9

# 클라우드 컴퓨팅을 위한 분산 파일 시스템 기술

한국전자통신연구원 ▌민영수* · 김홍연 · 김영균

## 1. 서 론

클라우드 컴퓨팅은 차세대 인터넷 비즈니스의 핵심 분야로 부각되고 있으며 미래 친환경 IT 신성장 전략의 하나로도 꼽힌다. 클라우드 컴퓨팅에 대한 다양한 정의들이 존재하지만, 공통적으로 인터넷을 기반으로 하여 IT 자원들을 서비스 형태로 제공하는 컴퓨팅으로 정의하고 있다[1,2].

최근 클라우드 컴퓨팅 시장에 진출했거나 진출을 선언한 Google, IBM, Microsoft, Sun microsystems 등과 같은 글로벌 IT 기업들을 살펴보면 이미 보유하고 있는 기반 기술들을 활용하거나 상호 협력을 통해 다양한 클라우드 서비스들을 제공함으로써 급격하게 성장하고 있는 클라우드 컴퓨팅 시장에서 자신들의 영역을 지속적으로 확장해나가고 있다.

클라우드 컴퓨팅은 인터넷을 통해 소비자가 필요로 하는 저장 공간과 전송 용량을 가상화하여 제공하는 가장 하위 계층인 dSaaS(data-Storage-as-a-Service), dSaaS 상위에서 가상화된 컴퓨터와 같은 처리 능력을 제공하는 Iaas(Infrastructure-as-a-Service), IaaS 상위에서 가상 머신이나 DBMS와 같은 특별한 운영체제와 응용들의 집합체를 제공하는 PaaS(Platform-as-a-Service), 마지막으로 사용자가 필요로 하는 소프트웨어를 원격에서 제공하는 가장 상위 계층인 Saas(Software-as-a-Service)로 구성된다[3]. 이러한 계층에 따라 제공되고 있는 대표적인 서비스들을 살펴보면 표 1과 같다.

표 1 클라우드 컴퓨팅의 계층과 대표적인 서비스들

| | |
|---|---|
| SaaS | Google Apps, Microsoft "Software+Services" |
| PaaS | IBM IT Factory, Google AppEngin, Force.com |
| IaaS | Amazon EC2, IBM Blue Cloud, Sun Grid |
| dSaaS | Nirvanix SDN, Amazon S3, Cleversafe dsNet |

클라우드 컴퓨팅을 제공하기 위해서는 막대한 물리적인 IT 인프라 구축이 필수적이다. 클라우드 컴퓨팅의 성장과 더불어 폭발적으로 증가하고 있는 데이터의 효율적인 저장과 관리뿐만 아니라 점점 다양해지고 있는 사용자의 요구를 충족시키기 위해서는 이러한 IT 인프라를 저렴하게 구축하고 효율적 활용할 수 있는 기술들이 요구된다. 데이터의 저장과 관리를 위한 분산 데이터 저장 기술과 데이터의 병렬 처리를 위한 분산 처리 기술은 소프트웨어적인 측면에서 클라우드 컴퓨팅을 제공하기 위한 필수적인 요소이다. 특히 클라우드 컴퓨팅에서 다루어지는 막대한 양의 데이터를 수많은 서버들에 분산 저장하고 관리하는 분산 데이터 저장 기술은 클라우드 컴퓨팅의 핵심 요소이다. 분산 데이터 저장 기술은 단순히 데이터의 저장과 관리만을 하는 것이 아니라, 데이터를 저장하고 있는 하드웨어의 장애에 유연하게 대처하고 서비스가 요구하는 충분한 성능도 보장해야 한다. 지금부터 살펴보고자 하는 분산 파일 시스템은 분산 데이터 저장 기술에서 대부분의 역할을 수행하며 분산 데이터 저장 기술의 전부라고도 할 수 있다.

기존에 존재하는 분산 파일 시스템들은 적용 분야나 특정 목적에 따라 다양하게 분류되고 있으며, 상당히 많은 분산 파일 시스템이 존재한다. 하지만 최근까지 클라우드 컴퓨팅을 위한 분산 데이터 저장 기술로서 활용이 되고 있는 분산 파일 시스템은 그리 많지 않다.

본 논문에서는 현재 클라우드 컴퓨팅에서 대표적으로 사용되면서 구조가 공개되어 있는 Google의 Google 파일 시스템[4]과 Apache의 Hadoop 분산 파일 시스템[5]을 살펴보고 ETRI에서 개발한 GLORY 파일 시스템(GLORY-FS[1])을 소개한다. GLORY-FS는 수천에서

* 정회원

1) GLORY-FS : 지식경제부 IT 성장 동력 기술 개발 사업의 일환으로 2007년부터 2012년까지 수행중인 "저비용 대규모 글로벌 인터넷 서비스 솔루션" 과제의 결과물임

수만 대의 저비용 스토리지 서버들을 이용하여 스토리지 비용을 최소화하고 높은 성능과 장애에 대한 효율적인 통제 능력을 갖는 대규모 인터넷 서비스를 위한 분산 파일 시스템으로써 향후 클라우드 컴퓨팅에서의 활용 가능성 등을 살펴본다.

본 논문의 2장에서는 최근 클라우드 컴퓨팅 시장에서 활용되고 있는 대표적인 분산 파일 시스템인 Google 파일 시스템과 Hadoop 분산 파일 시스템의 구조를 살펴보고, 3장에서는 GLORY-FS의 구조와 특징들을 상세하게 소개한다. 4장에서는 클라우드 컴퓨팅을 위한 분산 파일 시스템의 향후 연구 과제에 대한 구상을 끝으로 5장에서 본 논문의 결론을 기술한다.

## 2. 클라우드 컴퓨팅을 위한 분산 파일 시스템 기술 동향

클라우드 컴퓨팅을 위해 분산 파일 시스템이 갖추어야 할 사항들은 실로 다양하다. 데이터의 효율적인 저장과 관리뿐만 아니라 빈번하게 발생하는 하드웨어 장애에 유연하게 대처해야 하며 최적의 데이터 입출력을 위한 성능도 보장해야 한다. 이러한 사항들은 분산 파일 시스템의 모습을 결정짓는 매우 중대한 요소이다. 본 장에서는 최근 클라우드 컴퓨팅 시장에서 활용되고 있는 대표적인 분산 파일 시스템인 Google 파일 시스템과 Hadoop 분산 파일 시스템의 구조를 소개한다.

