PROGETTO SISTEMI OPERATIVI

(Tocci - Principe)

Panoramica

Abbiamo realizzato la specifica richiesta utilizzando una matrice di char [n x m] che memorizza, per ogni entry, un char , ed un array di dimensione sempre [n x m] di cui

ogni entry mantiene un codice alfanumerico relativo ad una prenotazione ed un puntatore all'array contenente i posti riservati relativi a quella prenotazione.
Quando un client vuole riservare dei posti, viene generata una chiave relativa a questa prenotazione, i posti vengono occupati e la chiave viene resa nota al client. Questa chiave sarà utile al client nel caso in cui voglia disdire una prenotazione, in quanto soltanto tramite questa potrà liberare i posti precedentemente occupati.
Per ogni client, viene staccato un thread che si occuperà delle operazioni richieste dal client stesso (e quindi aprirà un socket per questa connessione).

Accesso concorrenziale

Per gestire la concorrenza riguardo le prenotazioni di posti, abbiamo utilizzato, oltre ad un array
(n x m) di semafori binari, altri due semafori per evitare scenari di starvation. E' presente inoltre un semaforo per l'accesso all'array delle prenotazioni.
Quando un client vuole prenotare un posto, vengono richiesti immediatamente i token relativi ai semafori di ogni singolo posto desiderato. Una volta ricevuti tutti i token, si controlla se i posti siano liberi ed, in caso positivo, vengono riservati. A questo punto, si rilasciano i token precedentemente acquisiti e si prova ad accedere all'array delle prenotazioni, inserendo nella prima posizione libera (free-pointer) la chiave generata contestualmente alla prenotazione.
Una volta terminata questa operazione, la prenotazione può considerarsi effettuata con successo e viene consegnata la conferma al client.
In caso di rinuncia di prenotazione, si prova accedere in primo luogo all'array delle prenotazioni, rimuovendo la chiave relativa alla prenotazione da disdire, ed in secondo luogo ai singoli posti della matrice, settandoli a liberi (0).
Abbiamo pensato tuttavia che con queste procedure si potesse degenerare in situazioni di starvation nel caso in cui un client A voglia prenotare molti posti ed altri client vogliano invece prenotare alcuni dei posti che A vorrebbe. Il client A, dato che deve aspettare che tutti i semafori nella matrice siano liberi contemporaneamente, ha la possibilità di aspettare per un lasso di tempo molto ampio, anche solo per venire a sapere che i posti non sono disponibili.Un client viene considerato in starvation se in attesa per più di T_OUT. Se si verifica questa situazione viene attivata una procedura speciale:l l'accesso alla matrice viene negato a tutti i nuovi client. Per i client coinvolti nella situazione di stallo l'accesso alla matrice viene garantito in maniera sequenziale. Ciò viene effettuato tramite l'utilizzo di ulteriori 2 semafori: uno utilizzato per bloccare l'accesso alla matrice da parte di tutti i thread (da noi chiamato "master semaphore") ed un altro per serializzare gli accessi alla matrice da parte di altri thread attualmente in starvation (binario). Il client A quindi, aggiungerà un token al master semaphore in modo da bloccare gli altri thread, aspetterà di poter prelevare un token sul secondo semaforo e solo dopo averlo preso potrà agire sulla matrice indisturbato. Il processo inverso, ovviamente, viene effettuato per rilasciare il tutto dopo aver acceduto alla matrice attraverso questa procedura.

Chiusura e crash

Per la chiusura del programma, abbaimo mantenuto una lista all'interno della quale sono presenti i TID dei thread attualmente in esecuzione, in modo che qualsiasi thread catturi il segnale di chiusura possa, scandendo la lista, far scattare la cleanup (chiusura del socket relativo al thread e rilascio della memoria allocata) per gli altri thread attivi. I segnali che portano alla chiusura del processo server vengono gestiti tramite una funzione handler, che comprende inoltre: la rimozione dei semafori utilizzati, la chiusura del socket (principale) e del file su cui vengono effettuati i salvataggi. Il programma è resistente ad alcuni segnali e cerca di limitare i danni in caso di crash, grazie a salvataggi che permettono di ripristinare il contesto precedente.

