Progetto Filesystem Distribuito

Gabriele Savoia

Abstract

Il progetto consiste nella creazione di un filesystem distribuito implementato tramite la tecnologia ad oggetti distribuiti RMI in Java. I componenti di base sono: un insieme di ServerReplica ciascuno contenente una copia del filesystem; un ServerMaster che, oltre a gestire l'accesso alle risorse per evitare problemi di concorrenza, comunica con il client e gli indica quale ServerReplica contattare per fare una certa operazione; il client che permette di effettuare operazioni a livello di file e directory sul filesystem distribuito.

1 Specifiche Requisiti Software (SRS)

1.1 Obiettivo

L'obiettivo del progetto consiste nell'implementare un filesystem distribuito con replicazione dei dati su più ServerReplica (è possibile utilizzarne un numero arbitrario) rendendo trasparente all'utente finale tutta la parte delle comunicazioni remote.

Il client infatti è costituito da un'interfaccia a linea di comando con lo scopo di accedere (tramite un ristretto insieme di comandi) a file/directory remote con la stessa facilità con cui si accede a file locali nascondendo quindi la complessità del filesystem distribuito.

1.2 Descrizione generale

1.2.1 Funzionalità del sistema

Il sistema è composto da più `client`, un `ServerMaster` e diversi `ServerReplica`. Il principio di funzionamento è il seguente: un generico client che vuole fare una qualsiasi operazione contatta il ServerMaster; questo ritornarà al client il riferimento al ServerReplica da contattare (scelto sulla base di un criterio specifico); il client contatta direttamente il ServerReplica per effettuare l'operazione e nel caso serva, questa sarà propagata a tutti gli altri ServerReplica del sistema; una volta terminata l'operazione (e la propagazione se necessaria), il client informa il ServerMaster che l'operazione è terminata. Il ServerMaster si occupa di gestire l'accesso concorrente ai file / directory implementando meccanismi di lock.

Le funzionalità principali del filesystem distribuito sono le seguenti:

- `creazione file/directory`: l'utente può creare un nuovo file o directory specificando la locazione remota in cui si vuole creare;
- `rimozione file/directory`: è possibile eliminare un file o directory che esiste nel filesystem distribuito;
- `rinomina file`: l'utente può rinominare un file presente nel filesystem distribuito;
- `spostamento file`: è possibile muovere un file in una locazione target remota diversa da quella di partenza;
- `scrittura file`: l'utente può scrivere una sequenza di byte in un certo file remoto;
- `lettura file`: è possibile leggere il contenuto di un file remoto sottoforma di stringa;
- `listaggio file di una directory`: l'utente può visualizzare la lista dei file e directory presenti all'interno di una certa directory remota;

1.2.2 Vincoli e assunzioni

Per l'implementazione si ritiene opportuno definire i seguenti vincoli e assunzioni:

- il client non salva i dati remoti in file locali (es. la lettura di un file remoto consiste nel mantenere il contenuto in memoria principale del client);
- le dimensioni dei file sono tali da non eccedere la memoria principale del server e dei client. Se quindi un client vuole scrivere o leggere un file, si suppone che il contenuto non ecceda le capacità di memoria del server e del client;
- non ci sono permessi particolari sui file (i client possono accedere (in lettura e scrittura) a tutte le risorse del filesystem distribuito);
- non è possibile aggiungere ServerReplica a runtime (in quanto servirebbero tecniche di sincronizzazione dei file). Il numero di ServerReplica è infatti definito solo in fase di start dell'intero sistema;
- se un ServerReplica fallisce il sistema continua correttamente a funzionare ma non è più possibile riattivarlo (in quanto servirebbero tecniche di sincronizzazione dei file);

1.3 Requisiti specifici

1.3.1 Requisiti interfaccia utente

Per quanto riguarda il client, è richiesta un'interfaccia a linea di comando in grado di rilevare ed elaborare l'input dell'utente. La lettura degli input resta continuamente attiva fino a che l'utente non digita la keyword "exit" oppure preme "ctrl+c". Di seguito sono riportati i possibili comandi riconosciuti dal sistema:

