에 연결되어 있고, 상기 저장 도메인을 구성하는 관리 인터페이스를 포함하는 것을 특징으로 하는 저장 네트워크 내의 저장 도메인을 관리하는 장치,

청구항 2.

제 1 항에 있어서,

상기 하나 이상의 클라이언트가 논리적 저장 위치를 식별하는데 충분한 정보를 반송하는 각각의 저장 채널 프로토콜을 실행하고, 상기 논리적 저장 위치에 응답하여 저장 트랜잭션을 저장 도메인 내로 경로지정하는 로직을 포함하는 것을 특징으로 하는 저장 네트워크 내의 저장 도메인을 관리하는 장치,

청구항 3.

제 1 항에 있어서,

상기 네트워크 내에서 어떤 저장 위치로부터 다른 저장 위치로의 데이터 세트의 이동을 관리하는 로직을 포함하는 것을 특징으로 하는 저장 네트워크 내의 저장 도메인을 관리하는 장치,

청구항 4.

제 1 항에 있어서,

상기 관리 인터페이스는, 상기 네트워크로써 복수의 저장 도메인을 구성하기 위한 자원들을 포함하는 것을 특징으로 하는 저장 네트워크 내의 저장 도메인을 관리하는 장치,

청구항 5.

네트워크 내의 클라이언트들과 저장 자원들 사이에 중간 시스템을 상기 네트워크 내에 설치하는 단계;

상기 중간 시스템 내에 있는 로직을 이용하여 상기 네트워크 내에 클라이언트들에게 논리적 저장 범위들을 할당하는 단계;

상기 중간 시스템 내에 있는 로직을 이용하여 상기 논리적 저장 범위들에게 상기 네트워크 내의 저장 자원들을 할당하는 단계; 및

상기 클라이언트들에 할당된 상기 논리적 저장 범위들에 따라, 그리고 상기 논리적 저장 범위들에 할당된 상기 저장 자원들에 따라 상기 중간 시스템을 통하여 저장 트랜잭션을 경로지정하는 단계를 포함하는 것을 특징으로 하는 저장 네트워크 내의 저장 자원의 구성 및 관리를 위한 방법.

청구항 6.

저장 트랜잭션 통신 채널을 지원하는 통신 인터페이스;

상기 저장 트랜잭션 통신 채널에 걸쳐 수신된 저장 트랜잭션을 내부 포맷으로 번역하는 로직; 및

상기 내부 포맷으로 된 상기 저장 트랜잭션을 가상 회로에 경로지정하는 로직을 포함하는 저장 서버로서,

상기 가상 회로는 상기 저장 서버와 통신하여 각각의 데이터 저장부에의 연결을 관리하는 것을 특징으로 하는 저장 서버.

청구항 7.

제 6 항에 있어서,

상기 가상 회로는, 대응하는 하나 이상의 데이터 저장부에 대한 하나 이상의 통신 프로토콜로 상기 내부 포맷을 번역하는 로직을 포함하는 것을 특징으로 하는 저장 서버.

청구항 8.

제 7 항에 있어서,

데이터 소스 각각에 대응하는 상기 통신 프로토콜 각각은, 표준 "지능적 입출력"(I_20) 메시지 포맷에 상응하는 프로토콜을 포함하는 것을 특징으로 하는 저장 서버.

청구항 9.

제 6 항에 있어서,

상기 저장 트랜잭션을 가상 회로에 경로지정하는 로직은, 상기 통신 채널에 특정된 어드레스 범위와 상기 가상 회로 사이의 대응 관계를 나타내는 복수의 엔트리를 가진 테이블을 포함하는 것을 특징으로 하는 저장 서버.

청구항 10.

제 6 항에 있어서,

상기 저장 트랜잭션을 가상 회로에 경로지정하는 로직은, 각각의 데이터 소스와 상기 가상 회로 사이의 대응 관계를 나타내는 복수의 엔트리를 가진 테이블을 포함하는 것을 저장 서버.

청구항 11.

제 6 항에 있어서,

캐시를 포함하고,

상기 가상 회로는 상기 캐시와 통신하는 것을 특징으로 하는 저장 서버.

청구항 12.

제 6 항에 있어서,

상기 각각의 데이터 저장부는 비휴발성 메모리를 포함하는 것을 특징으로 하는 저장 서버.

청구항 13.

제 6 항에 있어서,

상기 각각의 데이터 저장부는 하드디스크들의 배열을 포함하는 것을 특징으로 하는 저장 서버.

청구항 14.

제 6 항에 있어서,

구성 데이터의 입력을 지원하는 사용자 인터페이스를 포함하는 것을 특징으로 하는 저장 서버.

청구항 15.

제 14 항에 있어서,

상기 사용자 인터페이스는 저장 서버에 연결되어 있는 그래픽 사용자 인터페이스를 포함하는 것을 특징으로 하는 저장 서버.

청구항 16.

제 14 항에 있어서,

상기 사용자 인터페이스는 저장 서버에 연결되어 있는 터치 스크린을 포함하는 것을 특징으로 하는 저장 서버.

청구항 17.

저장 트랜잭션을 위한 요구를 발생시키는 적어도 하나의 클라이언트 시스템과, 상기 클라이언트 시스템으로의 및 상기 클라이언트 시스템으로부터의 클라이언트 통신 채널과, 복수의 저장 장치와, 상기 저장 장치로의 및 상기 저장 장치로부터의 복수의 각각의 통신 채널을 포함하는 저장 네트워크를 위한 서버에 있어서,

버스 시스템을 포함하는 프로세서;

상기 버스 시스템에 연결되어 있는 클라이언트 인터페이스;

상기 버스 시스템에 연결되어 있고 상기 각각의 통신 채널에 대한 복수의 인터페이스;

상기 버스 시스템에 연결되어 있는 비휴발성 캐시 메모리; 및

상기 프로세서에 의해 제어되고, 상기 서버 인터페이스 상에 저장 트랜잭션을 위한 요구를 수신하고, 상기 요구된 저장 트랜잭션을 상기 복수의 저장 장치로 가도록 디렉팅하고, 상기 저장 트랜잭션에서 이용하기 위한 상기 비휴발성 캐시 메모리를 할당하는 자원들을 포함하는 것을 특징으로 하는 저장 네트워크를 위한 서버.

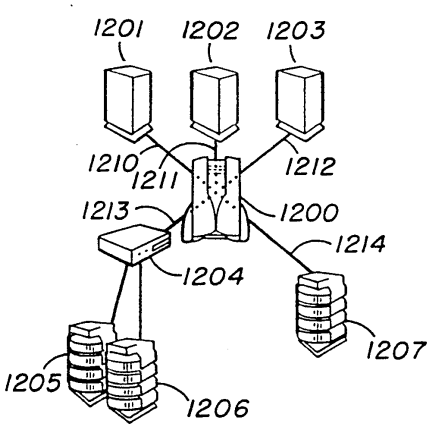
청구항 18.

