Progetto per monitoraggio di sistemi TuCSoN

Sitemi Autonomi

Giacomo Dradi 726868
giacomo.dradi@studio.unibo.it

14 marzo 2016

Sommario

Il lavoro svolto consiste nella realizzazione di un sistema software in grado di monitorare, tramite una GUI interattiva, lo stato di uno o più *tuple center TuCSoN* allocati su uno o più nodi fisici. Il sistema è in grado di monitorare *tuple center*, tuple contenute in essi e trasferimenti di tuple. Può essere inoltre utilizzato per effettuare simulazioni.

Indice

1 Overview

Il progetto prevede la realizzazione di un sistema software *Java* in grado di monitorare lo stato di un sistema *TuCSoN* composto da uno o più tuple center, allocati su uno o più nodi fisici, e di mostrare le informazioni ottenute in forma grafica attraverso una GUI. Il software può essere utilizzato, per estensione, per monitorare tutti quei sistemi più complessi costruiti sull'infrastruttura *TuCSoN*, come ad esempio *Molecules of Knowledge*.

Il sistema è composto dai seguenti progetti:

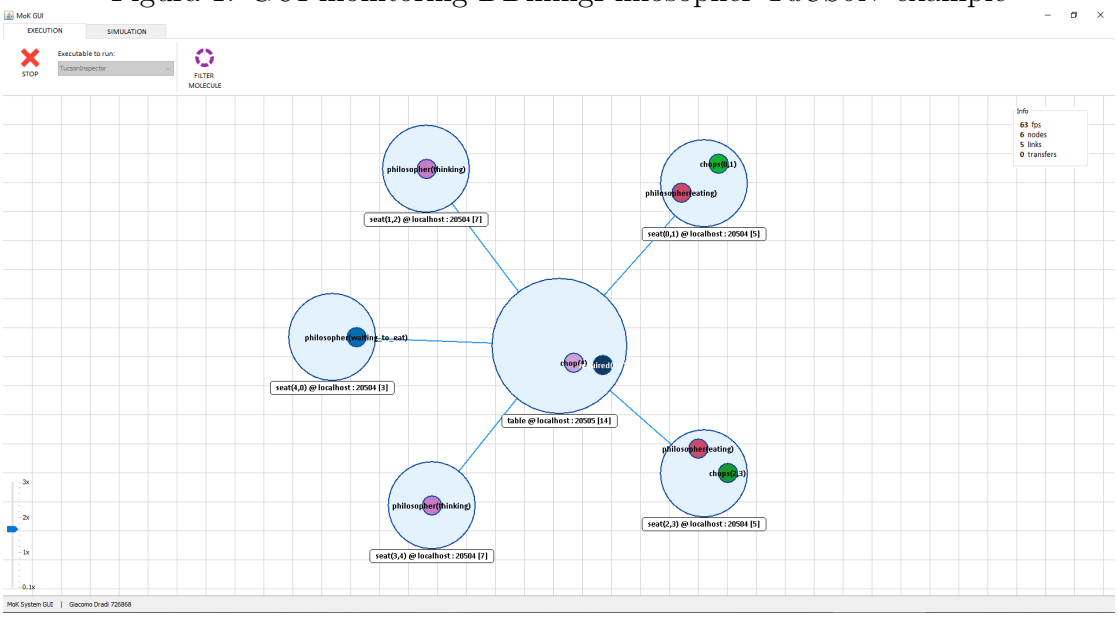
1. *it.unibo.mok.gui*, che comprende la GUI vera e propria, utilizzabile come libreria tramite apposite API.

2. *it.unibo.mok.inspector*, contenente la business logic con cui l'ispezione viene effettuata. Richiede *it.unibo.mok.gui*.

In aggiunta, l'*Inspector* default di *TuCSoN* è stato leggermente modificato ed esteso. Tutto il codice sorgente è disponibile nel branch *feature/inspector4gui* del repository ufficiale di *TuCSoN*:

https://bitbucket.org/smariani/tucson

Figura 1: GUI monitoring DDiningPhilosopher *TuCSoN* example



2 Inspector GUI

Il progetto *it.unibo.mok.gui* contiene l'implementazione della GUI. Essa si basa su *Swing* e non richiede l'utilizzo di librerie esterne.

2.1 API

Le API offerte dalla GUI (*it.unibo.mok.gui.interfaces.GUI*) sono le seguenti:

- aggiunta e rimozione di un nodo, inteso come *tuple center*.
- aggiunta e rimozione di un collegamento (link) tra due nodi. NB: due nodi devono essere collegati per poter comunicare.
- aggiunta e rimozione di una tupla (molecola) da un nodo.
- cambio di concentrazione di una tupla in un nodo.
- trasferimento di una tupla da un nodo a un altro.

Le tuple vengono identificate tramite una stringa. L'aggiunta di una tupla a un nodo in cui esiste già una tupla con lo stesso identificatore genera un aumento della concentrazione della tupla già esistente del valore di concentrazione della tupla da aggiungere.

La chiamata di trasferimento permette di specificare la "quantità" di tupla da trasferire, indicata come valore di concentrazione da spostare. Permette inoltre di scegliere se aggiungere automaticamente la nuova tupla al nodo destinazione, oppure no. In caso positivo, verrà eseguita l'operazione di aggiunta al nodo destinazione con lo stesso comportamento della aggiunta manuale, in caso negativo tale operazione dovrà essere effettuata manualmente. La chiamata di trasferimento è bloccante se il link sul quale viene effettuato è al momento utilizzato da un altro trasferimento (non importa la direzione). Per questo motivo, è bene delegare la gestione dei trasferimenti a un thread diverso per ogni link.

