Squid Storage

Muhamad H. Salman

Gabriele Basigli

Architettura Generale

Data Nodes: cluster di nodi usati come storage passivo, ricevono le nuove versioni dei file dal server e rispondono alle richieste.

Server Node: coordinatore centrale, gestisce le richieste dei client e le replicazione dei file sui datanode.

Client Node: offre una UI per l'utente tramite la quale modificare file e inviare gli aggiornamenti al server.

File Distribution

- Ogni file è replicato sui datanode secondo un fattore di **ridondanza** per avere più copie di back-up a cui accedere.
- Il server si occupa di tenere **sincronizzate** ed aggiornate tutte le copie di ridondanza.
- I datanode su cui replicare il file vengono scelti tramite algoritmo **Round Robin** per bilanciare il carico.
- Di conseguenza ogni datanode ha solo alcuni dei file gestiti dal sistema.
- Il server non ha nessun file in locale, per accedere ad un file richiesto da un client deve comunicare con i data node.

Failure Detection

- Per rilevare i fallimenti, il server invia dei **ping** per controllare lo stato dei datanode.
- Se un datanode va in **crash**, i file vengono replicati dinamicamente sui datanode rimanenti per mantenere la ridondanza.
- La connessione con i client non è monitorata perché i client possono **scollegarsi** liberamente.
- Nel caso in cui i client o i datanode perdano la connessione al server, tentano la **ri-connessione** in loop.

Squid Protocol

- Protocollo **testuale** utilizzato per comunicare modifiche sui file o per gestire la connessione.
- Un **formatter** costruisce i messaggi dati gli input oppure fa il **parsing** dei messaggi ricevuti.
- Dato che i messaggi sono di **lunghezza variabile**, per ogni messaggio viene prima inviata la sua lunghezza e poi il messaggio concreto.
- Il **contenuto dei file** viene invece trasmesso in richieste successive senza un formato, inviando prima la lunghezza del file.