제 17 항에 있어서,

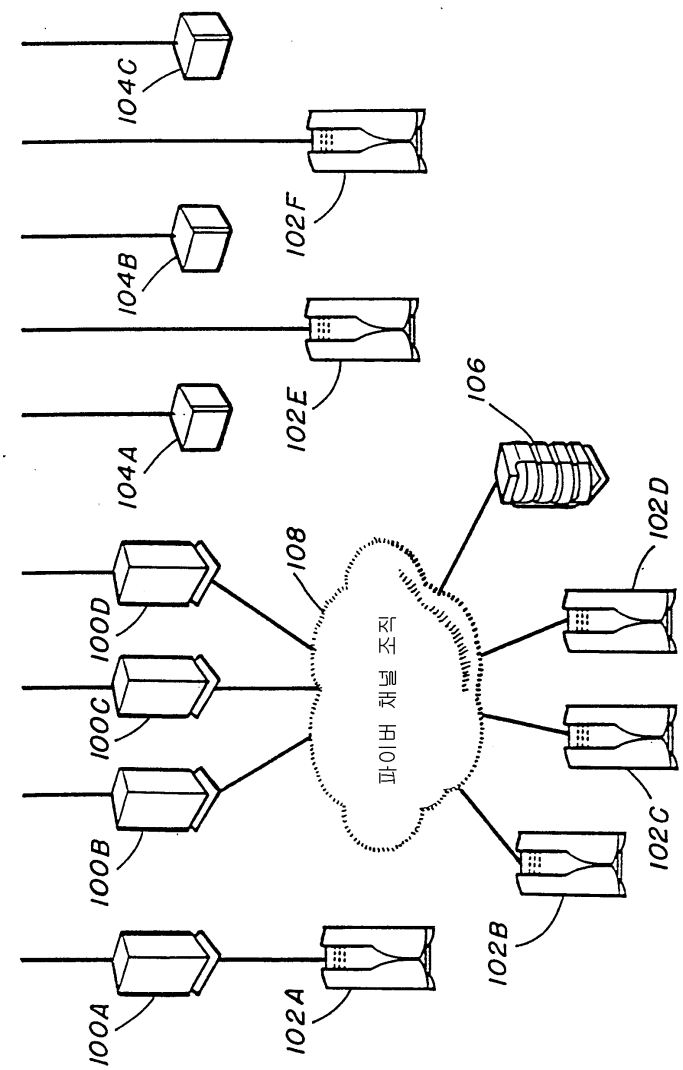
상기 프로세서에 의해 제어되는 상기 자원들은, 저장 트랜잭션에 대한 액세스 허가를 인증하고 확인하는 프로세싱부를 포함하는 것을 특징으로 하는 서버.

도면

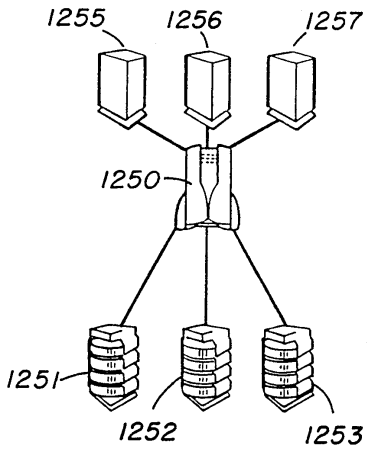
도면1a



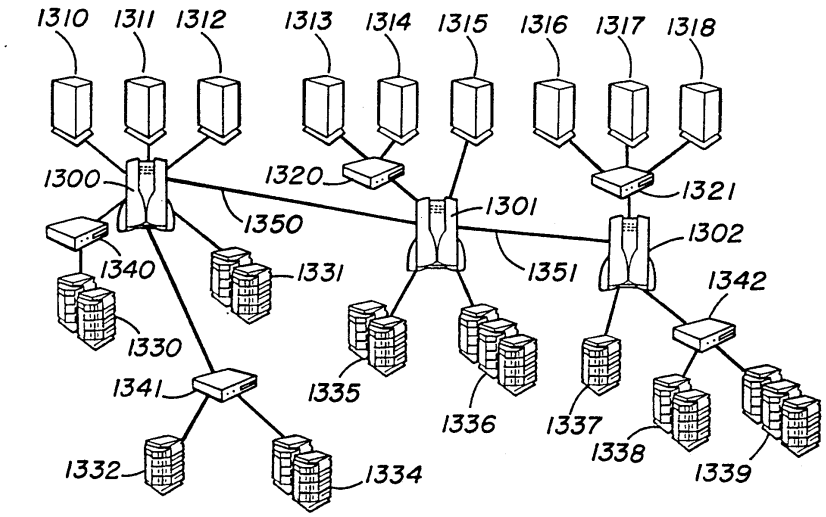
도면1b


104C
102F
104B
102E
104A
106
102D
108
파이버 채널 조직
100D
100C
100B
102C
102B
100A
102A

