Relazione progetto SO 2021/22

Simulazione transazioni

versione minima

Carmine Monaco, matricola 998832, lab T3 (Prof. Enrico Bini)

Tipi di processi

Master: Gestisce gli oggetti ipc, avvia i processi utenti e nodo, gestisce inizio e fine della simulazione, stampando la maggior parte delle informazioni richieste.
Nodo: Riceve richieste di invio transazioni dagli utenti. "Elabora" le transazioni e le inserisce nel libro mastro, notificando l'utente dell'esito della transazione.
Utente: Tenta periodicamente di inviare transazioni ad altri utenti tramite i nodi. Termina nel caso non riesce a mandare nessuna transazione per SO_RETRY volte consecutive.

Scelte progettuali principali

Accesso a libro mastro (memoria condivisa):

- A mutua esclusione tra scrittori (nodi) – usando mutex: *mutex_lm*
- Libero accesso fino a ultimo blocco correttamente scritto per i lettori (utenti)

Movimento transazioni tra utenti e nodi:
Message queue usate:

1. msq_trans: usata da utenti per inviare transazione ai nodi
2. msq_feedback: usata da nodi per inviare esito delle transazioni ricevute a utenti

I destinatari dei messaggi sono specificati inserendo il pid(destinatario) nel tipo del messaggio.

-Comportamento utente:

- Invia transazione tramite msgsnd(msq_trans,…,IPC_NOWAIT) a nodo x;
- Riceve esito transazione tramite msgrcv(msq_feedback,…,getpid(),IPC_NOWAIT) se può inviare altre transazioni;
- Riceve esito transazione tramite msgrcv(msq_feedback,…,getpid(),0) se **non** può inviare altre transazioni.

-Comportamento nodo:

- Riceve transazioni tramite msgrcv(msq_trans,…,getpid(),0) se ha bisogno di ulteriori transazioni per poter elaborare e inserire un blocco nel libro;
- Riceve transazioni tramite msgrcv(msq_trans,…,getpid(),IPC_NOWAIT) se può già elaborare e inserire blocchi (ovvero tp.size>=SO_BLOCK_SIZE-1);
- Invia esito transazione a utente y tramite msgsnd(msq_feedback,…,0).

Osservazione: utente durante la sua vita consuma tutti gli esiti delle transazioni mandati dai nodi, evitando un eventuale blocco dei nodi per via del riempimento di msq_feedback.

Pro: nessun bisogno di semafori esterni per evitare deadlock.
Contro: In caso di msq_trans piena, l'utente non riesce a inviare la transazione.

Tipi di processi - approfondimento

Master

Gestisce gli oggetti ipc, avvia i processi utenti e nodo, gestisce inizio e fine della simulazione, stampando la maggior parte delle informazioni richieste.
Alla fine della simulazione, chiama la funzione **void** clean_exit(int status) che termina eventuali utenti e nodi ancora funzionanti, elimina le risorse ipc assegnate, e effettua la stampa finale della simulazione. Per eliminare le risorse ipc in clean_exit, gli id delle risorse ipc (e altre variabili) si trovano in variabili globali per evitare di passare troppe variabili alla funzione void clean_exit(int status).

Segnali

- SIGALRM: aumenta contatore tempo simulazione, stampa informazioni sull'andamento della simulazione.
- SIGINT: chiude tutti gli utenti, portando il master ad eseguire clean_exit()

Nodo

Riceve richieste di elaborazioni di transazioni dagli utenti che inserisce nella transaction pool nel caso ci sia spazio in essa, altrimenti invia agli utenti l'esito negativo della transazione.
Se nella transaction pool sono presenti >=SO_BLOCK_SIZE-1 transazioni disponibili, dopo averle elaborate e scritte nel libro mastro, invierà l'esito positivo della transazione agli utenti che hanno inviato le transazioni "processate".

Segnali

- SIGINT: pone la variabile globale run a 0, iniziando la procedura di terminazione del nodo

Utente

Tenta periodicamente di inviare transazioni ad altri utenti tramite i nodi. Termina nel caso non riesce a mandare nessuna transazione per SO_RETRY volte consecutive.
Per implementare questo comportamento, viene mantenuto il numero del tentativo di invio di transazione attuale e il numero dell'ultimo tentativo che ha avuto successo. Viene anche usato un array che mantiene le transazioni pendenti che sono state inviate dopo l'ultima transazione che ha avuto successo. Quando non si possono più inviare transazioni, e non si hanno transazioni pendenti nell'array interno, l'utente termina.
L'utente inoltre tenta di inviare una transazione aggiuntiva se riceve il segnale SIGUSR1, a patto che non sia in attesa di un esito positivo di transizioni precedentemente inviate. La transazione inviata con il segnale non aumenta il numero dei tentativi di invio di transazione e l'esito viene ignorato in caso di successo.

Segnali

- SIGUSR1: pone la variable globale trans_request=1, usata per tentare di generare e inviare la transazione.

Scelte progettuali secondarie

- Tutti i processi ottengono i parametri assegnabili a runtime tramite il file "*conf.txt*" e la funzione **int** read_config_from_file(…);

Gestione oggetti ipc: creati da master via flag IPC_PRIVATE nelle funzioni (msg/shm/sem)get. Chiusi da master in funzione **void** clean_exit(int status);

Avvio utenti e nodi effettuato con **int** execve(const char *pathname, char *const argv[],...) da master.

Parametri passati ad utenti e nodo tramite char *const argv[] argv in execve:

- ID di oggetti ipc necessari.

Strutture dati usate

Principali	typedef struct{ pid_t sender,receiver; int qty,reward; struct timespec time; }transaction;	typedef struct{ int index; transaction transactions [SO_REGISTRY_SIZE] [SO_BLOCK_SIZE]; }libro_mastro;	typedef struct{ long mtype; transaction t; int val; }msg_main;	typedef struct{ long mtype; int type; int val; }msg_setup;
Ausiliarie	typedef struct{ char **argv; int argc; }exec_args;	typedef enum{ TIME,LM_FULL,USERS_TERMI NATED,SIG_INT,ERROR }termination_status;	typedef struct{ int num; transaction t; }pending_transaction;	

Oggetti ipc utilizzati:

1. msq_trans - Message queue usata da utenti per inviare transazioni ai nodi
2. msq_feedback - Message queue usata da nodi per inviare esito delle transazioni ricevute a utenti
3. msq_setup usata da master e utenti/nodi per:
   - invio informazioni necessarie per l'avvio della simulazione da master a utenti/nodi.
     - Esempio: invio di lista utenti e nodi a ciascun utente
   - invio informazioni da nodi a master.
     - Esempio: nodo invia a master il num. di transazioni rimanenti nella transaction pool a fine esecuzione.
4. shm_lm - Memoria condivisa contenente il libro mastro
5. mutex_lm_id - Mutex usato da nodi per sincronizzarsi nella scrittura del libro mastro

Nota: in msq_trans e msq_feedback viaggiano messaggi di tipo msg_main, in msq_setup messaggi di tipo msg_setup.

Convenzioni:

- Quando utente riceve l'esito della transazione tramite msq_feedback, l'esito della transazione è negativo se l'attributo *val* del messaggio ricevuto è uguale a -1, altrimenti l'esito è positivo.
- I messaggi msq_setup si distinguono in base all'attributo "type" (mtype non disponibile perché usata per specificare il pid del destinatario). I tipi di messaggi di setup sono i seguenti:
  - 0: sentinella fine messaggi di setup
  - 1: val contiene utente
  - 2: val contiene nodo