객체기반 저장시스템을 위한 백업시스템 설계 및 구현

(Design and Implementation of a Backup System for Object based Storage Systems)

윤 종 현 [†] 이 석 재 [††] 장 수 민 [††]
(Jong Hyeon Yun) (Seok Jae Lee) (Su Min Jang)

유 재 수 [†††] 김 홍 연 [††††] 김 준 [††††]
(Jae Soo Yoo) (Hong Yeon Kim) (Jun Kim)

요 약 최근 많은 연구가 이루어지고 있는 객체기반 저장시스템은 기존 블록기반 저장시스템과 달리 데이타 접근 방법, 데이타 속성 정보, 데이타 보안 방법 등을 포함할 수 있는 논리적인 객체를 데이타 저장 단위로 한다. 객체는 블록과 달리 크기의 제약이 없고, 저장장치의 내부 저장 구조에 종속적이지 않다. 객체기반 저장시스템은 이러한 객체 저장구조를 기반으로, 여러 저장장치에 쉽게 객체를 분산 처리해 데이타 입출력 성능을 향상시키고, 무제한적인 확장성을 제공한다. 그러나 객체기반 저장시스템의 오류 복구를 위한 백업 기술은 기존 백업 기술을 그대로 활용하는 정도에 그치고 있어 백업 시간이 길고, 객체기반 저장시스템의 특성을 활용할 수 없다. 따라서 본 논문에서는 객체기반 저장시스템의 특성을 정확히 파악하고 이를 잘 활용하여, 객체기반 저장시스템에서 높은 백업 성능과 효율을 제공할 수 있는 객체기반 백업시스템을 설계하고 구현한다. 본 논문에서 구현한 백업시스템은 블록 또는 파일이 아닌 객체를 백업의 기본 단위로 하여 기존 백업시스템에 비해 작업 시간을 단축시키고, 오류 발생 시 복구를 빠르게 할 수 있는 장점을 갖는다.

키워드 : 백업 시스템, 객체기반저장장치, SAN, NAS

Abstract Recently, the object based storage devices systems(OSDs) have been actively researched. They are different from existing block based storage systems(BSDs) in terms of the storage unit. The storage unit of the OSDs is an object that includes the access methods, the attributes of data, the security information, and so on. The object has no size limit and no influence on the internal storage structures. Therefore, the OSDs improve the I/O throughput and the scalability. But the backup systems for the OSDs still use the existing backup techniques for the BSDs. As a result, they need much backup time and do not utilize the characteristics of the OSDs.

In this paper, we design and implement a new object based backup system that utilizes the features of the OSDs. Our backup system significantly improves the backup time over existing backup systems because the raw objects are directly transferred to the backup devices in our system. It also restores the backup data much faster than the existing systems when system failures occur. In addition, it supports various types of backup and restore requests.

Key words : Backup System, Object based Storage Device, SAN, NAS

· 본 연구는 중소기업청 2007년도 산학연협력 기업부설연구소 설치지원사업 지원 및 정부(과학기술부)의 재원으로 한국과학재단(특정기초연구 과제번호 R01-2006-000-1080900) 지원에 의해 연구되었음

† 학생회원 : 충북대학교 정보통신공학과
bluette@netdb.cbnu.ac.kr
†† 학생회원 : 충북대학교 전기전자컴퓨터공학부
sjlee@netdb.cbnu.ac.kr
smjang@netdb.cbnu.ac.kr
††† 정 회 원 : 충북대학교 전기전자컴퓨터공학부 교수
yjs@chungbuk.ac.kr
†††† 종신회원 : 한국전자통신연구원 디지털홈연구단 인터넷서버그룹 저장시스템연구팀
kimhy@etri.re.kr
jkim@etri.re.kr

논문접수 : 2007년 6월 21일
심사완료 : 2007년 11월 22일

Copyright@2008 한국정보과학회 : 개인 목적이나 교육 목적인 경우, 이 저작물의 전체 또는 일부에 대한 복사본 혹은 디지털 사본의 제작을 허가합니다. 이 때, 사본은 상업적 수단으로 사용할 수 없으며 첫 페이지에 본 문구와 출처를 반드시 명시해야 합니다. 이 외의 목적으로 복제, 배포, 출판, 전송 등 모든 유형의 사용행위를 하는 경우에 대하여는 사전에 허가를 얻고 비용을 지불해야 합니다.
정보과학회논문지 : 시스템 및 이론 제35권 제1호(2008.2)

1. 서 론

인터넷 환경의 급속한 발전과 유비쿼터스 컴퓨팅 환경으로의 진화는 대량의 데이타에 대한 고성능 처리와 대용량 저장장치 그리고 플랫폼에 독립적이고 안전한 데이타 공유를 제공할 수 있는 저장장치 시스템을 요구하고 있다. 하지만 기존의 대표적인 대용량 저장시스템인 NAS(Network Attached Storage)와 SAN(Storage Area Network)은 성능과 확장성 그리고 데이타에 대한 보안 등의 요구사항을 만족시키기에 한계를 드러내고 있다[1].

최근 많은 연구가 진행되고 있는 객체기반 저장장치(Object-based Storage Device systems: OSDs) 기술은 기존 저장장치들이 갖는 한계를 해결할 수 있는 방법으로 고성능, 유연하고 무제한적인 확장, 데이타 보안, 플랫폼 독립적인 데이타 공유 등을 제공할 수 있는 기술로 인식되고 있다[1-3]. 객체기반 저장장치 기술은 카네기멜론 대학의 Parallel Data Lab.에서 수행한 NASD (Network Attached Secure Disk) 프로젝트와 Active Disk 프로젝트에서 연구되었으며, 현재는 ANSI T10에서 객체기반 저장장치에 대한 인터페이스를 SCSI 인터페이스에 추가한 SCSI-OSD 인터페이스 표준화 작업이 진행 중에 있다. 그리고 Intel, Seagate, IBM, EMC, Panasas, Cluster File System 등과 같은 기업들이 객체기반 저장장치 기술과 함께 이를 기반으로 하는 저장시스템에 대한 연구를 진행하고 있다[4].

그러나 객체기반 저장시스템 환경을 고려한 백업 기술에 대한 연구는 아직 활발히 진행되지 않고 있다. 현재 객체기반 저장시스템의 오류 복구를 위한 백업 기술은 객체기반 저장시스템의 특성이 고려되지 않은 기존 백업 기술을 그대로 활용하고 있다. 기존 백업 장치 관리 기법들은 객체기반 저장시스템의 기반이 되는 객체의 특성을 지원하지 못한다. 또한 객체기반 저장시스템과 백업 저장장치간의 데이타 전송 과정에서 객체를 블록기반 또는 파일로 변환해 전송하기 때문에 백업 또는 복구에 많은 시간이 필요하게 되는 문제점이 있다. Minosota 대학에서는 이러한 기존 백업시스템의 문제점을 지적하고 OSD에 직접 백업 명령을 전송하여 객체 정보에 대한 백업을 수행하는 백업 인식 OSD 구조를 소개하였다[5]. 하지만 제안된 시스템은 OSD의 객체 정보만을 백업하며, 기존 시스템이 제공하는 다양한 백업 기능을 제공하지 않는다.

본 논문에서는 이와 같은 문제점들을 해결하기 위해 객체기반 저장시스템에 적합한 백업 및 복구 기술에 대한 연구를 수행하고, 연구 결과를 기반으로 백업시스템을 설계하고 구현한다. 본 논문는 리눅스 환경에서 운영되는 객체기반 클러스터 파일시스템을 분석하고, 객체 데이타의 특성을 고려한 백업시스템의 요구사항 분석 및 기능을 정의한다. 또한 상용화 또는 연구 중인 객체기반 클러스터 파일시스템과 이를 지원하는 백업시스템들을 분석하고, 이를 통해 객체기반 클러스터 파일시스템 환경에 효율적인 백업시스템의 기능 및 요구 사항을 도출한다. 도출한 기능 및 요구사항을 만족할 수 있는 백업시스템의 구조와 처리 프로세스를 설계한다. 그리고 실제 객체기반 클러스터 파일시스템에서 동작하는 백업시스템을 구현하고 성능 평가를 통해 설계된 백업시스템의 적절성과 실현 가능성을 검증한다.

본 논문의 구성은 다음과 같다. 2장에서 기존 대용량 저장시스템과 객체기반 저장장치시스템을 소개한다. 3장에서 객체기반 저장시스템을 위한 백업시스템의 구조를 기술한다. 4장에서는 객체기반 저장시스템을 위한 백업시스템의 설계 및 구현 내용을 기술한다. 5장에서 구현한 백업시스템의 성능을 비교 분석하고, 6장에서 결론을 맺고 향후 연구방향을 기술한다.

2. 관련연구

2.1 대용량 저장장치 관리 기술

오늘날 데이타를 저장, 관리하는 대표적인 저장장치 관리 기술은 DAS(Direct Attached Storage), NAS(Network Attached Storage), SAN(Storage Area Network) 등이 있다. DAS는 서버에 저장장치를 직접 연결하는 가장 표준적인 방법이다[6,7]. DAS는 일반적인 컴퓨터 시스템에 연결된 저장장치를 말하며, 전용 케이블 등을 사용하여 직접적인 일대일 연결을 사용한다. NAS는 TCP/IP와 같은 일반적인 통신 프로토콜을 이용하여 기존의 네트워크에 저장장치를 직접 연결시켜 네트워크를 통해 들어오는 파일 I/O 프로토콜에 대응하는 특별한 목적의 저장시스템이다. SAN은 인프라에 대한 중복 투자 방지 및 이기종간 데이타 공유를 목적으로 제안된 장치로, 서버와 저장장치 사이의 네트워크로 파이버 채널(Fiber Channel)을 사용하여 연결한 저장시스템이다. 표 1은 대표적인 저장 기술들의 특징을 정리하고 있다.

표 1 대표적인 저장장치 관리 기술

	DAS	SAN	NAS
장치 관리	High/Low	High	Medium
보안	High	Medium	Low
장치 및 데이타 공유	Low	Medium	High
저장 성능	High	High	Low
확장성	Low	Medium	Medium
장치 편리성	Low	Low	Medium

2.2 객체기반 저장장치 시스템(OSDs : Object-based Storage Device systems)

기존의 저장장치 관리 기술들은 블록 단위의 입출력 관리와 파일시스템을 통한 파일 관리를 사용하여 디스크나 다른 저장장치들을 접근하고 관리한다. 하지만 이러한 방식은 논리적인 파일시스템과 물리적인 블록 단위 입출력 관리의 강결합을 초래하여 기존 저장장치 관리 기술의 확장성을 크게 제한한다. 따라서 기존 저장시스템의 논리적인 파일 관리 부분과 물리적인 블록 관리 부분을 효과적으로 분리하고, 보다 확장하기 쉬운 저장장치 시스템을 만들기 위해 객체기반 저장장치 모델이 제시되었다[1,2].

