Distributed File System을 위한 Java 기반 Management System 설계 및 구현

김태형, 정규식
숭실대학교 정보통신전자공학부
iris@q.soongsil.ac.kr, kchung@q.soongsil.ac.kr

A Design and its Implementation of Java based Management System for Distributed File Systems

Tae Hyoung Kim, Kyusik Chung
School of Electronic Engineering, Soongsil University
iris@q.soongsil.ac.kr, kchung@q.soongsil.ac.kr

Abstract

최근 인터넷 사용자가 늘어나면서 클러스터링 기반의 대형 서버가 등장하게 되었고 또한 분산파일시스템(DFS)에 대한 관심이 커지게 되었다. 기존 분산파일 시스템들은 파일시스템 측면의 많은 기능들이 제공되고 뛰어난 성능을 발휘하지만 사용자 인터페이스 및 관리 측면에서 본다면 미흡한 점이 많다.
본 논문에서는 분산파일 시스템의 종류와 플랫폼에 상관없이, 사용하기 쉬운 DFS Management System을 설계하고 구축한다. 전체 시스템 구조를 파악 할 수 있도록 GUI 환경을 제공하며, 모듈화/계층화 구조로 설계하는데 기존 DFS와 연동하기 위한 하부 통신모듈이 포함된다. 본 논문에서는 Java기반으로 DFS Management System을 구축하였으며, Coda DFS가 설치된 클러스터링 서버와 연동하여 시험하였다.

I. 서론

최근 인터넷 사용자가 늘어나면서 대형 웹서버에 대한 요구가 증가하게 되고 클러스터링 기반의 대형 서버가 등장하게 되었다. 이러한 클러스터링에 관심이 집중되면서 클러스터링 서버에 사용될 분산파일시스템(Distributed File System, DFS)에 대한 관심 또한 커지게 되었다.

분산파일 시스템은 파일 시스템 관점에서 본다면 많은 기능들이 추가되고 뛰어난 성능을 발휘하지만 사용자 인터페이스 및 관리(Management) 측면에서 본다면 미흡한 점이 많다. 또한 각 분산파일 시스템의 사용법이 달라 분산파일시스템이 바뀌면 사용자는 그 사용법을 다시 학습해야 하는 불편함들이 뒤따랐다. 대표적인 분산파일시스템들을 보면 AFS, Coda, NFS, InterMezzo 등을 꼽을 수 있다.
본 논문에서는 분산파일 시스템의 종류와 플랫폼(Platform)에 상관없이 사용자에게 쉽게 다가갈 수 있는, 클러스터링 서버 환경에서의 분산 파일 시스템을 위한 Java 기반의 매니지먼트 시스템을 설계하고 구현한다.

II. 연구배경

2.1 AFS(Andrew File System)[1]

AFS는 File Replication이 가능하므로 Data 접근성이 뛰어나고, 서비스 중에도 파일 서버, 클라이언트의 추가 및 제거가 가능하며 그 확장성이 뛰어나다. 또한 File Replication을 사용하여 Single Point Failure를 방지할 수 있고, Backup 및 시스템 장애 시에도 서비스를 지속할 수 있다. 클라이언트 캐시를 사용하기 때문에 네

트워크 서버의 부하를 줄일 수 있으며 빠른 파일 액세스가 가능하다.

2.3 NFS(Network File System)[2]

NFS는 가장 많은 플랫폼을 지원한다는 장점이 있으나 클라이언트 캐시를 지원하지 않기 때문에 네트워크 서버에 많은 부하를 주게 되며 항상 서버에 접속해서 파일을 액세스하기 때문에 액세스 속도 또한 느려지게 된다. File Replication을 지원하지 않고 많은 클라이언트들이 데이터 파일에 동시에 접근할 수 없다는 문제점들을 안고 있어 제한적인 확장성을 갖는다.

2.4 InterMezzo[3]

InterMezzo는 High Availability에 초첨을 맞춘 새로운 분산 파일 시스템으로 디렉터리의 복제, Disconnected Operation, 클라이언트 캐시 등의 기능을 지원하지만 현재 베타버전의 상태에 있으며 LINUX 플랫폼만을 지원한다는 단점이 있다.