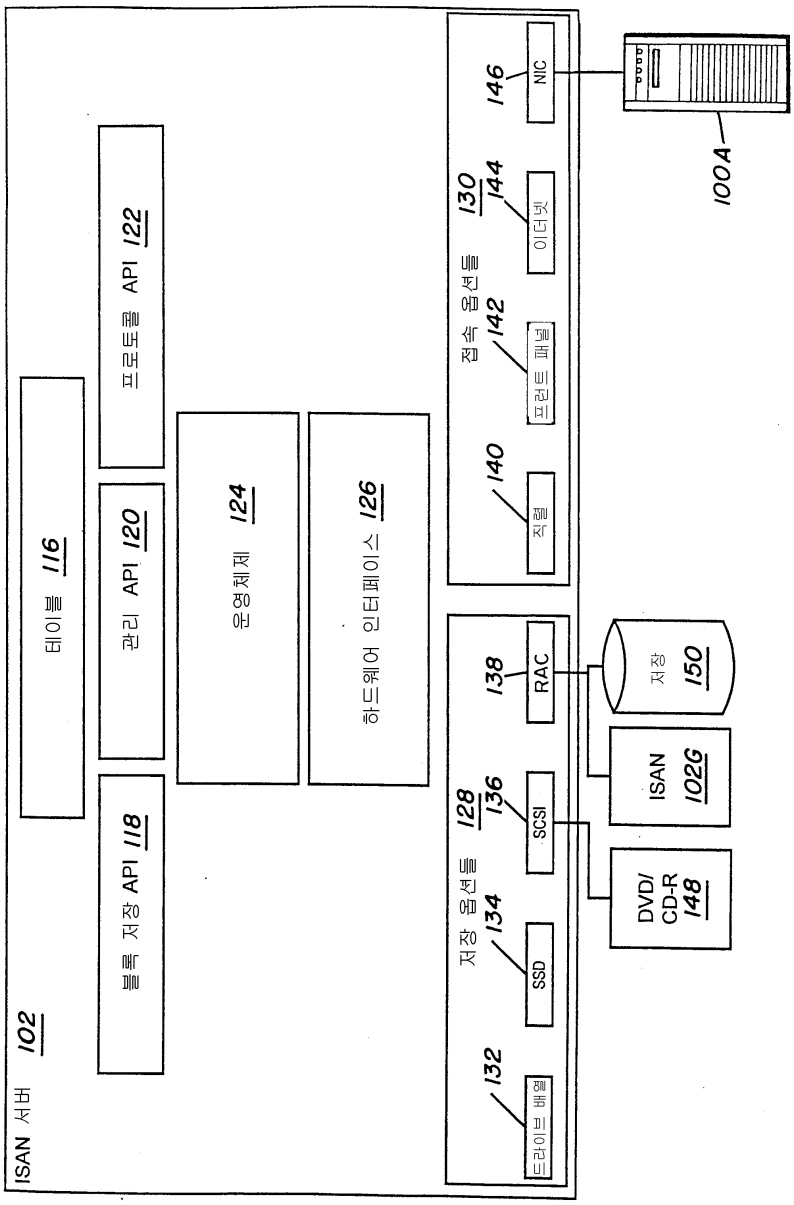
도면2


1255
1256
1257
1250
1251
1252
1253

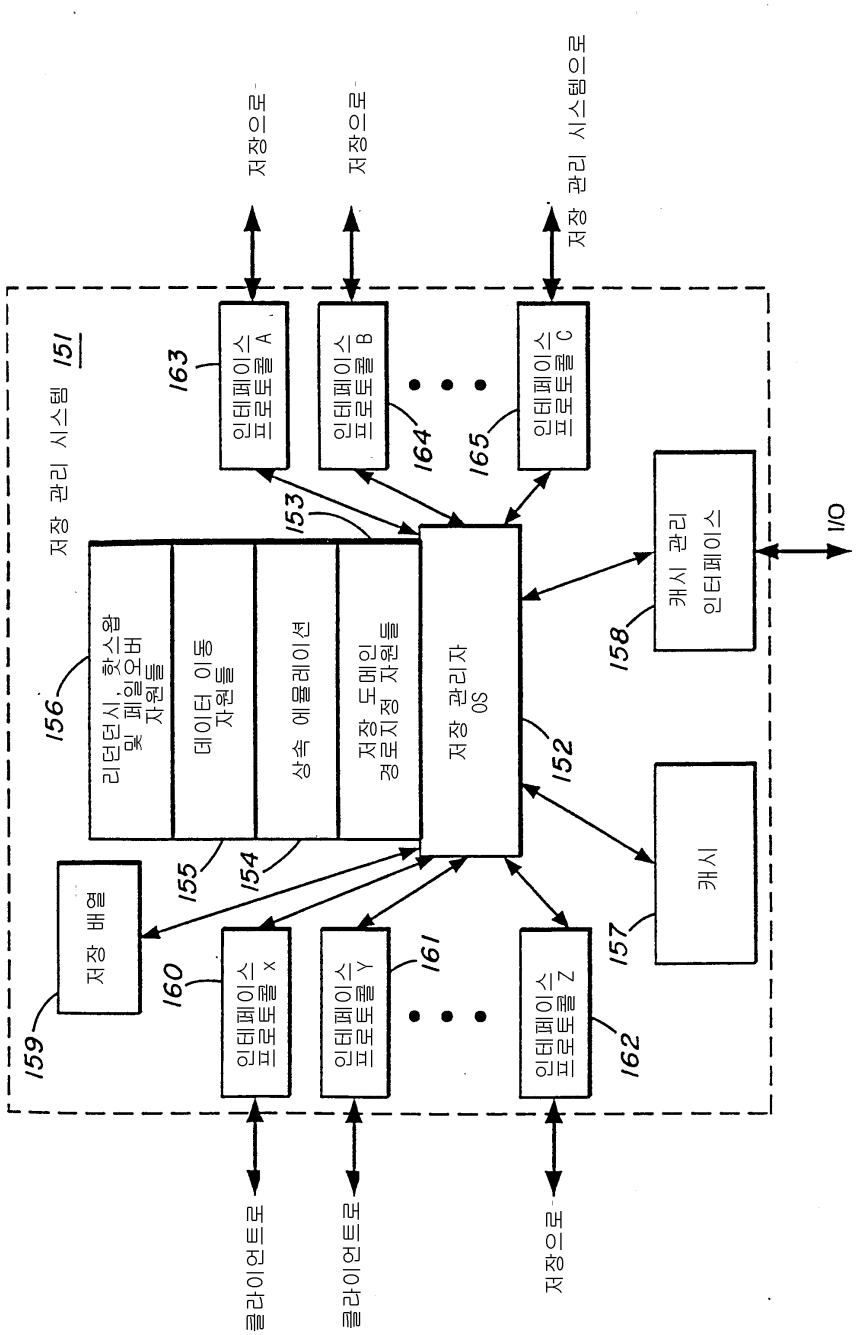
도면3


1310
1311
1312
1313
1314
1315
1316
1317
1318
1300
1320
1321
1301
1350
1302
1340
1331
1351
1342
1330
1341
1335
1336
1337
1338
1339
1332
1334

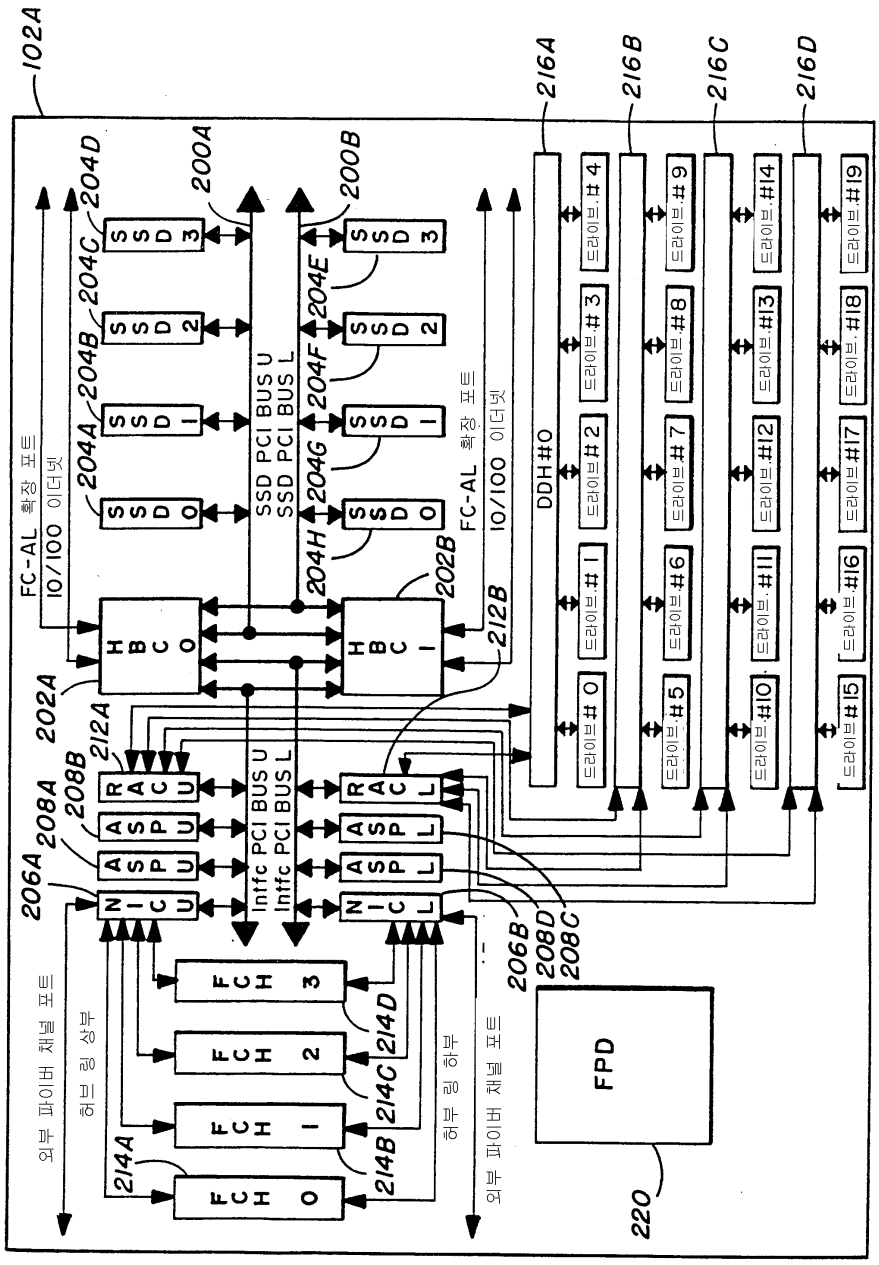
도면4


ISAN 서버 102
블록 저장 API 118
관리 API 120
프로토콜 API 122
테이블 116
운영체제 124
하드웨어 인터페이스 126
저장 옵션들 128
드라이브 배열 132
SSD 134
SCSI 136
RAC 138
접속 옵션들 130
직렬 140
프런트 패널 142
이더넷 144
NIC 146
DVD/ CD-R 148
ISAN 102G
저장 150
100A