- `create [path]`: crea un nuovo file;
- `write [path] [content]` : scrive il contenuto "content" nel file localizzato in "path";
- `read [path]` : lettura del file localizzato in "path";
- `rm [path]` : rimozione file (o directory) localizzato in "path";
- `move [sourcePath] [targetPath]` : sposto file da "sourcePath" a "targetPath";
- `rename [path] [newName]` : rinomina file localizzato in "path";
- `mkdir [path]` : creazione directory localizzata in "path";
- `ls [path]` : visualizzazione dei contenuti della directory localizzata in "path";
- `help` : visualizzazione di tutti i possibili comandi.

Per quanto riguarda il ServerMaster, durante la sua creazione è necessario specificare il numero di ServerReplica che si intende utilizzare.

Per il ServerReplica invece, è opportuno specificare durante la sua creazione il relativo ID univoco, ovvero un numero intero incrementale da 0 a numeroServerReplica-1.

1.3.2 Requisiti funzionali

Creazione file

Introduzione Il client crea un nuovo file (vuoto) con un certo nome e in una certa locazione.

Input

- `Path`: path assoluto del file da creare (es. "/folder/file.txt").

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.

2. client contatta il ServerReplica ed effettua la creazione del file (vuoto): se il file non ha una estensione, oppure il path non esiste o il file da creare esiste già, genera errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);

3. il ServerReplica propaga questa operazione a tutti gli altri ServerReplica raggiungibili. Se qualcuno non è raggiungibile, non sono generati errori (così il sistema continua a funzionare anche quando alcuni ServerReplica non funzionano);

4. il client informa il ServerMaster che l'operazione è terminata.

Output Viene restituito un messaggio che fa capire al client se l'operazione è andata a buon fine oppure se ci sono stati problemi.

Scrittura file

Introduzione Il client scrive su un file un certo contenuto di testo. Se il file non esiste ne viene creato uno nuovo con il contenuto specificato.

Input

- `Path`: path assoluto del file da scrivere (es. "/folder/file.txt");
- `Content`: Testo da scrivere nel file.

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.

2. client contatta il ServerReplica ed effettua la scrittura del file con il contenuto: se il path non è valido (es. sottodirectory non esistenti) genera errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);

3. il ServerReplica propaga questa operazione a tutti gli altri ServerReplica raggiungibili. Se qualcuno non è raggiungibile, non sono generati errori (così il sistema continua a funzionare anche quando alcuni ServerReplica non funzionano);

4. il client informa il ServerMaster che l'operazione è terminata.

Output Viene restituito un messaggio che fa capire al client se l'operazione è andata a buon fine oppure se ci sono stati problemi.

Lettura file

Introduzione Il client legge il contenuto (sottoforma di testo) di un determinato file.

Input

- `Path`: path assoluto del file da leggere (es. "/folder/file.txt").

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.
2. client contatta il ServerReplica ed effettua la lettura del contenuto del file: se il path non esiste genera errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);
3. il client informa il ServerMaster che l'operazione è terminata.

Output Viene visualizzato nella console il contenuto del file oppure un messaggio se ci sono stati problemi.

Rimozione file / directory

Introduzione Il client elimina un file oppure una directory (solo nel caso questa sia vuota).

Input

- `Path`: path assoluto del file / directory da eliminare (es. "/folder/file.txt" o "/folder/").

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.
2. client contatta il ServerReplica ed effettua l'eliminazione del file/directory: se il path non esiste oppure se la directory non è vuota genera errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);
3. il ServerReplica propaga questa operazione a tutti gli altri ServerReplica raggiungibili. Se qualcuno non è raggiungibile, non sono generati errori (così il sistema continua a funzionare anche quando alcuni ServerReplica non funzionano);
4. il client informa il ServerMaster che l'operazione è terminata.

