Progetto SO 2021/22
Simulazione transazioni

Versione finale

Bini, Radicioni, Schifanella C.

December 2021

Indice

1 Composizione gruppo di studenti

Il progetto si deve svolgere in gruppo composto al **massimo 3 da componenti**. È possibile anche svolgere il progetto da soli.

Si raccomanda che il gruppo sia composto da studenti dello **stesso turno**, i quali discuteranno con il docente del proprio turno. È consentita anche la realizzazione del progetto di laboratorio da parte di un gruppo di studenti di turni diversi. In questo caso, **tutti** gli studenti del gruppo discuteranno con **lo stesso docente**. Esempio: Tizio (turno T1) e Caio (turno T2) decidono di fare il progetto insieme. Lo consegnano e vengono convocati dal prof. Radicioni il giorno X. Tale giorno Tizio e Caio si presentano e ricevono entrambi una valutazione dal Prof. Radicioni (docente del T1, anche se Caio fa parte del turno T2 il cui docente di riferimento è il prof. Bini).

2 Consegna

Il progetto è costituito da:

1. il codice sorgente
2. una breve relazione che sintetizzi le scelte progettuali compiute

Il progetto si consegna compilando la seguente Google Form, cui si accede con credenziali istituzionali,

- `https://forms.gle/2tYN2VW5V18aYp5WA`

la quale richiederà, oltre al caricamento del progetto stesso (un unico file in formato .tgz o .zip NON .rar), anche i seguenti dati per ciascun componente del gruppo: cognome, nome, matricola, email. Dopo il caricamento del progetto, verrete convocati dal docente con cui discuterere (si veda Sezione 1 in caso di gruppo composto da studenti di turni diversi). Attenzione: **compilare una sola form per progetto** (e **non una per ogni componente del gruppo**). Una eventuale ulteriore consegna prima dell'appuntamento **annullerà la data dell'appuntamento**.

La consegna deve avvenire almeno **10 giorni prima** degli appelli scritti per dare modo al docente di pianificare i colloqui:

- se consegnate 10 giorni prima di un appello, il docente propone una data per la discussione entro l'appello seguente
- altrimenti, la data sarà dopo l'appello seguente.

Esempio: per avere la certezza di un appuntamento per la discussione di progetto entro l'appello del 08/02/2022, si deve consegnare entro le ore 24:00 del **29/01/2022**.

3 Valutazione e validità

Il progetto descritto nel presente documento potrà essere discusso se consegnato **entro il 30 Novembre 2022**. Dal Dicembre 2022 sarà discusso il progetto assegnato durante l'anno accademico 2022/23.

Tutti i membri del gruppo devono essere presenti alla discussione. La valutazione del progetto è **individuale** ed espressa in 30-esimi. Durante la discussione

- verrà chiesto di illustrare il progetto;
- verranno proposti quesiti sul programma "Unix" del corso anche non necessariamente legati allo sviluppo del progetto.

È necessario ottenere una votazione di almeno **18** su 30 per poter essere ammessi allo scritto. In caso di superamento della discussione del progetto, la votazione conseguita consentirà di partecipare allo scritto per i **cinque appelli successivi** alla data di superamento.

In caso di mancato superamento, lo studente si potrà ripresentare soltanto dopo almeno **un mese** dalla data del mancato superamento

Si ricorda che il voto del progetto ha un peso di $\frac{1}{4}$ sulla votazione finale di Sistemi Operativi.

4 Pubblicazione del proprio progetto, plagio, sanzioni

Copiare altri progetti o parte di essi impedisce una corretta valutazione. Per questo motivo, gli studenti che consegnano il progetto sono consapevoli che:

- la condivisione con altri gruppi attraverso canali di pubblici o privati (a titolo di esempio: google drive, canali telegram, github, etc.) di tutto o parte del progetto non è consentita fino a tutto Novembre 2022;
- la copiatura di tutto o parte del progetto non è consentita;
- eventuali frammenti di codice estratti da parti di codice visto a lezione o da altri repository pubblici devono essere opportunamente dichiarati.

Nel momento in cui uno studente non rispettasse le sopra citate norme di comportamento, dopo essere stato convocato, sarà oggetto delle seguenti sanzioni:

- se lo studente avrà nel frattempo superato l'esame di Sistemi Operativi anche successivamente alla data di discussione del progetto, la verbalizzazione del voto verrà annullata
- se lo studente avrà soltanto superato la discussione del progetto ma non l'esame, la valutazione del progetto verrà annullata e lo studente non potrà accedere ad ulteriore discussione di progetto prima dei due appelli successivi alla data di evidenza di copiatura.