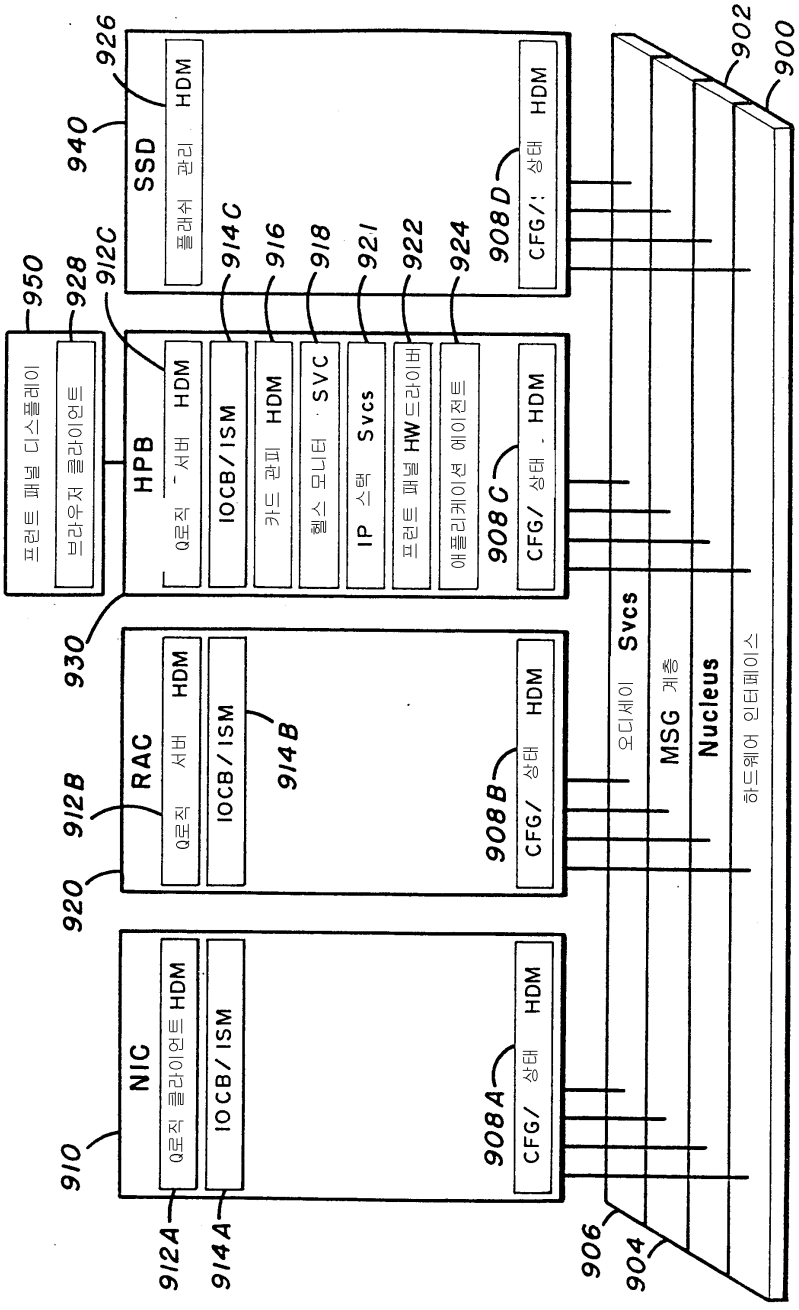
도면5


저장 관리 시스템 151
저장 관리자 OS 152
저장 도메인 경로지정 자원들 154
상속 에뮬레이션 155
데이터 이동 자원들 156
리던던시, 핫스왑 및 페일오버 자원들
캐시 157
캐시 관리 인터페이스 158
저장 배열 159
인터페이스 프로토콜 X 160
인터페이스 프로토콜 Y 161
인터페이스 프로토콜 Z 162
인터페이스 프로토콜 A 163
인터페이스 프로토콜 B 164
인터페이스 프로토콜 C 165
153
I/O
저장으로
저장으로
저장 관리 시스템으로
클라이언트로
클라이언트로
저장으로

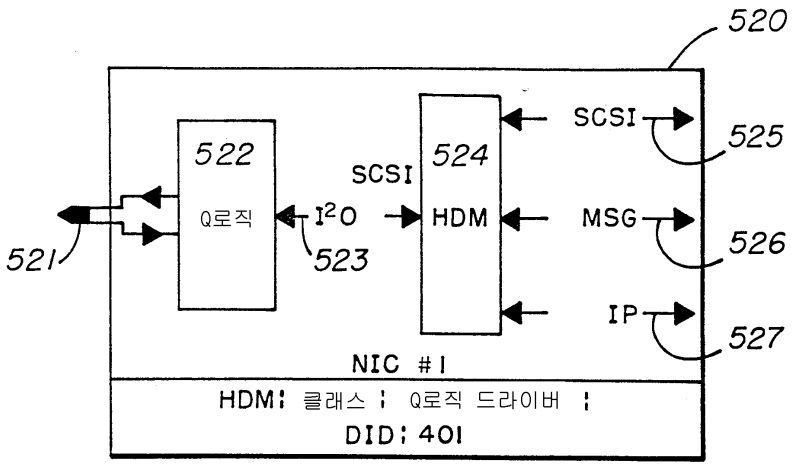
도면6


102A
216A
216B
216C
216D
200A
200B
204D
204C
204B
204A
204E
204F
204G
204H
202A
202B
212B
212A
208B
208A
206A
206B
208D
208C
214A
214B
214C
214D
220
FC-AL 확장 포트
10/100 이더넷
SSD PCI BUS U
SSD PCI BUS L
Intfc PCI BUS U
Intfc PCI BUS L
DDH #0
외부 파이버 채널 포트
허브 링 상부
허브 링 하부
FPD
드라이브 #0
드라이브 #1
드라이브 #2
드라이브 #3
드라이브 #4
드라이브 #5
드라이브 #6
드라이브 #7
드라이브 #8
드라이브 #9
드라이브 #10
드라이브 #11
드라이브 #12
드라이브 #13
드라이브 #14
드라이브 #15
드라이브 #16
드라이브 #17
드라이브 #18
드라이브 #19

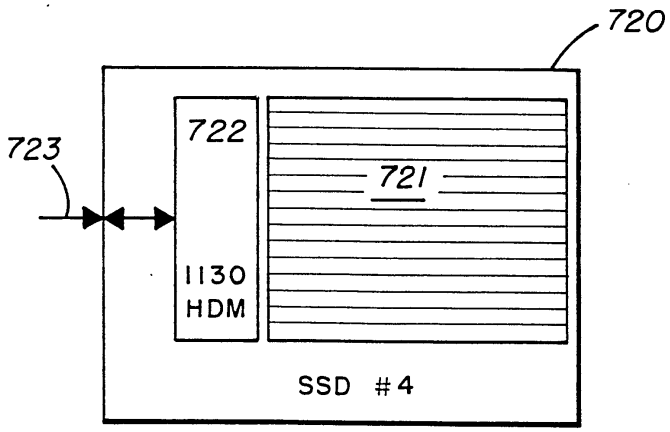
도면7


NIC
Q로직 클라이언트 HDM
IOCB/ISM
CFG/ 상태 HDM
RAC
Q로직 서버 HDM
IOCB/ISM
CFG/ 상태 HDM
HPB
Q로직 서버 HDM
IOCB/ISM
카드 관피 HDM
헬스 모니터 SVC
IP 스택 Svcs
프런트 패널 HW드라이버
애플리케이션 에이전트
CFG/ 상태 HDM
SSD
플래쉬 관리 HDM
CFG/ 상태 HDM
프런트 패널 디스플레이
브라우저 클라이언트
오디세이 Svcs
MSG 계층
Nucleus
하드웨어 인터페이스
910
912A
914A
908A
920
912B
914B
908B
930
928
912C
914C
916
918
921
922
924
908C
908D
940
926
950
906
904
902
900