Output Viene restituito un messaggio che fa capire al client se l'operazione è andata a buon fine oppure se ci sono stati problemi.

Spostamento file

Introduzione Il client sposta un file in una posizione target (che non deve corrispondere ad un file già esistente).

Input

- `SourcePath`: path assoluto del file da spostare (es. "/folder/file.txt");
- `TargetPath`: path assoluto della posizione target in cui voglio spostare il file ("/otherFolder/file.txt").

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.

2. client contatta il ServerReplica ed effettua lo spostamento del file: se anche solo uno tra il SourcePath e il TargetPath non è valido, oppure il TargetPath corrisponde ad un file esistente o il SourcePath corrisponde ad una directory e non ad un file, viene generato un errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);

3. il ServerReplica propaga questa operazione a tutti gli altri ServerReplica raggiungibili. Se qualcuno non è raggiungibile, non sono generati errori (così il sistema continua a funzionare anche quando alcuni ServerReplica non funzionano);

4. il client informa il ServerMaster che l'operazione è terminata.

Output Viene restituito un messaggio che fa capire al client se l'operazione è andata a buon fine oppure se ci sono stati problemi.

Rinomina file

Introduzione Il client rinomina un file con un nome arbitrario (che non esista già nella stessa directory).

Input

- `Path`: path assoluto del file da rinominare (es. "/folder/file.txt");
- `NewName`: nuovo nome da assegare al file specificato dal path immesso (deve contenere l'estensione).

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.

2. client contatta il ServerReplica ed effettua la rinomina del file: se esiste già un file con lo stesso nome oppure il path corrisponde ad una directory e non ad un file o il nuovo nome non contiene l'estensione viene generato un errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);

3. il ServerReplica propaga questa operazione a tutti gli altri ServerReplica raggiungibili. Se qualcuno non è raggiungibile, non sono generati errori (così il sistema continua a funzionare anche quando alcuni ServerReplica non funzionano);

4. il client informa il ServerMaster che l'operazione è terminata.

Output Viene restituito un messaggio che fa capire al client se l'operazione è andata a buon fine oppure se ci sono stati problemi.

Creazione directory

Introduzione Il client crea una nuova directory con un certo nome (deve essere senza estensione e non deve già esistere un'altra directory con lo stesso nome).

Input

- `Path`: path assoluto della directory da creare (es. "/folder/");

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.
2. client contatta il ServerReplica ed effettua la creazione della directory: se esiste già un file o directory con lo stesso nome oppure nel nome è presente anche l'estensione viene generato un errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);
3. il ServerReplica propaga questa operazione a tutti gli altri ServerReplica raggiungibili. Se qualcuno non è raggiungibile, non sono generati errori (così il sistema continua a funzionare anche quando alcuni ServerReplica non funzionano);
4. il client informa il ServerMaster che l'operazione è terminata.

Output Viene restituito un messaggio che fa capire al client se l'operazione è andata a buon fine oppure se ci sono stati problemi.

Listaggio contenuto directory

Introduzione Il client visualizza il contenuto (file e directory) di una certa directory.

Input

- `Path`: path assoluto della directory di cui vedere il contenuto (es. "/folder/");

Elaborazione

1. client contatta il ServerMaster dicendogli che vuole iniziare una nuova operazione e questo risponde con il riferimento al ServerReplica da contattare. Se ServerMaster rileva problemi di concorrenza, è riportato un errore nella console. Se il ServerMaster non è raggiungibile, è riportato un errore nella console e poi il client è arrestato. Se il ServerMaster vede che nessun ServerReplica è raggiungibile, genera un errore e il client viene arrestato.
2. client contatta il ServerReplica e ritorna il contenuto in termini di file e directory della directory specificata: se il path non esiste viene generato un errore. Se il ServerReplica da contattare non è raggiungibile, è riportato un errore nella console (che dice all'utente di riprovare);
3. il client informa il ServerMaster che l'operazione è terminata.

