대용량 분산 파일 시스템 상에서의 메타데이터 서버 분석

김소연 O, 노홍찬, 박치현, 박상현
연세대학교 컴퓨터과학과
{sykim, fallsmal, tianell, sanghyun}@cs.yonsei.ac.kr

Analysis of Metadata Server on Clustered File Systems

So-Yeon Kim O, Hong-Chan Roh, Chi-Hyun Park, Sang-Hyun Park
Dept. of Computer Science, Yonsei University

요 약

인터넷 서비스가 갈수록 대형화됨에 따라 발생하는 대용량의 데이터를 저장할 수 있는 저장 공간이 요구된다. 따라서 안정된 인터넷 서비스를 제공하고 대용량 데이터를 처리할 수 있는 대용량 분산 파일 시스템의 성능 개선이 필요하다. 대용량 분산 파일 시스템의 성능은 메타데이터 서버가 얼마나 효율적으로 데이터 서버들에 대한 메타데이터를 관리하고 클라이언트의 요청을 분산시킬 수 있는지에 달려있다. 메타데이터 서버의 향상된 설계를 위해서 최근에 각광받고 있는 메타데이터 서버에 대한 분석이 필요하다. 따라서 최근 연구, 개발되어 주목받고 있는 대용량 분산 파일 시스템인 GFS, HDFS, LakeFS을 중심으로 메타데이터 서버를 분석하고자 한다.

1. 서 론

인터넷 서비스를 사용하기 위해 수백 명의 클라이언트가 동시에 접속함에 따라 발생하는 대용량 데이터를 저장할 수 있는 저장 공간이 요구된다. 따라서 안정된 인터넷 서비스와 대용량 데이터를 처리할 수 있는 대용량 분산 파일 시스템이 필요하다.

대용량 분산 파일 시스템은 수천 개의 대용량 하드디스크를 가지고 있으며 메타데이터 서버, 데이터 서버, 클라이언트로 구성되어 대용량 분산 응용 프로그램을 효율적으로 구현할 수 있다. 대용량 분산 파일 시스템의 메타데이터 서버는 메타데이터를 관리하고 클라이언트 및 데이터 서버와 연계되어 작동함으로써 전체 시스템을 통솔하는 역할을 한다. 이러한 메타데이터 서버의 구조와 연산 처리 방법들은 메타데이터 서버가 다수의 클라이언트의 요청을 처리할 때 나타나는 병목 현상을 해결하며, 전체 시스템 성능 및 저장 공간에 대한 확장성, 서버 오류에 대한 복구 능력을 향상시키는데 기여한다.

최근에 다양한 방법으로 메타데이터 서버들 간에 부하를 균등하게 분산시켜 성능을 향상시킨 대용량 분산 파일 시스템이 연구, 개발되고 있다. 예로는 GFS(Google file system)[1], HDFS(Hadoop file system)[2], LakeFS[3], Ceph[4], Lustre[5], OASIS[6] 등이 있다. 특히 GFS, HDFS, LakeFS는 기존의 하드웨어에 대한 호환성이 높으며, 응용 프로그램에서 빠른 데이터 접근이 가능하고, 저비용 하드웨어를 통해 배포할 수 있어 최근 주목 받고 있는 대용량 분산 파일 시스템이다. 이런 대용량 분산 파일 시스템들의 성능은 메타데이터 서버가 얼마나 효율적으로 메타데이터를 관리하고 클라이언트의 요청을 분산시킬 수 있는지에 달려있다. 메타데이터 서버의 향상된 설계를 위해서 최근 주목 받고 있는 이들 대용량 분산 파일 시스템들의 메타데이터 서버에 대한 비교, 분석이 필요하다.

대표적으로 대용량 분산 파일 시스템을 분석한 연구로는 [7]이 있다. [7]은 대용량 분산 파일 시스템의 전반적인 구성에 대해서는 분석하였으나, 시스템의 핵심인 메타데이터 서버에 대한 기술은 하지 않았다. 따라서 본 논문에서는 GFS, HDFS, LakeFS의 메타데이터 서버의 구조와 서버들의 연산 처리 방법을 비교, 분석해보고자 한다.