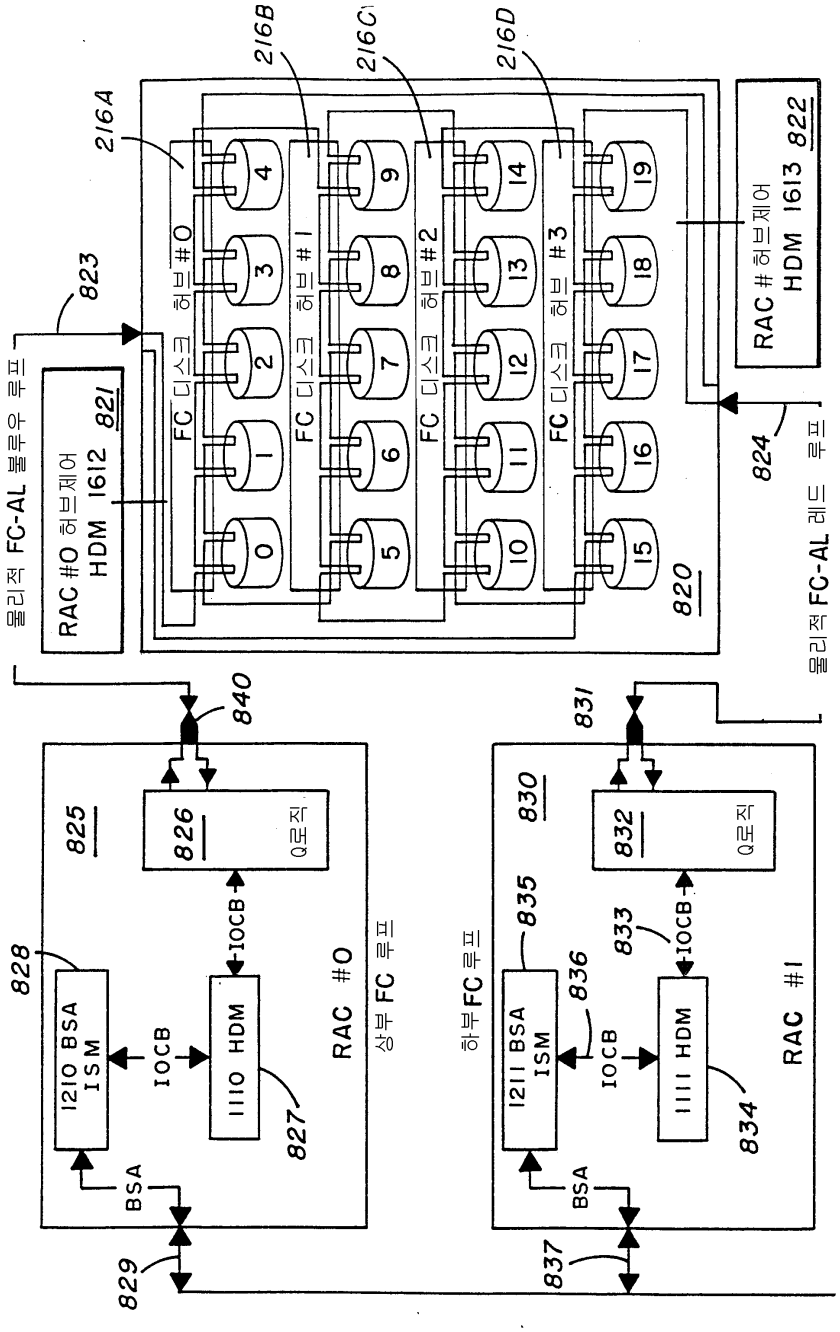
도면8


520
522
Q로직
SCSI
I²O
523
524
HDM
SCSI
525
MSG
526
IP
527
521
NIC #1
HDM: 클래스 : Q로직 드라이버 :
DID: 401

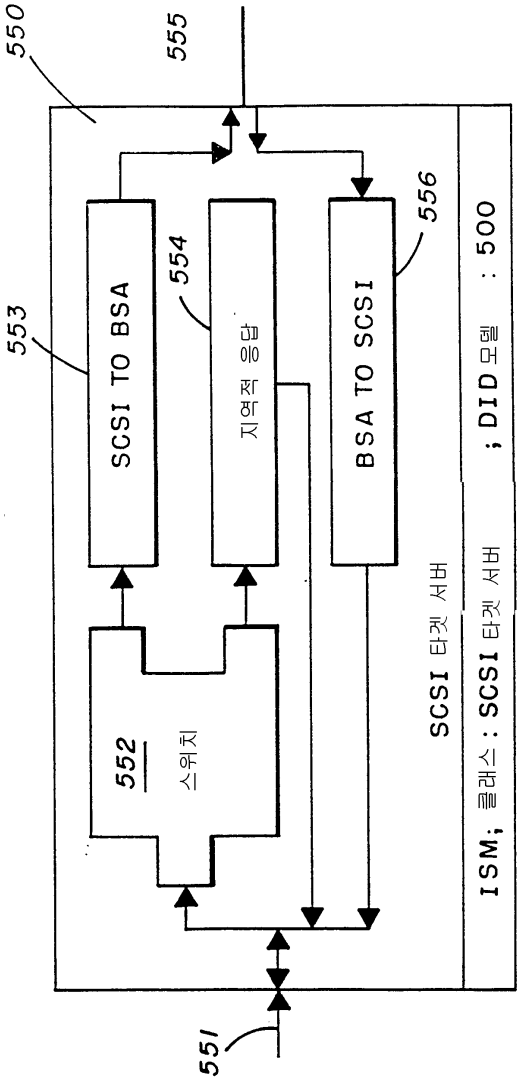
도면9


720
723
722
721
1130
HDM
SSD #4

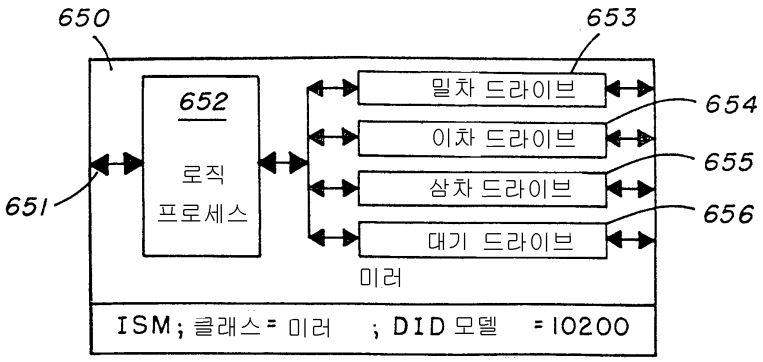
도면10


물리적 FC-AL 볼루우 루프
RAC #0 허브제어
HDM 1612
821
823
2164
2168
216C
216D
FC 디스크 허브#0
FC 디스크 허브#1
FC 디스크 허브#2
FC 디스크 허브#3
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
820
RAC #허브제어
HDM 1613
822
824
물리적 FC-AL 레드 루프
840
831
825
826
Q로직
830
832
Q로직
828
IOCB
IOCB
1210 BSA
ISM
IOCB
1110 HDM
827
RAC #0
상부 FC 루프
835
833
836
1211 BSA
ISM
IOCB
1111 HDM
834
RAC #1
하부 FC 루프
BSA
BSA
829
837

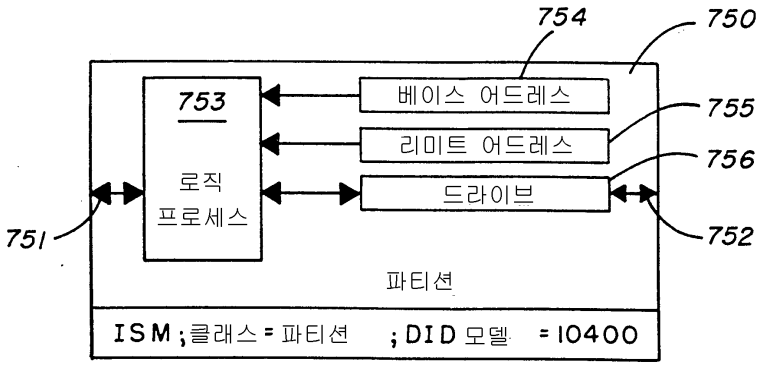
도면11


550
555
556
554
553
SCSI TO BSA
지역적 응답
BSA TO SCSI
552
스위치
SCSI 타겟 서버
ISM; 클래스 : SCSI 타겟 서버 ; DID 모델 : 500
551