객체기반 저장장치 모델은 그림 1과 같이 블록기반의 저장시스템에서 파일과 디렉토리에 대한 논리적인 데이타 관리를 수행하는 부분과 디스크에 데이타 블록을 할당하고 매핑하는 것과 같은 물리적인 저장장치 관리 부분을 분리한다. 물리적인 저장장치 관리 부분을 OSD에서 수행하고, 기존의 블록기반 인터페이스를 객체기반 인터페이스로 변경한다. 이렇게 저장장치 관리 부분을 OSD에서 수행하도록 함으로써 저장시스템 서버의 작업부하를 줄일 수 있으며, 클라이언트들 간에 저장시스템의 물리적인 구조에 독립적인 데이타 공유가 가능하다. 또한 OSD에 저장되는 논리적인 객체에 대한 별도의 보안 정책을 설정할 수 있다. 그림 1은 객체기반 저장장치 모델의 개념을 나타낸다[1-3].

객체기반 저장장치 모델은 데이타 접근 방법, 데이타 속성 정보, 데이타 보안 방법 등을 포함할 수 있는 논리적인 객체를 데이타 저장 단위로 한다. 객체는 블록과 달리 크기의 제약이 없고, 저장장치의 내부 저장 구조에 종속적이지 않다. 객체기반 저장시스템은 이러한 객체 저장구조를 기반으로, 여러 저장장치에 쉽게 객체를 분산 처리해 데이타 입출력 성능을 향상시키고, 무제한적인 확장성을 제공한다.

객체기반 저장장치는 네트워크상에 분산된 클라이언트-서버 구조로 클라이언트 저장시스템, 메타데이타 서버, 객체 저장 서버로 구성된다. 클라이언트 저장시스템은 클라이언트 응용에 운영체제에 관계없이 동일한 파일 접근 인터페이스를 제공하는 인스톨 가능한 저장시스템 관리자(installable File system Manager: FM)로 구성된다.

메타데이타 서버(Meta-Data Server: MDS)는 OSD에 저장된 객체에 대한 메타데이타를 관리하며 FM에 대한 인증과 접근 제어를 담당한다. MDS는 FM이 적절한 권한을 갖고 있으면 객체에 접근할 수 있는 인증서를 발행한다. FM은 해당 인증서를 이용해 객체 저장 서버에 객체를 읽거나 쓰는 작업을 수행할 수 있다.

객체 저장 서버는 OSD에 FM으로 전달된 객체 데이타를 저장하고 관리하는 작업을 수행한다. FM이 OSD에 객체 데이타를 읽기 또는 쓰기 작업을 수행하는 경우, 병렬 동시 접근이 가능하므로 읽기/쓰기 성능을 크게 향상 시킬 수 있다. 또한 OSD의 데이타에 대한 능동적인 접근제어가 가능해지므로 보안성이 향상된다.

이와 같은 구조에 의해 클라이언트 응용은 기존과 동일한 파일 접근 인터페이스를 통하여 데이타를 읽고 쓸 수 있다. FM은 메타데이타 서버로부터 인증과 파일에 대한 메타데이타를 얻고 객체 저장 서버로부터 직접 파

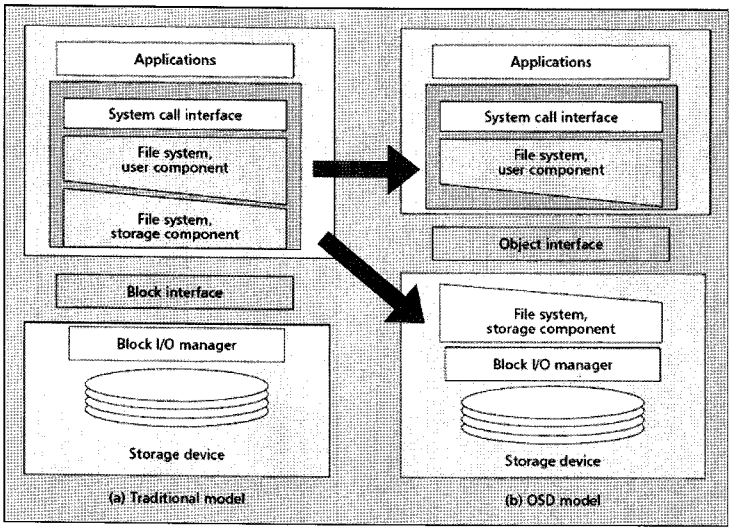


그림 1 기존 저장장치 모델과 객체기반 저장장치 모델

일 객체를 병렬 입출력을 통해 클라이언트 응용에 전달한다.

2.3 백업

백업은 전산장비의 고장이나 다른 불의의 사고에 대비하여 파일 또는 데이타베이스를 복사해 두는 행위를 말한다. 데이타의 백업은 대형 컴퓨터를 운영하는 대규모 사업체는 물론 개인 컴퓨터에서도 필수적이고 일상적인 업무이다. 백업 유형은 표 2와 같이 백업의 형태, 기능, 백업시스템의 구조 등에 따라서 백업 방식을 구분할 수 있다[8-11].

3. 제안하는 백업시스템 구조

현재 객체기반 저장시스템의 오류 복구를 위한 백업 기술은 객체기반 저장시스템의 특성이 고려되지 않은 기존 백업 기술을 그대로 활용하고 있다. 기존 백업 장치 관리 기법들은 객체기반 저장시스템의 기반이 되는 객체의 특성을 지원할 수 없으며 객체기반 저장시스템과 백업 저장장치간의 데이타 전송 과정에서 객체를 블록기반 또는 파일로 변환해 전송하기 때문에 백업 또는 복구에 많은 시간이 필요하다. 본 논문에서는 이와 같은 문제점들을 해결하기 위해 객체기반 저장시스템에 적합한 백업 및 복구 기술을 연구하고, 이를 적용시킨 백업 시스템을 설계하고 구현한다.

3.1 요구사항

객체기반 저장시스템의 구조는 기본적으로 SAN 시스템과 유사하다. 그러나 객체기반 저장시스템에서는 객체 단위로 데이타 접근이 이루어지기 때문에 객체기반 백업시스템은 SAN 환경의 백업시스템과 다른 구성과 처리 과정을 필요로 한다.

객체기반 저장시스템에서 백업 대상이 되는 데이타는 크게 저장시스템 메타데이타, 객체 메타데이타, 객체의 실제 콘텐츠 데이타로 나눌 수 있다. 객체기반 저장시스템에서 메타데이타들은 독립된 MDS에 위치하기 때문에 메타데이타와 객체 데이타를 한 번에 백업할 수 없다. 따라서 객체기반 백업시스템은 저장시스템 메타데이타와 객체 관련 데이타를 각각 백업하고 관리할 수 있는 기능을 지원해야 한다. 또한 복구 작업 수행 시 저장시스템 메타데이타와 객체 관련 데이타를 MDS와 OSD 장치에 각각 동시에 복구하는 기능도 지원해야 한다. 또한 백업 또는 복구 과정에서 MDS에서 관리하는 저장시스템 메타데이타와 OSD 장치에서 관리하는 객체 데이타 간의 동기화가 정확하게 이루어지지 않으면 전체 시스템의 데이타에 문제가 발생될 수 있으므로, 백업시스템은 백업/복구 작업이 진행되는 동안 외부 사용자가 접속하지 못하도록 오프라인 상태를 유지하거나, 온라인 상태에서 백업을 수행할 수 있도록 온라인 백업 기능을 지원해야 한다. OSD는 대용량 다수의 데이타를 저장하기 위한 저장장치 관리 기술이므로 대용량 데이타를 효과적으로 백업, 복구 할 수 있는 부분 백업이 필요하다. 또한 이러한 백업 데이타를 빠른 속도로 주고받을 수 있는 데이타 전송 기술이 요구된다.

기존 백업시스템과 다른 객체기반 저장시스템의 특징을 지원하기 위해 백업시스템은 다음과 같은 기능을 제공해야 한다.

- 객체 단위의 데이타 백업 및 복구
- 대상 객체의 접근 권한에 따른 백업 및 복구 권한 제어
- MDS의 메타데이타 정보 백업 및 복구
- OSD의 객체 메타데이타 정보 백업 및 복구
- MDS 및 OSD 수준에서 동기화된 온라인 백업
- MDS 및 OSD 수준에서의 부분 백업
- 백업 데이타 전송 속도를 보장하는 데이타 전송 기능

3.2 백업시스템 구조

객체기반 저장시스템에 적합한 백업시스템의 구조는 그림 2와 같다. 제안하는 백업시스템은 크게 메타데이타 서버에 위치하여 백업 인터페이스 명령, 백업 인터페이스 서버, 백업 클럭, 백업 작업 관리자로 구성되는 백업 관리자 부분, 백업 장치에 위치하여 백업 정보를 저장하기 위한 백업 장치 서버, 객체기반 저장시스템의 저장장

표 2 백업 방식의 구분

구분	백업 방식	
형태	전체 백업	백업을 수행할 시스템에 존재하는 모든 파일 백업
	부분 백업	이전의 백업 데이타와 비교하여 변경된 파일만 백업
대상	파일 기반 백업	파일 구조를 파악하여 파일과 디렉토리 구조를 백업
	장치 기반 백업	파일 구조를 무시하고 디스크블록을 백업
기능	온라인 백업	작업 수행 중에도 백업 지원
	오프라인 백업	백업을 위해 일반적인 작업을 중지
시스템 구조	S/W 구조	중앙집중식백업 지역관리백업
	H/W 구조	LAN 백업 LAN-Free 백업 Serverless 백업

치를 구성하는 각각의 OSD에 위치하여 저장된 객체 정보를 백업, 복구하는 백업 에이전트로 구성된다.

