한국컴퓨터정보학회 동계학술대회 논문집 제31권 제1호 (2023. 1)

다중 사용자 동기화 지원 개인용 클라우드 서버 개발

최효현*, 이우현O
*인하공업전문대학 컴퓨터정보과,
O인하공업전문대학 컴퓨터정보과
e-mail: hchoi@inhatc.ac.kr*, lwh8762@gmail.comO

Development of a Personal Cloud Server supporting Multi-User Synchronization

Hyo Hyun Choi*, Woo Hyun LeeO
*Dept. of Computer Science, Inha Technical College,
ODept. of Computer Science, Inha Technical College

● 요 약 ●

본 논문에서는 개인용 클라우드 서버를 구현하였다. 서로 다른 컴퓨터간의 파일 공유를 쉽게 해 주고 접속한 클라이언트간의 상태를 동기화하여 사용자 경험과 접근성을 높였다. 서버는 NodeJS환경으로 실행되며 solidjs로 만들어진 웹 페이지를 vite를 통하여 빌드하고 클라이언트에 전송하여 렌더링하고 클라이언트와 서버는 웹 소켓으로 연결되어 서버의 변경사항을 실시간으로 반영한다.

키워드: 클라우드(Cloud), 동기화(Synchronization), 원격 드라이브(Remote Drive), SolidJS(SolidJS)

I. Introduction

사용자가 여러 PC를 사용하여 개인 PC에서 공용 PC로 파일을 전송하거나 다른 사람에게 파일을 전송해야 할 때 간단한 파일을 전송하더라도 로그인이 필요한 이메일, 클라우드 서비스 등을 이용해야 하는 경우가 많다. 그러나 이러한 서비스들은 기업에서 제공하기에 용량 제한이 있으며 대용량 파일을 공유하기 위해서는 유료 플랜을 구매해야 한다.

이러한 제한점을 해결하기 위한 개인용 클라우드 서버를 연구 개발하였다. 용량 제한이 없으며 원하는 파일에 대한 URL만 가지고 있으면 언제 어디서든 해당 파일을 다운로드 받을 수 있으며, 다른 사람과 파일을 공유하는 용도로도 쓰이기 위해 파일을 업로드하거나 폴더 및 파일의 이름 바꾸기 등의 수정이 이루어지면 나머지 사람들도 즉각적으로 알 수 있는 동기화 기능까지 지원하도록 하였다. 또한 사용자 손쉽게 개인 PC에 명령어 몇 줄로만으로 클라우드 서버를 구축할 수 있도록 개발하였다.

II. Development

1. 전체 구조

사용자 간의 현재 저장소의 상태를 실시간으로 공유할 수 있도록 시스템을 설계하였다. 현재 저장소의 상태를 제공해주는 API 서버, 웹 서버, 업로드 요청 서버, 웹 소켓 서버가 논리적으로 분리되어있다. 클라이언트가 접속할 때 http 요청은 URL 구조에 따라 express.js[1]를 통해 각각의 서버로 요청이 분리된다. 웹 서버는 solidjs[2]로 만들어진 페이지를 vite[3]로 빌드하여 페이지를 제공한다. 웹 페이지 렌더링이 완료되면 클라이언트는 서버와 웹 소켓 연결을 socket.io[4] 라이브러리를 이용해 연결을 수행한다. 이 프로젝트에서는 socket.io의 rooms 기능을 이용하여 특정 디렉토리에 접속한 클라이언트들을 관리한다. 동시에 서버에 현재 디렉토리에 대한 정보 요청을 API 서버에 보낸다. API 서버는 RESTful [5]하게 설계되어 있으므로 GET 요청으로 특정 디렉토리의 정보를 가져올 수 있다. 또한 POST, DELETE, PUT 요청을 통해 API 서버로 특정 파일 또는 폴더에 대한 업로드 요청, 삭제, 수정요청을 보낼 수 있다. 서버에 이러한 수정 요청이 보내지고 수정이 이루어지면 클라이언트가 서버에 접속할 때 연결된 웹 소켓을 통하여 모든 클라이언트에게 알림을 보낸다. 알림을 받은 클라이언트는 API 서버로 GET 요청을 다시 보내 현재 상태를 다른 클라이언트들과 동기화 할 수 있다.