2.5 CODA[4]

Coda file system은 1983년에 AFS-2에서 분류되어 나온 것으로 File Replication이 가능하므로 Data 접근성이 뛰어나고, 그 확장성이 뛰어나다. File Replication을 사용가능하며, Backup 및 시스템 장애 시에도 서비스를 지속할 수 있다. Disconnected Operation 기능을 이용하면 네트워크 자체의 이상에 대해서도 대처 할 수 있으며 클라이언트 캐시를 사용하기 때문에 네트워크 서버의 부하를 줄일 수 있으며 빠른 파일 액세스가 가능하다.

이처럼 분산파일 시스템은 파일 시스템 관점에서 본다면 많은 기능들이 추가 되고 뛰어난 성능을 발휘하지만 사용자 인터페이스 및 관리(Management) 측면에서 본다면 Command Interpreter 형태만을 지원하는 등 미흡한 점이 많다. 또한 사용법이 달라 분산파일시스템이 바뀌면 사용자는 그 사용법을 다시 학습해야 하는 불편함 들이 뒤따랐다. 이에 본 논문에서는 실제 사용자가 사용하기 쉽고 전체 시스템 구조를 한눈에 파악 할 수 있도록 GUI 환경을 적용하여 DFS Management System을 구현하도록 한다.

III. DFS Management System 설계

3.1 전체 시스템 구조

기존에 구성되어 있는 분산파일 시스템 환경에 본 논문에서 구현하고자 하는 DFS Management Server와 DFS GUI로 전체적인 구조를 이루게 된다. 그림 1과 같이 DFS Management Server는 하나의 머신을 할당하여 기존의 분산파일 시스템과 같은 네트워크 상에 설치되며 DFS GUI는 분산파일 시스템을 관리하는 관리자 머신에 설치되어 사용된다.

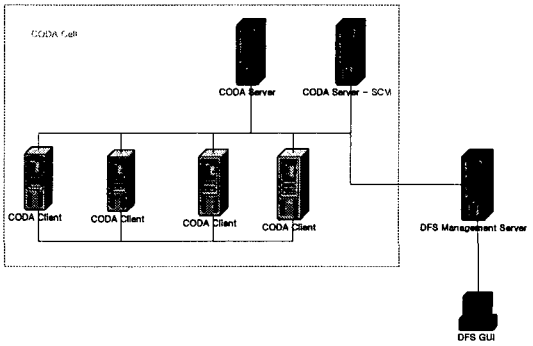

그림 1 시스템 전체 구성도

전체 시스템은 작업 수행에 따라 그림 2와 같은 총 4개 Layer의 구조로 표현할 수 있다.

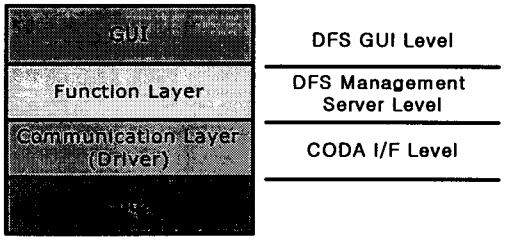

그림 4 시스템 Layer

DFS GUI Level은 직접적으로 사용자의 입력을 받는 부분이며, DFS Management Server Level은 사용자의 입력을 기존의 Coda 시스템으로 전달하기 위해 Coda I/F Level을 호출한다. 마지막으로 Coda I/F는 실제적으로 DFS Management Server로부터 받은 명령을 Coda 시스템으로 전달하는 역할을 하며 Coda I/F 부분을 변경함으로써 다른 종류의 분산파일 시스템으로의 포팅이 용이하도록 설계하였다.

3.2 구현기능

기본적으로 Coda에서 제공하는 기능들을 좀더 편리하고 효율적인 방법으로 관리하기 위해 DFS Management System에서 다음과 같은 기능을 사용자에게 제공한다.

- Machine, VSG, Volume, Directory 단위의 View
- Server add, remove, start, stop
- VSG add, remove
- Volume add, remove, move, mount

3.2 DFS Management Server

DFS Management Server는 Coda/IF와 DFS GUI를 연결시켜 주는 중간자적인 역할을 담당하며 그림 3과 같이 총 5개의 state로 동작한다. Command Wait State에서 DFS GUI의 명령을 받아 Process State에서 이를 처리해 DFS GUI에 응답하는 과정을 반복해서 동작하게 된다.