백업 인터페이스 명령은 백업 및 복구 작업을 수행하기 위한 명령을 내린다. 사용자는 백업 인터페이스 명령을 사용하여 백업 인터페이스 서버에 백업 및 복구 명령을 전달한다. 또한 백업 및 복구된 정보를 사용자가 확인할 수 있도록 백업 인터페이스 서버에 정보를 요청한다. 백업 인터페이스 서버는 사용자가 백업 인터페이스 명령을 통해 내린 백업 및 복구 작업 정보를 분석하여 작업 명령 파일을 작성하고, 이를 백업 작업 관리자에게 전달한다. 또한 백업 및 복구된 작업 정보를 백업 인터페이스 명령의 요청에 따라 전달하는 기능을 담당한다. 백업 클럭은 사용자가 지정한 시간에 동작되는 예약 및 스케줄 백업에 따라 해당 백업 작업 정보를 백업 작업 관리자에게 전달한다. 백업 클럭은 1분 단위로 동작하며, 매 동작시마다 수행할 작업 정보를 검색하고, 검색된 작업 정보를 전달한다.

백업 작업 관리자는 백업 및 복구 작업을 스케줄링 하고, 백업 장치 서버에 작업 명령을 전달하는 기능을 담당한다. 또한 백업 시 메타데이타 서버의 메타데이타를 백업하여 백업 장치 서버로 전달하고, 복구 시에는 백업된 메타데이타 정보를 다시 복구하는 기능도 담당한다. 백업 장치에 위치하는 백업 장치 서버는 백업된 객체 정보 및 메타데이타 정보를 백업 작업 단위 별로 관리한다. 백업 장치 서버는 백업 작업 관리자에게 작업 명령 정보를 수신하고, 이를 백업 에이전트에게 전달하여 처리한다. 백업 작업 시 백업 장치 서버는 백업 에이전트로부터 백업할 객체 데이타를 수신하여 지정된 위치에 저장하고, 백업된 객체 정보에 대한 히스토리 정보를 백업 작업 관리자에게 전달한다. 백업할 모든 객체 데이타를 백업 한 후 백업 관리자로부터 백업된 객체 데이타에 대한 메타데이타 정보 또한 수신하여 저장한다. 복구 작업 시 백업 장치 서버는 백업 에이전트에 백업된 객체 데이타를 전송한다. 또한 백업 작업 관리자에게 백업된 메타데이타 정보를 전송하여 복구하도록 한다.

객체기반 저장시스템의 OSD에 위치하는 백업 에이전트는 백업 장치 서버의 요청에 따라 OSD의 객체 정보를 백업 장치 서버로 전송하고 수신한다. 백업 시 백업 장치 서버는 백업 에이전트에게 백업할 객체 정보의 위치를 보낸다. 이 정보를 수신한 백업 에이전트는 지정된 객체 정보를 백업 장치 서버로 전송한다. 복구 시 백업 장치 서버는 백업 에이전트에게 복구시킬 객체 정보를 보낸다. 이 정보를 수신한 백업 에이전트는 이후 백업 장치 서버가 전송하는 객체 데이타를 지정된 위치에 저장한다.

4. 제안하는 백업시스템의 설계 및 구현

4.1 제안하는 백업시스템의 주요 특성

기존 백업시스템들은 대부분이 백업 대상인 파일 단위의 백업을 지원한다. 따라서 객체기반 저장시스템에서 사용되는 기존 백업시스템들은 대부분 FM에서 동작하게 된다. 하지만 객체기반 저장시스템은 기존의 파일시스템과 다르게 파일을 구성하는 실제 데이타, 즉 객체가 OSD라는 별도의 호스트에 존재한다. 또한 각 파일에 대한 메타데이타 정보를 MDS에 별도로 저장한다. 기존 백업시스템은 이들 OSD에 저장된 객체들을 FM에서

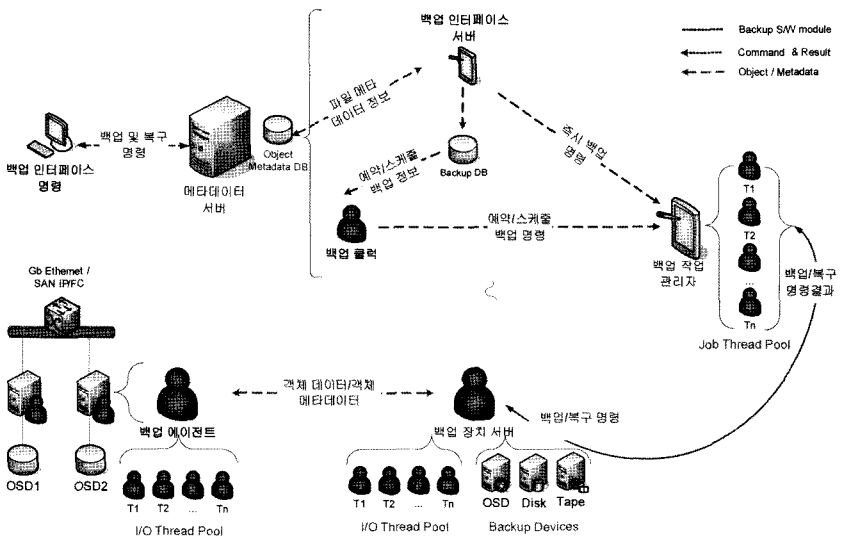


그림 2 백업시스템 구조

파일로 조합하고, 이를 백업 장치에 백업하는 순서로 백업이 진행된다. 이 경우 객체기반 저장시스템에 저장된 파일들을 안전하게 백업할 수 있지만, 각 파일에 설정된 보안 또는 접근 방법과 같은 메타데이타 정보가 손실되는 문제점이 발생한다. 또한 백업을 위해 파일을 구성하는 객체 정보를 FM까지 읽어야하는 추가 비용 문제가 발생한다.

제안하는 백업시스템은 이러한 객체기반 저장시스템의 특성을 고려하여 MDS에서 백업을 진행한다. 제안하는 백업시스템은 각 OSD에 백업 에이전트를 두고, 백업할 객체 정보를 직접 전달하여 백업을 진행한다. 또한 MDS에 있는 메타데이타 정보를 별도로 백업한다. 따라서 객체기반 저장시스템에 저장된 데이타를 보다 완벽하게 백업할 수 있다. 기존 파일시스템과 다르게 객체 정보를 FM까지 읽을 필요 없이, OSD에서 백업 장치로 직접 백업이 가능하기 때문에 보다 빠른 백업이 가능하다. 또한 복구 시에도 기존 백업시스템과 달리 복구된 파일 데이타에 대한 별도의 메타데이타 설정이 필요 없으므로 객체에 대한 보안 및 접근 방법, 그리고 객체에 대한 속성 정보와 같이 MDS에 저장되는 메타데이타 정보를 다시 설정할 필요가 없다. 따라서 복구 시간 역시 기존 백업시스템과 비교하여 단축될 수 있다.

4.2 백업 과정

백업 작업의 시작은 백업 인터페이스 명령으로부터 시작된다. 백업 인터페이스 명령을 통해 사용자는 백업에 필요한 여러 정보들을 입력한다. 백업에 필요한 정보로는 백업할 데이타가 위치한 경로, 백업할 데이타를 저장할 백업 장치 서버의 위치 및 백업 장치 서버 내의 저장 경로, 그리고 다양한 백업 옵션들로 구성된다. 사용자가 입력한 백업 정보는 백업 인터페이스 서버로 전달된다. 백업 인터페이스 서버는 사용자가 입력한 백업 정보의 유효성을 판단한다. 즉 백업할 데이타가 위치한 경로가 유효한지, 사용자가 입력한 백업 옵션들이 유효한지 확인한다. 유효한 백업 정보로 판별되면 백업 인터페이스 서버는 사용자가 입력한 백업 정보를 메타데이타 서버 내의 지정된 위치에 저장한다. 사용자가 입력한 백업 정보의 타입이 즉시 백업이면 백업 작업 관리자에게 바로 전달된다. 백업 정보의 타입이 예약 백업 또는 스케줄 백업이면 백업 정보는 백업 작업 관리자에게 바로 전달되지 않는다. 예약 백업 또는 스케줄 백업 정보는 사용자가 지정한 시간까지 대기하며, 백업 클록에 의해 지정된 시간에 백업 작업 관리자에게 전달되어 처리된다. 그림 3은 이와 같은 처리 과정을 보여준다.

백업 인터페이스 서버가 전달한 백업 작업 정보는 백업 작업 관리자에게 분석된다. 백업 작업 관리자는 백업 인터페이스 서버가 전송한 백업 정보를 바탕으로 백업할 객체에 대한 작업 명령 파일을 작성한다. 백업 작업 관리자는 백업 작업 파일을 작성한 후, 백업 장치 서버에 백업을 위한 기본적인 백업 정보를 먼저 전송한다. 기본적인 백업 정보를 수신한 백업 장치 서버는 백업 에이전트에게 백업 준비 명령을 전달하고, 각 에이전트의 상태가 백업 가능한지 확인한다. 각 에이전트로부터 상태 정보를 수신한 후 백업 장치 서버는 백업 작업 관리자에게 백업 준비가 완료되었음을 알린다. 백업 작업 관리자는 백업 준비의 완료를 수신하면 백업할 객체 정보를 순차적으로 백업 장치 서버에 전송한다. 그림 4는

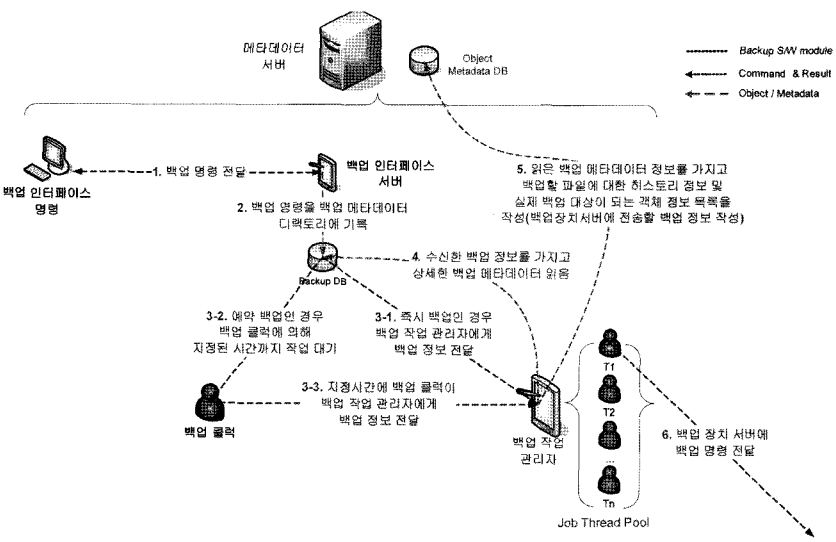


그림 3 백업시스템의 백업 작업 처리 과정(Step1)