### 2.1 Google File System(GFS)

클라우드 서비스를 제공하기 위한 Google의 클라우드 컴퓨팅 플랫폼은 잘 알려진 것처럼 분산 데이터 저장 기술인 GFS를 기반으로 하고 있다. Google은 자사 응용들의 작업부하 패턴과 기술적인 환경 등을 철저하게 분석하여 현재 플랫폼의 기반이 되는 GFS를 설계하고 구현하였다.

GFS의 설계와 구현을 위해 도출된 사항들을 살펴보면 다음과 같다. 첫째로 막대한 양의 데이터를 저장하기 위해 수많은 저비용 스토리지 서버들을 사용하며 그로 인해 고장이 빈번하게 발생하다는 것이다. 따라서 빈번하게 발생하는 다양한 장애 상황에 유연하게 대처할 수 있는 장애 대처 방안을 필요로 한다. 둘째로 다루어지는 파일의 크기는 대부분은 100메가 바이트 이상이며 다수의 클라이언트가 대용량을 연속적으로 읽는 연산과 순차적으로 데이터를 추가하는 연산을 주로 요구한다는 것이다. 따라서 대용량 파일을 효과적으로 관리할 수 있는 방법과 효율적인 동기화 방법이 필요하며 빠른 응답 시간보다는 전체적으로 입출력 전송량을 최대한 활용하는 것이 무엇보다 중요하다는 것이다.

GFS는 그림 1에서처럼 여러 클라이언트에 의해 접근되는 단일 마스터(master)와 여러 청크서버(chunkserver)들로 구성되며 저비용의 리눅스 서버 상에서 사용자 레벨 프로세스로 구동된다. 성능과 확장성, 가용성을 높이기 위해 메타데이터와 데이터의 흐름을 분리한 비대칭형 구조인 GFS는 새로운 청크서버를 추가하는 것만으로 저장 공간의 확장을 손쉽게 이룰 수 있다.

파일들은 생성 시에 마스터에 의해 부여되는 고유한 64비트 청크 핸들(chunk handle)을 가지는 64메가 바이트 크기의 청크(chunk)들로 이루어진다. 크기가 큰 청크를 사용함으로써 마스터가 유지해야할 메타데이터의 수를 감소시키고 네트워크 부하를 줄일 수 있는 장점을 가진다. 또한 각 청크에 대한 다수의 복제본을 여러 청크서버에 분산하여 저장함으로써 청크서버의 장애에 효과적으로 대처할 수 있다. 마스터는 장애가 발생한 청크서버에 포함된 청크들에 대해 복제본을 가진 다른 청크서버들을 활용하여 재복제와 재분산 같은 회복 동작을 수행한다.

GFS는 파일과 디렉토리 트리의 사본을 손쉽게 생성할 수 있는 스냅샷(snapshot)을 제공하며 동일한 파

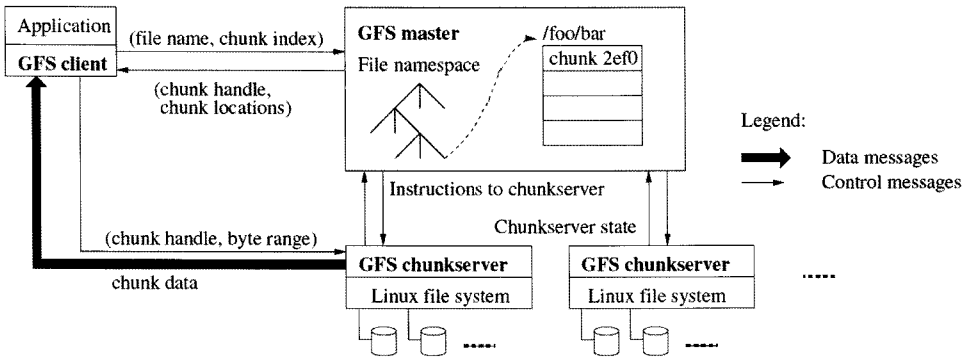



그림 1 GFS의 구조

일에 대해 동시에 데이터를 추가할 수 있도록 각 클라이언트의 원자적 추가 연산을 보장한다.

#### 2.1.1 GFS 클라이언트

클라이언트는 POSIX API를 지원하지 않으며 파일 시스템 인터페이스와 유사한 자체적인 인터페이스를 제공한다. 응용 프로그램에 연결된 클라이언트는 파일 시스템 API를 수행하며, 응용 프로그램을 대신하여 마스터와 청크서버들과 통신한다. 클라이언트는 메타데이터 연산을 위해서 마스터와 통신하며, 데이터 입출력을 위해서 청크서버들과 직접 통신한다. 대부분의 GFS 응용들은 대용량 파일을 지속적으로 읽는 연산을 주로 수행하기 때문에 클라이언트에서 파일 데이터 캐시를 유지해서 얻을 수 있는 이득이 거의 없다. GFS는 이러한 캐시를 없애서 캐시 일관성 문제를 제거하였으며, 그 결과 클라이언트와 전체 시스템이 간결해졌다.

#### 2.1.2 GFS의 메타데이터 관리

마스터는 네임스페이스, 접근 제어 정보, 파일과 청크의 매핑 정보, 청크의 위치 정보와 같은 모든 파일 시스템 메타데이터를 관리하고 청크와 복제본의 생성, 복제본 수 조정, 사용되지 않는 저장 공간의 회수 등 시스템 전반적인 동작들을 제어한다.

마스터는 메타데이터를 메모리에서 유지하기 때문에 초기 기동이나 장애의 발생으로 인한 재기동 시에 청크서버들로부터 청크에 대한 정보를 받아서 메모리 상에 메타데이터를 구축하고 로깅을 통해 구축 이후에 발생하는 메타데이터에 대한 변경을 로컬 디스크에 저장한다. 이러한 로그는 마스터의 장애에 대비하여 원격의 다른 서버에 복제된다. 마스터에 장애가 발생하면 로그를 복제하고 있는 다른 서버가 마스터로 동작하여 장애에 신속하게 대처한다. 또한 읽기 전용 모드로 동작하는 쉐도우(shadow) 마스터를 두어 메타데이터에 대한 가용성을 향상시킨다.

