Alma Mater Studiorum · Università di Bologna

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA E SCIENZE INFORMATICHE

campus di Cesena

Relazione PCD Assignment02: Parte 1

Brini Tommaso

Buizo Manuel

Rambaldi Riccardo

Anno Accademico 2023-2024

Indice

Capitolo 1

Introduzione

La prima parte del secondo assignment prevedeva l'implementazione della simulazione realizzata nel assignment precedente, ma questa volta gestita con task asincroni concorrenti sfruttando l'Executor Framework di Java.

Capitolo 2

Strategia Risolutiva e Architettura Proposta

L'approccio risolutivo è suddiviso in due parti: **SpecificWorker** e **GenericWorker**. Entrambi gli approcci sono stati implementatati a task asinconi, tuttavia con una piccola differenza.

Il **GenericWorker** è basato sull'idea di identificare i singoli agenti come task, rendendo quindi sequenziale il singolo comportamento (inteso come Sense, Decide e Act) di ogni agente. Dopodichè si era pensato di gestire la sincronizzazione dei vari agenti con l'environment sfruttando sempre una barriera (implementata come monitor) che fosse ciclica per ogni step dei vari agenti. Tuttavia si è ritenuto più corretto sfruttare il meccanismo proprio delle futures, aspettando la terminazione di ogni agente prima di eseguire lo step successivo.

Lo **SpecificWorker**, al contrario del Generic, cerca di sfruttare il parallelismo delle varie azioni eseguibili da un agente, scorporando ognuna come task. Infatti in questo caso, ogni singolo comando di ogni agente che sta eseguendo il proprio step viene eseguito parallelamente a quello degli altri agenti coinvolti. In sostanza, ad ogni step vengono eseguiti (nel giusto ordine) i tre comandi che compongono lo step di ogni agente coinvolto nella simulazione, aspettandone la terminazione prima di procedere.

L'architettura adottata per entrambe le soluzioni è Master-Worker, dove il master utilizza un ***newScheduledThreadPool executor*** inizializzato di default con un core pool di 7 thread, per eseguire i vari task. Tuttavia, sia

il tipo di executor che il numero di thread del pool possono essere cambiati dalla GUI della simulazione scegliendo tra un ***newFixedThreadPool*** e un ***newSingleThreadExecutor***.

Capitolo 3

Descrizione Comportamento del Sistema

La simulazione è gestita dalla classe AbstractSimulation che è il Thread principale che si occupa di inizializzare e lanciare la simulazione. Inizialmente, dopo aver inizializzato l'environment e il numero corretto di agenti, esegue il primo step dell'environment per poi lanciare tramite il master worker i vari agenti coinvolti. Il master, una volta eseguiti i task, si mette in wait a sua volta sul monitor *startStopMonitor*.
Una volta che anche questi hanno terminato svegliano il thread della simulazione tramite il master per eseguire lo step successivo. All'interno del monitor, la wait viene chiamata su una *condition variable* di una mutex *ReentrantLock*. Questo monitor permette inoltre di gestire la funzione di *pause* e *resume* del sistema. All'inizio di ogni step infatti, il thread dell'AbstractSimulation controlla se è stato premuto il tasto pause, e nel caso si mette in wait. Come anticipato sopra, la sincronizzazione dei vari task è gestita tramite le future: ogni task eseguito dall'executor tramite il metodo *submit(Runnable task)* restituisce una future che rappresenta il risultato della computazione che "prima o poi" sarà disponibile. Per ognuna, viene chiamata la *get()* dal master che quindi attende che sia disponibile il risultato di tutti prima di poter procedere allo step successivo. In questo caso, l'environment esegue il proprio step sempre prima di tutti gli agenti. Questo vale per entrambi i master descritti: anche lo SpecificWorker chiama la *get()* su ogni task (che in questo caso rappresenta il singolo comando) prima di passare allo step o al comando successivo.
Dalla GUI è inoltre possibile, solo per lo SpecificWorker, decidere quanti thread dedicare all'esecuzione di ciascun tipo di comando (Sense, Decide e Act).