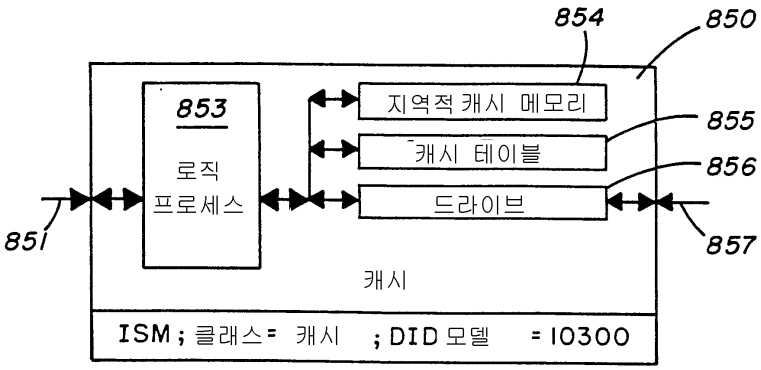
도면12


650
653
652
로직
프로세스
651
밀차 드라이브
이차 드라이브
삼차 드라이브
대기 드라이브
654
655
656
미러
ISM; 클래스 = 미러 ; DID 모델 = 10200

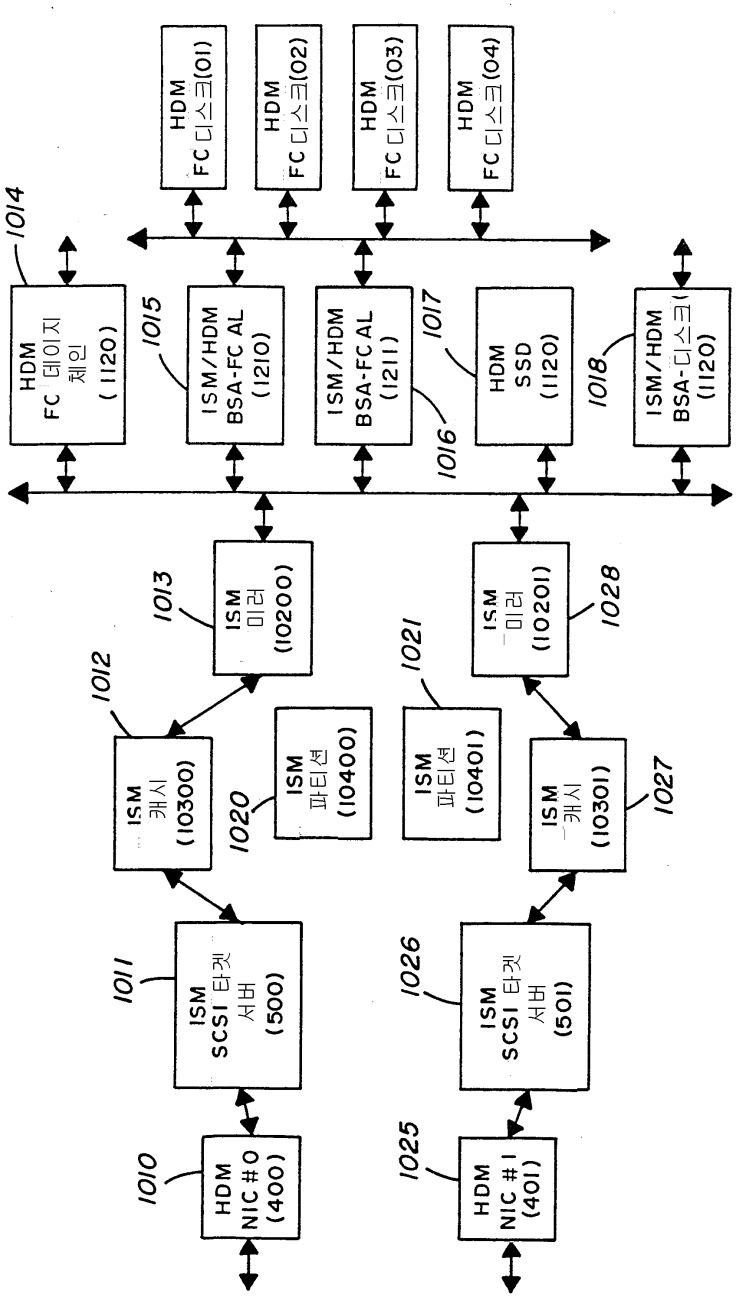
도면13


754
750
753
베이스 어드레스
755
리미트 어드레스
756
로직
프로세스
드라이브
751
752
파티션
ISM;클래스=파티션 ;DID 모델 =10400

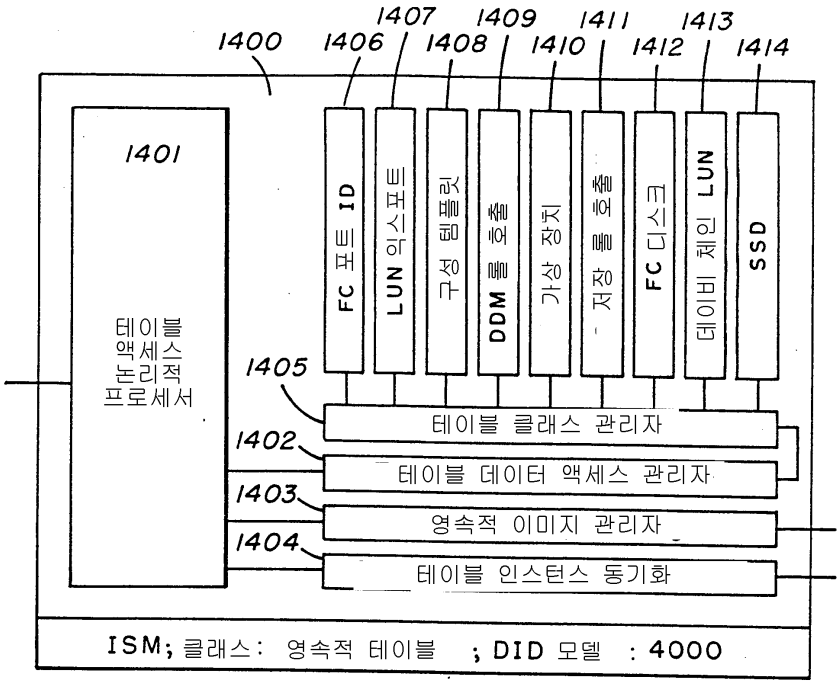
도면14


854
850
853
지역적 캐시 메모리
855
캐시 테이블
856
로직
프로세스
드라이브
851
857
캐시
ISM; 클래스= 캐시 ; DID 모델 =10300

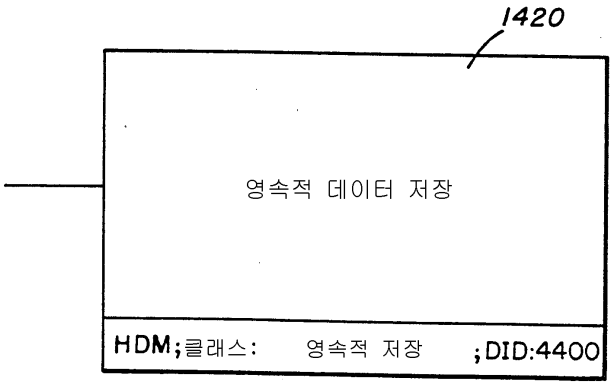
도면15


HDM FC 디스크(01)
HDM FC 디스크(02)
HDM FC 디스크(03)
HDM FC 디스크(04)
1014
HDM FC 데이지 체인 (1120)
1015
ISM/HDM BSA-FC AL (1210)
ISM/HDM BSA-FC AL (1211)
1017
HDM SSD (1120)
1016
1018
ISM/HDM BSA-디스크 (1120)
1013
ISM 미러 (10200)
ISM 미러 (10201)
1028
1012
1021
ISM 캐시 (10300)
1020
ISM 파티션 (10400)
ISM 파티션 (10401)
ISM 캐시 (10301)
1027
1011
ISM SCSI 타겟 서버 (500)
1026
ISM SCSI 타겟 서버 (501)
1010
HDM NIC #0 (400)
1025
HDM NIC #1 (401)