마스터는 주기적으로 heartbeat 메시지를 통해 청크서버들의 상태를 수집하고 필요 없어진 청크의 삭제와 같은 명령들을 전달한다. heartbeat 메시지를 통해 장애가 발생한 청크서버가 검출되면 그 청크서버에 포함된 청크들에 대해 다른 청크서버에 있는 복제본을 활용하여 재복제를 수행하여 장애에 대처한다. 만일 한 랙(rack) 안에 있는 청크서버들에 모든 복제본을 배치했을 때 랙에 장애가 발생하면 그 복제본들을 서비스할 수 있는 청크서버들이 없게 된다. 이러한 장애로 인해 장애가 복구될 때까지 서비스가 지연될 수도 있으며 최악의 경우에는 복구가 불가능하여 데이터의 손실을 야기할 수도 있다. 이것을 방지하기 위해 마스터는 청크의 복제본들을 여러 청크서버들에 배치할 때 한 랙 안에 있는 청크서버들뿐만 아니라 다른 랙에 있는 청크서버들도 고려한다.

마스터는 주기적으로 현재 복제본들이 분산된 정도를 검사하고 디스크 공간의 활용과 부하의 분산을 위해 복제본들에 대한 재분배를 수행한다. 새로 추가된 청크서버의 디스크 공간을 채우기 위해 새로 생성되는 청크들을 이 청크서버로 보내어 과도한 부하를 주기보다는 이러한 재분배를 수행하여 점차적으로 균형을 이룰 수 있도록 한다.

어떤 청크서버에 장애가 발생한 동안 그 청크서버가 포함하고 있는 청크들이 클라이언트 요청에 의해 갱신되면서 다른 청크서버들이 가진 청크들과 버전 번호가 달라질 수 있다. 마스터는 장애가 발생했던 청크서버가 재기동하면 버전 번호를 비교하여 필요 없어진 청크들을 검출하고 삭제하도록 명령한다.

마스터는 파일이나 디렉토리에 대한 삭제 요청에 대해 곧바로 삭제하지 않고 지워진 시간을 이름으로 하는 숨겨진 파일로 변경한다. 마스터가 파일 시스템 네임스페이스를 검사하면서 일정 시간이 지난 숨겨진 파일을 삭제하고 메모리 상에 유지하는 메타데이터로부터 제거한다. 또한 파일과 매핑되어 있지 않은 고아 청크들을 검출하고 그 청크를 포함하고 있는 청크서버로부터 받은 heartbeat에 대해 고아 청크들을 삭제하도록 응답을 보낸다.

#### 2.1.3 GFS의 데이터 관리

청크서버는 청크의 관리와 클라이언트가 요청하는 입출력을 담당하며 주기적으로 자신과 청크의 상태 정보를 마스터에게 heartbeat 메시지를 통해 보고한다. 청크서버는 청크를 로컬 파일 시스템 파일로 저장하며 저장된 청크의 데이터 오류를 검출하기 위해 체크섬(checksum)을 사용한다. 청크서버는 오류가 검출된 청크에 대해 마스터에게 보고하고 마스터는 그 청크의 새로운 복제본을 다른 청크서버에 생성한 후 오류가 검출된 청크를 지우도록 청크서버에게 명령을 보낸다. 청크 서버에서는 청크를 로컬 파일들로 저장하기 때문에 운영체제의 버퍼 캐시가 이미 자주 접근되는 데이터를 메모리에 유지한다. 따라서 별도로 파일 데이터를 캐시할 필요가 없다.

### 2.2 Apache Hadoop Distributed File System(HDFS)

HDFS는 신뢰성과 확장성을 갖춘 분산 컴퓨팅을 위한 오픈 소스 소프트웨어 개발 프로젝트인 Hadoop에서 분산 컴퓨팅 프레임워크를 지원하기 위해 개발된 분

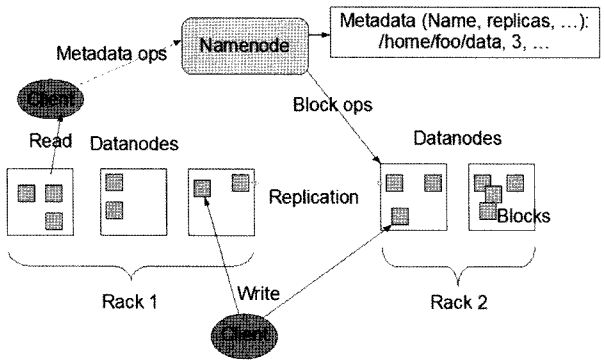



그림 2 HDFS의 구조

산 파일 시스템이다. HDFS는 현재 Amazon, IBM, Yahoo 등과 같은 글로벌 IT 기업들의 클라우드 컴퓨팅 플랫폼의 기반이 되는 분산 파일 시스템으로 가장 널리 활용이 되고 있다[7].

HDFS의 설계와 구현을 위해 도출된 사항들을 살펴보면 GFS와 대부분 동일하며 플랫폼 간의 손쉬운 이식성을 보장하기 위해 자바를 사용하여 구현되었다는 점이 크게 다르다. 높은 이식성을 지닌 자바 언어의 사용은 자바를 지원하는 다양한 서버들에서 HDFS가 구동할 수 있다는 장점을 지닌다.

HDFS는 그림 2에서처럼 네임스페이스를 관리하고 클라이언트의 파일에 대한 접근을 통제하는 단일 네임노드(NameNode)와 데이터 저장소를 관리하는 수많은 데이터노드(DataNode)들로 구성되며, 주로 GNU/Linux를 운영체제로 하는 저비용 서버에서 구동되도록 설계되었다.

HDFS는 대부분의 파일 시스템들과 유사한 네임스페이스 계층 구조를 제공하여 디렉토리나 파일의 생성, 삭제, 이동, 재명명 등이 가능하지만 사용자에 대한 저장 공간 크기 제어, 접근 권한, 링크를 지원하지 않고 있다. 또한 한번 생성된 파일은 읽기와 추가(append)만 가능하다. 따라서 데이터의 일관성을 보장하기 위한 복잡한 동시성 모델을 필요로 하지 않으며 간단한 동시성 모델을 통해 데이터 읽기 성능을 높이는데 초점을 맞추고 있다.