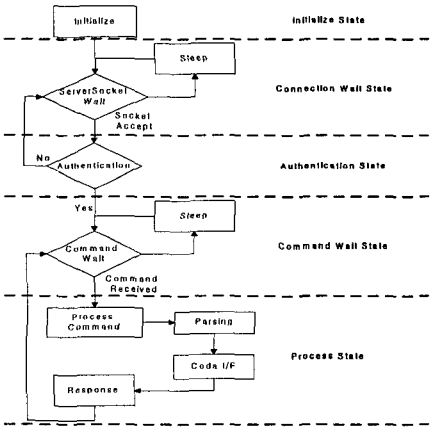

그림 3 Server 전체 흐름도

DFS Management Server의 각 모듈별 전체구조는 그림 4와 같이 DFS GUI 와의 통신을 위한 GUI Communicator와 GUI로부터 받은 명령을 처리하기 위한 Command Communicator, CODA I/F에 명령을 전달하기 위한 CODA I/F Communicator로 구성된다.

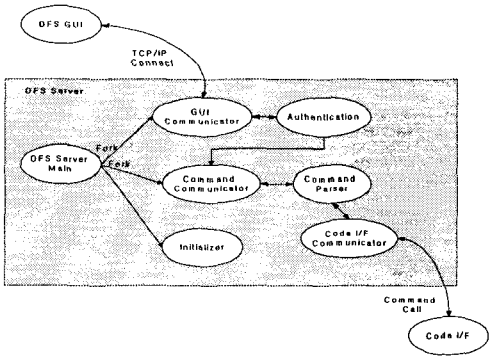

그림 6 DFS Management Server 전체 구조도

3.3 DFS GUI

DFS GUI는 실제적으로 사용자와 접하는 부분으로 사용자가 내리는 명령을 DFS Management Server에 전달하는 Control Information(CI) 부분과 화면 인터페이스부분으로 나누어 질 수 있다. DFS GUI 전체 구조도를 살펴보면 그림 5와 같다. Control Info(CI) Manager는 각 자료구조를 유지하고 변화된 자료구조에 대한 갱신을 하는 역할을 한다. CI Manager는 자료구조의 갱신을 위해 Command Communicator를 호출해서 사용자로부터의 입력을 Protocol 형태로 변경하고 Command Communicator는 Connection Manager를 호출해 Client Socket을 통해 DFS Management Server에 명령을 전달한다. DFS Management Server는 이에 따른 작업 후에 응답을 돌려주게 되고 Client Socket은 DFS Management Server로부터 들어온 응답을 Parser에게 넘겨주어 제대로 된 응답인지의 여부를 체크한 후 주어진 Protocol에 따라 Parsing작업을 수행한다. 이는 다시 Connection Manager를 통해 Command Communicator로 올라가고 CI Manager에 도착하게 되면 CI Manager는 받아들인 데이터를 유지하고 있던 자료구조의 형태로 변화시켜 새로운 자료구조를 생성하거나 갱신하게 된다.

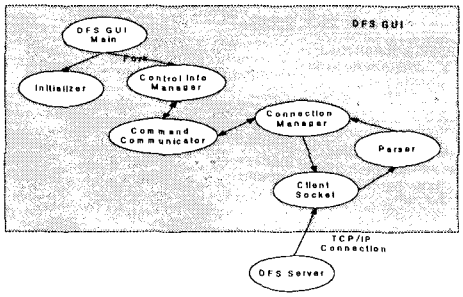

그림 5 DFS GUI 전체 구조도

3.4 화면 인터페이스

화면 인터페이스의 전체 구조는 그림 6과 같다.

화면 인터페이스는 크게 MainFrame, Menu, Left Tree Pane, Right Information Pane의 4가지 부분으로 나뉠 수 있다.

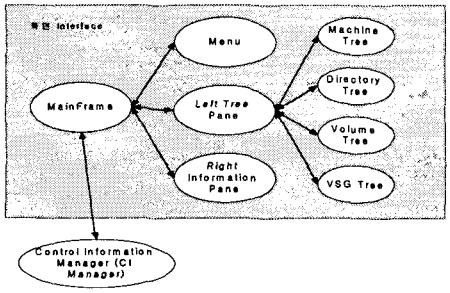

그림 6 화면 인터페이스 구조

3.5 명령 프로토콜 (Command Protocol)

DFS GUI에서 사용자의 입력에 따라 내려진 명령은 프로토콜의 형태로 변경된 후 Client Socket에서 TCP/IP 소켓 스트림을 사용해서 DFS Management Server로 전달된다. 이를 위해 DFS GUI와 DFS Management Server 사이는 정해진 프로토콜로 명령의 교환이 이루어진다. 프로토콜은 Text 형식으로 전달되며 프로토콜 자체도 일정한 형식을 갖게 된다. 그림 7은 프로토콜의 구조이다.