2장에서 분산 파일 시스템의 구조에 대해서 알아보고 주목받고 있는 대용량 분산 파일 시스템인 GFS, HDFS, LakeFS을 중심으로 그외의 대용량 분산 파일 시스템인 Ceph, Lustre, OASIS에 대해 소개한다. 3장에서는 GFS, HDFS, LakeFS의 메타데이터 서버들에의 연산 처리 방법을 소개한다. 4장에서 메타데이터 서버들의 특징을 비교, 분석하고, 5장에서 결론을 맺는다.

2. 본 론

2.1 분산 파일 시스템 구조

분산 파일 시스템들의 구조는 대칭형과 비대칭형으로

본 연구는 한국학술진흥재단의 2008년도 기초 연구지원 기초과학 사업(D00849)의 지원을 받아 수행되었습니다.

구분할 수 있다. 대칭형 구조는 모든 노드들이 클라이언트와 서버 기능을 동시에 가지고 있어 노드들 사이에 통신하는 비용이 전체 대용량 분산 파일 시스템의 확장성을 제약한다는 단점이 있다. 반면에 비대칭형 구조는 메타데이터 서버가 독립적으로 존재하므로, 메타데이터와 데이터가 따로 처리된다. 따라서 비대칭형 구조는 데이터 서버, 클라이언트 등을 대규모로 운영하는 경우 입출력 성능과 확장성이 증가하는 장점을 가진다. 이러한 장점으로 인해 현재 대용량 분산 파일 시스템들은 대부분 비대칭형 구조를 사용한다[8].

2.2 대용량 분산 파일 시스템 상에서의 메타데이터 서버의 구조

주목받고 있는 대용량 분산 파일 시스템인 GFS, HDFS, LakeFS을 중심으로 그외의 대용량 분산 파일 시스템인 Ceph, Lustre, OASIS에 대해 소개한다 .

GFS, HDFS, LakeFS는 비대칭형 분산 파일 시스템 구조를 가진다. 실제 데이터 입출력 작업을 메타데이터 서버에서 처리하지 않고, 데이터 서버와 클라이언트가 직접 처리할 수 있도록 하여, 수백 명의 클라이언트에 의해 접속되고 있는 환경으로 인해 생기는 메타데이터 서버의 부하를 줄여 성능을 높였다.

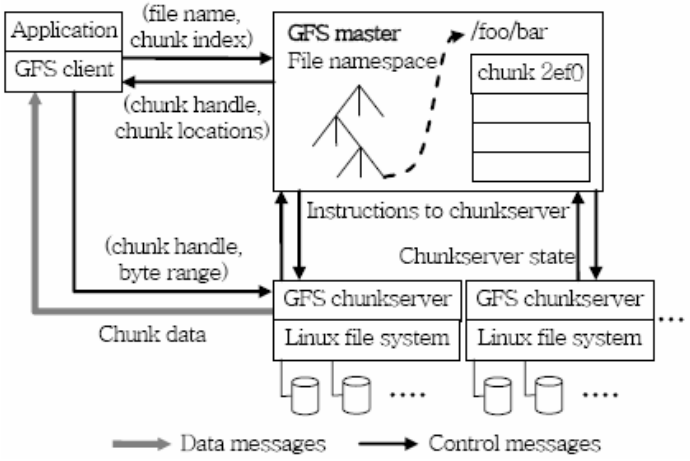

그림 1. GFS 구조

GFS는 그림1과 같이 메타데이터 서버(GFS master), 클라이언트(GFS client), 여러 개의 chunk서버(GFS chunkserver)로 구성된다. 파일은 64MB의 고정된 크기의 chunk로 나누어져 chunk서버에 저장된다. 그리고 각 chunk의 복제본(replica)도 나누어져 chunk서버에 저장된다. 메타데이터 서버는 파일의 네임스페이스, 파일과 chunk 간의 매핑 정보, 각 chunk 복제본들의 위치 정보를 저장하고 있는 메타데이터를 관리한다. GFS의 메타데이터 서버는 chunk서버로부터 하트비트(Heartbeat) 메시지를 주기적으로 받아 chunk서버에 있는 chunk의 상태를 체크하고 chunk들의 상태에 따라 chunk의 재복제, 재분산 동작을 수행한다. 메타데이터 서버의 상태를 섀도 메타데이터 서버(shadow master)에 복제하여 메타데이터 서버의 장애 처리와 회복에 사용한다. 이러한 메타데이터를 메인 메모리에 저장함으로써 메타데이터 서버가 작업을 신속하게 할 수 있다. 메타데이터 서버는 클라이언트에게 파일 이름, chunk 인덱스를 포함한 요청을 받고 해당 chunk 핸들과 chunk 위치를 반환한다.