HDFS에서 하나의 파일은 GFS의 청크와 동일한 개념의 블록들로 이루어지며 데이터노드들에 분산되어 저장된다. 마지막 블록을 제외하고는 한 파일을 구성하는 모든 블록들은 동일한 크기를 가지며 데이터노드의 장애에 대처하기 위해 블록들의 복제본이 여러 데이터노드들에 중복되어 저장된다. 파일마다 블록의 크기와 복제본의 수를 조절할 수 있으며 복제본의 수는 파일을 생성할 때 지정할 수도 있고 나중에 변경하는 것도 가능하다.

### 2.2.1 HDFS 클라이언트

파일을 생성하기 위해 클라이언트는 한 블록의 크기를 가지는 로컬 파일 시스템의 파일에 임시로 데이터를 저장한다. 한 블록의 크기가 데이터로 모두 채워지거나 닫기 연산이 수행되면 클라이언트는 네임노드에게 메타데이터에 반영해줄 것을 요청하고 복제본이 배치될 데이터노드들의 리스트를 받아온다. 클라이언트는 네임노드로부터 받아온 리스트에 따라 자신과 가장 근접한 첫 번째 데이터노드로 임시 파일과 데이터노드들의 리스트를 전송한다. 첫 번째 데이터노드는 4킬로바이트 단위로 데이터를 받아 로컬 파일 시스템의 파일에 저장하면서 곧바로 두 번째 데이터노드에게 전송한다. 두 번째 데이터노드도 동일하게 저장하고 세 번째 데이터노드에게 전송한다. 이러한 파이프라인 형태로 임시 파일의 데이터가 모두 전송되고 나면 3개의 데이터노드에 동일한 복제본이 저장되며, 클라이언트는 저장이 완료된 것을 확인한 후에 임시 파일에 대한 전송을 끝낸다.

### 2.2.2 HDFS의 메타데이터 관리

간결하게 설계된 메타데이터 구조를 기반으로 모든 메타데이터를 메모리에 유지하는 네임노드는 메타데이터의 영속성을 위해 ext3와 같은 로컬 파일 시스템을 사용하여 두 개의 파일을 저장한다. 첫 번째 파일은 HDFS 파일과 블록들 간의 매핑, 파일 시스템 속성들과 같은 모든 메타데이터를 저장하는 FsImage 파일이며 두 번째 파일은 메타데이터에 발생하는 모든 변경 사항을 기록하는 EditLog라는 트랜잭션 로그 파일이다. 네임노드는 재기동 시에 메모리에 유지하는 메타데이터를 구축하기 위해 FsImage 파일과 EditLog 파일을 사용하며 두 파일에 대해 체크포인트(checkpoint)와 같은 갱신이 함께 수행된다.

복제본 배치를 어떻게 할지와 읽기 연산을 위해 어떤 복제본에 접근할 지를 결정하는 정책은 데이터에 대한 신뢰성과 가용성, 네트워크 대역폭 활용을 높이는데 매우 밀접한 관련을 가진다. 네임노드는 복제본 배치와 접근할 복제본을 결정하기 위해 랙 인지(rack awareness)를 활용한다. HDFS에서는 기본적으로 내정하고 있는 3개의 복제본에 대하여 한 랙의 서로 다른 서버에 1개씩, 그리고 다른 랙에 있는 서버에 나머지 1개를 배치하는 정책을 사용한다. 이러한 배치 정책과 더불어 읽기 연산을 위해 랙 인지를 활용하여 읽기 요청을 한 클라이언트에서 가장 가까운 복제본을 선택하는 정책을 사용한다.

네임노드는 주기적으로 모든 데이터노드들로부터 자신의 상태를 나타내는 heartbeat과 자신이 포함하

고 있는 블록들의 리스트인 blockreport를 받는다. 네임노드는 heartbeat을 활용하여 장애가 발생한 데이터노드를 검출하고 데이터의 가용성을 보장하기 위해 재복제를 통한 장애 처리 루틴을 수행한다. 또한 네임노드는 blockreport를 활용하여 메타데이터로 유지하고 있는 블록들의 정보와 데이터노드가 포함한 블록들 간의 불일치를 검출하고 불일치하는 블록들을 삭제하도록 데이터노드에게 명령한다.

네임노드는 파일이나 디렉토리에 대한 삭제 요청에 대해 곧바로 삭제하지 않고 임시 디렉토리로 옮겨서 지워진 것처럼 보이도록 하고 설정된 일정 시간이 지나면 실제로 삭제를 수행한다. 이러한 지연 삭제는 실제로 삭제되지 않은 데이터의 복구를 빠르게 할 수 있으며, 곧바로 데이터를 삭제할 때 발생하는 부하를 줄일 수 있는 장점을 지닌다.

#### 2.2.3 HDFS의 데이터 관리

64메가바이트 크기의 블록들로 구성되는 HDFS 파일들은 서로 다른 여러 데이터노드에 분산되어 저장되며 해당 데이터노드는 각 블록을 로컬 파일 시스템의 파일로 저장하고 관리한다. 데이터노드는 디렉토리당 최적의 파일 수를 고려하여 서브 디렉토리들을 구성하고 블록들을 저장하는 로컬 파일 시스템의 블록 파일들을 이러한 디렉토리 트리에 적절히 배치한다. 데이터노드는 기동할 때 디렉토리 트리를 스캔하여 blockreport라 불리는 모든 블록들의 리스트를 생성하고 네임노드에게 이 리스트를 전송한다.

데이터노드는 데이터의 무결성을 보장하기 위해 블록 파일을 저장할 때 그 블록 파일에 대한 체크섬을 별도의 숨겨진 파일로 저장한다. 클라이언트가 블록 파일을 요청하면 이러한 체크섬은 블록 파일과 함께 제공되며 클라이언트는 두 개를 비교하여 블록 파일의 오류 여부를 판단하게 된다. 오류가 있다면 클라이언트는 동일한 복제본을 가진 다른 데이터노드로 요청을 하게 된다.

## 3. GLORY-FS

ETRI에서 개발중인 GLORY-FS는 저비용으로 대규모 스토리지를 구축할 수 있는 솔루션이라는데 그 기술의 의의가 있다. 기존의 엔터프라이즈급 스토리지가 SAN, FC, HBA 등 고가의 스토리지 하드웨어에 기반하여 구축 비용이 높음에 비해 GLORY-FS 솔루션은 여러대의 범용적인 서버들을 전문적인 스토리지 장비로 활용할 수 있도록 하여 스토리지 구축 비용을 절감시켜 준다.