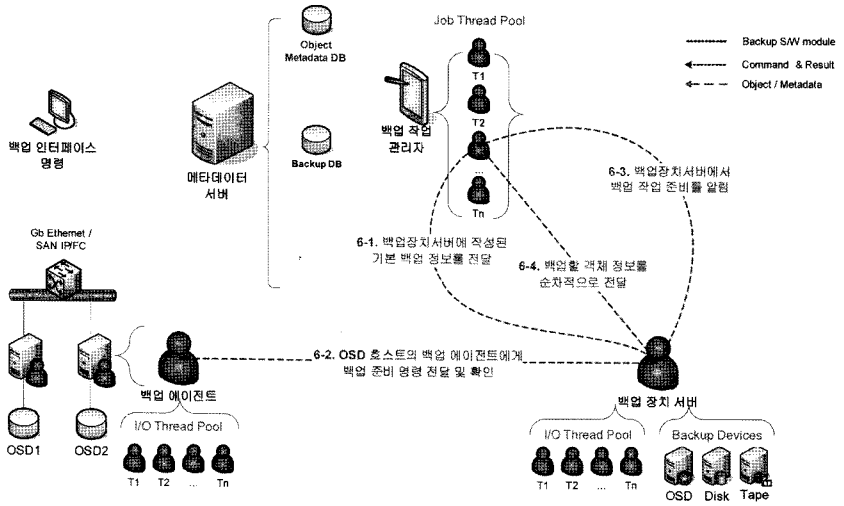


그림 4 백업시스템의 백업 작업 처리 과정(Step2)

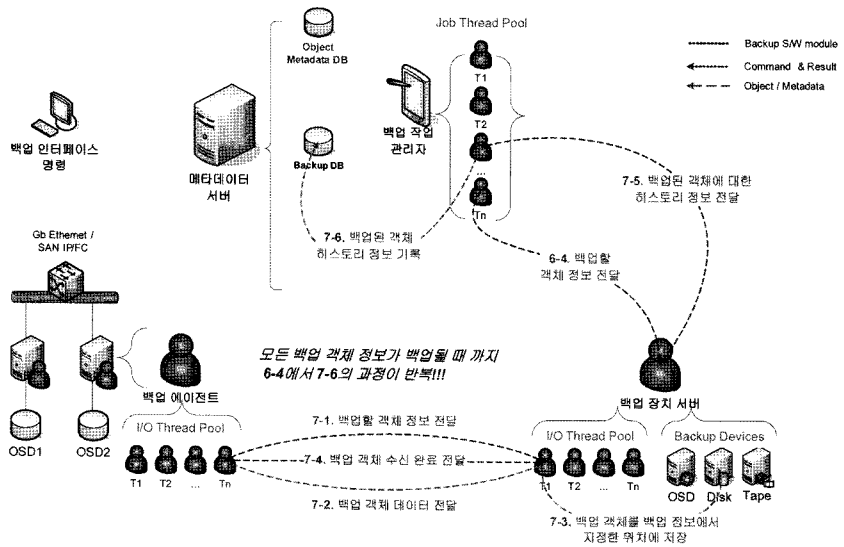


그림 5 백업시스템의 백업 작업 처리 과정(Step3)

이와 같은 처리 과정을 보여준다.

백업 인터페이스 서버로부터 백업할 객체 정보를 수신한 백업 장치 서버는 백업 에이전트에게 백업할 객체에 대한 정보를 전달한다. 백업 장치 서버로부터 백업할 객체에 대한 정보를 수신한 백업 에이전트는 백업할 객체 데이타의 위치를 확인한 후, 객체 데이타를 백업 장치 서버에게 전송한다. 백업 장치 서버는 백업 에이전트가 전송한 백업 객체 데이타를 백업 장치 서버의 지정된 위치에 저장한다. 백업 에이전트는 백업 장치 서버에게 하나의 객체 데이타에 대한 전송이 완료된 다음, 객체 데이타의 전송이 완료됐음을 백업 장치 서버에게 전송한다. 백업 장치 서버는 작업이 끝난 객체 데이타에 대한 백업 히스토리 정보를 백업 작업 관리자에게 전달하여 기록하도록 한다. 이 과정은 백업할 모든 객체 데이타가 백업 장치 서버로 백업될 때까지 반복된다. 그림 5는 이와 같은 처리 과정을 보여준다.

백업할 모든 백업 객체가 백업 장치 서버로 백업되면, 백업 장치 서버는 객체 데이타의 백업 완료를 백업 작업 관리자에게 전달한다. 이 정보를 수신한 백업 작업 관리자는 백업한 객체 데이타에 대한 메타데이타 정보를 TAR 백업을 사용하여 묶는다. 이후 이 정보를 백업 장치 서버에 전송한다. 이 정보를 수신한 백업 장치 서버는 해당 데이타는 지정된 위치에 저장하고 메타데이타 정보에 대한 백업이 완료되었음을 백업 작업 관리자에게

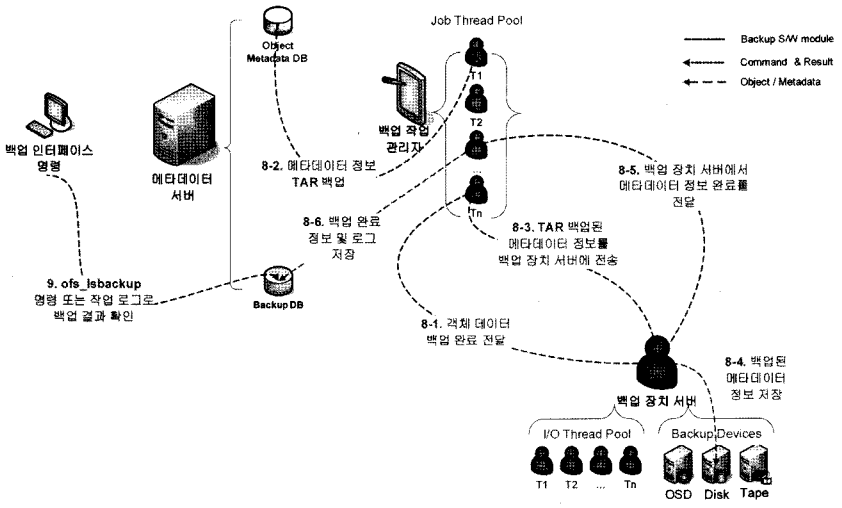


그림 6 백업시스템의 백업 작업 처리 과정(Step4)

전달한다. 백업 작업 관리자는 백업이 완료되었음을 기록한다. 그림 6은 이와 같은 처리 과정을 보여준다.

4.3 복구 과정

복구 작업의 시작 역시 백업 인터페이스 명령으로부터 시작된다. 백업 인터페이스 명령을 통해 사용자는 복구에 필요한 정보들을 입력한다. 복구에 필요한 정보로는 복구할 백업 작업의 작업 ID와 다양한 복구 옵션들로 구성된다. 사용자가 입력한 복구 정보는 백업 인터페이스 서버로 전달된다. 백업 인터페이스 서버는 사용자가 입력한 복구 정보의 유효성을 판단한다. 즉 복구할 백업 작업의 작업 ID가 유효한지 확인한다. 유효한 복구 정보로 판별되면 백업 인터페이스 서버는 사용자가 입력한 복구 정보를 메타데이타 서버 내의 지정된 위치에 저장한다. 사용자가 입력한 복구 정보는 백업 작업 관리자에게 바로 전달된다. 백업 작업 관리자는 백업 인터페이스 서버로부터 전달받은 복구 정보를 바탕으로, 복구할 백업 작업 정보를 검색하여 작업 명령 파일을 작성한다. 그림 7은 이와 같은 처리 과정을 보여준다.

백업 작업 관리자는 복구 작업 파일을 작성한 후, 백업 장치 서버에 복구를 위한 기본적인 복구 정보를 먼저 전송한다. 기본적인 복구 정보를 수신한 백업 장치 서버는 백업 에이전트에게 복구 준비 명령을 전달하고,

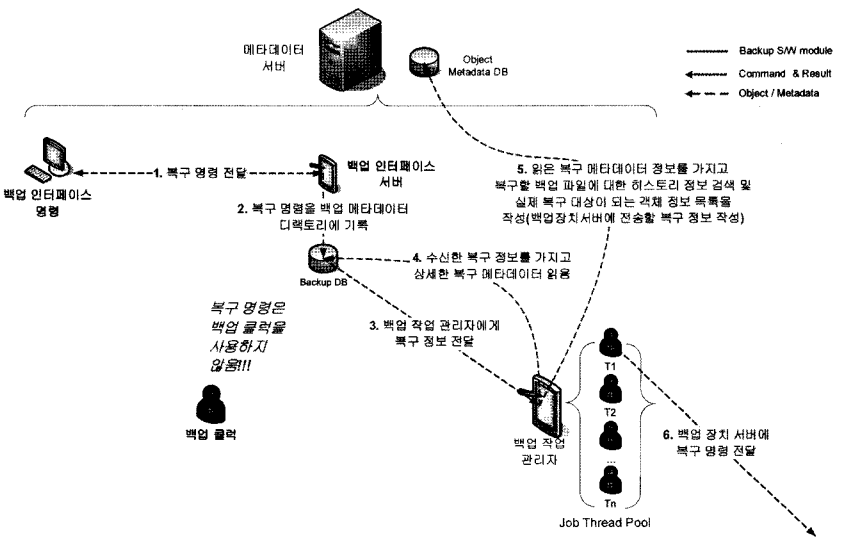


그림 7 백업시스템의 복구 작업 처리 과정(Step1)

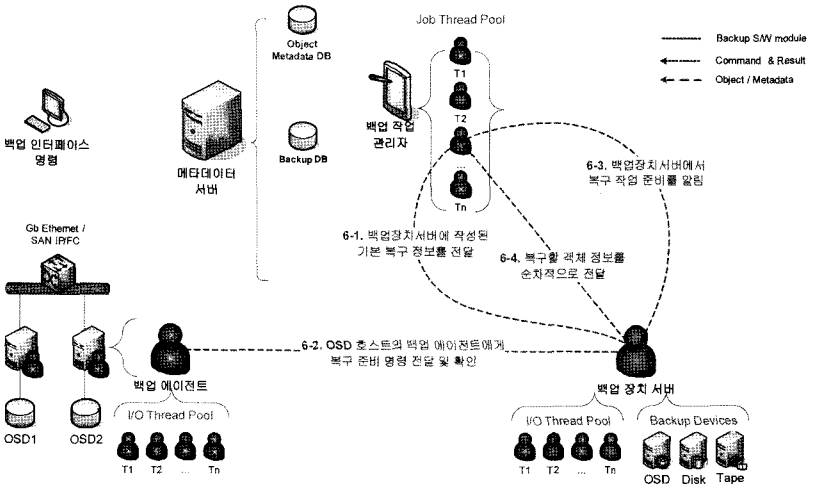


그림 8 백업시스템의 복구 작업 처리 과정(Step2)

각 에이전트의 상태가 복구 가능한지 확인한다. 각 에이전트로부터 상태 정보를 수신한 후 백업 장치 서버는 백업 작업 관리자에게 복구 준비가 완료되었음을 알린다. 백업 작업 관리자는 복구 준비의 완료를 수신하면 복구할 객체 정보를 순차적으로 백업 장치 서버에 전송한다. 그림 8은 이와 같은 처리 과정을 보여준다.