5 Descrizione del progetto: versione minima (voto max 24 su 30)

Si intende simulare un libro mastro contenente i dati di transazioni monetarie fra diversi utenti. A tal fine sono presenti i seguenti processi:

- un processo *master* che gestisce la simulazione, la creazione degli altri processi, etc.
- `SO_USERS_NUM` processi *utente* che possono inviare denaro ad altri utenti attraverso una *transazione*
- `SO_NODES_NUM` processi *nodo* che elaborano, a pagamento, le transazioni ricevute.

5.1 Le transazioni

Una transazione è caratterizzata dalle seguenti informazioni:

- *timestamp* della transazione con risoluzione dei nanosecondi (si veda funzione `clock_gettime(...)`)
- *sender* (implicito, in quanto è l'utente che ha generato la transazione)
- *receiver*, utente destinatario della somma
- *quantità* di denaro inviata
- *reward*, denaro pagato dal sender al nodo che processa la transazione

La transazione è inviata dal processo utente che la genera ad uno dei processi nodo, scelto a caso.

5.2 Processi utente

I processi utente sono responsabili della creazione e invio delle transazioni monetarie ai processi nodo. Ad ogni processo utente è assegnato un buget iniziale `SO_BUDGET_INIT`. Durante il proprio ciclo di vita, un processo utente svolge iterativamente le seguenti operazioni: SO_RETRY condizione di verifica nel ciclo for

1. Calcola il bilancio corrente a partire dal budget iniziale e facendo la somma algebrica delle entrate e delle uscite registrate nelle transazioni presenti nel libro mastro, sottraendo gli importi delle transazioni spedite ma non ancora registrate nel libro mastro.
   - Se il bilancio è maggiore o uguale a 2, il processo estrae a caso:
     - Un altro processo utente destinatario a cui inviare il denaro
     - Un nodo a cui inviare la transazione da processare
     - Un valore intero compreso tra 2 e il suo bilancio suddiviso in questo modo:
       * il reward della transazione pari ad una percentuale `SO_REWARD` del valore estratto, arrotondato, con un minimo di 1,
       * l'importo della transazione sarà uguale al valore estratto sottratto del reward

       Esempio: l'utente ha un bilancio di 100. Estraendo casualmente un numero fra 2 e 100, estrae 50. Se `SO_REWARD` è pari al 20 (ad indicare un reward del 20%) allora con l'esecuzione della transazione l'utente trasferirà 40 all'utente destinatario, e 10 al nodo che avrà processato con successo la transazione.
   - Se il bilancio è minore di 2, allora il processo non invia alcuna transazione

1) Receiver, serve una memoria con i pid dei fratelli users.
2) Memoria con i pid dei processi nodes.

bilancio: + entrate nel libro mastro + uscite nel libro mastro - uscite NON nel libro mastro

srand(...)

2. Invia al nodo estratto la transazione e attende un intervallo di tempo (in nanosecondi) estratto casualmente tra `SO_MIN_TRANS_GEN_NSEC` e massimo `SO_MAX_TRANS_GEN_NSEC`.

nanosleep(...)

Inoltre, un processo utente deve generare una transazione anche in risposta ad un segnale ricevuto (la scelta del segnale è a discrezione degli sviluppatori).

Se un processo non riesce ad inviare alcuna transazione per `SO_RETRY` volte **consecutive**, allora termina la sua esecuzione, notificando il processo master.

bilancio < 2 per SO_RETRY volte consecutive, terminazione del processo e notifica al master.

5.3 Processi nodo

fare una List o array chiamata transaction pool

Ogni processo nodo memorizza privatamente la lista di transazioni ricevute da processare, chiamata *transaction pool*, che può contenere al massimo `SO_TP_SIZE` transazioni, con `SO_TP_SIZE > SO_BLOCK_SIZE`. Se la *transaction pool* del nodo è piena, allora ogni ulteriore transazione viene scartata e quindi non eseguita. In questo caso, il sender della transazione scartata deve esserne informato.

