하둡을 이용한 온라인 대용량 저장소 구현

엄세진*, 임승호*
*한국외국어대학교 디지털정보공학과
e-mail:slim@hufs.ac.kr

Implementation on Online Storage with Hadoop

Se-Jin Eom*, Seung-Ho Lim*
*Dept of Digital Information Engineering, Hankuk University of Foreign Studies

요 약

최근 페이스북이나 트위터와 같은 소셜네트워크 서비스를 포함하여 대용량의 빅데이터에 대한 처리와 분석이 중요한 이슈로 다뤄지고 있으며, 사용자들이 끊임없이 쏟아내는 데이터로 인해서 이러한 데이터들을 어떻게 다룰 것인지, 혹은 어떻게 분석하여 의미 있고, 가치 있는 것으로 가공할 것인지가 중요한 사안으로 여겨지고 있다. 이러한 빅데이터 관리 도구로써 하둡은 빅데이터의 처리와 분석에 있어서 가장 해결에 근접한 도구로 평가받고 있다. 이 논문은 하둡의 주요 구성요소인 HDFS(Hadoop Distributed File System)와 JAVA에 기반하여 제작되는 온라인 대용량 저장소 시스템의 가장 기본적인 요소인 온라인 데이터 저장소를 직접 설계하고 제작하고, 구현하여 봄으로써 대용량 저장소의 구현 방식에 대한 이슈를 다뤄보도록 한다.

1. 서론

최근 IT산업의 가장 큰 화두중 하나로 빅데이터의 처리와 분석을 꼽지 않을 수는 없을 것이다. 페이스북이나 트위터와 같은 소셜네트워크 서비스를 이용하는 사용자들이 끊임없이 쏟아내는 단문의 텍스트들로부터, 대중교통의 실시간 서비스 정보 혹은 주식시장에서의 각종 금융정보에 이르기까지 현대 사회에는 매 순간마다 그 수를 헤아릴 수 없을 정도의 다량의 데이터가 생산되고 있으며, 이러한 데이터들을 어떻게 다룰 것인지, 혹은 어떻게 분석하여 의미 있고, 가치 있는 것으로 가공할 것인지에 대한 고민은 대단히 중요한 사안으로 여겨지고 있다. 그리고 하둡은 이러한 빅데이터의 처리와 분석에 있어서 가장 해결에 근접한 도구로 평가받고 있다.

이 논문은 Hadoop의 주요 구성요소인 HDFS(Hadoop Distributed File System)와 JAVA에 기반하여 제작되는 온라인 대용량 저장소 시스템의 가장 기본적인 요소를 직접 설계하고 제작하는 과정과 고찰에 대하여 다루고 있다. HDFS라는 최신의 거대한 가상의 파일 시스템 위에서 서버를 동작시키고 클라이언트를 제작하여 서버와 클라이언트간의 통신이 이루어지기까지에 대한 내용으로, 이 시스템의 구성과 제작과정, 그리고 이 시스템이 가지는 나름대로의 기술적 의미에 대하여 고찰할 것이다.

본 논문에서는 온라인상에서 마치 로컬 하드 드라이이브의 대명사인 C드라이브와 같이 마음껏 데이터를 쓰고 읽을 수 있는 시스템을 직접 제작 및 구현해 봄으로써 하둡을 이용한 온라인 저장소의 활용방안 및 쉬운 구현 방안에 대해서 알아보도록 한다.

2. C-drive 디자인 및 구현

본 논문에서 구현한 온라인 저장소의 이름을 클라우드 시스템 드라이브라는 의미로 C-drive라고 하였으며, 이의 구현사항에 대해서 기술하도록 한다. C-drive의 기본적인 시스템의 컨셉은 매우 간단하다. C-drive는 근본적으로 온라인에서 대용량의 데이터를 저장할 수 있는 가상의 디스크를 제공하는 개념의 서비스를 구축하는 것으로, 서버와 클라이언트가 존재해야하고, 서로 간에 데이터를 송수신할 수 있어야 하며, 클라이언트 쪽에선 사용자를 위한 UI가 존재해야하고, 서버 측에서는 사용자의 계정정보 및, 저장된 파일을 관리할 수 있어야한다는 전제를 확보해야 한다. 시스템 컨셉을 바탕으로 도식화한 시스템 구성은 그림 1과 같으며, 크게 네 부분으로 구성되어 있다.