백업 인터페이스 서버로부터 복구할 객체 정보를 수신한 백업 장치 서버는 백업 에이전트에게 복구할 객체에 대한 정보를 전달한다. 백업 장치 서버로부터 복구할 객체에 대한 정보를 수신한 백업 에이전트는 복구할 객체 데이타의 위치를 지정한다. 백업 장치 서버는 백업 에이전트에게 백업 장치 서버에 백업되어있는 객체 데이타를 전송한다. 백업 에이전트는 백업 장치 서버로부터 하나의 객체 데이타에 대한 수신이 완료된 다음, 객체 데이타의 수신이 완료됐음을 백업 장치 서버에게 알린다. 백업 장치 서버는 작업이 끝난 객체 데이타에 대한 상태 정보를 백업 작업 관리자에게 전달한다. 이 과정은 복구할 모든 객체 데이타가 백업 에이전트로 전송될 때까지 반복된다. 그림 9는 이와 같은 처리 과정을 보여준다.

복구할 모든 객체 데이타가 백업 에이전트에 의해 원위치로 복구되면, 백업 장치 서버는 객체 데이타의 복구 완료를 백업 작업 관리자에게 전달한다. 이 정보를 수신한 백업 작업 관리자는 복구한 객체 데이타에 대한 메

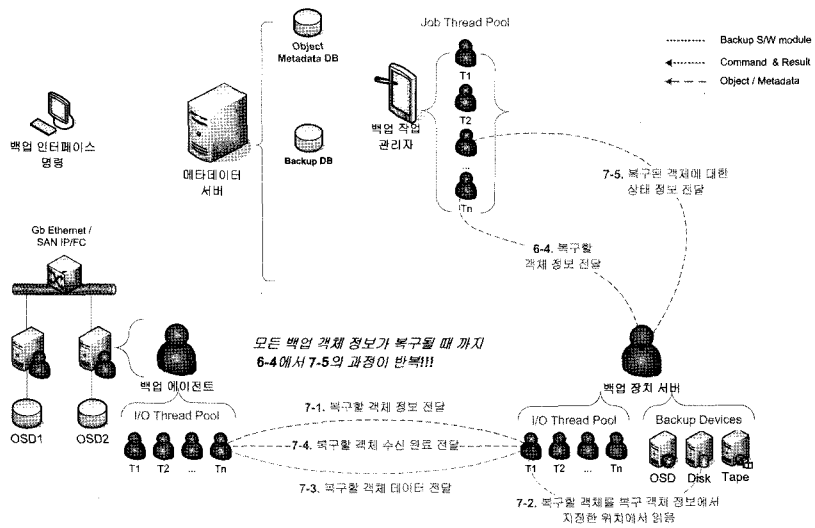


그림 9 백업시스템의 복구 작업 처리 과정(Step3)

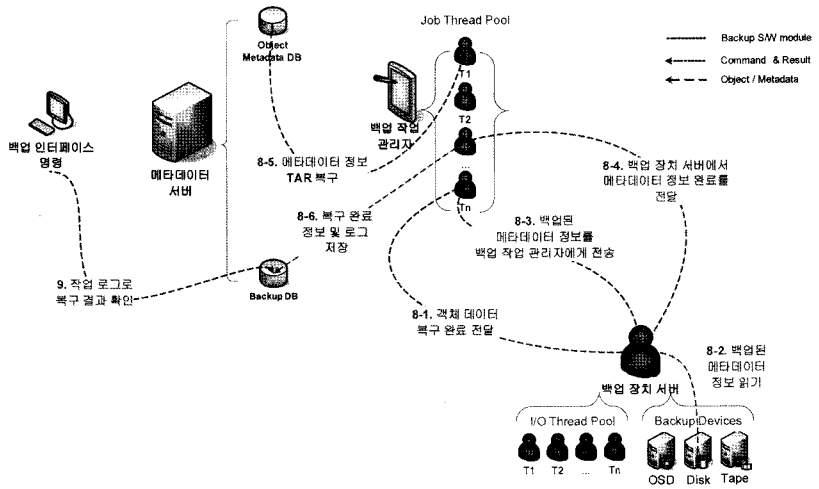


그림 10 백업시스템의 복구 작업 처리 과정(Step4)

타데이타 백업 정보를 백업 장치 서버로부터 전달받는다. 복구된 객체 데이타에 대한 메타데이타를 모두 수신한 후 백업 작업 관리자는 백업 장치 서버에서 메타데이타 정보 수신이 완료되었음을 알리고 이후 이 정보를 TAR 복구한다. 이 작업이 완료된 후, 백업 작업 관리자는 복구가 완료되었음을 기록한다. 그림 10은 이와 같은 처리 과정을 보여준다.

4.4 온라인 백업

온라인 백업은 사용자가 시스템을 사용하는 중에도 백업이 가능한 방법이다. 대부분의 SAN 백업시스템은 온라인 백업을 지원하며 이를 수행하기 위해 스냅샷(Snapshot), COW(Copy-On-Write) 등의 기술들을 사용하여 백업을 진행한다.

스냅샷은 특정 시점의 저장장치 상태를 그대로 복제하여 유지하는 방법으로 온라인 상태에서 특정 시점의 백업을 작성하는데 유용하다. 스냅샷은 백업 대상이 되는 데이타 셋과 동일한 크기의 저장 공간을 추가적으로 필요로 한다. 상용 백업시스템에서는 원본 데이타 셋을 특정 시점에서 스냅샷으로 만들고, 이후 변경 내용은 COW 방법을 사용하여 저장 공간 사용량을 줄이고 빠르게 스냅샷을 생성할 수 있도록 구현하고 있다.

COW 방법은 스냅샷 이후에 변경되는 블록들을 임시 저장소에 저장하여 스냅샷을 유지하는 방법이다. 스냅샷이 설정된 볼륨의 데이타에 대한 쓰기, 변경, 삭제 등의 요청이 발생하면 저장시스템의 I/O 관리자는 해당 요청을 볼륨의 비어 있는 공간을 이용해 처리하고 로그를 기록해 둔다. 이렇게 기록된 임시 정보들은 스냅샷 해제 명령을 입력하거나 임시저장 공간이 꽉 찬 경우, 스냅샷에 기록된 정보들을 원본 데이타에 적용하고 스냅샷을 해제한다.

제안하는 백업시스템의 온라인 백업은 스냅샷 관리 모듈과 백업 실행 모듈을 통해 처리된다. 스냅샷 관리 모듈은 MDS와 OSD에 온라인 백업을 위한 스냅샷을 생성하고, 스냅샷 데이타를 백업 실행 모듈에서 처리할 수 있도록 준비 작업을 한다. 백업 실행 모듈은 스냅샷 관리 모듈이 생성한 MDS와 OSD의 스냅샷을 사용하여 온라인 백업을 수행한다. 백업 실행 모듈은 기존의 백업 프로세스를 처리하는 모듈들과 동일하게 백업을 처리하기 때문에 MDS의 백업 작업 관리자, 백업 서버의 백업 장치 서버, OSD의 백업 에이전트를 그대로 사용한다. 따라서 백업 실행 모듈은 기존 모듈과 기능상 차이점은 없으며, 단지 백업할 원본 데이타를 스냅샷 관리 모듈에서 생성한 스냅샷에서 가져와 처리하는 것이 다르다. 백업 실행 모듈이 처리를 완료하면 스냅샷 관리 모듈은 생성한 스냅샷을 모두 해제하고 백업을 종료한다. 또한 스냅샷 관리 모듈은 스냅샷 생성 과정에서 오류가 발생하면 온라인 백업을 중지시키고 에러를 처리하는 기능도 수행한다.

그림 11과 12는 온라인 백업의 처리 흐름을 나타낸다. 백업 인터페이스 명령에 의해 온라인 백업 명령이 지시되면 백업 작업 관리자는 백업 작업을 생성한다. 백업 작업 관리자는 MDS에 온라인 백업을 위한 준비를 요청한다. MDS는 온라인 백업을 위해 현재 시스템 메타데이타에 대한 스냅샷을 생성한다. 그리고 MDS는 각 OSD에 위치하는 백업 에이전트에게 메시지를 보내 온라인 백업을 위한 준비를 명령한다. 각각의 백업 에이전트 역시 OSD에 저장된 객체 데이타에 대한 스냅샷을 생성하고, MDS에 스냅샷이 생성되었음을 알린다. 모든 OSD의 스냅샷이 생성되어 온라인 백업 준비가 완료되면 MDS의 백업 작업 관리자는 온라인 백업의 수행을