2. 동기화 과정

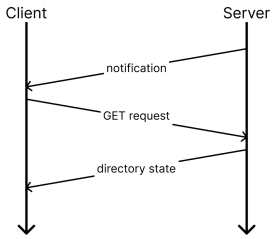

Fig. 1. Synchronization process

업로드가 이루어지는 경우 클라이언트는 서버에 파일을 일정 청크 단위로 전송한다. 청크 단위 전송이 완료될 때마다 서버는 클라이언트에게 알림을 보내고 클라이언트는 다시 서버에 GET 요청을 보내게 된다.

3. 동기화의 최적화

각 청크의 전송이 빠르게 수행되면 브라우저는 많은 GET 요청을 시도하고 브라우저의 자원을 많이 점유하는 문제가 생긴다. 수 밀리초 단위로 동기화 과정을 수행하면 큰 의미가 없는 요청이 발생하고 이는 웹 페이지 성능과 서버에 큰 부하를 준다. 이 문제의 해결을 위해 프로젝트에서 http 요청을 최적화할 수 있는 알고리즘을 설계했다. 이 알고리즘은 무시할 수 있는 요청은 수행하지 않고 필요한 요청만 보내는 방식이다. 업로드의 경우 필요한 요청은 가장 최신 상태의 파일 상태를 가져오는 요청이고 그 요청이 끝나기 전까지 서버로부터 받은 알림은 무시할 수 있는 요청이 된다. 업로드 상태 동기화를 위해 보낸 요청이 완료될 때까지 스택에 알림을 저장하고 이전 요청이 끝나면 스택의 가장 마지막에 저장된 알림을 처리하며 이전에 들어온 알림은 무시된다.

이러한 방식으로 불필요한 http 요청은 처리하지 않게 최적화를 진행하였고 전체적인 요청 처리 과정은 그림 3과 같다.

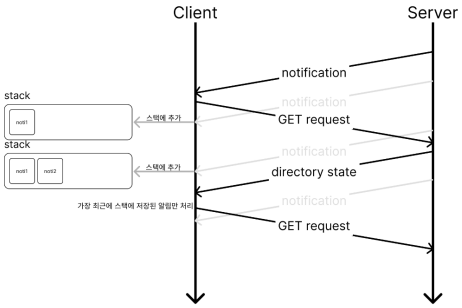

Fig. 2. Optimized request method

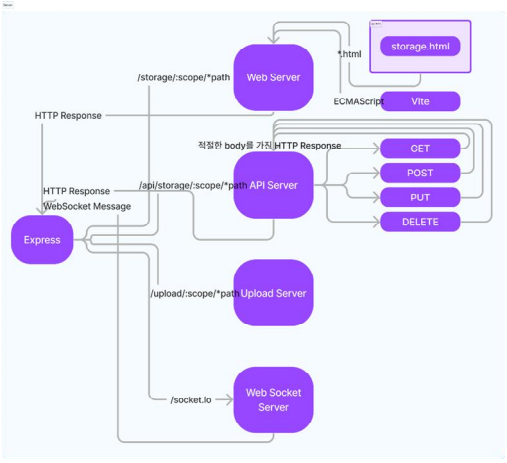

Fig. 3. Request Processing Flow

III. Conclusions

서로 다른 컴퓨터 간의 파일을 기존의 방식보다 더 빠르고 쉽게 공유할 수 있으며 동기화를 지원하는 웹 기반의 개인용 클라우드 서버를 구축하였다. API서버가 웹 서버와 논리적으로 분리되어 있으므로 통신 규약에 맞게 요청을 보내는 방식만 맞추면 별도의 써드파티 프로그램을 만들어 자동으로 파일의 동기화를 진행해주는 확장 기능을 만들 수 있다. 이 프로젝트는 git에 의해 버전관리가 이루어지며 GitHub에 전체 프로젝트를 공개하고 있다.[6]

REFERENCES

[1] Express, https://expressjs.com/
[2] SolidJS, https://www.solidjs.com/docs/latest/api
[3] Vite, https://vitejs.dev/guide/
[4] Socket.IO, https://socket.io/docs/v4/
[5] REST API, https://restfulapi.net/
[6] https://github.com/lewohy/cloud