Output Viene visualizzato nella console il contenuto della directory oppure un messaggio se ci sono stati problemi.

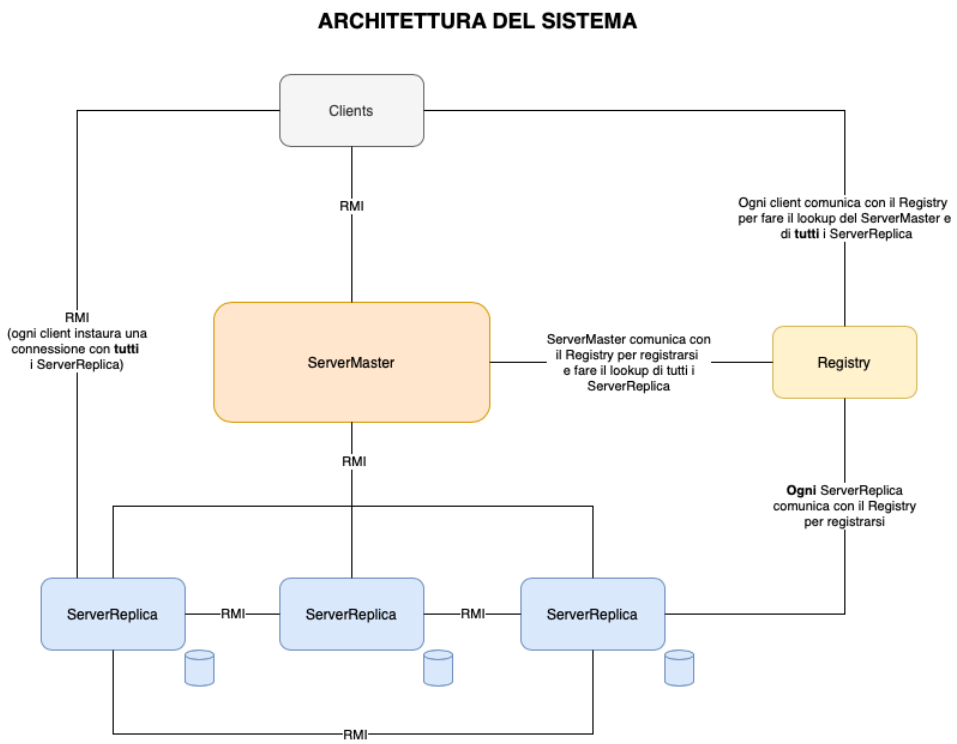

Figure 1: Architettura del sistema.

2 Architettura

2.1 Componenti del sistema

L'architettura del sistema è fondamentalmente client-server e sfrutta i vantaggi forniti dalla tecnologia Java RMI. Come riportato in figura 1, i componenti principali sono i seguenti:

- `Client`: nel sistema possono esistere contemporaneamente un certo numero di client che interagiscono con il filesystem distribuito. Un generico client, dopo aver contattato il Registry RMI ed aver eseguito i dovuti lookup, è in grado di invocare metodi remoti sia del ServerMaster che di tutti i ServerReplica grazie appunto alla tecnologia RMI. Il concetto di base è che il client contatta direttamente un singolo ServerReplica (su cui eseguirà una certa operazione es. lettura / scrittura), mentre contatta il ServerMaster solo per comunicare lo stato (inizio e fine) dell'operazione da svolgere e per ricevere l'ID del ServerReplica da contattare;