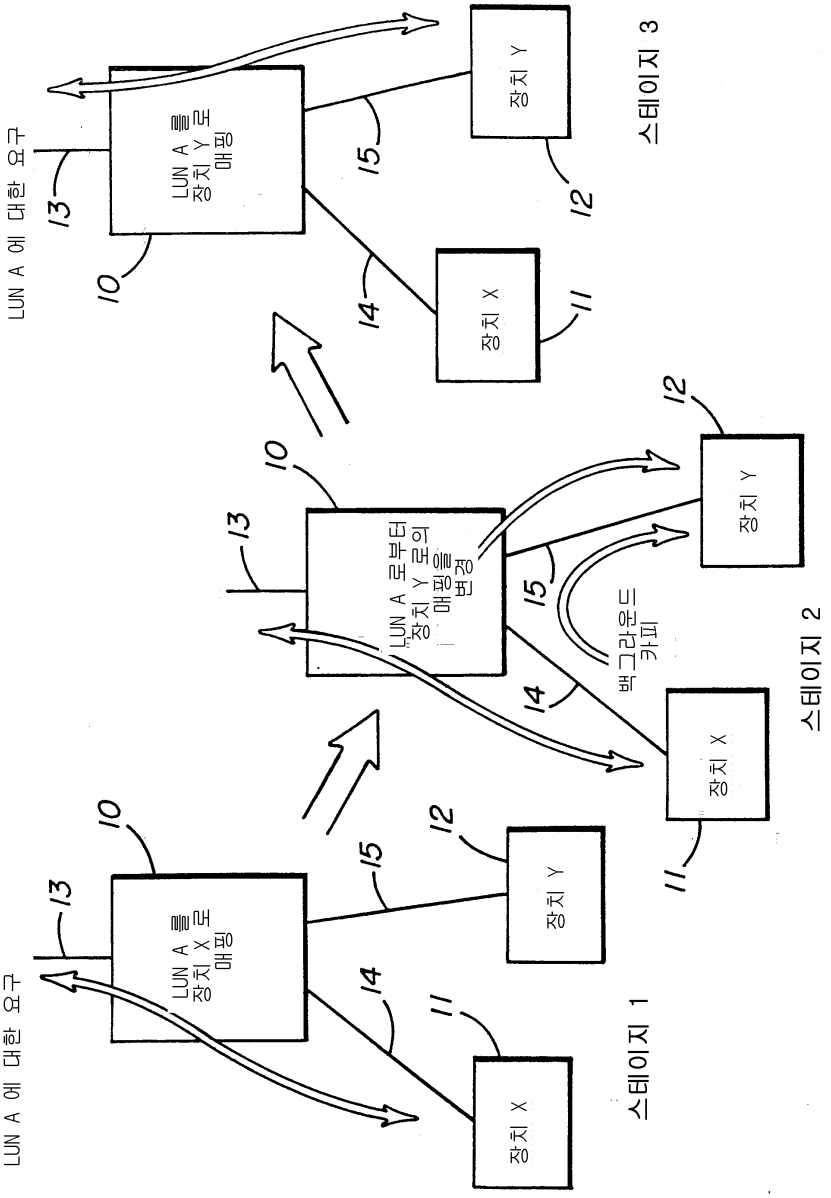
도면16


1400
1401
테이블 액세스 논리적 프로세서
1405
1402
1403
1404
1406
1407
1408
1409
1410
1411
1412
1413
1414
FC 포트 ID
LUN 익스포트
구성 템플릿
DDM 룰 호출
가상 장치
저장 룰 호출
FC 디스크
데이비 체인 LUN
SSD
테이블 클래스 관리자
테이블 데이터 액세스 관리자
영속적 이미지 관리자
테이블 인스턴스 동기화
ISM; 클래스: 영속적 테이블 ; DID 모델 : 4000

