Relazione assignment 1

Antonio Iannotta

March 2022

Contents

1 Analisi del problema

1.1 Descrizione del problema

Il problema consiste nello sviluppare una simulazione del fenomeno fisico dell'interazione tra corpi all'interno di una superficie ben delimitata.
Tali corpi sono soggetti di conseguenza alla forza che viene esercitata da tutti gli altri corpi, con tale forza che viene influenzata dalla variazione della posizione in cui i corpi vengono a trovarsi. La variazione della posizione è determinata sia dalla forza e sia dalla collisione dei corpi con i bordi.

1.2 Descrizione del problema concorrente

Il problema della concorrenza risiede nel fatto che ciascun corpo deve monitorare la propria situazione. In particolare ciascun corpo si ritroverà a calcolare la forza a cui è soggetto in relazione alla posizione di tutti gli altri corpi, così come dovrà gestire la situazioni lungo i bordi della superficie. Questa è una situazione concorrente dal momento che, come sarà maggiormente spiegato nella sezione successiva, ciascun corpo dovrà calcolare la propria situazione sulla base del comportamento degli altri (e sulla base dell'eventuale collisione con i bordi) in un certo istante.

2 Strategia risolutiva e architettura del sistema

Come da specifica, il primo passo risolutivo è stato quello di ragionare in maniera modulare piuttosto che monolitica. Nell'ottica di un programma concorrente si sono individuati i corpi come Thread. Tuttavia si è ritenuto che creare un'unica classe che estendesse la classe astratta Thread potesse in qualche modo compromettere la chiarezza del programma.
Per la soluzione del problema è possibile individuare 4 classi chiave:

- **Body**
- **BodyThread**
- **Monitor**
- **Barrier**

Monitor e Barrier saranno maggiormente spiegate nel seguito. Ci si vuole soffermare ora sulle classi **Body** e **BodyThread**.
Come da specifica il primo passo è stato quello di attuare una modularizzazione del sistema, per questo motivo si è deciso di creare due classi. La prima classe, Body, presenta un generico corpo con le caratteristiche che lo contraddistinguono.
La seconda classe, BodyThread, rappresenta i corpi come Thread. In particolar modo il costruttore di questa classe è il seguente:
BodyThread(Monitor, Barrier, Body)

dove Monitor e Barrier sono i due elementi che si occupano della gestione della concorrenza che saranno meglio spiegati dopo mentre il Body rappresenta il corpo a cui il Thread fa riferimento. Le operazioni che vengono eseguite dal BodyThread (chiamate quindi all'interno del metodo *run()*) sono le due operazioni definite all'interno del Monitor, ovvero quelle che consentono di aggiornare la velocità con cui ciascun corpo si muove e di controllare la situazione lungo i bordi. Dopo ciascuna operazione il BodyThread effettua un aggiornamento della sua posizione. Prima di parlare di Monitor e Barrier è opportuno chiarire come si è pensato di organizzare l'architettura del sistema. L'architettura consiste in:

- un model che raggruppa le due classi di riferimento per il Body ed il Monitor
- un'interfaccia grafica che si occupa di mostrare a video quello che effettivamente avviene
- entità Barrier che opera come Monitor e che gestisce le diverse iterazioni con cui le operazioni dei vari corpi devono susseguirsi

2.1 Monitor

Dopo aver effettuato la modularizzazione dei corpi ci si è soffermati sulla gestione dell'aspetto concorrente. In particolar modo la scelta è ricaduta sul monitor come strumento in quanto consentiva atomicità e mutua esclusione delle due sezioni critiche che sono state individuate: **aggiornamento della velocità** e **gestione della situazione lungo i bordi**. La semantica che si è deciso di adottare è la seguente:

- In un certo istante un corpo prende possesso del monitor e vuole eseguire una delle due operazioni. Tuttavia in un certo istante un solo corpo può essere dentro al monitor e, conseguentemente, abbiamo una sola operazione possibile per questo corpo. In particolar modo le due operazioni sono atomiche, iniziano e finiscono. Si è voluto evitare situazioni di interleaving tra le diverse operazioni. Il motivo che ha portato a questa scelta è che è lecito pensare che in questo modo la gestione delle forze, il calcolo delle forze e le posizioni che i corpi vanno ad assumere saranno calcolati in maniera più precisa. Inoltre si è voluto seguire un ragionamento di natura matematica. Nella risoluzione di sistemi lineari (come può essere un sistema che consenta di calcolare la posizione di corpi soggetti ad una certa forza) quello che si fa è sempre risolvere per variabili scegliendone sempre prima una.
- Quando un corpo entra ad eseguire una certa posizione marca la sua presenza incrementando un contatore che fa in modo che qualsiasi altro processo voglia entrare ad eseguire quella porzione di codice venga posto in attesa. Alla fine della computazione mediante una notify il corpo comunica

ad uno dei Thread in attesa la possibilità di poter entrare nel monitor. Un aspetto importante è la totale indifferenza nei confronti dell'ordine con cui si susseguono i corpi nell'accedere al monitor. Questo è un aspetto che abbiamo considerato secondario dal momento che l'impoortanza era sull'atomicità delle due operazioni.