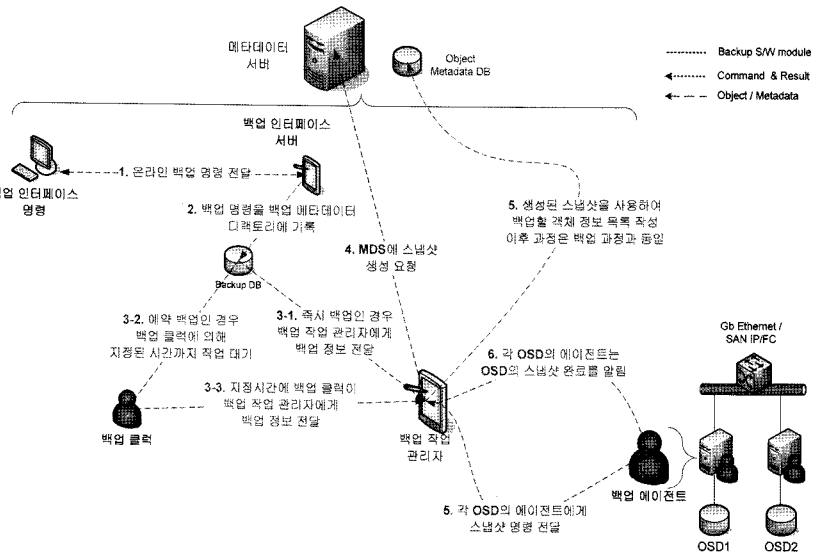


그림 11 온라인 백업 프로세스(Step 1)

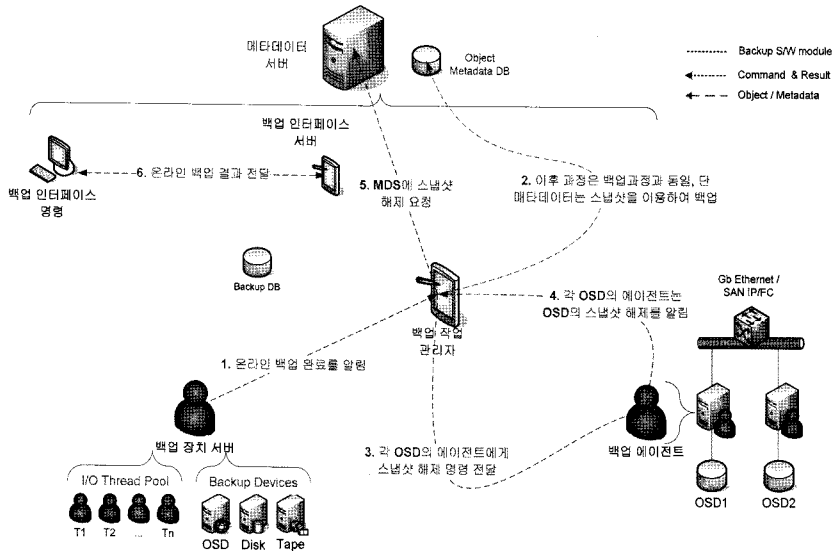


그림 12 온라인 백업 프로세스(Step 2)

알리고 백업 대상 객체 목록을 작성하여 백업 장치 서버에게 전달한다. 백업 장치 서버는 전달 받은 객체 목록을 이용해 백업을 기존 백업 프로세스와 동일하게 처리한다.

백업 작업이 완료되면 백업 장치 서버는 백업 작업 관리자에게 백업 결과를 전송한다. 백업 작업 관리자는 온라인 백업이 완료되면 각 OSD의 백업 에이전트에게 생성된 OSD의 스냅샷을 해제할 것을 지시한다. 각 OSD의 스냅샷이 모두 해제되고, 이 정보가 백업 작업 관리자에게 전달되는 즉시 MDS의 스냅샷이 해제되고 온라인 백업 프로세스의 전체 과정이 종료된다. 이후 온라인 백업의 결과를 백업 인터페이스에 전송한다.

4.5 부분 백업

백업 대상이 되는 파일 시스템 또는 데이타베이스의 크기가 매우 큰 경우 전체 백업을 수행하는 것은 시스템에 엄청난 부하를 가져온다. 따라서 주기적으로 전체 백업을 실시하되, 백업 수행의 부하를 줄이기 위해 전체 백업 이후 변경된 데이타에 대한 백업만 수행하는 부분 백업 기법을 사용한다. 부분 백업은 대용량 데이타에 대한 백업 시간을 매우 효과적으로 단축할 수 있고, 백업

장치의 저장용량을 적게 사용할 수 있기 때문에 객체기반 저장시스템과 같은 대용량 저장시스템에 대한 백업 시스템에서는 필수적으로 지원해야 하는 기능이다.

부분 백업을 수행하기 위해서는 먼저 전체 백업이 수행되어야 한다. 전체 백업이 수행된 이후의 데이타에 대한 변경은 모두 부분 백업 관리 모듈에 의해 감시되고, 부분 백업이 수행되는 경우 변경된 데이타에 대한 부분만 백업을 수행한다. 부분 백업 기법은 일정 주기로 전체 백업을 한 번 수행한 후, 다음 전체 백업 주기가 되기 전 까지는 전체 백업 이후 변경된 부분에 대한 정보만 추가적으로 백업을 수행하는 형태로 구현한다. 부분 백업은 마지막 전체 백업 이후 변경된 모든 데이타들을 찾아 매번 백업을 수행하는 차등(Differential)백업과 가장 마지막 부분 백업 이후 변경된 데이타들만 찾아 백업을 수행하는 점진(Incremental)백업으로 나눌 수 있다. 차등 백업은 점진 백업에 비해 부분 백업을 수행하는데 걸리는 시간이 상대적으로 더 길다. 하지만 복구 시 가장 최근 전체 백업 본을 먼저 복구하고 가장 마지막 차등 백업 본을 복구하는 2단계로 복구가 끝나기 때문에, 전체 백업 본을 복구하고 이후 모든 점진 백업 본을 차례로 모두 복구해야하는 점진 백업에 비해 짧은 시간에 복구를 완료할 수 있다.

객체기반 저장시스템에서는 두 개의 레벨에서 부분 백업의 처리가 가능하다. 먼저 MDS 메타데이타 레벨에서 부분 백업을 처리할 수 있다. 즉 메타데이타의 변경 시간을 기준으로 부분 백업을 진행한다. 메타데이타 레벨에서의 부분 백업은 변경이 발생한 파일 단위의 부분 백업을 진행하기 때문에 처리가 간단하지만 백업 시간이 늘어나고, 변경되지 않은 파일의 일부 객체 데이타를 백업하게 되므로, 백업 장치의 저장 공간을 많이 사용하게 된다.

OSD 객체 레벨에서도 부분 백업을 처리할 수 있다. OSD 객체 레벨에서의 부분 백업은 변경된 객체 데이타만 백업하므로 백업 시간과 백업 장치의 저장 공간을 더 효율적으로 사용할 수 있으나, MDS와 OSD간의 시스템 시간 동기화 및 변경 정보에 대한 실시간 메타데이타 업데이트와 같은 추가적인 처리과정이 필요하게 된다. 따라서 본 논문에서는 MDS 메타데이타 레벨에서의 부분 백업을 지원한다.

MDS 메타데이타 레벨에서 부분 백업을 지원하기 위해 기존 백업 작업 관리자 모듈에 부분 백업 관리 모듈이 추가된다. 부분 백업이 MDS 메타데이타 레벨에서 처리되는 경우 백업 장치 서버나 백업 에이전트 등은 추가적인 모듈을 필요로 하지 않는다. 부분 백업 관리 모듈은 MDS의 메타데이타 정보를 이용해 부분 백업의 전체 백업 또는 이전 점진/차등 백업 이후 변경된 데이타들을 찾아내고 백업할 대상을 결정하여 백업 관리자에게 넘겨준다. 백업 관리자는 넘겨받은 백업 목록을 기존 백업 처리 모듈들을 그대로 활용하여 처리한다.

그림 13과 14는 부분 백업 프로세스를 나타낸다. 전체 백업 프로세스의 경우 앞 절의 백업 과정을 통해 이루어지므로 설명을 생략한다. 부분 백업의 경우 스케줄러에 의해 백업이 수행되는 경우가 대부분이 된다. 이전 전체 백업 기록이 존재하는 경우 동일한 경로의 백업 작업이 시작되면 파일 메타데이타 백업 후 변경이 발생한 파일들의 정보를 분류해 낸다. 이후 백업 작업은 변경이 발생한 파일들에 대해서만 이루어진다. 백업 작업 관리자는 변경이 발생한 파일들에 대해서만 백업 대상

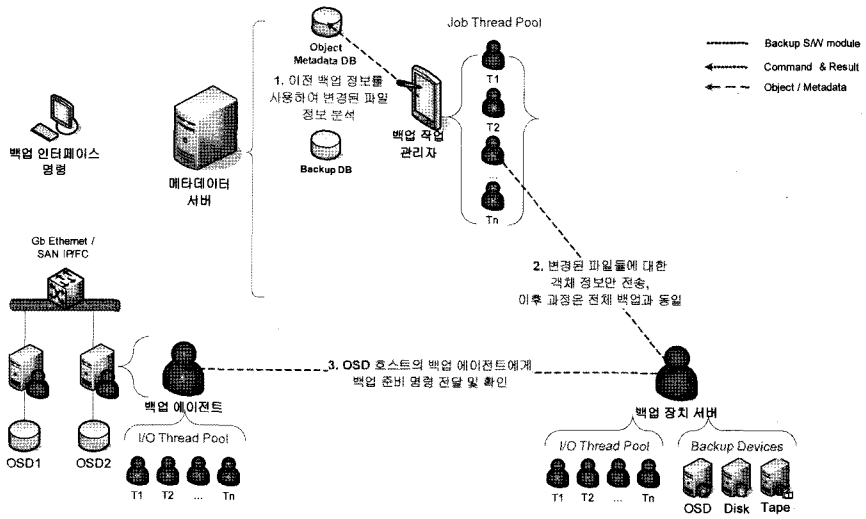


그림 13 부분 백업 프로세스(Step 1)

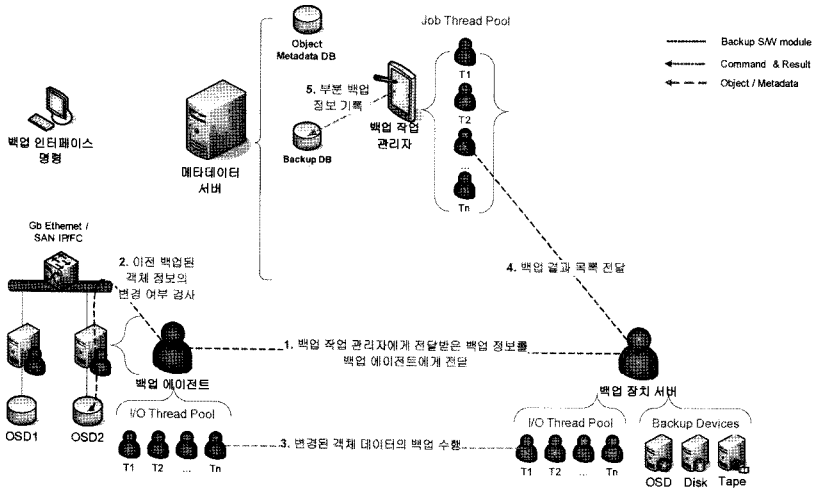


그림 14 부분 백업 프로세스(Step 2)

객체 목록을 작성하고, 이를 백업 장치 서버에게 전송하여 백업을 수행한다. 백업 대상 객체 목록에는 이전 작업에 의해 객체가 백업된 시간 정보를 포함한다.