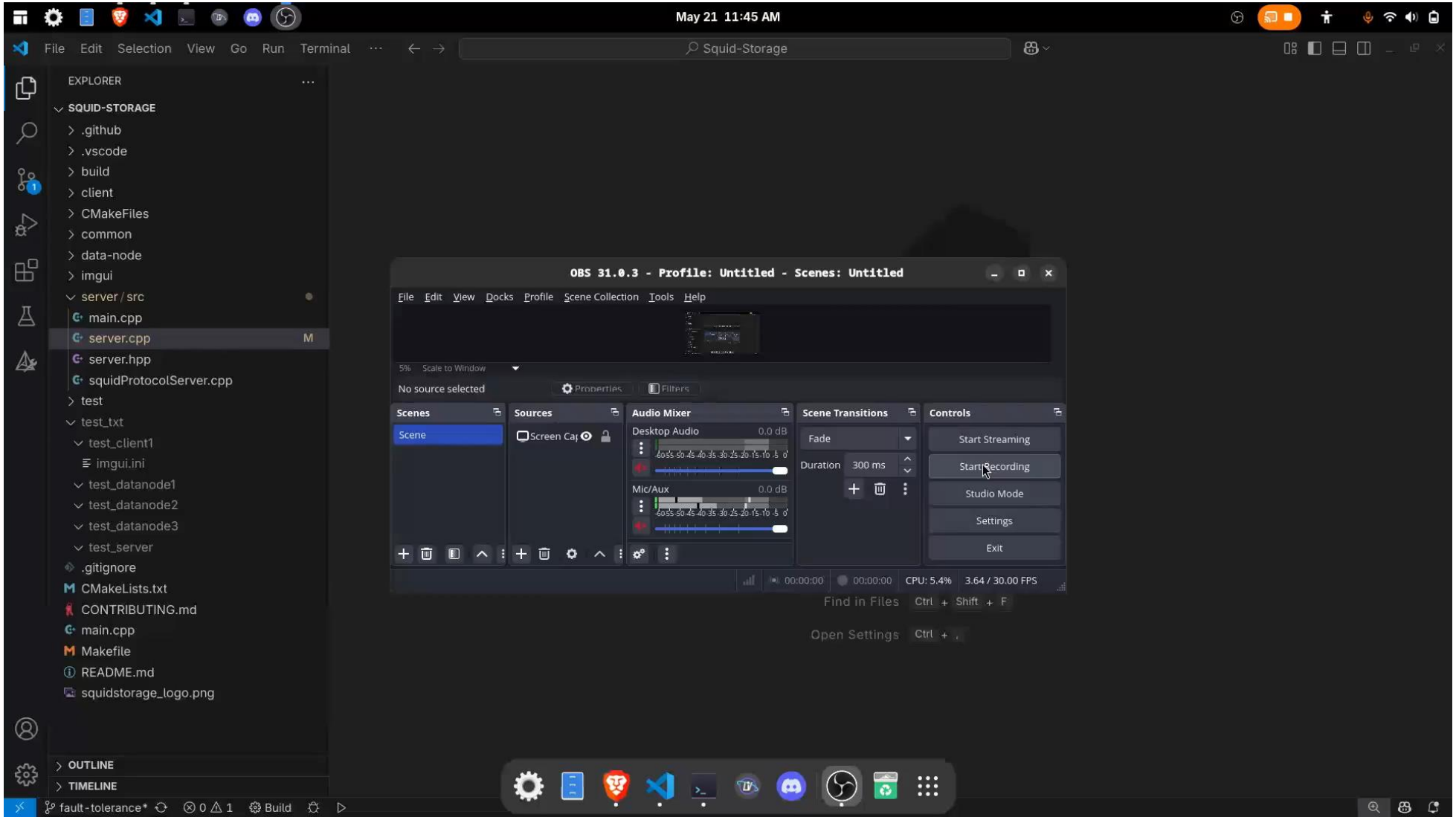
Struttura di un messaggio

- Viene utilizzato il formato chiave-valore: Keyword<Arg1:Val1, Arg2:Val2>
- Esempio: CreateFile<filePath:/folder/test.txt, fileVersion:1>

Table 1: Protocol Messages

Keyword	**Arguments**	**Response**
CreateFile	filePath, fileVersion	ACK
TransferFile	filePath	ACK
ReadFile	filePath	ACK
UpdateFile	filePath, fileVersion	ACK
DeleteFile	filePath	ACK
AcquireLock	filePath	isLocked
ReleaseLock	filePath,	ACK
Heartbeat	-	ACK
SyncStatus	-	fileVersions
Identify	-	nodeType, processName
Close	-	ACK

Demo Replication & Resilience



Consistenza e Disponibilità

Lock distribuiti: solo un client alla volta può modificare un file, acquisendo un lock sul server per quel file.

Quando un client salva le modifiche, queste vengono propagate asincronicamente agli altri client.

Le modifiche vengono anche salvate in modo persistente sulle repliche del file presenti nel datanode.

I lock hanno un timeout in modo che un crash di un client con un lock non blocchi il sistema.

Versionamento dei file

- Per tenere traccia di aggiunta, rimozione e aggiornamento di un file
- Versionamento, una sorta di logical clock
- Utilizzato per confrontare due copie dello stesso file