예를 들어 시장에서 널리 유통되는 1 테라바이트 SATA 하드 디스크 100장을 HP DL380급의 서버 10대에 나누어 장착하고 GLORY-FS 소프트웨어를 설치한 후 기가비트 이더넷에 연결하기만 하면 100 테라바이트급의 대규모 단일 볼륨이 만들어 진다. GLORY-FS 스토리지에 접근하고자 하는 호스트의 경우 기존의 서버 그대로 GLORY-FS 소프트웨어를 설치하고 기가비트 이더넷에 접속하는 것으로 모든 준비가 완료된다. GLORY-FS에 접근하는 호스트는 물리적으로 스토리지 서버가 10대임에도 마치 한 대의 큰 100 테라바이트 서버인 것처럼 사용하게 된다.

용량 추가 또한 이미 운용중인 서비스의 중단 없이 손쉽게 지원된다. 단지 추가하고자 하는 용량만큼의 하드 디스크와 서버를 구매하여 GLORY-FS 소프트웨어를 설치한 후 기존의 스토리지와 기가비트 이더넷으로 연결하고 등록시켜주면 온라인으로 스토리지 용량이 늘어난다. 사실상 네트워크가 허용하는 한 무한대로 용량을 추가할 수 있는 셈이다. 또한 용량 추가와 함께 스토리지 전체의 처리 성능 또한 같이 업그레이드되는 효과를 가진다. 참고로, 스토리지 서버 및 호스트의 대수는 1만대까지 확장, 용량은 이론상 엑사바이트 수준까지 확장 가능하다.

범용적인 서버 사용으로 인한 신뢰성 저하 우려를 불식시키기 위한 다양한 기술도 탑재했다. 데이터가 저장되면 GLORY-FS 소프트웨어가 자동으로 여러 디스크 및 서버에 복사본을 생성하여 디스크 및 서버 장애에 대비한다. 만약 일부 디스크 및 서버에 장애가 발생하더라도 복사본이 있는 한 스토리지에 지속적으로 접근 가능하며, 또한 유실된 복사본은 일정 시간이 지나면 자동으로 정상적인 디스크 및 서버에 재복제하는 등의 자체 치유 기능을 탑재하였다. 이 모든 절차는 사용자의 개입 없이 완전히 자동화되어 이루어지므로 서버 대수가 증가함에 따른 관리자의 관리 부담을 최소화하였다.

이외에도 GLORY-FS는 핫-스팟(hot-spot) 회피 기능, 그림 3에서 볼 수 있는 웹기반 모니터링 및 관리 기능 등을 제공하여 순간적으로 특정 서버 또는 데이터에 대한 집중된 부하를 다수의 서버로 자동 분산시킬 수 있으며, 다수의 서버, 디스크, 네트워크 자원 등의 중앙 집중적인 모니터링 및 관리 기능 등을 편리하게 수행할 수 있도록 하는 등 본격적인 스토리지 시스템으로서의 다양한 기능을 탑재하였다.

### 3.1 GLORY-FS의 특징

GLORY-FS가 갖는 특징들은 클라우드 컴퓨팅을 위

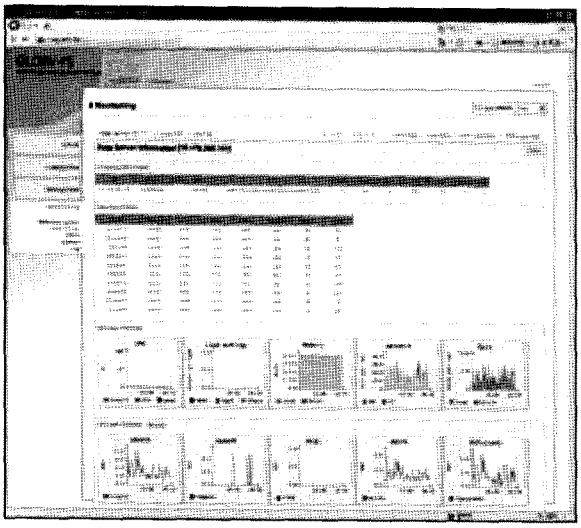


그림 3 GLORY-FS의 웹기반 모니터링 및 관리도구

한 분산 파일 시스템과 상당히 유사하며 다음과 같이 요약할 수 있다.

· **스토리지의 가상화**

데이터를 저장하는 수많은 서버들의 저장 공간을 하나의 저장 공간인 것처럼 가상화하여 클라이언트에게 제공함으로써 클라이언트가 대용량의 로컬 스토리지를 가지고 사용하는 것과 같은 기능을 제공한다. 이러한 가상화를 통하여 클라이언트에게 엑사바이트 이상의 대용량 스토리지를 제공할 수 있다.

· **온라인 공간 확장과 선형적인 입출력 성능 향상**

서비스의 중단 없이 자유롭게 스토리지를 증설, 제거, 유지 보수할 수 있는 기능을 제공하며 온라인 스토리지 증설을 통해 저장 공간의 확장과 부가적인 입출력 성능의 향상을 기대할 수 있다.

· **높은 가용성**

수많은 저비용 스토리지 서버들을 이용하므로 빈번하게 장애가 발생할 수 있다. 장애가 발생한 서버나 디스크로 인해 서비스가 지연되거나 중단되는 것을 방지하기 위해 장애를 감지하고 복구하는 기능이 제공된다. 장애의 발생으로 데이터가 유실되는 것을 방지하기 위해 모든 데이터는 일정한 수만큼 다른 서버들에 복제되어 유지된다.

· **POSIX에 준하는 파일 시스템 API**

별도의 고유한 API를 제공하는 것이 아니라 POSIX에 준하는 파일 시스템 API를 제공함으로써 GLORY-FS로 파일 시스템을 변경하더라도 기존의 애플리케이션들을 수정할 필요가 없다.

· **네트워크 구조 인식**

서버, 스위치, 랙 등의 네트워크 위상 구조를 인식하고 이를 이용하여 데이터를 최적으로 배치하는 기능을 제공한다.

· **Hot Spot 회피**

접근이 빈번한 서버나 파일을 검출하여 데이터 재분배나 복제본 증가를 통해 Hot Spot을 회피할 수 있는 기능을 제공한다.