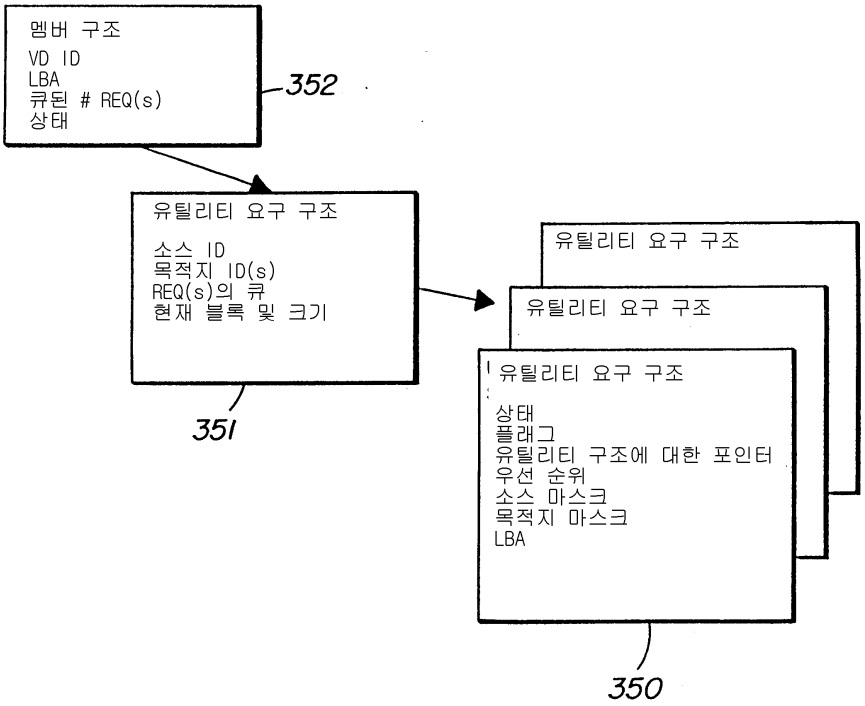
도면17


1420
영속적 데이터 저장
HDM;클래스: 영속적 저장 ;DID:4400

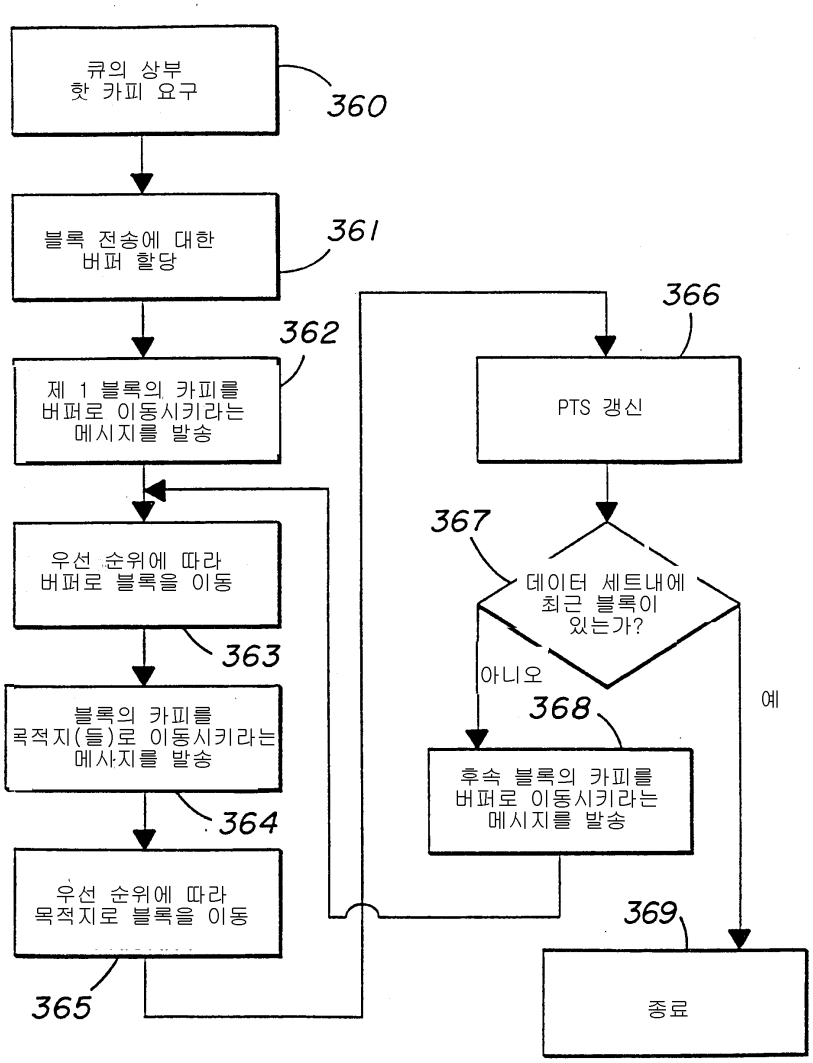
도면18


LUN A 에 대한 요구
LUN A 를 장치 X 로 매핑
장치 X
장치 Y
스테이지 1
LUN A 로부터 장치 Y 로의 매핑을 변경
백그라운드 카피
스테이지 2
LUN A 를 장치 Y 로 매핑
스테이지 3
10
11
12
13
14
15

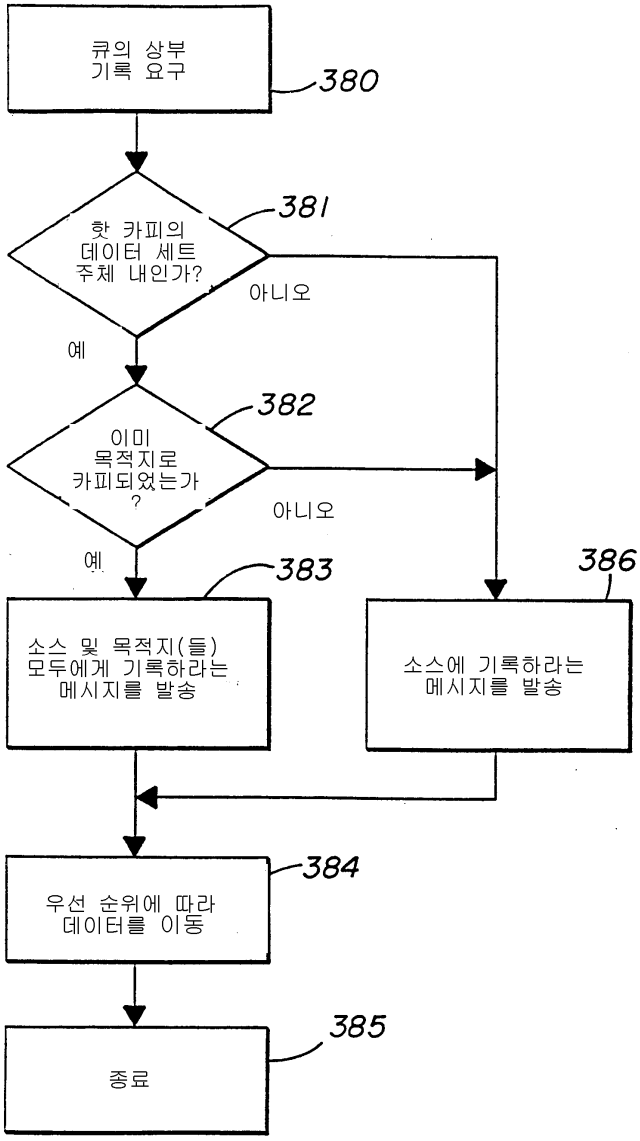
도면19


멤버 구조
VD ID
LBA
큐된 # REQ(s)
상태
352
유틸리티 요구 구조
소스 ID
목적지 ID(s)
REQ(s)의 큐
현재 블록 및 크기
351
유틸리티 요구 구조
유틸리티 요구 구조
유틸리티 요구 구조
상태
플래그
유틸리티 구조에 대한 포인터
우선 순위
소스 마스크
목적지 마스크
LBA
350

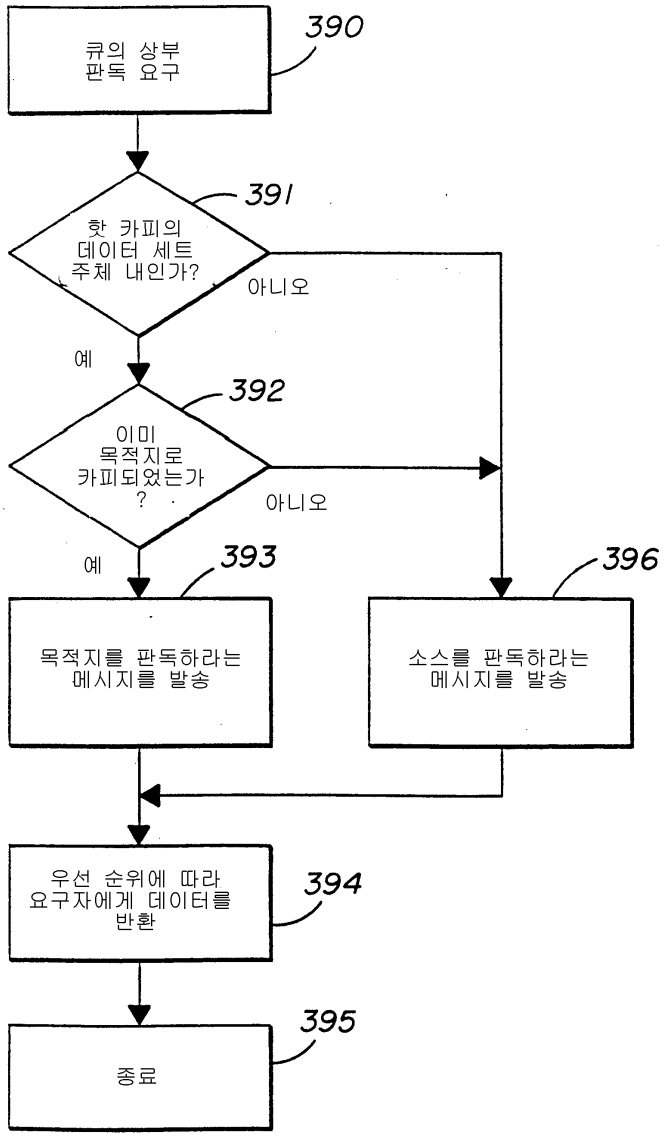
도면20


큐의 상부
핫 카피 요구
360
블록 전송에 대한
버퍼 할당
361
제 1 블록의 카피를
버퍼로 이동시키라는
메시지를 발송
362
우선 순위에 따라
버퍼로 블록을 이동
363
블록의 카피를
목적지(들)로 이동시키라는
메시지를 발송
364
우선 순위에 따라
목적지로 블록을 이동
365
366
PTS 갱신
367
데이터 세트내에
최근 블록이
있는가?
아니오
예
368
후속 블록의 카피를
버퍼로 이동시키라는
메시지를 발송
369
종료

도면21


큐의 상부 기록 요구
380
핫 카피의 데이터 세트 주체 내인가?
381
아니오
예
이미 목적지로 카피되었는가 ?
382
아니오
예
383
386
소스 및 목적지(들) 모두에게 기록하라는 메시지를 발송
소스에 기록하라는 메시지를 발송
우선 순위에 따라 데이터를 이동
384
종료
385

도면22


390
큐의 상부
판독 요구
391
핫 카피의
데이터 세트
주체 내인가?
아니오
예
392
이미
목적지로
카피되었는가
?
아니오
예
393
목적지를 판독하라는
메시지를 발송
396
소스를 판독하라는
메시지를 발송
우선 순위에 따라
요구자에게 데이터를
반환
394
395
종료