Le transazioni sono processate da un nodo in *blocchi*. Ogni blocco contiene esattamente `SO_BLOCK_SIZE` transazioni da processare di cui `SO_BLOCK_SIZE`−1 transazioni ricevute da utenti e una transazione di pagamento per il processing (si veda sotto).

Il ciclo di vita di un nodo può essere così definito:

- Creazione di un blocco candidato
  - Estrazione dalla transaction pool di un insieme di `SO_BLOCK_SIZE`−1 transazioni non ancora presenti nel libro mastro
  - Alle transazioni presenti nel blocco, il nodo aggiunge una transazione di reward, con le seguenti caratteristiche:
    * *timestamp*: il valore attuale di `clock_gettime(...)`
    * *sender*: -1 (definire una MACRO...)
    * *receiver*: l'dentificatore del nodo corrente
    * *quantità*: la somma di tutti i reward delle transazioni incluse nel blocco
    * *reward*: 0
- Simula l'elaborazione di un blocco attraverso una attesa non attiva di un intervallo temporale casuale espresso in nanosecondi compreso tra `SO_MIN_TRANS_PROC_NSEC` e `SO_MAX_TRANS_PROC_NSEC`.

nanosleep(...)

- Una volta completata l'elaborazione del blocco, scrive il nuovo blocco appena elaborato nel libro mastro, ed elimina le transazioni eseguite con successo dal transaction pool.

5.4 Libro mastro

shared memory

Lista dinamica con SO_REGISTRY_SIZE dimensione massima, con ogni nodo della lista contenente esattamente SO_BLOCK_SIZE transazioni

Il libro mastro è la struttura condivisa da tutti i nodi e gli utenti, ed è deputata alla memorizzazione delle transazioni eseguite. Una transazione si dice confermata solamente quando entra a far parte del libro mastro. Più in dettaglio, il libro mastro è formato da una sequenza di lunghezza massima `SO_REGISTRY_SIZE` di blocchi consecutivi. All'interno di ogni blocco sono contenute esattamente `SO_BLOCK_SIZE` transazioni. Ogni blocco è identificato da un identificatore intero progressivo il cui valore iniziale è impostato a 0.

Una transazione è univocamente identificata dalla tripletta $(timestamp, sender, receiver)$. Il nodo che aggiunge un nuovo blocco al libro mastro è responsabile anche dell'aggiornamento dell'identificatore del blocco stesso.

5.5 Stampa

Ogni secondo il processo master stampa:

- numero di processi utente a nodo attivi (users e nodes attivi)
- il budget corrente di ogni processo utente e di ogni processo nodo, così come registrato nel libro mastro (inclusi i processi utente terminati). Se il numero di processi è troppo grande per essere visualizzato, allora viene stampato soltanto lo stato dei processi più significativi: quelli con maggior e minor budget.

5.6 Terminazione della simulazione

La simulazione terminerà in uno dei seguenti casi:

- sono trascorsi `SO_SIM_SEC` secondi,
- la capacità del libro mastro si esaurisce (il libro mastro può contenere al massimo `SO_REGISTRY_SIZE` blocchi),
- tutti i processi utente sono terminati.

Alla terminazione, il processo master obbliga tutti i processi nodo e utente a terminare, e stamperà un riepilogo della simulazione, contenente almeno queste informazioni:

- ragione della terminazione della simulazione
- bilancio di ogni processo utente, compresi quelli che sono terminati prematuramente
- bilancio di ogni processo nodo
- numero dei processi utente terminati prematuramente
- numero di blocchi nel libro mastro
- per ogni processo nodo, numero di transazioni ancora presenti nella transaction pool

6 Descrizione del progetto: versione "normal" (max 30)

All'atto della creazione da parte del processo master, ogni nodo riceve un elenco di `SO_NUM_FRIENDS` nodi amici.

Il ciclo di vita di un processo nodo si arricchisce quindi di un ulteriore step:

- periodicamente ogni nodo seleziona una transazione dalla transaction pool **che è non ancora presente nel libro mastro** e la invia ad un nodo amico scelto a caso (la transazione viene eliminata dalla transaction pool del nodo sorgente)

Quando un nodo riceve una transazione, ma ha la transaction pool piena, allora esso provvederà a spedire tale transazione ad uno dei suoi amici scelto a caso. Se la transazione non trova una collocazione entro `SO_HOPS` l'ultimo nodo che la riceve invierà la transazione al processo master che si occuperà di creare un nuovo processo nodo che contiene la transazione scartata come primo elemento della transaction pool. Inoltre, il processo master assegna al nuovo processo nodo `SO_NUM_FRIENDS` processi nodo amici scelti a caso. Inoltre, il processo master sceglierà a caso altri `SO_NUM_FRIENDS` processi nodo già esistenti, ordinandogli di aggiungere alla lista dei loro amici il processo nodo appena creato.