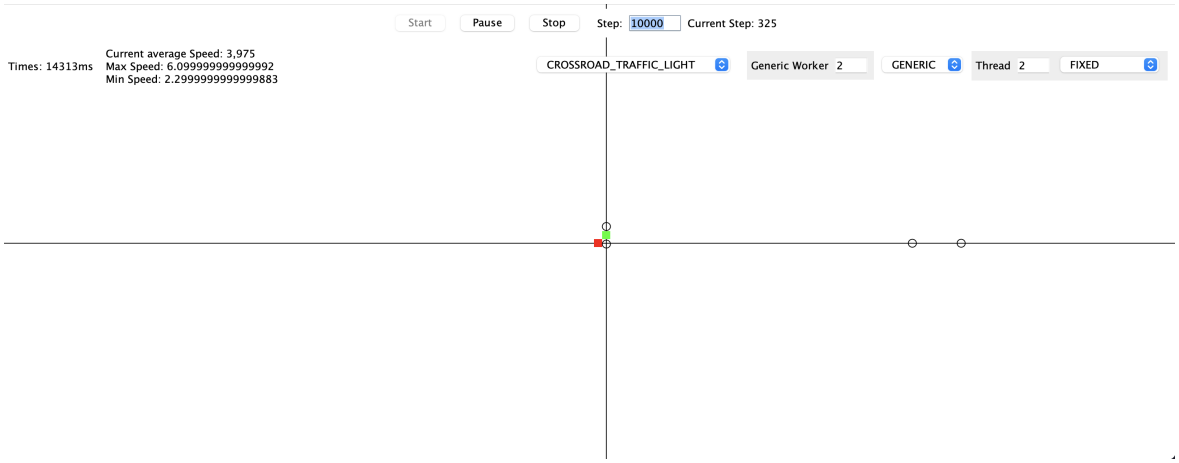

Figura 3.1: GUI realizzata (GenericWorker)

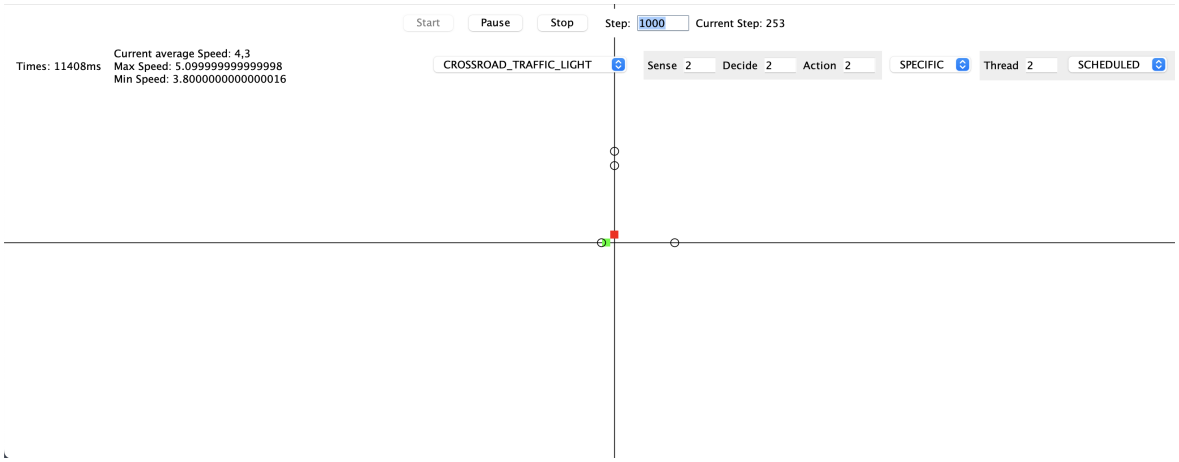

Figura 3.2: GUI realizzata (SpecificWorker)

Capitolo 4

Prove di Performance

Sono stati testati i tempi di esecuzione al variare del tipo di executor scelto.