HDFS는 그림2와 같이 메타데이터 서버(Namenode), 클라이언트(Client), 노드에 있는 저장 공간을 관리하는 많은 데이터 노드들(Datanodes)로 구성된다. HDFS는 파일이 한 번 쓰고 여러 번 읽어진다고(write-once-read-many) 가정한다. 따라서 웹 스크롤러 응용 프로그램을 대상으로 사용된다. 파일은 64MB의 고정된 크기의 블록으로 나누어져 데이터 노드에 저장된다. 블록들은 안정성을 위해 다시 복제되어 다른 데이터 노드에 저장된다. 메타데이터 서버는 파일 디렉터리의 열기, 닫기, 이름 변경과 같은 파일 시스템 동작을 수행하며 블록 생성, 삭제, 복제를 지시한다. 메타데이터 서버는 데이터 노드와 목적지 데이터 블록에 대한 클라이언트 요구에 응답하고 데이터 노드로의 블록 매핑을 판단한다. 그리고 데이터 노드로부터 하트비트 메시지를 주기적으로 받아 데이터 노드와 메타데이터 서버의 연결을 유지한다. 최근 하트비트 메시지가 없는 데이터 노드는 연결이 끊어진 것으로 표시하고 새로운 입출력 요청에 대해 그 데이터 노드를 사용하지 않는다. 그리고 메타데이터의 변경 사항들을 영구 저장하기 전에 로그를 하드 디스크에 저장한다.

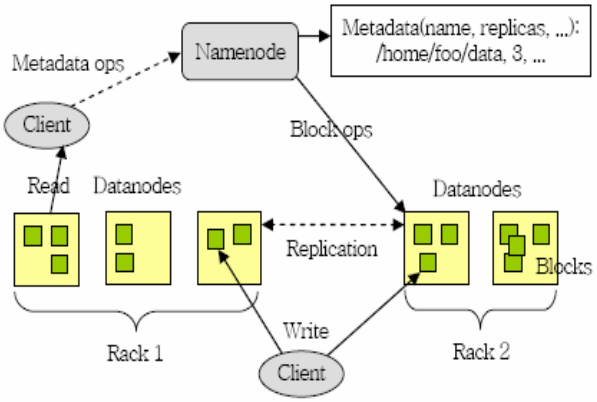

그림 2. HDFS 구조

LakeFS는 메타데이터 서버, 클라이언트, 데이터 서버로 구성된다. 파일은 64MB의 고정된 크기의 chunk로 나누어져 데이터 서버에 저장된다. LakeFS의 메타데이터 서버는 하드디스크를 그림3의 구조로 설계하여 메타데이터를 저장하고 관리한다. 이를 통해 1억 개 이상의 파일 및 디렉터리를 지원하고 로컬 파일 시스템을 메타데이터 저장소로 활용함에 따라 안정성을

확보한다. 데이터 서버로부터 하트비트 메시지를 주기적으로 받아 데이터 서버에 있는 chunk들의 상태를 체크하고 chunk들의 상태에 따라 chunk의 재복제, 재분산 동작을 수행한다. N-way replica 기반 장애 복구 구조를 통해 수동, 자동 복구 운영 환경 특성에 따라 사용자가 변경할 수 있는 다양한 복제 모드를 제공한다.

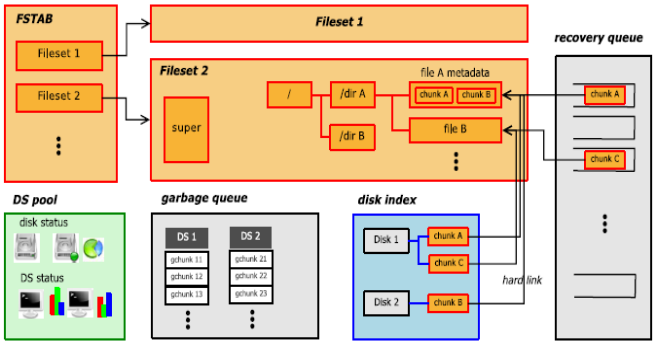

그림 3. LakeFS 구조

기타 대용량 분산 파일 시스템인 Ceph, Lustre, OASIS를 소개한다.