7 Configurazione

I seguenti parametri sono letti a **tempo di esecuzione**, da file, da variabili di ambiente, o da `stdin` (a discrezione degli studenti):

- `SO_USERS_NUM`: numero di processi utente
- `SO_NODES_NUM`: numero di processi nodo
- `SO_BUDGET_INIT`: budget iniziale di ciascun processo utente
- `SO_REWARD`: la percentuale di reward pagata da ogni utente per il processamento di una transazione
- `SO_MIN_TRANS_GEN_NSEC`, `SO_MAX_TRANS_GEN_NSEC`: minimo e massimo valore del tempo (espresso in nanosecondi) che trascorre fra la generazione di una transazione e la seguente da parte di un utente

- `SO_RETRY`, numero massimo di fallimenti consecutivi nella generazione di transazioni dopo cui un processo utente termina
- `SO_TP_SIZE`: numero massimo di transazioni nella transaction pool dei processi nodo
- `SO_MIN_TRANS_PROC_NSEC`, `SO_MAX_TRANS_PROC_NSEC`: minimo e massimo valore del tempo simulato (espresso in nanosecondi) di processamento di un blocco da parte di un nodo
- `SO_SIM_SEC`: durata della simulazione (in secondi)
- `SO_NUM_FRIENDS` (solo versione max 30): numero di nodi amici dei processi nodo (solo per la versione full)
- `SO_HOPS` (solo versione max 30): numero massimo di inoltri di una transazione verso nodi amici prima che il master creai un nuovo nodo

Un cambiamento dei precedenti parametri non deve determinare una nuova compilazione dei sorgenti.

Invece, i seguenti parametri sono letti a **tempo di compilazione**:

- `SO_REGISTRY_SIZE`: numero massimo di blocchi nel libro mastro
- `SO_BLOCK_SIZE`: numero di transazioni contenute in un blocco

La seguente tabella elenca valori per alcune configurazioni di esempio da testare. Si tenga presente che il progetto deve poter funzionare anche con altri parametri.

parametro	letto a ...	"conf#1"	"conf#2"	"conf#3"
`SO_USERS_NUM`	run time	100	1000	20
`SO_NODES_NUM`	run time	10	10	10
`SO_BUDGET_INIT`	run time	1000	1000	10000
`SO_REWARD` [0–100]	run time	1	20	1
`SO_MIN_TRANS_GEN_NSEC` [nsec]	run time	100000000	10000000	10000000
`SO_MAX_TRANS_GEN_NSEC` [nsec]	run time	200000000	10000000	20000000
`SO_RETRY`	run time	20	2	10
`SO_TP_SIZE`	run time	1000	20	100
`SO_BLOCK_SIZE`	compile time	100	10	10
`SO_MIN_TRANS_PROC_NSEC` [nsec]	run time	10000000	1000000	
`SO_MAX_TRANS_PROC_NSEC` [nsec]	run time	20000000	1000000	
`SO_REGISTRY_SIZE`	compile time	1000	10000	1000
`SO_SIM_SEC` [sec]	run time	10	20	20
`SO_FRIENDS_NUM`	run time	3	5	3
`SO_HOPS`	run time	10	2	10

8 Requisiti implementativi

Il progetto (sia in versione "minimal" che "normal") deve

- essere realizzato sfruttando le tecniche di divisione in moduli del codice,
- essere compilato mediante l'utilizzo dell'utility `make`
- massimizzare il grado di concorrenza fra processi
- deallocare le risorse IPC che sono state allocate dai processi al termine del gioco
- essere compilato con almeno le seguenti opzioni di compilazione:

```
gcc -std=c89 -pedantic
```

- poter eseguire correttamente su una macchina (virtuale o fisica) che presenta parallelismo (due o più processori).

Per i motivi introdotti a lezione, ricordarsi di definire la macro `_GNU_SOURCE` o compilare il progetto con il flag `-D_GNU_SOURCE`.