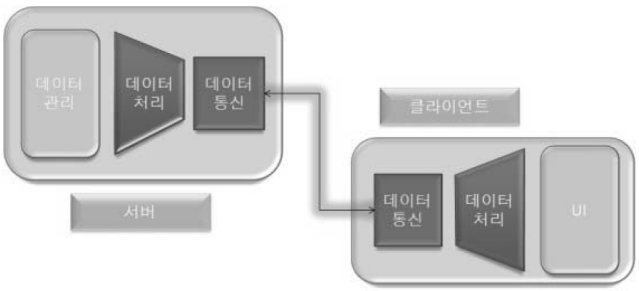

그림 1 C-drive의 기본 구성

1) 데이터 통신부

서버와 클라이언트 간의 데이터가 송수신되는 부분으로, 소켓프로그래밍을 통해 제작되며, 통신 프로토콜은

TCP/IP를 사용한다. 기본적으로 사용자의 계정 및 사용자가 원하는 데이터의 송수신이 일어나는 부분이다.

2) 데이터 처리부

서버와 클라이언트가 데이터 통신부를 통하여 전달한 혹은, 전달받은 데이터를 각자 주어진 과정을 통하여 사용자의 요구에 맞게 데이터를 처리하는 부분을 담당하는 부분이다.

3) 데이터 관리

서버의 경우, 클라이언트로부터 전달 받은 데이터를 어떻게 보관하고 관리할 것인지 결정하는 부분이다. 계정정보와 계정마다 할당되는 계정공간 내 데이터 관리를 담당하는 부분이다.

4) UI

클라이언트의 경우, 서버로부터 전달받은 계정정보 및 저장되어 있는 데이터들을 어떻게 시각화 시키고, 또 그러한 데이터를 사용하기 위한 사용자의 명령과 그에 따른 결과를 어떻게 보여줄 것인지 결정하는 부분이다.

기본적으로 C-drive에서의 데이터의 흐름은 그림 2와 같이 TCP/IP를 통한 데이터 송수신을 1:1 통신을 전제로 하고 있다. 따라서 서버 컴퓨터에 접속하고자 하는 클라이언트의 숫자가 기하급수적으로 늘어나거나 몇몇 소수의 클라이언트가 과도한 데이터 송수신을 요구할 경우, 데이터의 대역폭을 감당할 수 없을 것이다. 이러한 데이터의 병목현상을 어느 정도 해소시켜줄 수 있는 솔루션을 가지고 있어야 한다. 소켓 프로그래밍에서는 이러한 문제에 대응하기 위하여 기본적으로 멀티플렉싱, 멀티쓰레딩과 같은 자체적인 솔루션을 가지고 있으나 이것만으로는 현실적으로 문제를 해결할 수 없는 것이 사실이다. 한 개의 프로세스에서 다수의 소켓을 열어 대응하는 멀티플렉싱은 대용량 데이터를 안정적으로 송수신하는데에 문제점이 있으며, 멀티 쓰레드는 프로세스가 동작하는 서버의 프로세스에 과도한 부하를 일으킨다는 문제가 있다.

이러한 데이터의 병목현상을 해소하기 위해서 기본적인 멀티쓰레드의 사용 외에도 C-drive는 일종의 로드밸런싱의 개념을 추가하였다. C-drive 시스템의 서버는 메인서버와 서브서버가 나뉘어져 있으며, 모든 클라이언트는 최초로 메인서버를 방문하게 된다. 이때, 메인서버는 방문한 클라이언트들에게 자신이 알고 있는 서브서버들 중 하나의 IP주소를 클라이언트에게 돌려준다. 클라이언트는 앞선 과정을 통해 새롭게 알게 된 서브서버의 IP주소를 이용하여 자신에게 배정된 서브서버로 접속하여 자신의 ID, 및 패스워드 정보를 전달한다. 서브서버는 클라이언트로부터 받은 계정정보를 조회하여, 해당 계정에 할당된 데이터 공간을 찾아가 얻어낸 계정정보를 클라이언트에게 전달해주고, 클라이언트의 이후 요청명령을 위해 대기한다. 만약 클라이언트에게서 어떠한 요청이 있다면 그에 맞는 동작을 수행한다.