- `ServerMaster`: è presente un singolo ServerMaster nel sistema. Questo espone una serie di metodi che possono essere invocati in maniera remota (grazie alla tecnologia RMI) dai client dopo che la registrazione nel Registry è avvenuta. Il ServerMaster si occupa principalemnte di accettare le richieste (da parte dei client) di inizio / fine delle operazioni e di gestire la concorrenza sui file del sistema distribuito (tramite il mantenimento di una apposita hashmap). Dopo aver fatto gli opportuni lookup nel Registry, il ServerMaster è in grado di comunicare direttamente con tutti i ServerReplica e di conseguenza riesce a capire quali tra questi sono raggiungibili e quali no. Quando il client comunica l'inizio di un'operazione, il ServerMaster ha il compito di ritornare l'id del ServerReplica più opportuno in funzione alla richiesta (ad esempio è possibile ritornare l'id del ServerReplica raggiungibile più 'vicino' (in senso geografico) al client). Nel contesto di questo progetto però è ritenuto sufficiente ritornare in maniera random l'id di un singolo ServerReplica raggiungibile;

- `ServerReplica`: nel sistema sono presenti un numero arbitrario di ServerReplica (3 nello schema di esempio riportato). Un ServerReplica espone un insieme di metodi remoti (accessibili tramite RMI dopo l'opportuna registrazione nel Registry) che il client, il ServerMaster e gli altri ServerReplica possono invocare. Il concetto di base è che ogni ServerReplica contiene una copia

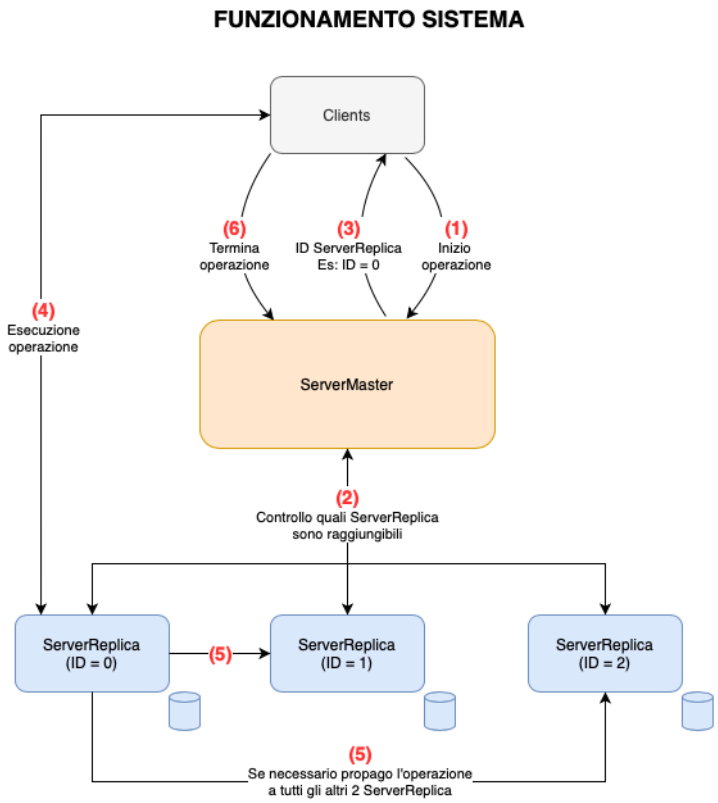

Figure 2: Funzionamento del sistema.

dell'intero filesystem e che quindi è necessario garantire la consistenza tra i ServerReplica. Durante un'operazione di modifica chiesta dal client infatti, il ServerReplica dovrà propagare l'operazione agli altri ServerReplica mediante tecnologia RMI che richiede quindi una fase iniziale di lookup al Registry;

- `Registry`: si tratta del servizio di naming del Java RMI che permette di registrare oggetti remoti (come ServerMaster e ServerReplica) e di effettuarne il lookup permettendo così l'invocazione remota dei relativi metodi.