### 3.2 GLORY-FS의 구조

GLORY-FS는 다수의 클라이언트와 메타데이터를 저장하고 관리하는 메타데이터 서버(MDS)들의 클러스터, 데이터를 저장하는 다수의 데이터 서버(DS)들이 네트워크로 연결되어 있는 비대칭형 구조로 그림 4와 같다.

GLORY-FS에서는 모든 DS들의 저장 공간을 하나의 저장 공간인 것처럼 가상화하여 클라이언트에게 제공한다. 서비스를 중단하지 않고 온라인 상으로 새로운 DS를 추가하여 저장 공간을 확장할 수 있으며, 디스크마다 고유하게 부여되는 ID를 통해 서버간 디스크의 이동을 가능하게 하여 장애가 발생한 서버의 디스크들을 다른 서버로 이동하여 서비스를 제공할 수 있다. GLORY-FS에서 하나의 파일은 일정한 크기의 청크들로 나뉘어져서 다수의 DS들로 분산되어 저장되며 가용성을 위해 동일한 청크에 대해 일정 수의 복제본들이 서로 다른 DS들에 저장된다. 이러한 청크들과 복제본들의 위치 정보는 해당 파일에 대한 메타데이터로서 MDS에 저장된다.

#### 3.2.1 GLORY-FS 클라이언트

GLORY-FS 클라이언트는 그림 5에서 볼 수 있는 것

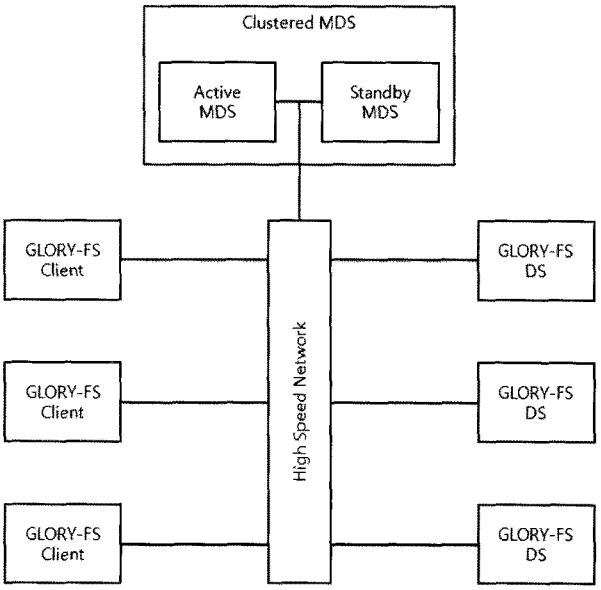



그림 4 GLORY-FS의 구조

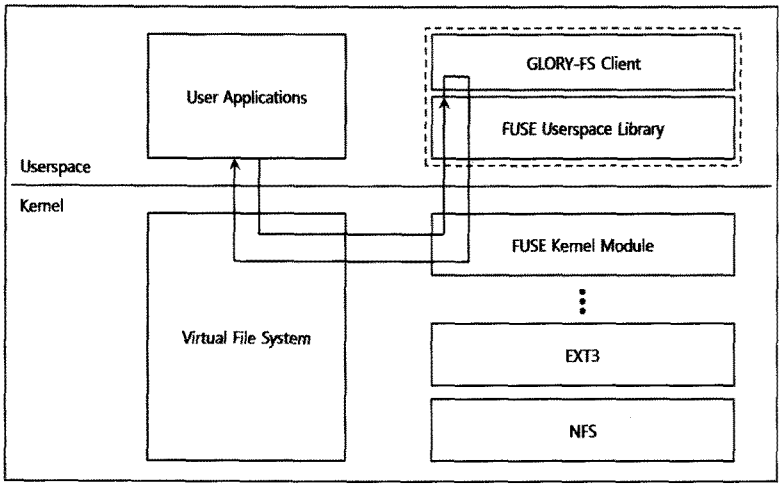



그림 5 FUSE 기반 클라이언트

과 같이 FUSE[6]를 기반으로 사용자 레벨에서 동작하며 GLORY-FS를 가상 시스템 상에 마운트하여 네임스페이스 연산과 파일 입출력 연산을 수행한다. 네임스페이스 연산은 파일, 디렉토리, 심볼릭(symbolic) 링크의 생성, 삭제, 검색, 재명명 연산뿐만 아니라 ACL (access control list)와 같은 속성과 관련된 연산을 포함한다.

GLORY-FS 클라이언트는 파일 입출력 연산을 수행하기 위해 MDS에게 해당 파일의 위치 정보를 요청하고 MDS는 그 파일을 구성하는 청크들이 위치한 DS들의 리스트를 반환한다. GLORY-FS 클라이언트는 반환받은 DS들의 리스트에 따라 DS들에 직접 접근하여 파일 입출력을 수행한다.

### 3.2.2 GLORY-FS 메타데이터 관리

MDS는 DBMS를 활용하여 메타데이터를 저장하고 관리한다. 메타데이터 저장소로 DBMS를 사용하면 대량의 파일들을 접근할 때 일정한 성능을 확보할 수 있으며 별도의 인덱스를 유지할 필요가 없다. MDS는 네임스페이스, 속성, 위치 정보, 통계 정보와 같은 메타데이터를 저장하기 위해 적절한 DB 모델링을 통해 DBMS에 저장한다. 이러한 적절한 DB 모델링을 통해 메타데이터 연산과 관리의 효율성을 극대화 하였다. DB에 있는 메타데이터를 접근하는 모든 연산은 ODBC를 통하여 이루어지며 내부적으로 SQL을 사용한다.

### 3.2.3 GLORY-FS의 데이터 관리

DS는 데이터의 무결성 보장과 안전한 보관, 그리고 효율적 관리와 최적의 입출력 성능 보장을 최우선으로 한다. 사용자의 파일은 DS에서 청크의 단위로 저장이 되며 각 청크는 잘 구성된 디렉토리의 임의 위치에 청크의 아이디를 이름으로 하는 일반 파일로 저장된다. 청크와 파일의 크기에 따라 하나의 파일을 저장하는 청크의 수가 달라지며, 각 청크들은 시스템의 부하 분산과 저장 공간의 효율적인 사용을 위해 네트워크 위상을 인지하여 여러 DS들 중 가장 적합한 곳에 위치하게 된다.