2.2 Controller

Si è accennato prima al fatto che nell'architettura fosse presente un'entità tra il model rappresentato dalla coppia di classi **Body-BodyThread** e dal Monitor da una parte e l'interfaccia grafica dall'altra. Tale entità è una Barriera che si occupa di gestire le diverse interazioni (NSteps) dei gruppi di corpi in modo da garantire che non ci sia overlapping. Mediante la barriera quindi si gestisce il numero di iterazioni che i corpi devono eseguire impedendo che questi muoiano ma ponendoli in una coda e risvegliandoli tutti per un numero di volte pari a quello delle iterazioni. La soluzione apportata quindi viene ad essere una soluzione che comprende due monitor:

- Il primo, implementato dalla classe Monitor è quello che si occupa di gestire la concorrenza delle operazioni.
- Il secondo, quello implementato dalla classe Barrier è quello che si occupa di garantire che le iterazioni avvengano in maniera corretta.

2.2.1 Grafica

La terza parte dell'architettura che si è deciso di adottare è quella relativa alla grafica. In particolar modo l'interfaccia grafica opera andando a selezionare il controller con il relativo numero di corpi e iterazioni per far partire la parte visuale della simulazione.

3 Comportamento del sistema

Per quanto riguarda il comportamento del sistema quello che si è deciso di fare è di modellare l'interazione di maggiore interesse dal punto di vista della concorrenza, vale a dire quella tra i corpi e il monitor che contiene le operazioni da eseguire.

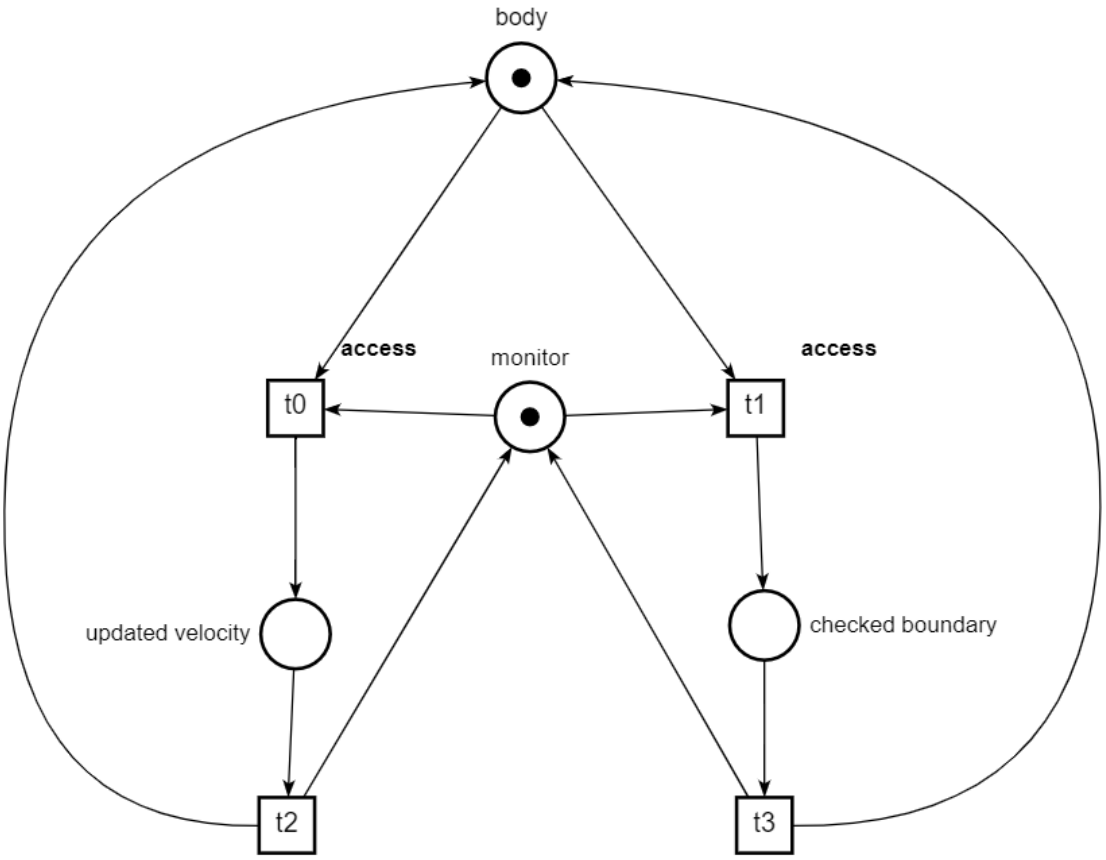

Figure 1: Caption

4 Performance

Per quanto riguarda le performance sono stati eseguiti i seguenti test con 100,1000 e 5000 corpi:

100 corpi:
1000 iterazioni: Tempo di esecuzione = 9008 ms
10000 iterazioni: Tempo di esecuzione = 25808 ms
50000 iterazioni: Tempo di esecuzione = 63869 ms

1000 corpi:
1000 iterazioni: Tempo di esecuzione = 35141 ms
10000 iterazioni: Tempo di esecuzione = 238218 ms
50000 iterazioni: Tempo di esecuzione = 1010631 ms

5000 corpi:
1000 iterazioni: Tempo di esecuzione = 388795 ms
10000 iterazioni: Tempo di esecuzione = 5560709 ms
50000 iterazioni: Tempo di esecuzione = 20188743 ms

4.1 Considerazioni sulle performance

I precedenti test sono stati eseguiti su diverse macchine per valutare l'eventuale differenza nella potenza di calcolo. Quello che si è rilevato è che oltre la potenza di calcolo quello che effettivamente incide per quanto riguarda le performance è il numero di Thread che vengono istanziati. Questo è evidente andando a considerare le differenze che si riscontrano nel caso di 100,1000 e 5000 Thread istanziati a parità di numero di iterazioni. La conclusione che si è tratta è che indipendente dal numero di iterazioni esiste un numero di Thread massimo oltre il quale il sistema non scala in maniera estremamente efficace.

5 Analisi con TLA+