C-drive의 송수신을 중심으로 한 데이터의 전반적인 흐름을 살펴보았다면, 이제는 데이터의 처리를 살펴보도록 한다. 앞서 언급한 대로, 데이터의 송수신은 메인서버 및 서브서버를 통하여 클라이언트와의 통신하고 있다면. 데이터의 처리는 서브서버와 하둡 HDFS간의 데이터 읽기/쓰기로 설명할 수 있다. 하둡은 HDFS를 포함한 대부분의 기능과 관련된 API를 제공하고 있다. 따라서, 그림 3에서와 같이 서버는 이러한 API를 이용하여 데이터가 저장된 HDFS에 접근하여 원하는 동작을 수행한다.

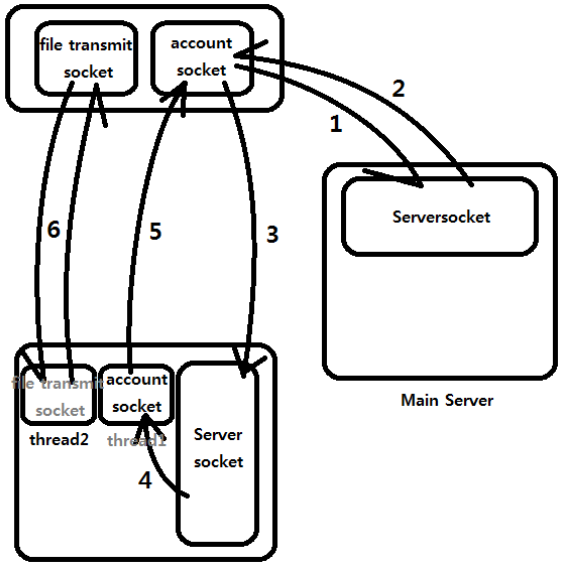

그림 2 C-drive의 데이터 흐름

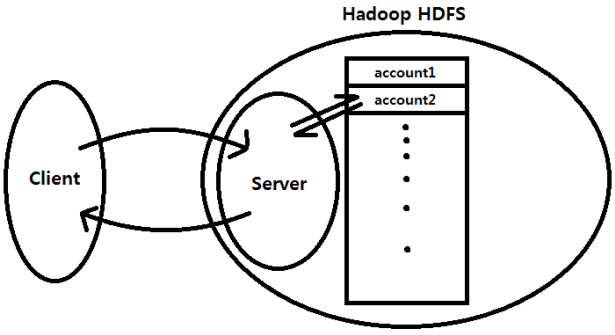

그림 3 HDFS에 대한 C-drive 서버의 접근

C-drive와 같은 서비스를 제공하는 시스템에 있어서 두 개의 서로 다른 어플리케이션이 데이터를 주고받으며 그것을 사용자가 의도대로 사용하기 위해선, 데이터를 관리하기 위한 규칙이 필요하다. 그 중 첫째는 데이터를 구조화시켜 보관해야하며, 둘째로 각각의 사용자에 대한 각각 정보를 가지고 있어야한다는 것인데, 이는 일종의 메타데이터, 즉 데이터에 관한 구조화된 데이터로서 다른 데이터를 설명해주는 데이터 혹은 그에 준하는 형식으로 구현이 가능하다. 즉, 각 개인 사용자 소유의 데이터들의 속성정보, 사용내역과 관련된 기록들을 또 다른 데이터로 구조화

하여 기록하는 것이다. 그리고 그것을 읽어냄으로써 사용자는 원하는 정보를 얻어갈 수 있다.

메타데이터 관리는 HDFS가 제공하는 API의 기능을 이용하는 것과 약간의 아이디어를 더하는 것으로 데이터 관리를 구성하였다. C-drive가 제공해야하는 데이터 정보는 다음과 같다.

1) 계정 ID
2) 계정 비밀번호
3) 계정 공간 할당
4) 계정 공간 내 데이터리스트
5) 로그 기록을 위한 로그파일.

이는 아래와 같은 방법으로 메타데이터를 표현하여 관리하였다.