청크 파일은 두 개의 버전 정보와 실제 데이터로 구성되며, 두 개의 버전은 초기에 0으로 설정되고 청크 파일의 모든 변경사항이 발생할 때마다 버전이 증가한다. DS는 청크 파일의 변경사항이 발생할 때마다 pre-version 증가, 변경사항 갱신, post-version 증가의 3단계를 수행한다. DS는 데이터의 무결성을 보장하기 위해 청크 파일의 버전 정보를 활용한다. 청크 파일의 변경이 3단계의 순서로 수행되는 도중 오류로 인한 중단이 발생하면 pre-version과 post-version의 값이 다를 것이다. 간단하게 두 버전 값을 비교함으로써 청크 파일이 올바르게 저장되었는지를 판단해 볼 수 있다.

GLORY-FS 파일 시스템은 DS의 장애로 인한 데이터의 유실을 방지하기 위하여 동일한 청크 파일을 여러 DS에 중복 저장하는 복제 정책을 사용한다. DS는 이러한 복제 정책을 지원하기 위한 데이터 복제 기능을 지연된 복제라는 형태로 수행한다. 하나의 파일에 상응하는 다수의 청크 파일들은 우선적으로 primary DS 한곳에 저장이 되며, 복제의 수행은 이러한 primary DS가 존재하는 환경에서 동일한 데이터를 단지 다른 DS로 복제를 하는 것이다. 지연된 복제는 클라이언트가 파일을 저장할 때 즉시 복사본이 생성되는 것이 아니고, primary DS에 저장이 완료된 후에 별도로 복제를 수행한다는 의미이다. 그림 6은 복제가 수행되는 절차를 나타낸 것이다.

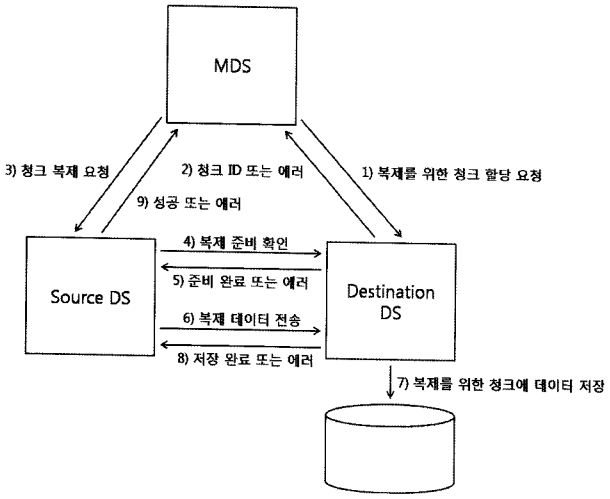



그림 6 복제 절차

### 3.2.4 MDS 클러스터링

GLORY-FS가 지원하는 MDS 클러스터링은 그림 7에서처럼 가장 간단한 active-standby 형태이다. active 상태의 마스터 MDS는 모든 메타데이터 연산을 처리하며, standby 상태의 슬레이브(slave) MDS는 마스터 MDS에 장애가 발생했을 때 active 상태로 전환하여 read-only 모드로 동작한다. 클러스터 MDS는 DBMS의 비동기적인 복제 기능을 활용하여 마스터 MDS의 DB에 저장된 메타데이터를 standby MDS의 DB로 복제함으로써 메타데이터 동기화를 수행한다. 마스터와 슬레이브 MDS는 heartbeat을 통해 상호간에 서버 장애 발생을 감지하고 장애 발생이 감지되었을 때 장애에 대처하기 위한 장애 처리 루틴을 수행한다. 클라이언트와 DS는 마스터 MDS의 가상 IP를 통해 정보들을 교환하며, 마스터 MDS와 슬레이브 MDS는 실제 IP를 통해서 정보들을 교환한다.

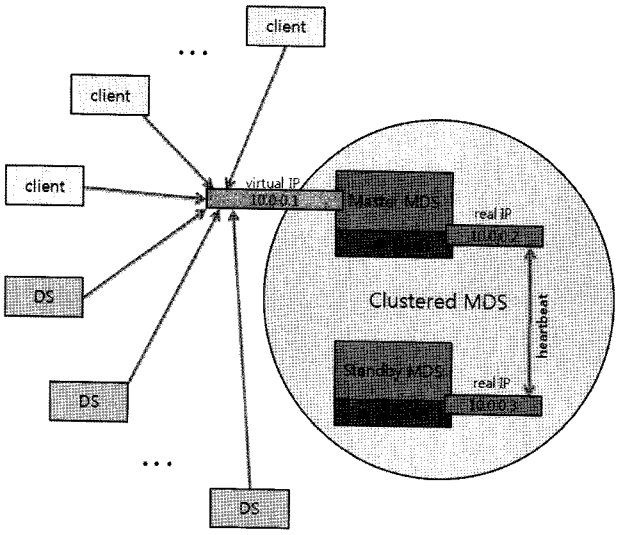



그림 7 Active-Standby MDS 클러스터 구조

## 4. 클라우드 컴퓨팅을 위한 분산 파일 시스템 기술 연구 과제

클라우드 컴퓨팅을 위한 대표적인 분산 파일 시스템들은 성능과 확장성, 그리고 신뢰성과 가용성 측면에서 대규모 시스템을 구축할 수 있는 비대칭형 구조를 취하고 있다. 비대칭형 구조는 메타데이터와 데이터의 흐름을 분리하여 데이터 입출력 성능을 높이면서 독립적인 확장과 안전한 서비스를 제공하는 것이 가능하다. 하지만 이 구조에서 가장 문제가 되는 부분은 메타데이터를 관리하는 단일 서버로 인해 발생하는 것들이다. 메타데이터를 관리하는 단일 서버의 장애 발생으로 인해 서비스가 지연될 수 있으며, 최악의 경우에는 서비스가 중단될 수도 있다. 또한 메타데이터의 양과 클라이언트 수의 증가로 인해 단일 서버가 처리할 수 있는 한계를 넘어서는 상황이 발생할 수도 있다. 단일 서버로 인해 발생할 수 있는 문제들에 대한 해결책으로 제시되고 있는 것이 메타데이터를 관리하는 서버의 클러스터링이다. 여러 서버가 클러스터를 구성하여 메타데이터를 관리하고 처리함으로써 장애 발생에 대한 대비와 메타데이터 처리 성능 향상을 꾀할 수 있다. Active-standby 형태의 메타데이터 서버 클러스터링 방법은 메타데이터 서버의 가용성을 보장할 수 있는 가장 기초적인 형태로 단일 서버가 처리할 수 있는 한계를 넘어서는 상황에는 적합하지 못하다. 따라서 위에서 언급한 문제들을 해결하기 위해서는 N개의 메타데이터 서버들이 전체 메타데이터를 나눠서 관리하는 active-active의 형태의 메타데이터 서버 클러스터링이 필요하다.

