Relazione sull'implementazione memBox : un memory storage concorrente

Dan Dorin Izvoreanu

Federica Currao

29 giugno 2016

Il processo principale

Thread principale (membox.c)

Il main di membox.c rappresenta il nucleo della parte server.

All'avvio il server esegue le seguenti operazioni:

- Setta le variabili globali.

- Acquisisce la configurazione che gli viene assegnata di default ed in seguito sovrascritta se passato come parametro all'avvio un PATH con un file.

- Maschera i segnali.

- Avvia il thread gestore dei segnali.

- Inizializza l'hash Table e threadpool.

- Si mette in attesa di connessioni.

Il pool di thread (threadpool.c threadpool.h)

Una delle strutture principali sulla quale si basa il progetto membox è il threadpool, esso svolge concretamente i vari task. Viene inizializzata tramite il

numero di thread che si vogliono far lavorare in contemporanea, tutti i thread vengono creati simultaneamente e iniziano con l'esecuzione della funzione thDo. Quest'ultima prende i vari task dalla coda e li esegue. Il passaggio dei parametri alla thDo avviene tramite una struct formata dalla funzione che si vuole mandare in esecuzione e i suoi argomenti, questo permette un riutilizzo di questa pool per scopi non necessariamente legati a questo progetto.

La queue (queue.c queue.h)

La gestione della queue dei lavori all'interno del pool di thread avviene tramite una particolare libreria che gestisce le code, fornita delle tipiche operazioni necessarie.

Il worker (funzione di membox.c)

L'unica funzione che viene passata al pool di thread è il worker. Esso ha come argomento un file descriptor della connessione del client che si vuole servire. Dopo aver inizializzato alcuni flag, il worker legge l'operazione che deve svolgere. Effettuati i controlli necessari, richiama la funzione richiesta dal client.

Funzioni che agiscono sul repository

Sono put, update, get e remove. Esse richiamano al loro interno, dopo i neccessari controlli, le funzioni della icl_hash, una libreria che ci è stata fornita.

La comunicazione con il client avviene tramite le funzioni di connections.c che tratteremo in seguito.

Funzioni di lock

Le funzioni di lock e unlock agiscono settando una variabile globale con il file descriptor del client che richiama la lock, permettendo così solamente a tale client di eseguire operazioni sul repository. Se al termine della connessione il client non ha rilasciato la lock, il worker la libera automaticamente.

Gestione dei segnali (gestore_segnali)

Il thread di gestione segnali si occupa di ricevere segnali inviati al processo e gestirli. Tutti i thread, infatti, hanno i segnali relativi al processo mascherati.

I segnali SIGINT, SIGTERM e SIGQUIT

Questi segnali richiedono la terminazione del server. Registrati dal thread gestore_segnali, si esegue un shutdown sul socket della accept per non accettare nuove connessioni, si è preferita quest'ultima alla close perchè ha un comportamento ben definito in un ambiente multithread.

Una volta impedito l'arrivo di nuove condizioni si distrugge il pool dei thread in maniera non gentile, indipendentemente dalla coda dei task viene fatta una join su tutti i thread, liberando dalla memoria tutto ciò che la struct ha allocato in precedenza.

La terminazione continua facendo un shutdown su tutte le connessioni attive, si libera tale lista, si fa un join del thread che gestisce i segnali e si deallocano la tabella hash che fa da repository, la struct usata per configurare il server e per passare i parametri al thread gestore dei segnali.

Il segnale SIGUSR2

Anche questo è un segnale di terminazione, ma a differenza dei precedenti esegue una terminazione gentile. Prima di fare la join di tutti i thread si attende che la coda dei task sia stata svuotata, ciò avviene tramite una spinlock, è stata fatta questa scelta infelice per motivi di tempo.

Il segnale SIGUSR1

Questo segnale comunica al processo di dover stampare le informazioni da monitorare su un file che è stato precedentemente impostato tramite l'attività di configurazione che avviene all'inizio del processo principale.

Lista delle connessioni attive (intList.h)

Le connessioni attive vengono salvate su una lista in modo da poterle arrestare in caso di terminazione del server. Tale lista è utilizzata anche per conoscere il delle connessioni concorrenti, utile per le statistiche.

Informazioni da monitorare (stats.h)

Il monitoraggio delle informazioni richieste avviene tramite la struct statistics che ci è stata fornita. Essa viene utilizzata anche per il controllo dei limiti imposti dalla configurazione del server.

Connecitons (connections.c)

Il principale metodo di comunicazione tra server è client sono le funzioni:

- readReader / writeReader
- writeData / readData

Il client invia le operazioni richieste tramite writeHeader allo stesso modo il server manda i messaggi di risposta. Quando è necessario l'invio di oggetti si usano le writeData e readData. Queste funzioni sono simili tra loro: per far sì che venga scritto/letto tutto il messaggio su usa un ciclo while dal quale non si esce fino all'invio di tutti i dati necessari o in caso di errore.