1) 계정 ID : /id hdfs 공간 내 id명 디렉토리.
2) 계정 비밀번호 : /id/pass.dat id명 디렉토리 내부의 비밀번호 정보파일.
3) 계정 공간 할당 : /id/home/ id명 디렉토리 내부 홈디렉토리.
4) 계정 공간 내 데이터리스트 : /id/home/ 디렉토리의 파일리스트 조회 API사용.
5) 로그 기록을 위한 로그파일 : /id/log/log.txt 계정디렉토리 내 로그폴더/파일

이로써 계정정보와 계정 내 데이터의 로그는 해결되었고, 그 외의 간단한, 이를테면 동기화 정도의 로드를 요구하는 로그를 기록하길 원한다면 log폴더 내에 로그기록을 메타데이터로 남기면 되는 것이다.

3. UI 구성 및 실험

C-drive는 JAVA로 GUI를 구현하였으며, 클라이언트와 서버 부분의 GUI화면으로 구성되어 있다. 클라이언트의 간단한 첫 화면 구성으로, ID와 비밀번호를 넣을 수 있는 텍스트박스와 로그인 요청 버튼, 계정생성버튼, 종료버튼이 만들어져 있다. 계정생성버튼을 누를 경우 동일한 화면에서 버튼의 기능과 텍스트가 변화한다. 클라이언트의 계정을 이용해서 온라인 저장소로 접속한 메인화면 GUI의 예를 그림 4에서 확인할 수 있다.

메인서버의 경우 실행 시, Text화면이 실행된다. 메인서버가 동작중, 클라이언트로부터 접속 요청이 오거나, 그로 인하여 새로운 쓰레드 및 소켓을 생성과 같은 동작이 일어날 경우, 해당 동작의 정보가 Text화면에 표시된다. 클라이언트와 메인서버의 통신 및 데이터 전송 GUI 테스트를 통해서 C-drive가 제대로 동작중인지를 확인할 수 있다.

4. 결론

본 논문에서는 하둡을 이용한 온라인 대용량 저장소 시스템, C-drive를 직접 제작하고 구현해 보았다. C-drive는 결과적으로 서비스가 가능한 최소한의 기능을 만족시키는 정도로 구현되었으며 테스트되었다. 실제로 데이터가 송수신되고 있음을 확인하였고, 계정 간 데이터 구조가 설계했던 대로 동작하고 있음 역시 간단히 확인해 볼 수 있었다. 오픈소스인 하둡을 이용하여 대용량 온라인 저장소를 구현하였다는 데 의의를 찾을 수 있을 것이다.

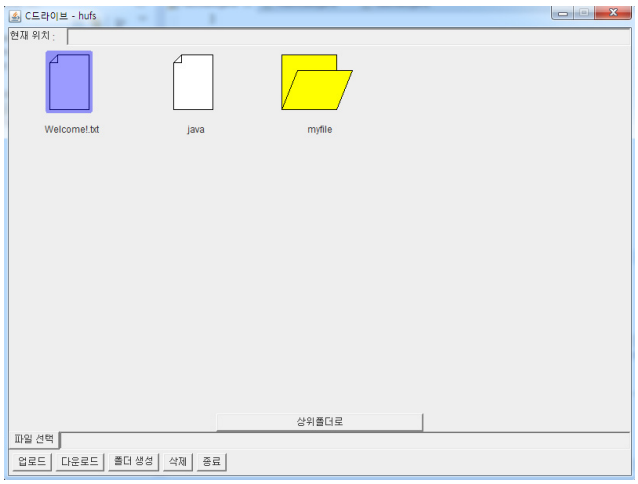

그림 4 클라이언트의 C-drive 접속 화면

참고문헌

[1] Apache hadoop, http://http://hadoop.apache.org
[2] Ashlee Vance. "Hadoop, a Free Software Program, Finds Uses Beyond Search". New York Times. Archived from the original on 11 February 2010.
[3] Apache hadoop, "HDFS User Guide". Hadoop. apache.org.
[4] 정재화 "시작하세요1하둡프로그래밍", pp. 6-9, 위키북스, 2012.
[5] 톰화이트 "Hadoop 완벽가이드", pp. 36-39, 한빛미디어, 2011.