가용성을 위한 데이터의 중복 저장은 실제 데이터의 양보다 더 많은 저장 공간을 요구하고 있다. 이러한 저장 공간의 요구는 인프라 구축비용의 증가에 따른 서비스의 비용 증가로 이어져서 서비스 경쟁력을 저하시키는 요인이 된다. 결과적으로 가용성을 유지하면서 데이터의 중복 저장으로 인한 저장 공간을 최소화시키는 것이 커다란 과제이다. 데이터의 중복 저장은 장애 발생으로 인한 데이터의 복구를 위해 유지하는 복제본과 여러 사용자들이 동일한 데이터를 다른 이름으로 저장하는 두 가지 측면을 볼 수 있다. 현재 이러한 데이터의 중복 저장을 개선하기 위한 방법들이 지속적으로 연구가 되고 있으며 대표적으로 이중화 패리티 기술과 데이터 중복 제거 기술이 있다[7,8].

## 5. 결 론

본 논문에서는 클라우드 컴퓨팅에서 사용되는 대

표적인 분산 파일 시스템들을 살펴보고 ETRI에서 개발한 GLORY-FS 파일 시스템 기술을 소개함으로써 클라우드 컴퓨팅을 지원하기 위한 분산 파일 시스템 기술 이슈를 제시하였다.

클라우드 컴퓨팅에서 분산 파일 시스템이 갖추어야할 대표적인 사항들을 요약해보면, 첫째로 저비용의 스토리지 서버들을 활용하여 막대한 양의 데이터를 효율적으로 저장하고 관리할 수 있어야 하며 어떠한 장애에도 서비스의 지연이나 중단이 발생하지 않도록 적절하게 대처해야 한다는 것이다. 무엇보다 안정성이 절실하게 요구된다. 둘째로 서비스가 필요로 하는 데이터 입출력 성능을 보장해야 한다는 것이다. 네트워크 위상 인지(network topology aware)를 통한 데이터 배치와 선택, 효과적인 캐시의 사용 등과 같은 데이터 입출력 성능을 개선하기 위한 방법들이 제공되어야 한다. 셋째로 지능적인 시스템 관리를 통해 관리자의 개입을 최소화하고 서비스의 중단 없이 유지 보수가 가능해야 한다는 것이다.

대용량 인터넷 서비스를 위해 개발된 GLORY-FS는 클라우드 컴퓨팅을 목표로 하지는 않았지만 클라우드 컴퓨팅에서 분산 파일 시스템이 갖춰야할 사항들을 이미 갖추고 있으며 현재 클라우드 컴퓨팅에서 사용되고 있는 분산 파일 시스템들을 살펴보았을 때 향후 클라우드 컴퓨팅을 위한 분산 파일 시스템으로서 충분히 활용할 수 있는 가능성을 가지고 있다.

현재까지 GLORY-FS는 여러 인터넷 서비스 업체들과 함께 하면서 BMT와 ROI 분석을 통해 안정적인 성능과 저렴한 구축비용을 인정받았으며 최근에는 국내 주요 인터넷 포털에서 실제 서비스를 지원하는 분산 파일시스템으로 운용되고 있다.

클라우드 컴퓨팅을 위한 분산 파일 시스템 기술은 아직까지 해결해야할 과제들이 남아있다. 특히 메타데이터를 관리하는 서버의 클러스터링과 데이터의 중복 저장으로 인한 저장 공간 낭비의 최소화는 중요한 기술적 이슈로 남아있다.

## 참고문헌


[1] 한국소프트웨어진흥원, "클라우드 컴퓨팅의 현재와 미래, 그리고 시장전략", 2008. 10.

[2] Cloud computing, http://en.wikipedia.org/wiki/Cloud_computing

[3] Cloud computing with Linux, http://www.ibm.com/developerworks/linux/library/l-cloud-computing

[4] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, "The Google File System," In Proc. of ACM Symp. on Operating Systems Principles, 2003, pp. 20-43.

[5] HDFS, http://hadoop.apache.org/core/docs

[6] FUSE, http://fuse.sourceforge.net

[7] Who uses Hadoop, http://wiki.apache.org/hadoop/PoweredBy

[8] Non-standard RAID levels, http://en.wikipedia.org/wiki/Non-standard_RAID_levels

[9] Data deduplication, http://en.wikipedia.org/wiki/Data_deduplication

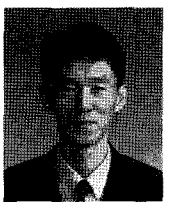


**민 영 수**

1998 충북대학교 정보통신공학 학사
2001 충북대학교 정보통신공학 석사
2006 충북대학교 정보통신공학 박사
2006~현재 ETRI 인터넷플랫폼연구부 저장시스템연구팀
관심분야 : 파일 시스템, 데이터베이스 시스템, 다차원 색인 구조 등
E-mail : minys@etri.re.kr

**김 홍 연**

1992 인하대학교 통계학과 학사
1994 인하대학교 전자계산학과 석사
1999 인하대학교 전자계산학과 박사
1999~현재 ETRI 인터넷플랫폼연구부 저장시스템연구팀
관심분야 : 스토리지 시스템, 파일 시스템, 데이터베이스 시스템 등
E-mail : kimhy@etri.re.kr



**김 영 균**

1991 전남대학교 전산통계학과 학사
1993 전남대학교 전산통계학과 석사
1995 전남대학교 전산통계학과 박사
1995~현재 ETRI 인터넷플랫폼연구부 저장시스템연구팀장
관심분야 : 스토리지 시스템, 파일 시스템, 데이터베이스 시스템 등
Email: kimyoung@etri.re.kr