백업 장치 서버는 전달받은 객체 목록을 분석하고 백업 작업을 수행한다. 백업 장치 서버는 객체 데이타 요청과 함께 해당 객체의 이전 백업 정보를 OSD의 백업 에이전트에게 전송한다. OSD의 백업 에이전트는 전송받은 객체의 이전 백업 정보와 현재 객체의 상태 정보를 비교하여 변경이 발생하지 않았으면, 변경 사항이 없음을 백업 장치 서버에게 알린다. 백업 장치 서버는 해당 객체 정보는 백업하지 않고 목록에서 상태 정보만 업데이트 한다. OSD의 백업 에이전트는 변경이 발생한 객체에 대해서는 기존 백업 작업과 마찬가지로 객체 데이타를 백업 장치 서버에게 전송하여 백업을 수행한다. 백업 장치 서버는 작업이 완료된 후 백업 결과 목록을 백업 작업 관리자에게 전송한다. 백업 작업 관리자는 수행된 부분 백업에 대한 정보를 기록한다. 이후의 처리과정은 백업 과정과 동일하다.

4.6 백업 데이타 전송을 위한 전송 기법

객체기반 저장시스템은 대용량의 데이타를 효과적으로 처리하기 위해 제안된 시스템이므로 백업되는 데이타 역시 대용량인 파일들이 될 수 있다. 따라서 백업이나 복구 진행 시, 백업 장치 서버와 OSD 사이에 백업 데이타를 빠른 속도로 전송할 수 있는 방법이 필요하다.

따라서 제안하는 백업시스템은 백업 작업 관리자와 백업 장치 서버, 다수의 백업 에이전트 사이에 동시작업이 가능하도록 쓰레드(Thread)를 사용하여 멀티세션을 유지한다. 이를 통해 다수 파일의 동시 전송과 작업 제어가 가능하다. 그림 15는 제안하는 쓰레드의 구조를 나타낸다. 각 프로세스는 크게 세 개의 다른 기능을 수행

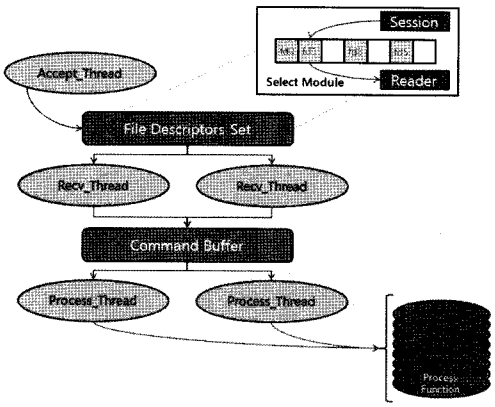


그림 15 성능 향상을 위한 프로세스의 쓰레드 구성도

하는 쓰레드로 구성된다. 세 개의 쓰레드는 멀티세션을 연결해주는 Accept_Thread와 명령어를 수신하는 Recv_Thread, 수신한 명령어를 처리하는 Process_thread로 구성된다. 또한 소켓 연결된 세션의 파일 디스크립터를 저장하는 File Descriptors Set과 패킷을 저장하는 Command Buffer가 있다. 이러한 구성을 기본으로 백업 및 복구를 위해 여러 개의 데이타를 전송할 때 다수의 쓰레드를 이용하여 멀티세션을 생성하고 전송속도를 극대화 한다. 또한 백업이 이루어지고 있는 동안 필요한 제어 작업이나 추가 정보를 요청할 수 있도록 하여 멀티프로세싱을 극대화했다.

또한 파일의 전송 속도를 늘리기 위해 소켓 통신을 통한 파일 전송 시 일반적으로 사용되는 read()와 write() 시스템 호출 함수를 사용하지 않고 sendfile() 시스템 호출 함수를 사용한다. 그림 16은 read(), write() 함수 사용 시의 처리 과정을 나타낸다. 위쪽은 함수 호

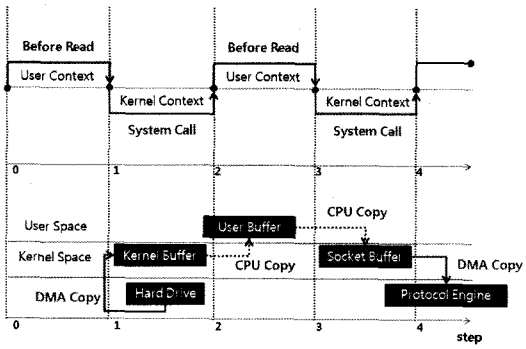


그림 16 read(), write() 함수를 사용한 데이타 전달 과정

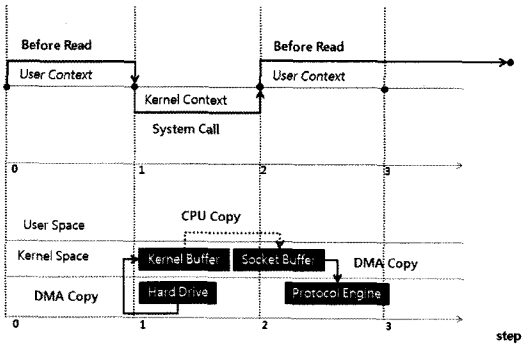


그림 17 sendfile() 함수를 사용한 데이타 전달 과정

출 시 사용자(user)/커널(kernel) 간의 문맥교환(context switching)을 보여주며, 아래쪽은 함수 호출에 따른 데이타의 복사과정을 보여주고 있다. 보통 이 2개의 시스템 호출을 가지는 부하가 크지 않다고 생각할 것이다. 그러나 실제로 이 2개의 시스템 호출이 진행 되면, 데이타는 적어도 4번 복사가 된다. 또한 그 만큼 사용자/커널 간에 문맥교환이 실행된다. 저장장치에 저장된 파일 데이타를 읽는 과정에서 저장장치의 데이타를 커널 버퍼(buffer)에 복사하고 이를 다시 사용자 버퍼에 복사한다. 사용자 버퍼에 복사된 데이타는 보내질 소켓(socket) 버퍼에 복사하여 보내진다. 소켓 버퍼에서는 복사된 데이타를 프로토콜을 사용하여 전송하게 된다. 따라서 최소 4번의 데이타 복사가 이루어지며, 이는 대용량 데이타 전송 시, 매우 많은 작업 시간을 낭비하게 한다.

제안하는 백업시스템은 백업 데이타 전송 시 sendfile() 시스템 호출 함수를 사용한다. sendfile() 함수는 두 개의 파일 디스크립터를 사용하여 자료의 전송을 보다 간단히 수행하시 위해 소개되었다. sendfile() 함수는 데이타의 복사 횟수를 감소시킬 뿐만 아니라 사용자/커널 간의 문맥교환도 감소시킨다. 그림 17은 sendfile() 호출 시의 처리 과정을 나타낸다. sendfile() 함수는 이전의 사용자 버퍼로 복사하는 과정이 없이 바로 커널 버퍼에서 소켓 버퍼로 바로 복사되기 때문에 파일을 전송하는데 속도가 이전의 방법 보다 빠르다.

표 3 객체기반 저장시스템을 위한 백업시스템의 구현 환경

항 목	환 경
CPU	Intel Xeon 2.4Ghz × 1
RAM	1GB
HDD	80GB ATA × 1
OS	Linux (Fedora Core4, Kernel ver.2.6.10)
RAID	RAID 구성 없음
Compiler	gcc ver.3.2.2-5
노드 수	6 노드
네트워크	TCP/IP 1Gb ethernet switch

5. 실험 및 성능 평가

5.1 구현 환경

본 논문에서 정의한 객체기반 백업시스템에 대한 요구사항 및 설계 내용의 적절성을 검증하고, 기존 백업시스템 대비 성능 향상 정도를 평가하기 위해 백업시스템을 구현하였다. 구현한 백업시스템은 한국전자통신연구원에서 개발한 객체기반 클러스터 파일시스템인 OASIS 시스템에서 백업 및 복구 기능을 지원한다[12]. OASIS는 일반적인 객체기반 분산 저장시스템과 유사하게 MDS, OSD, FM으로 구성되어 있는 객체기반 클러스터 저장시스템이다. 구현한 객체기반 백업시스템의 백업 인터페이스 및 인터페이스 서버, 백업 클럭, 그리고 백업 작업 관리자는 MDS에 함께 설치되어 동작하고, 백업 에이전트는 OSD에 설치된다. 백업 장치 서버는 별도의 백업 전용 시스템 또는 FM에 설치되어 백업 및 복구를 테스트하도록 한다.

백업시스템의 구현 환경은 표 3에 정리하였다. 구현은 총 6개의 노드에 설치된 OASIS 시스템에서 이루어졌다. MDS와 FM은 각각 별도의 노드에 설치되며, 나머지 노드들에는 모두 OSD가 설치되어 하나의 OASIS 시스템을 구성하였다. FM이 설치된 노드에는 백업 장치 서버가 설치되어 백업 데이타를 저장한다.