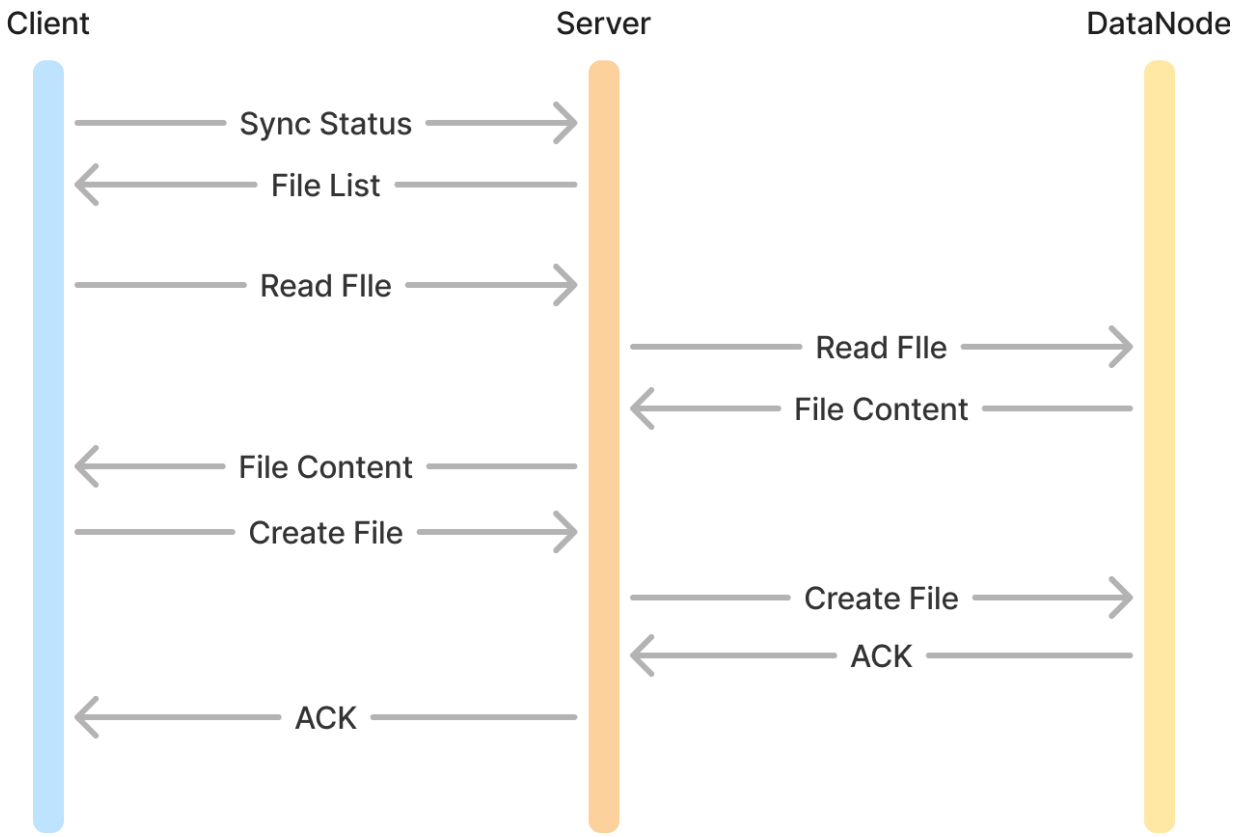
Esempio:

/folder/test.txt 5

/folder/test1.txt 2

Sincronizzazione dei file

- Operazione eseguita quando un client si collega al server
- Utilizza il protocollo