Ceph는 메타데이터 서버, 클라이언트, 데이터 정보를 객체화시켜 저장하는 객체 저장 서버로 구성된 객체 기반 대용량 분산 파일 시스템이다. 유저 레벨 파일 시스템 인터페이스인 FUSE(filesystem in userspace)[9]를 사용하여 POSIX 인터페이스를 준수한다. Ceph는 동적 서브트리 분할 방법(dynamic subtree partition)[10]을 사용하여 메타데이터 서버를 클러스터링한다. 메타데이터 서버가 관리하는 네임스페이스 공간의 파티션을 동적으로 재배치하고 이에 따라 메타데이터를 이동시키는 방법이다. 이름 공간을 서브트리별로 분할하여 사용함으로써 메타데이터 서버들끼리 협력적 캐싱(collaborative caching)이 가능하도록 한다. Lustre는 특정 데이터를 다른 메타데이터 서버에 복제할 수 있고 클라이언트가 메타데이터 서버들의 클러스터에 접근할 때 좀 더 유연한 대응을 할 수 있다. 메타데이터 서버는 다수의 클라이언트 요청에 의해 생길 수 있는 병목 현상(hot spot)문제를 해결하고자 클라이언트에 write-back 캐시를 사용한다. 일반적으로 메타데이터에 대한 접근 요청이 적을 때는 write-back 캐시를 사용하고 많을 때는 클라이언트 캐시를 사용함으로써 쓰기로 인한 오버헤드를 줄인다. OASIS는 메타데이터 서버, 클라이언트(FM), 객체 저장 서버(OST)로 구성된 객체 기반 대용량 분산 파일 시스템이다. 메타데이터 서버는 파일 객체가 저장되어 있는 객체 저장 서버의 정보도 저장하고 있다. 메타데이터 서버는 객체 저장 서버를 묶어서 Linear, RAID-0, RAID-1, RAID-5의 형태로 구성하고 관리한다. 메타데이터 서버는 커널 모듈로 구현되어 리눅스 커널의 VFS 함수를 호출함으로써 하위 로컬 파일 시스템에 독립적이다[8].

3. 메타데이터 서버 연산 처리 방법

3.1 네임스페이스(namespace) 연산 처리

네임스페이스는 여러가지 식별자(함수)들을 이름이 있는 범위를 사용하여 그룹화한 것으로 모든 식별자들을 네임스페이스 안에 선언함으로써 소프트웨어를 각각의 라이브러리나 컴포넌트와 결합하는 과정에서 이름 충돌 문제를 해결하는 파일 시스템 이름 공간이다. 네임스페이스 연산은 다른 기존 파일 시스템들과 유사하며, 파일을 생성하고 지울 수 있고 하나의 디렉터리에서 다른 디렉터리로 복사할 수 있으며 파일 이름을 변경한다.

GFS의 메타데이터 서버는 전체 경로 이름(full pathname)을 메타데이터로 매핑하여 네임스페이스를 표현하며 Prefix 압축을 통해 매핑 테이블을 효율적으로 메모리에 보관한다. 그리고 GFS의 메타데이터 서버는 다중 연산시에 네임스페이스에 락(lock)을 거는 것을 허용한다. 예를 들어 /home/user/foo 변경 중에는 /home/user에 read 락을 걸고 /home/user/foo에 write 락을 걸어서 데이터의 내용을 안전하게 수정할 수 있도록 한다.

HDFS의 메타데이터 서버는 GFS처럼 디렉터리 트리를 통해 네임스페이스 연산을 수행한다. 메타데이터 서버는 메모리에서 네임스페이스를 관리하며 네임스페이스의 속성에 대한 변화를 기록한다. 그리고 HDFS의 메타데이터 서버는 하드 링크와 소프트 링크를 지원하지 않고 사용자 한도 량(quotas)과 접근 허가는 구현하지 않았다.

LakeFS의 메타데이터 서버는 그림3의 구조로 메타데이터를 디스크에 저장하여 관리한다. LakeFS는 디렉터리 구조로 되어 있는 다수 개의 파일 세트들로 구성된 FSTAB(file system table)을 가진다. 네임스페이스 연산은 FSTAB을 이용하여 파일 시스템을 목적에 맞게 사용할 수 있도록 정보들을 저장하고 부팅 시에 자동적으로 적용될 수 있도록 한다.