Saving & restoring

Il programma supporta salvataggio/caricamento da file. In particolare, attraverso il parametro "-f <nomeFile> " si può specificare il nome di un file dove verrà salvata la configurazione attuale, l'array delle prenotazioni e gli eventuali cambiamenti di questo (delta). Abbiamo pensato infatti che fosse inutile salvare, ad ogni operazione effettuata dai vari client, l'intero array (che potrebbe raggiungere dimensioni importanti) e che salvare, dopo ogni operazione, soltanto i cambiamenti dell'array dovuti a queste fosse più performante. Abbiamo definito un protocollo di salvataggio dei delta per distinguere le modalità di caricamento (vedi immagine). L'accesso al file di salvataggio, essendo anch'esso concorrenziale, è gestito attraverso un semaforo.
Il ripristino, attivato attraverso lo stesso parametro, permette di ricaricare una sessione precedentemente salvata su un file, reinstallando i parametri di configurazione e lo stato delle strutture dati. Questo è possibile grazie al processamento dei "delta changes".

Caratteristiche

Generiche

MakeFile

Abbiamo creato un makefile per facilitare la compilazione del programma. La funzione generica (all) compila sia client che server e posiziona i file da eseguire nella cartella /bin.

Protocolli proprietari

I protocolli, essendo stati definiti da noi, presentano un overhead minimo. Le specifiche dei protocolli utilizzati si trovano nella cartella /doc.

Chiave di prenotazione dinamica

La chiave di prenotazione è un codice alfanumerico che presenta la seguente struttura:

- ➢ una parte lunga log_10((n x m) + 1) che rappresenta l'indice della posizione della prenotazione all'interno dell'array delle prenotazioni
- ➢ una parte (la restante) costituita da una password generata casualmente.

La lunghezza della chiave è settabile dal client lanciando il programma con l'opzione -s

Output colorato

Possibilità di scegliere una formattazione colorata dell'output.

Command-line interface

Tramite l'utilizzo della libreria "Argp" sia client che server possono essere facilmente gestiti tramite riga di comando. La libreria è offerta da GNU.org (già presente nelle distro Linux) e segue gli standard UNIX per l'immissione dei parametri.

Help esplicativo

L'help message comprende esempi ed informazioni utili sul progetto.

Client

Selezione posti

Possibilità di scegliere la posizione dei posti da prenotare.

Visualizzazione protocolli interattiva

Tramite il parametro "-v" è possibile seguire passo passo le operazioni protocollari relative alla richiesta effettuata.

Porta & IP dinamici

Possibilità di scegliere il server di destinazione (IP). Resistente a input errati (range di porta).

Server

Multithread

Il server accetta più connessioni contemporaneamente: per ognuna viene generato un nuovo thread che la gestisce.

Minimizzazione sezione critica

Grazie all'utilizzo di numerosi semafori le sezioni critiche risultano essere ridotte al minimo.

Prevenzione starvation

Il server prevede una procedura speciale da applicare nel caso in cui si inneschi una situazione di starvation

Lista dei thread attivi

Viene mantenuta una lista collegata dei thread attivi utile per le routine di chiusura.

Resistenza e robustezza

Il server risulta resistente a problematiche di rete (e.g: caduta di connessioni) e robusto rispetto a dati inconsistenti.

Delta changes

Al fine di minimizzare le scritture su disco, si scrivono sul file soltanto i cambiamenti effettuati dai vari client.

Free pointer

L'utilizzo del sistema UNIX-like per la ricerca di una locazione libera in un array permette di ridurre i tempi di attesa dei client.

Modulare

Il codice è suddiviso in librerie e/o moduli per favorire la collaborazione e la comprensibilità.

Allocazione efficiente delle ADT

Sebbene il server supporti impostazioni dinamiche, le strutture dati risultano di dimensione fissata (no realloc).

Thread nominativi

I thread sono stati etichettati con ip e porta del client in modo da rendere più semplice il debugging.

Systemd integration

E' stata implementata l'integrazione con il nuovo gestore di avvio systemd. I file necessari per installare il server come demone sono situati nella cartella systemd/

Log connessioni

Il server presenta a video informazioni sulle connessioni in corso.

Controllo posti

Il server controlla la validità (e.g.: doppioni, out of range, etc) dei posti ricevuti in prenotazione.