Figura 4.1: Fixed Thread Pool executor

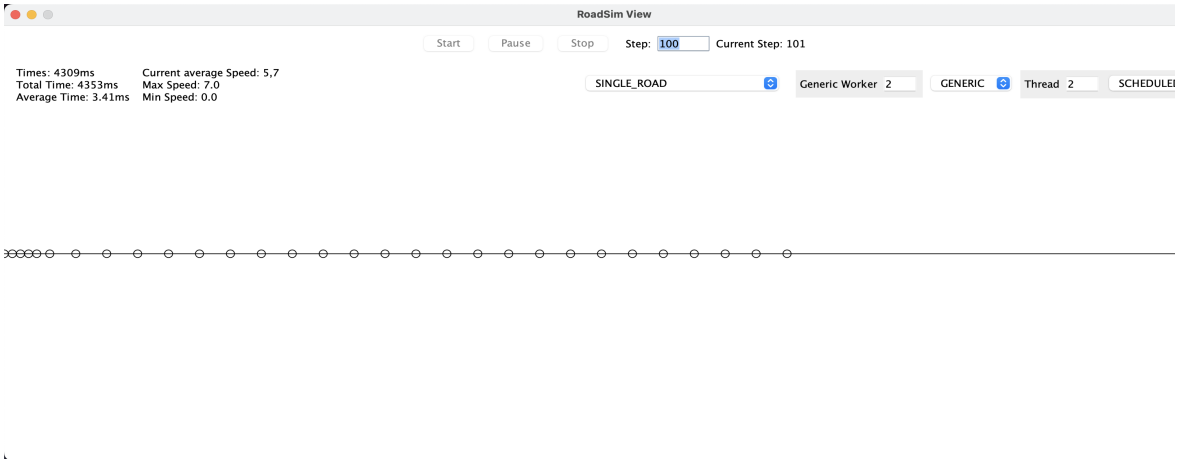

Figura 4.2: Scheduled Thread Pool executor

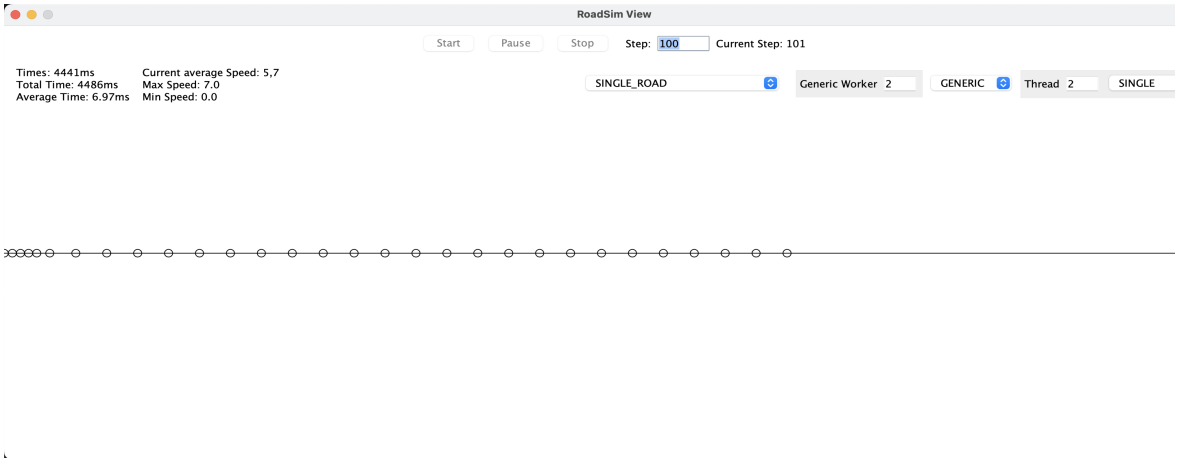

Figura 4.3: Single Thread executor