3.2 복제(replication) 연산 처리

GFS의 메타데이터 서버는 파일 데이터 저장 시에 3개의 chunk 복제본을 생성하여 다른 데이터 서버에 저장한다. 데이터 서버가 고장 나는 경우 기존의 복사본을 이용하여 손실된 chunk 복제본과 동일한 chunk복사본을 만들어 3개의 복사본을 유지한다. 손실된 데이터에 대한 접근을 보장하여 데이터 신뢰성, 가용성을 증대시킨다.

HDFS의 메타데이터 서버는 rack-aware 복제 배치 정책을 이용하여 chunk 복제를 배치한다. 메타데이터

서버는 각 데이터 노드로부터 랙 아이디를 받아서 랙의 위치를 파악한다. 같은 랙에 두개의 chunk 복제본을 넣고 다른 랙에 마지막 chunk 복제본을 넣는다. 같은 랙 안의 기계 간에 네트워크 대역폭은 다른 랙 안의 기계 간의 네트워크 대역폭보다 훨씬 크다. 따라서 chunk 복제본을 서로 다른 3개의 랙 대신 2개의 랙에 위치시킴으로써 데이터를 읽을 때 사용되는 총 네트워크 대역폭을 감소시킨다.

LakeFS의 메타데이터 서버는 chunk 복제본의 상태를 6가지 타입으로 분류하여 관리한다. Chunk 복제본의 인덱스를 저장하고 있는 그림3의 Recovery queue을 통해 chunk 복제본의 상태를 체크한다. Recovery queue는 각 데이터 서버를 위한 분리된 sub queue들로 구성되고 각 queue는 독립적으로 구성되어 동시에 다수의 데이터 서버가 병렬적으로 복제 연산을 수행할 수 있도록 한다.

Failed	데이터 서버에 존재하는 chunk 복제본이 없음
Critical	데이터 서버에 하나의 chunk 복제본이 존재함
Under	데이터 서버에 존재하는 chunk 복제본들의 수가 복제인수 임계값을 충족함
Degraded	데이터 서버에 존재하는 chunk 복제본들의 수가 복제인수 임계값을 충족하나 유효하지 않은 chunk 복제본이 존재함
Normal	데이터 서버에 존재하는 chunk 복제본들의 수가 복제인수 임계값을 충족하고 모두 유효함
Over	데이터 서버에 존재하는 chunk 복제본들의 수가 복제 인수 임계값을 초과함

표 1. LakeFS의 chunk복제본 복제 상태 종류

3.3 회복(recovery) 연산 처리

GFS의 메타데이터 서버는 chunk서버로부터 하트비트 메시지를 주기적으로 받아 메타데이터 서버와 chunk서버의 연결을 유지한다. 하트비트 메시지가 없는 chunk서버는 연결이 끊어진 것으로 인식하고 입출력 요청을 그 chunk서버로 보내지 않는다. 손실된 chunk는 다른 chunk서버에 있는 기존의 chunk 복제본을 복제하여 chunk서버에 저장하고 복제 인수가 3개가 되도록 유지한다. 메타데이터 서버의 고장을 대비하여 네임스페이스와 chunk 매핑 변경 연산을 로깅(logging)하고 메타데이터 서버 전체를 주기적으로 섀도(shadow) 메타데이터 서버에 복제한다.

HDFS의 메타데이터 서버도 GFS의 메타데이터 서버와 같은 방식으로 회복 연산 처리를 수행한다. 하트비트 메시지를 주기적으로 받아 chunk의 상태를 체크하여 손실된 chunk를 회복한다. 하나의 데이터 노드에 저장된 데이터의 양이 많아져 디스크 공간이 특정 임계 값 아래로 떨어졌을 때는 데이터를 하나의 데이터 노드에서 다른 노드로 이동시켜 재분산시킨다.

LakeFS의 메타데이터 서버는 하트비트 메시지를 통해 데이터 노드가 일시적 중지 상태(TIMEOUT)인지 확인하고 하트비트 메시지가 없는 데이터 노드는 오프라인(OFFLINE) 상태로 판명한다. 데이터 노드가 온라인 상태에서 오프라인 상태로 변경되면 메타데이터 서버와 데이터 노드 간의 연결이 끊어진 것으로 판명한다. 연결이 끊어진 데이터 노드 안의 chunk들의 정보를 회복하기 위해 Recovery queue에서 잃어버린 chunk들의 재복제 작업을 수행한다.

4. 메타데이터 서버의 특징 비교, 분석

GFS, HDFS, Lake FS 의 메타데이터 서버의 특징은 표2와 같다.