Command	CMDSTART	COMMAND CODE	COMMAND INFORMATION	CMDEND
Response	RSPSTART	COMMAND CODE	RESPONSE INFORMATION	RSPEND

그림 7 Command Protocol 구조

3.6 CODA I/F (Coda Interface)

기존의 Coda 시스템과 DFS Management Server를 연결하기 위한 드라이버 계층으로 DFS Management Server로부터 받은 명령을 Coda 시스템에 실제적으로 전달하고 Coda System으로부터 받은 결과를 DFS Management Server로 넘겨주는 역할을 한다. CODA I/F는 스크립트 형태로 구현되어 있으며 SSH을 사용하여 Coda 시스템과 통신을 한다.

V. DFS Management System 구현

관리하기 위한 분산파일 시스템으로는 CODA를 사용하였다. 서버 2대와 클라이언트 3대를 사용하여 실험을 진행하였으며 설계시 구현 하고자 하였던 각 기능들이 제대로 동작함을 확인하였다. 그림 8은 실제 동작화면이다.

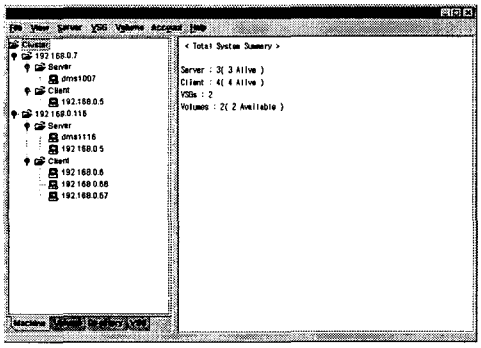

그림 8 실제 동작 화면

VI. 결론

본 논문에서는 Coda 시스템에서 Coda 서버가 하는 여러 가지 일련의 작업들을 사용자가 좀더 편리하게 할 수 있도록 관리부분에 초점을 맞춰 GUI형태의 통합 관리 도구를 제공하며, 트리형태의 뷰를 제공하기 때문에 현재 클라이언트/서버의 관계를 명확히 눈으로 알 수 있어 사용에 용이하고, 하위 분산파일 시스템의 동작과는 상관없이 동작하기 때문에 현재 분산파일시스템의 변경 없이 사용이 가능하다는 장점을 갖는다.

또한 분산 파일 시스템에 의존적이지 않은 독립적인 모듈로 구성하기 위해 계층적인 구조를 갖도록 설계되었다. 기능적인 추가만을 추구하던 비효율적인 분산파일 시스템의 매니지먼트에 효율적인 대안을 제시한다는 것에 이 논문의 의의가 있다고 할 수 있다.

이러한 장점들 이외에 현재 서버의 상태를 DFS Management Server에게 알려줄 수 있는 에이전트(Agent)를 사용한다면 사용자에게 현재 서버의 상태를 시각적인 메시지로 전달 할 수 있을 것이고, 이 에이전트를 통해 현재의 Load를 측정하여 Load가 많은 Server의 File을 다른 Server로 Replication 하여 Load가 적은 서버 쪽으로 Load를 분산시키는 Load balancing도 가능해 지므로 좀더 나은 통합 관리 도구로 사용할 수 있을 것으로 보인다. 이를 위해서는 서버의 상태를 알려줄 수 있는 에이전트의 개발이 선행 되어야 할 것이며, Load를 측정하는 방법에 대해 고찰할 필요가 있을 것이다.

참고문헌

[1] Satyanarayanan, M., Howard, J.H., Nichols, D.N., Sidebotham, R.N., Spector, A.Z., West, M.J., "The ITC Distributed File System: Principles and Design", Proceeding of the 10th ACM Symposium on Operating System Principles, Orcas Island., Dec. 1985.

[2] Sandberg, R., et al., "design and Implementation of the Sun Network File System.", Proc. Summer Usenix Conf., Portland, 1985, pp. 119-13

[3] http://www.inter-mezzo.org/docs/opc99_html/index.html

[4] M. Satyanarayanan et al., "Coda: A Highly Available File System for a Distributed Workstation Environment", IEEE Trans. Computers, Vol. 39, No. 4, Apr. 1990, pp. 447-459.