Esame Software Engineering (AA 2021/22)

4 Novembre 2022

Enrico Tronci
Computer Science Department, Sapienza University of Rome
Via Salaria 113 - 00198 Roma - Italy

`tronci@di.uniroma1.it` `http://mclab.di.uniroma1.it`

Esercizio 2 (20 punti)

L'unità di tempo per questo esercizio è il secondo.

Una possibile implementazione di una architettura a microservizi consiste di un DB che contiene i dati permanenti condivisi tra i microservizi e delle implementazioni dei microservizi.

Un microservizio, a suo volta, può essere realizzato con un *dispatcher* a da un certo numero di *istanze* del microservizio. Il dispatcher riceve le richieste di servizio e le passa ad una istanza che esegue il calcolo e ritorna il risultato al servizio che l'ha richiesta.

Nel nostro contesto, un *dispatcher* può essere implementato con N code di input (dove N è il numero di servizi che può accedere al dispatcher) e K outputs (dove K è il numero di istanze del microservizio disponibili).

Il blocco *dispatcher* periodicamente guarda se è presente (coda di input non vuota) una richiesta su qualcuno degli input e, nel caso, la passa ad una delle istanze.

Per semplicità ci concentriamo sul caso di un microservizio che prende come input due numeri interi e ritorna come output la loro somma.

L'ambiente `Env` per il microservizio consiste di processi che mandano richieste ad una delle code di input del dispatcher.

Una richiesta è una tripla di interi. I primi due elementi della tripla rappresentano gli interi da sommare ed il terzo elemento è l'id del processo (istanza di microservizio) che ha richiesto il servizio.

Questo esercizio si focalizza sulla modellazione dell'ambiente `Env` di cui sopra. A tal fine si sviluppino i seguenti blocchi.

1. Blocco `Env` nel file `env.mo` che modella un blocco che genera a richieste a random, cioè triple di interi a random e le inserisce in una FIFO (delle richieste al dispatcher). Si assuma:
   - Che ogni componente della tripla è un numero intero compreso tra -1000 e 1000 estremi inclusi;
   - Che il tempo medio tra una richiesta e l'altra è di 10 unità di tempo
2. Blocco `DQueue` nel file `dqueue.mo` che modella la FIFO delle richieste al dispatcher. Il blocco `Env` inserisce le richieste in questa FIFO.

3. Blocco `DQueueTerminator` nel file `dqueueterminator.mo` che modella un processo che ogni unita di tempo estrae un dato (se presente) dalla FIFO di input. Nel nostro setting `DQueueTerminator` è connesso sul lato di lettura alla FIFO `DQueue` (mentre il blocco `Env` è connesso sul lato di scrittura di `DQueue`). Il blocco `DQueueTerminator` permette di eseguire simulazioni di lunghezza arbitraria senza che la FIFO `DQueue` vada in overflow perchè c'è un processo che inserisce dati nella FIFO (`Env`) e nessuno che li estrae.

4. Blocco `monitor1` nel file `monitor1.mo` che prende come input le triple generate dal blocco `Env` e ritorna come output valor medio e deviazione standard per ogni componente della tripla e del tempo tra un richiesta e l'altra. Se tutto è realizzato correttamente il valor medio per ogni componente dovrebbe essere circa 0 ed il tempo medio tra un richiesta e l'altra dovrebbe essere circa 10.

Parametri del Modello

Il modello conterrà i seguenti parametri:

1. `HORIZON`, contenente l'orizzonte di simulazione, cioè il tempo simulato. Potete usare `HORIZON` = 10000.

Output della simulazione

Si usi l'istruzione Modelica `terminate` per terminare la simulazione quando la variabile Modelica `time` ha un valore maggiore del paramentro `HORIZON`.

Alla terminazione si stampino nel file `outputs.txt` valori medi e deviazioni standard per ogni componente della tripla e del tempo tra richieste, con il seguente formato.

La prima riga (di *intestazione*) del file `outputs.txt` contiene:

`Avg1 StdDev1 Avg2 StdDev2 Avg3 StdDev3 AvgWait StdWait` (`ID` = `aaa`, `MyMagicNumber` = `bbb`, `HORIZON` = `ccc`, `time` = `ddd`)

dove:

- `aaa` è il valore del parametro `ID` (numero di matricola),
- `bbb` è il valore del parametro `MyMagicNumber`,
- `ccc` è il valore del parametro Modelica HORIZON,
- `ddd` è il valore della variabile Modelica `time` quando la simulazione viene terminata dal comando `terminate`.

Le altre righe hanno il seguente formato:

<Valor medio della componente 1 della tripla> <Valore della deviazione standard della componente 1 della tripla> <Valor medio della componente 2 della tripla> <Valore della deviazione standard della componente 2 della tripla> <Valor medio della componente 3 della tripla> <Valore della deviazione standard della componente 3 della tripla> <Valor medio del tempo tra una richiesta e l'altra> <Valore della deviazione standard del tempo tra una richiesta e l'altra>

Si avranno quindi, a parte la prima riga di intestazione, 1 sola riga con 8 colonne.

Si usi un orizzonte di simulazione molto grande (maggiore di HORIZON). In particolare si verifichi che l'orizzonte di simulazione sia maggiore del valore del `time` quando la simulazione viene terminata dal comando `terminate`. Se questo non è verificato il modello è sbagliato. Questo valore di `time` è visibile su stdout.

NOTA

Si vedano le istruzioni ed in particolare la sezione *NOTA BENE* delle istruzioni.