분류	GFS	HDFS	LakeFS
Interface (API)	POSIX API기반 -Java API	POSIX API기반 -Java API	POSIX API기반 -FUSE
메타데이터 저장 공간의 위치	메인 메모리	메인 메모리	하드디스크
데이터 복제 알고리즘	파이프라인 (Pipelined) 특화된 복제	파이프라인 (Pipelined) 특화된 복제	파이프라인 (Pipelined) 특화된 복제
메타데이터 서버 공통 기능	네임스페이스(Namespace)관리 메타데이터 관련 연산 처리 재복제(Re-replication) 재분산(Cluster rebalancing) 복제(Replication) 가비지 컬렉션(Garbage collection)		

표 2. GFS, HDFS, LakeFS 메타데이터 서버 분석표

4.1 메타데이터 서버의 공통된 특징

GFS, HDFS, LakeFS의 메타데이터 서버, 클라이언트, 데이터 서버로 구성된 비대칭 구조이다.

GFS, HDFS, LakeFS의 메타데이터 서버는 공통적으로 데이터 서버 및 클라이언트와 연계되어 작동함으로써 그 둘 간에 정보를 전달하고 제어하는 역할을 한다. 또한 전체 경로 이름을 메타데이터로 매핑하여 네임스페이스로 표현하여 관리한다. 여러 개의 chunk서버에 동일한 데이터를 복제하여 분산시킴으로써

특정 chunk서버에 장애가 발생하더라도 다른 chunk서버에 저장된 데이터를 통해 관련 응용 프로그램이 장애에 영향을 받지 않고 작업을 수행할 수 있도록 복제 연산을 처리한다. 메타데이터 서버는 하트비트 메시지를 통해 데이터 서버에 있는 chunk 복제본의 손실이나 메타데이터 서버와 chunk서버의 연결 상태를 체크하고 이 상태를 보고 chunk를 재복제한다. 특정 서버가 관리하는 파일에 과부하가 걸리면 부하가 적은 서버에 chunk 복제본을 재분산하여 작업 균형을 맞춘다. 가비지 컬렉션 연산을 통해 필요 없어지거나 오랜 시간 쓰여지지 않은 파일을 발견해서 모아둔 뒤 삭제한다.

4.2 메타데이터 서버별 차별화된 특징

GFS와 HDFS의 메타데이터 서버는 POSIX 인터페이스를 모두 제공하는 대신 파일의 생성 및 삭제 등 일부 인터페이스만 제공하여 전체 시스템의 구현 복잡도를 제거한다. 응용 프로그램의 사용에 JAVA API을 제공한다. LakeFS의 메타데이터 서버는 FUSE를 사용한다. FUSE는 커널 모듈, 사용자 라이브러리, 마운트 유틸리티로 구성되어 일반 사용자가 단지 사용자 프로그램으로 자신의 파일 시스템을 구현하고 마운트 할 수 있는 방법을 제공한다. 기존의 파일 시스템 관련 응용 프로그램들이 수정 없이 사용 가능하며, 리눅스 커널 버전의 독립적인 운영 환경이 지원된다. GFS의 메타데이터 서버는 파일 및 chunk 이름 공간에 관련된 메타 데이터, 파일을 어떤 chunk에 매핑할 것인지에 대한 정보, 그리고 각 chunk 복제본에 대한 위치 정보를 관리한다. 메타 데이터 서버는 이름 공간과 매핑 정보에 대해서는 영구적으로 저장하나 chunk 복제본의 위치 정보는 영구적으로 저장하지 않는다. 그 대신 chunk의 위치 정보를 파악하기 위해서 메타데이터 서버가 사용될 때 모든 chunk서버에게 chunk 정보를 질의함으로써 그 내용을 파악해서 메모리에 보관하고 해시 테이블과 같은 구조로 이를 저장한다[8]. 이러한 구현을 통해 메타데이터 서버는 chunk서버 상호 간의 동기화를 할 필요가 없어 단순하게 관리할 수 있다. GFS의 메타데이터 서버는 메타데이터와 데이터를 분리하여 관리함으로써 데이터 서버에 접근하기 위해 메타데이터에 접근해야 하는 부하를 줄이고[7] 빠른 연산과 데이터에 대한 접근 속도 향상을 가져왔다는 장점을 가진다. 하지만, 메타데이터의 공유가 어려워서 사용성이 떨어지고 오작동으로 인해서 메인 메모리의 데이터 손실이 생길 수 있다.

