15장 구글 드라이브 설계

1단계 문제 이해 및 설계 범위 확정

요구사항

- 파일 업로드/다운로드, 파일 동기화, 알림이 가장 중요 기능
- 모바일 앱, 웹 지원
- 파일 암호화
- 파일 크기 10GB 제한
- DAU 천만

개략적 추정치

- 가입 사용자는 오천만, 천만 DAU 가정
- 모든 사용자에게 10GB 무료 저장공간 할당
- 매일 각 사용자가 평균 2개 파일 업로드한다고 가정. 각 파일 평균 크기는 500KB
- 읽기:쓰기 비율 1:1
- 필요한 저장공간 총량 = 5천만 사용자 X 10GB = 500PB
- 업로드 API QPS = 1천만 사용자 X 2회 업로드/24시간/3600초 = 약 240
- 최대 QPS = QPS X 2 = 480

2단계 개략적 설계안 제시 및 동의 구하기

필요한 API 종류

1. **파일 업로드 API**

- 단순 파일 업로드 : 파일 크기 작을 때
- 이어 올리기 : 파일 사이즈 크고 네트워크 문제로 업로드 중단될 가능성 높을 때

- `https://api.example.com/files/upload?uploadType=resumable`
  - 인자
    - uploadType=resumable
    - data : 업로드 할 로컬 파일

2. **파일 다운로드 API**

- `https://api.example.com/files/download`
- 인자
  - path

3. 파일 갱신 히스토리 API

- `https://api.exampel.com/files/list_revisions`
- 인자
  - path
  - limit : 히스토리 길이 최대치

서버 용량 제한 극복

AWS S3 도입

- 로드밸런서 : 네트워크 트래픽 고르게 분산
- 웹 서버 : 로드밸런서 추가하면 손 쉽게 추가 가능, 트래픽 폭증 대응 가능
- 메타데이터 DB : DB를 파일 저장 서버에서 분리하여 SPOF 회피, 다중화 및 샤딩
- 파일 저장소 : S3이용. 가용성 데이터 무손실 보장 위해 두 개 이상 지역에 다중화

동기화 충돌

- 한 사용자가 가진 로컬 사본과 서버의 최신 사본을 합칠지, 둘 중 하나를 대체할지 결정.

개략적 설계안

- 사용자 단말
- 블록 저장소 서버 (block server) : 파일 블록을 클라우드 저장소에 업로드하는 서버. block-level storage 라고도한다. 각 블록에는 고유한 해시값이 할당. 해시값은 메타데이터 DB에 저장.
- 클라우드 저자옷

- 아카이빙 저장소 (cold storage) : 오랫동안 사용되지 않은 비활성 데이터 저장
- 로드밸런서
- API 서버 : 파일 업로드 외에 거의 모든거 담당
- 메타데이터 DB : 사용자, 파일, 블록, 버전 등의 메타데이터 정보 관리
- 메타데이터 캐시
- 알림 서비스
- 오프라인 사용자 백업 큐 (offline backup queue) : 클라이언트가 접속 중 아니라 파일 최신 상태 확인할 수 없을 때 해당 정보를 두어 나중에 동기화될 수 있도록

3단계 상세 설계

블록 저장소 서버

- 델타 동기화 : 파일이 수정되면 전체 파일 대신 수정일 일어난 블록만 동기화 → 네트워크 절약
- 압축 : 블록 단위로 압축. 텍스트 파일의 경우 gzip 이나 bzip2 사용
- 각 블록 압축 후 암호화 후 클라우드 저장소로 보냄

높은 일관성 요구사항

- 캐시에 보관된 사본과 db의 원본(master) 일치해야
- db에 보관된 원본 변경시 캐시의 사본 무효화
- RDBMS는 ACID를 보장하지만 NoSQL은 동기화 로직 안에 프로그램을 넣어야.

업로드 절차

- 파일 메타데이터 추가
    - 추가되었음을 알림서비스에 통지, 알림 서비스는 관련 클라이언트에게 파일 업로드 되고 있음을 알림
- 파일을 클라우드 저장소에 업로드
    - 업로드 끝나면 완료 콜백 호출. 메타데이터 db에 기록된 해당 파일 상태를 uploaded로 변경
    - 알림 서비스에 통지, 알림

다운로드 절차

- 클라이언트 A 접속중이고 다른 클라이언트가 파일 변경하면 알림 서비스가 A에게 새 버전 끌어가야 한다고 알림
- 클라이언트 A가 네트워크에 연결된 상태가 아닐 경우에는 데이터가 캐시에 보관됨.

알림 서비스

- 롱 폴링 : 드롭박스. 긴 시간동안 연결을 유지. 특정 파일에 대한 변경 감지하면 연결을 끊는 방식 채택.
- 웹 소켓 : 지속적인 통신 채널. 양방향 통신, 실시간성 보장. 하지만 여기저는 이 두가지가 필요하지 않다.

저장소 공간 절약

- 중복 제거
- 지능적 백업 전략 도입
  - 한도 설정 : 보관해야 하는 파일 버전 개수에 상한 둔다. 상한 도달시 제일 오래된 버전 버림
  - 중요한 버전만 보관 : 아주 자주 바뀌는 파일의 경우 업데이트 될 때마도 새로 보관하면 낭비.
- 자주 쓰이지 않는 데이터는 아카이빙 저장소로 옮김. S3 glacier 는 이용료 훨씬 저렴.

장애 처리

- 로드밸런서 장애 : secondary 로드밸런서가 이어감. 로드밸런서끼리 heartbeat 로 체크.
- 블록 저장소 서버 장애 : 다른 서버가 이어 받음
- 클라우드 저장소 장애 : S3 여러 지역 다중화
- API 서버 장애 : 무상태 서버이므로 해당 서버로 요청 보내지 않기
- 메타데이터 캐시 장애 : 다중화하여 해결
- 메타데이터 DB 장애
  - 주 DB 서버 : 부 DB 중 하나를 주로 바꾸고, 부를 하나 추가
  - 부 DB 서버 : 다른 부 DB가 읽기 연산 처리하도록 하고 그 동안 다른 서버로 교체
- 알림 서비스 장애 : 롱 폴링 연결을 모두 다시 시작하는 것은 어려움