Comportamento GUI

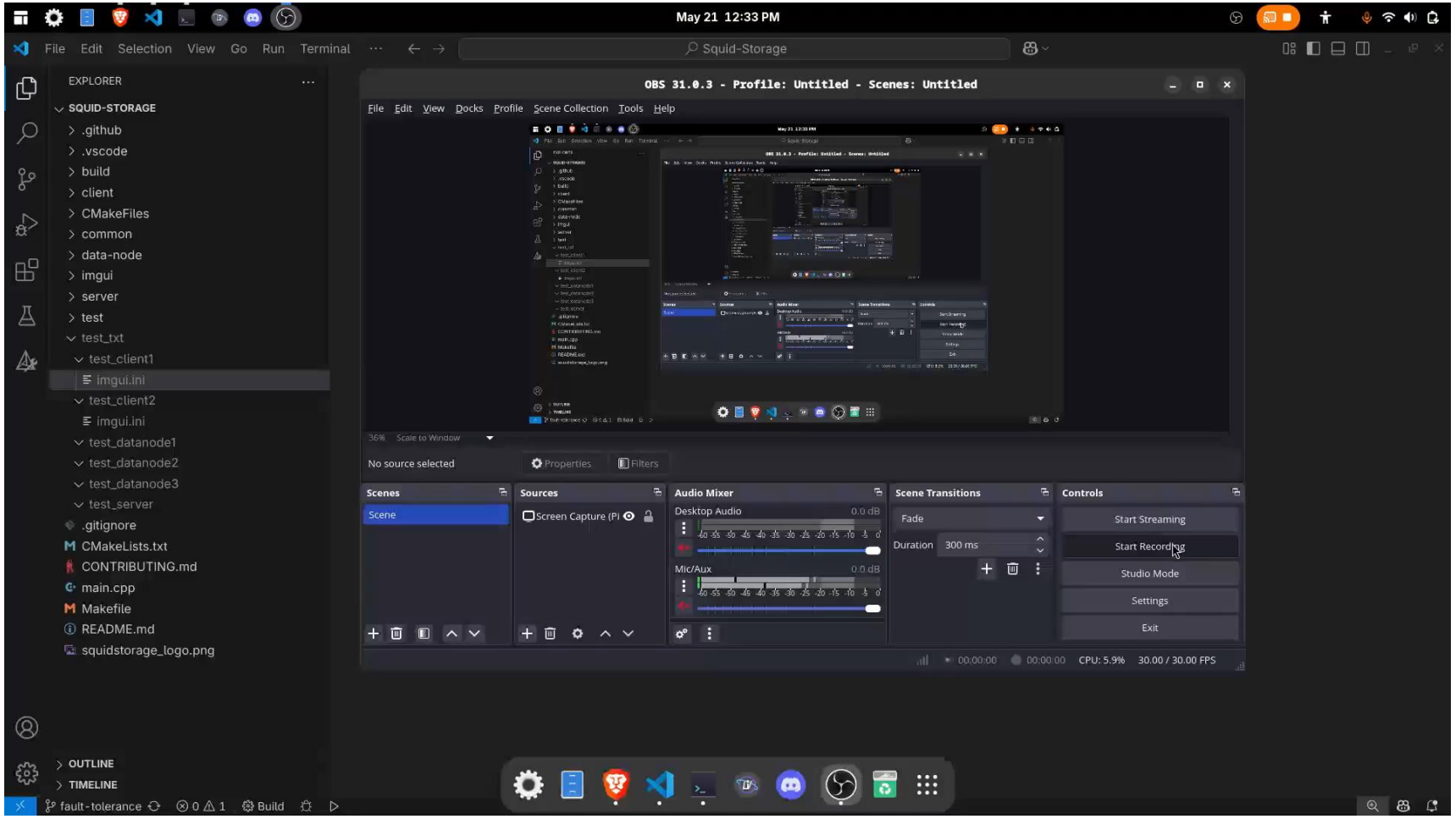
IN CASO DI DISCONNESSIONE, I CLIENT POSSONO ACCEDERE AI FILE SOLO IN LETTURA.

UN CLIENT PER MODIFICARE UN FILE NON DEVE SOLO AVERE IL LOCK MA ANCHE POSSEDERE L'ULTIMA VERSIONE DEL FILE.

IL SERVER COSTRINGE I CLIENT AD AGGIORNARSI TRAMITE UN BOTTONE REFRESH NELLA GUI PRIMA DI ACQUISIRE IL LOCK.

MENTRE IL CLIENT SI SINCRONIZZA CON IL SERVER, LA GUI MOSTRA UNA SCHERMATA DI CARICAMENTO E NON PERMETTE EVENTUALI MODIFICHE.

Demo Client concorrenti



Grazie per l'attenzione