HDFS의 메타데이터 서버는 메타데이터를 메모리에 저장하여 관리하고 메타데이터와 관련된 모든 변경 사항들을 영구 저장하기 전에 임시로 로그를 디스크에 저장함으로써 메타데이터의 신뢰성을 확보한다.

LakeFS의 메타데이터 서버는 메타데이터를 디스크에 저장하여 관리한다. 디스크 기반 메타데이터 관리 구조를 통해 1억 개 이상의 파일 및 디렉터리들을 관리할 수 있는 환경을 지원한다. 로컬 파일 시스템을 메타데이터 저장소로 활용함에 따라 메인메모리의 오작동으로 인한 데이터 손실을 막을 수 있어 안정성이 확보된다.

5. 결 론

메타데이터를 요구하는 클라이언트의 수가 수만 개 이상으로 급증함에 따라 페타바이트급의 데이터 저장 공간을 위한 대용량 분산 파일 시스템이 사용된다. 최근에 대용량 분산 파일 시스템의 성능을 좌우하는 메타데이터 서버에 대한 다양한 설계 방법이 제안되었다. 본 논문에서는 대용량 분산 파일 시스템인 GFS, HDFS, LakeFS, Ceph, Lustre, OASIS 의 메타데이터 서버를 소개하고 높은 성능으로 인해 주목 받고 있는 대용량 분산 파일 시스템인 GFS, HDFS, LakeFS을 중심으로 각 시스템들을 비교, 분석하였다.

이를 통해 메타데이터 서버의 관여를 최소화하고 복제 및 회복 연산하는 설계가 메타데이터 서버의 작업 부하를 줄이고 손실된 데이터의 복구 능력을 향상시킬 수 있음을 알 수 있었다.

다수의 클라이언트가 메타데이터 서버에 동시에 작업 요청을 보냄으로써 생기는 메타데이터 서버의 병목 현상에 대해 효율적으로 처리하는 방법은 개선의 여지가 있다. 본 논문에서 대용량 분산 파일 시스템의 메타데이터 서버들을 비교, 분석한 결과를 토대로 메타데이터 서버의 병목 현상을 해결할 수 있고 높은 확장성과 효율적인 분산 방법을 제공할 수 있는 메타데이터 서버에 관해 향후 연구할 예정이다.

참 고 문 헌

[1] S. Ghemawat, H. Gobioff, and S.T. Leung, “The Google File System,” In Proc. of the 19th ACM Symp. on Operating Systems Principles, Bolton Landing,2003.

[2] D. Borthakur, “The Hadoop Distributed File System:Architecture and Design,”http://lucene.apache.org/hadoop/hdfs_design.html, 2005.

[3] Keun-Tae Park, Hong-Yeon Kim, Young-Chul Kim, Sang-Min Lee, Young-Kyun Kim, and Myung-Joon Kim, “Lake: Towards highly manageable cluster storage for extremely scalable services”, International Conference on Computational Sciences and Its Applications ICCSA , 2008.

[4] Sage Weil, Scott A. Brandt, Ethan L. Miller, Darrell D.E. Long, and Carlos Maltzahn, “Ceph: A Scalable, High-Performance Distributed File System,” In Proc.

of Conf. on Operating Systems Design and Implementation, 2006.
[5] Lustre, “Lustre: A Scalable High Performance File System,” Cluster File System, Inc., http://www.lustre.org/docs/whitepaper.pdf, 2002.
[6] Y.K. Kim, H.Y. Kim, S.M. Lee, J. Kim, and M.J. Kim, “OASIS: Implementation of a Cluster File System Using Object-based Storage Devices,” In Proc. of the Int’l Conf. on Computational Science and Its Applications, 2006.
[7] 김영철, 박근태, 이상미, 김홍연, 김영균, “클러스터 파일 시스템 기술 동향”, [ETRI]전자통신동향분석, 제 22권, 제 6호, 2007.
[8] 차명훈, 이상민, 김 준, 김영균, 김명준 , “대규모 분산 파일 시스템 환경의 메타데이터 관리”, [ETRI]전자통신동향분석, 제 22권, 제 3호, 2007.
[9] http://fuse.sourceforge.net/
[10] S.A. Weil, K.T. Pollack, S.A. Brandt, and E.L. Miller, “Dynamic Metadata Management for Petabytescale File Systems,” In Proc. of the 2004 ACM/IEEE Conf.on Supercomputing, 2004.