2.2 Procedura di avvio del sistema

Il corretto avvio del sistema avviene rispettando l'ordine delle seguenti fasi:

1. Attivazione del Registry RMI;
2. Esecuzione di un certo numero di ServerReplica in cui per ciascuno si assegna un ID univoco da terminale. Questo ID deve partire da 0 e deve essere incrementale (es. se scelgo di avere 3 repliche, gli ID saranno 0, 1, 2 rispettivamente);
3. Attivazione del ServerMaster in cui si specifica da terminale il numero totale di ServerReplica;
4. Esecuzione un certo numero di client che interagiranno, a seconda delle esigenze, con il filesystem distribuito.

2.3 Dettaglio funzionamento

Come riportato in figura 2, il funzionamento del sistema (una volta attivato correttamente) può essere riassunto nei seguenti punti:

1. Il client invoca il metodo remoto di inizio operazione del ServerMaster (specificando il path della risorsa (file/directory) coinvolta e la tipologia: se di lettura o di scrittura);

HASHMAP CONCORRENZA

Key	Value
...	....
"/myPath/myFile.txt"	writeLock=True - numReaders=0
...	...

Figure 3: Hashmap concorrenza.

2. Il ServerMaster, dopo aver gestito la hashmap per la concorrenza, controlla che ci sia almeno 1 ServerReplica raggiungibile;
3. Il ServerMaster ritorna al client un id random (id = 0 nell'esempio in figura) tra quelli dei ServerReplica raggiungibili;
4. Il client contatta il ServerReplica (con id = 0 per questo esempio) ed esegue sullo stesso l'operazione;
5. Quando l'operazione è terminata sul ServerReplica con id = 0, se l'operazione è di tipo scrittura (es. creazione, scrittura, rinomina, spostamento, ...) allora questa deve essere propagata a tutti gli altri ServerReplica raggiungibili (id = 1 e id = 2 nell'esempio). Se invece l'operazione è di lettura (es. lettura contenuto file oppure listaggio contenuto directory) allora non serve alcun tipo di propagazione e il contenuto è ritornato al client;
6. Quando la propagazione (se necessaria) è terminata, allora il client comunica al ServerMaster che l'operazione è terminata e viene mostrato nel terminale del client il relativo risultato (successo o eventuali errori).

2.4 Gestione concorrenza

Per la gestione della concorrenza è stata implementata una hashmap *key-value* mantenuta ed aggiornata in memoria principale del ServerMaster. In particolare (vedi figura 3) una generica *key* è una stringa che rappresenta il path univoco di una risorsa (file o directory) mentre il relativo *value* è una struttura composta da 2 valori: un booleano che indica se è settato oppure no il write lock e un intero che tiene traccia del numero di reader.

Il concetto è che quando il client comunica al ServerMaster l'inizio di una nuova operazione che coinvolge una certa risorsa (vedi punto 2 della figura 2) allora l'hashmap viene aggiornata di conseguenza: se l'operazione è di scrittura (creazione, scrittura, rinomina, ...) allora il write lock verrà settato a true, se invece l'operazione è di lettura (lettura file o listaggio contenuto directory) allora avviene un incremento del numero di reader per la risorsa. In maniera speculare, quando viene comunicata la terminazione dell'operazione (vedi punto 6 della figura 2, ovvero quando anche le eventuali propagazioni sono terminate), la hashmap viene aggiornata settando a false il write lock oppure decrementando il numero di reader se si tratta di un'operazione di lettura.

Avendo quindi a disposizione queste informazioni, il ServerMaster implementa controlli per l'accesso alle risorse ed in particolare:

- Se ad una risorsa è settato il write lock a true allora non è possibile da parte di un client iniziare una nuova operazione (sia in scrittura che in lettura);
- Se ad una risorsa è settato il write lock a false allora è possibile far partire (anche contemporaneamente da parte di più client) delle nuove operazioni di lettura sulla risorsa;
- Se ad una risorsa è associato 1 o più reader allora non è possibile da parte di un client iniziare una nuova operazione in scrittura (è possibile iniziare solo operazioni in lettura);

2.5 Diagramma delle classi

In figura 4 è riportato il diagramma delle classi per l'implementazione del filesystem distribuito.