5.2 성능 평가

구현한 백업시스템의 성능을 평가하기 위해 다양한 종류의 백업 데이타를 사용하여 제안하는 백업시스템의 성능을 비교, 분석한다. 성능 평가는 백업 작업의 평균 작업 시간을 기준으로 수행한다. 백업 데이타는 다양한 동영상 및 텍스트 파일을 혼합하여 구성하며, 백업 데이타를 다양한 크기로 조정하여 성능 평가를 수행한다. 비교 대상은 Linux 환경에서 백업 목적으로 사용되는 TAR와의 평균 작업 시간을 비교한다. 이는 FM에서

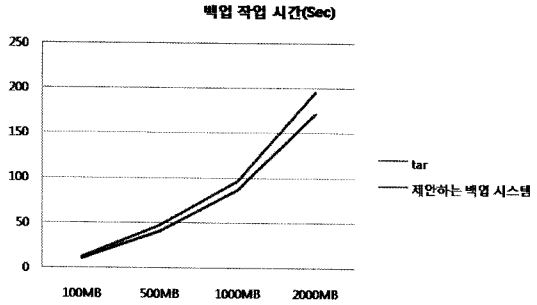


그림 18 제안하는 백업시스템의 백업 작업 시간(sec)

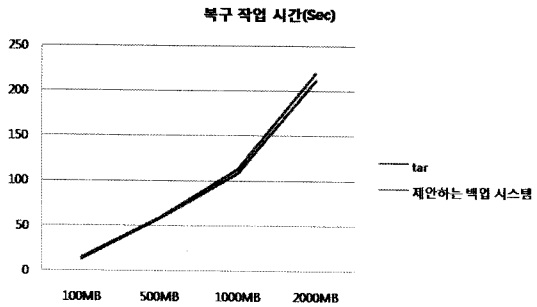


그림 19 제안하는 백업시스템의 복구 작업 시간(sec)

TAR를 통해 백업을 진행하는 것이 일반 백업시스템에서 백업을 진행하는 것과 유사하기 때문이다.

그림 18은 제안하는 백업시스템에서 다양한 유형의 실험 데이타에 대해 백업을 수행한 결과를 비교한다. 제안하는 백업시스템의 작업 시간이 TAR와 비교하여 우수함을 보인다. 제안하는 시스템은 다수의 쓰레드를 사용하여 백업할 객체 정보의 동시 전송 및 작업 제어가 가능하다. 따라서 대용량의 데이타를 전송하는 경우 TAR와 비교하여 나은 성능을 보인다.

객체기반 저장시스템은 파일을 구성하는 데이타가 각 OSD에 객체 단위로 저장된다. 즉 FM과 독립된 별도의 저장장치에 파일의 실제 내용을 구성하는 데이타들이 저장된다. TAR를 비롯한 기존 백업시스템은 FM에서 파일 단위의 백업을 지원한다. 따라서 파일을 구성하는 모든 객체 정보를 OSD들로부터 FM까지 읽은 후 파일로 조합하여 백업 장치로 백업을 수행한다. 반면 제안하는 백업시스템은 TAR나 기존 백업시스템보다 낮은 수준인 객체 단위의 백업을 수행한다. 각 OSD에 백업 에이전트를 두고 백업할 파일의 객체를 직접 백업 장치에 전송하여 백업한다. 따라서 TAR나 기존 백업시스템처럼 객체 정보를 파일로 조합하여 백업할 필요가 없다. 이를 통해 제안하는 백업시스템은 백업 작업 시간을 보다 단축할 수 있다.

그림 19는 제안하는 백업시스템에서 다양한 유형의 실험 데이타에 대해 복구를 수행한 결과를 비교한다. 복구의 경우 백업과 달리 작업 시간에 있어서 거의 차이를 보이지 않는다. 하지만 TAR의 경우 객체의 특성을 고려하지 않고 백업된 데이타를 복구하는 것이므로, 복구된 객체에 대한 후처리 작업이 반드시 필요하다. 즉 TAR 백업된 객체 정보는 MDS에 저장된 메타데이타에 대한 백업을 지원하지 않으므로 객체에 대한 접근 방법 및 보안 방법, 그리고 객체에 대한 속성 정보와 같이 MDS에 저장되는 메타데이타 정보를 다시 설정해야 한다. 하지만 제안하는 백업시스템의 경우 객체의 특성을 고려하여 MDS에 저장된 객체의 메타데이타 정보를 함께 백업해두기 때문에 TAR와 달리 후처리 작업이 필요하지 않다. 따라서 전체적인 작업 시간에서 기존의 백업 방법보다 우수한 작업 시간을 보인다.

표 4는 제안하는 백업시스템에서 제공하는 기능들을 다른 상용 백업시스템과 비교한 것이다. 일반적인 상용 백업시스템은 아직 객체기반 저장시스템의 특징을 고려하여 설계되지 않았기 때문에 객체기반 데이타 백업을 지원하지 않고 있다. 하지만 제안하는 시스템은 객체기반 데이타 백업을 지원할 뿐만 아니라 일부 기능의 수정을 통해 일반 파일시스템에 대한 백업 역시 지원이 가능하다. 또한 다른 상용 백업시스템이 제공하는 온라인 백업 및 부분 백업 등의 기능들을 모두 지원한다.

6. 결론 및 향후 연구

본 논문에서는 객체기반 저장시스템을 위한 효과적인 백업시스템의 요구사항을 정의하고, 객체기반 저장시스템의 특징을 반영한 백업시스템을 설계하였다. 그리고

표 4 백업시스템의 특징 비교(O : 지원함, X : 지원 못함, △ : 부분지원)

기능	Veitas NetBackup	Legato NetWorker	HP OmniBack-II	메릴랜드 AMANDA	제안하는 백업소프트웨어
중앙 집중식 관리	O	O	O	O	O
파일시스템 백업	O	O	O	O	△
객체기반 백업	X	X	X	X	O
부분 백업	O	O	O	X	O
장치 및 미디어 관리	O	O	O	O	△
온라인 백업	O	O	O	X	O

설계한 내용을 바탕으로 객체기반 백업시스템을 객체기반 저장시스템에서 구현하였다.

기존 백업 장치 관리 기법들은 객체기반 저장시스템의 기반이 되는 객체의 특성을 지원하지 못한다. 또한 객체기반 저장시스템과 백업 저장장치간의 데이타 전송 과정에서 객체를 블록기반 또는 파일로 변환해 전송하기 때문에 백업 또는 복구에 많은 시간이 필요했다.

본 논문에서는 이와 같은 문제점들을 해결하기 위해 객체기반 저장시스템의 특성을 고려한 백업 및 복구 기술에 대한 연구를 수행하고, 연구 결과를 기반으로 백업 시스템을 설계하고 구현하였다. 본 논문에서 설계하고 구현한 백업시스템은 기존 시스템에서 대표적인 백업 방법인 TAR와 비교하여 우수 또는 동등 이상의 성능을 보여준다. 또한 상용 소프트웨어와 비교하여 그들이 제공하는 다양한 형태의 기능을 제공한다.

향후 연구에서는 본 논문에서 개발한 객체기반 백업 시스템의 성능을 보다 개선하고, 그 기능 확장을 위한 추가적인 연구를 수행한다.

참 고 문 헌

[1] Thomas M. Ruwart, "OSD: A Tutorial on Object Storage Devices," Proceedings of the 19th IEEE / 10th NASA Goddard Conference on Mass Storage Systems and Technologies, 2002.

[2] Feng Wang, Scott A. Brandt, Ethan L. Miller, Darrell D. E. Long, "OBFS: A File System for Object-Based Storage Devices," Proceedings of the 21th IEEE / 12th NASA Goddard Conference on Mass Storage Systems and Technologies, April 2004.

[3] Storage Network Industry Association, "http://www.snia.org/ home"

[4] Lustre, "http://www.lustre.org/"

[5] Girish Moodalbail, Nagapramod Mandagere, Aravindan Raghuveer, Sunil Subramanya, and David H.C. Du, "Backup Aware Object based Storage," DTC Research Report 2007/25, 2007.

[6] 서대화, 민병준, 임기욱, "네트워크 연결형 스토리지의 기술 동향", 정보과학회지, 제19권, 제3호, pp. 6-13, 2001.

[7] W. C. Preston, Using SANs and NAS, pp. 66-188, O'REILLY, 2002.

[8] A. L. Chervenak, V. Vellanki, Z. Kurmas and V. Gupta. "Protecting File System, A Survey of Backup Techniques," Proceedings of the 6th NASA Goddard Conference on Mass Storage Systems and Technologies, 1998.

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

[10] 백업 및 복구 서비스 구성, http://www.microsoft.com/korea/technet/security/guidance/secmod201.asp

[11] Michael Kaczmarski, Tricia Jiang, David A. Pease, "Beyond backup toward storage management," IBM Systems Journal, Vol.42, No.2, pp. 322-337, 2003.

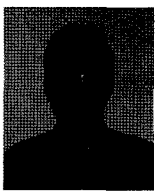
[12] ETRI, "http://www.etri.re.kr/"

윤 종 현

2003년 충북대학교 전기전자및컴퓨터공학부 정보통신공학(공학사). 2005년 충북대학교 정보통신공학과(공학석사). 2005년 3월~현재 충북대학교 정보통신공학과 박사과정. 관심분야는 DBMS, 저장시스템, 시공간 색인 구조, 이동 객체 DBMS, 센서 네트워크 등

이 석 재

2000년 충북대학교 정보통신공학과(공학사). 2002년 충북대학교 정보통신공학과(공학석사). 2006년 충북대학교 정보통신공학과(공학박사). 2006년~2007년 7월 충북대학교 BK21 Post-Doc. 2007년 8월~현재 애플테크(주) 기술연구소장. 관심분야는 DBMS, 저장 시스템, 주기억장치 DBMS, 센서 네트워크 등



장 수 민

1997년 목포대학교 전산통계학과(이학사). 1999년 충북대학교 정보통신공학과(공학석사). 2007년 8월 충북대학교 정보통신공학과(공학박사). 2007년 9월~현재 BK21 Post Doc. 관심분야는 온라인 게임서버, 분산처리, 데이타베이스, 정보검색 등

유 재 수

1989년 2월 전북대학교 컴퓨터공학과 공학사. 1991년 2월 한국과학기술원 전산학과 공학석사. 1995년 2월 한국과학기술원 전산학과 공학박사. 1995년 2월~1996년 8월 목포대학교 전산통계학과 전임강사. 1996년 8월~현재 충북대학교 전기전자컴퓨터공학부 및 컴퓨터정보통신연구소 정교수. 관심분야는 데이타베이스 시스템, 정보검색, 멀티미디어 데이타베이스, 분산 객체 컴퓨팅 등

김 홍 연

1992년 인하대학교 통계학과(이학사). 1994년 인하대학교 전자계산학과(공학석사) 1999년 인하대학교 전자계산학과(공학박사). 1999년 8월~현재 한국전자통신연구원선임연구원. 관심분야는 네트워크스토리지및파일시스템, 클러스터컴퓨팅, DBMS

김 준

1984년 부산대학교 계산통계학과(이학사). 1986년 KAIST 전산학과 석사(공학석사). 1986년~현재 한국전자통신연구원 책임연구원. 관심분야는 DBMS, 트랜잭션 처리, 클러스터 스토리지 SW