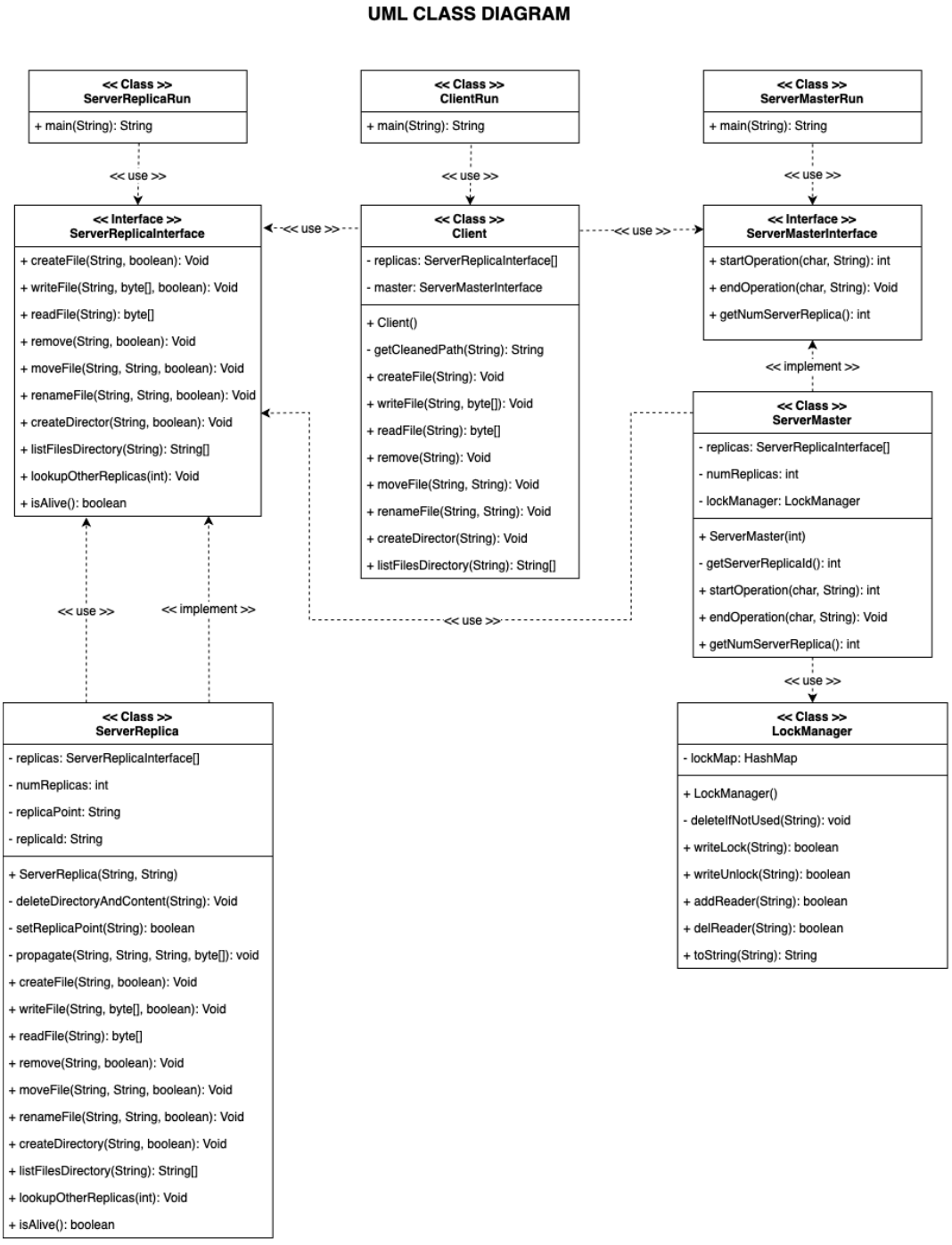

Figure 4: UML class diagram.