2.2 Comandi utente

La GUI supporta i seguenti comandi:

- zooming out e in, tramite slider o rotella del mouse
- panning dell'intera area, tramite drag con tasto destro
- spostamento di un nodo, tramite drag con tasto sinistro

2.3 Modalità di visualizzazione

La GUI offre due modalità di visualizzazione, a seconda del valore corrente di zoom: con tuple, o con grafici a torta.

Nella modalità con molecole, verranno visualizzate, per ogni nodo, tutte le tuple presenti al suo interno, con il valore di concentrazione all'interno di ognuna.

Nella modalità con grafici a torta, ogni nodo è rappresentato da un grafico a torta, in cui vengono mostrate solo le tuple che hanno concentrazione maggiore della media di tutte le concentrazioni di quel nodo. Per ogni tupla sarà presente una "fetta" del grafico, con grandezza proporzionale alla sua concentrazione rispetto al totale delle concentrazioni mostrate.

2.4 Modalità di funzionamento

La GUI ha due modalità di funzionamento: simulazione delle API o esecuzione di inspectors.

Nella modalità simulazione, è possibile testare ogni API della GUI, verificandone gli effetti.

Nella modalità esecuzione è possibile lanciare degli *Executable*, creati custom a seconda dello specifico sistema da testare, che a loro volta richiameranno le api della gui. Tali *executables* possono essere iniettati anche a *run-time*, e possono essere utilizzati anche per test delle performance.

2.5 Configurazione

La GUI è configurabile tramite un file di configurazione testuale, "*gui.config*", che deve trovarsi nella stessa directory dell'eseguibile che lancia la GUI. Alcuni parametri di particolare interesse sono:

- `NODE_BORDER_THICKNESS` specifica la dimensione del bordo dei nodi. Se 0, non viene disegnato alcun bordo, dimezzando il numero di cerchi presenti nella GUI (migliori performance).
- `MOLECULE_BORDER_THICKNESS` come sopra, ma specifica il bordo delle tuple (molecole).
- `DANCE_ANIMATION_FPS` se 0, le tuple non si muoveranno all'interno dei nodi, ottenendo migliori performance.

- `MOLECULE_SHOW_ID_INSTEAD_OF_CONCENTRATION` se true, ogni tupla (molecola) mostrerà al suo interno il suo identificatore invece che la concentrazione. Utile per debug.
- `NODE_PIE_CHART_ZOOM_THRESHOLD` indica il valore di zoom di soglia per cui la visualizzazione passa dalla modalità a tuple a quella con grafici a torta.

3 TuCSoN Inspector

Nel progetto *it.unibo.mok.inspector* sono contenuti alcuni esempi di *executables*:

- *SimpleMoKInstance*, usato principalmente per test
- *StressTest*, per test sulle performance
- *TucsonInspector*, il core dell'ispezione di *TuCSoN*

Tale progetto contiene anche la classe *Main*, eseguibile, che istanzia una GUI e inietta gli *executable* sopra elencati. Il progetto della GUI (o un artefatto generato da esso) deve essere perciò importato come dipendenza.

3.1 Funzionamento

Qualunque sistema *TuCSoN* può essere monitorato dall'*executable TucsonInspector*. Il suo funzionamento è il seguente:

1. al lancio, legge dal file di configurazione *"network.config"* i tuple center, secondo la nomenclatura *TuCSoN*, da ispezionare.
2. crea una comunicazione tra *Skeleton* e *Stub* di *Inspector4Gui*, istanziando un *Inspector4Gui* per ogni *tuple center*.
3. a ogni notifica (tramite pattern observer) di aggiornamento ricevuta da qualunque degli *Inspector4Gui* istanziati (nuova tupla, trasferimento, ecc), chiama l'opportuno metodo della GUI.

3.2 Inspector4Gui

Inspector4Gui è il componente che realizza fisicamente la comunicazione con il nodo *TuCSoN* a ci si riferisce. Esso, assieme alle altre classi necessarie, è contenuto nel package:

alice.tucson.introspection4gui

del progetto *TuCSoN*.

Il funzionamento è il seguente:

1. al lancio, istanzia un *InspectorContextStub* default, specificando come agent-role: inspector4gui.

2. l'*ACCProvider* di *TuCSoN*, ricevuta la request dallo Stub, istanzia un *Inspector4GuiContextSkel*, estensione dell'InspectorContextSkel default.

Inspector4GuiContextSkel invia un *Inspector4GuiContextEvent* a ogni "trigger" ricevuto dalla *RespectVM*. Di default, il contenuto informativo del messaggio prevede:

- l'intera lista delle tuple del *tuple center*
- le reazioni che hanno avuto successo
- altre informazioni non rilevanti in questo contesto

Inspector4GuiContextSkel, diversamente dallo skel default, non comunica tutte le tuple del nodo a ogni trigger. Esso memorizza le tuple correnti del nodo, e ad ogni update ricevuto trova le tuple da aggiungere e/o quelle rimosse. Nel messaggio da inviare allo Stub inserirà solamente queste, e non l'intero contenuto del *tuple center*.

Dal momento che l'infrastruttura su cui basa *Inspector4Gui* riusa in gran parte quella dell'*Inspector* default, alcune funzioni di quest'ultimo sono state mantenute, come ad esempio la possibilità di filtrare le tuple da ricevere secondo una pattern matching Prolog.

Per poter intercettare quando una tupla viene trasferita, *Inspector4Gui* controlla, tra le reazioni che hanno avuto successo, quelle contenenti l'operatore "?". Dai suoi termini, ricava poi le informazioni circa *tuple center* sorgente, destinazione, e molecola da scambiare.

3.3 Interazione

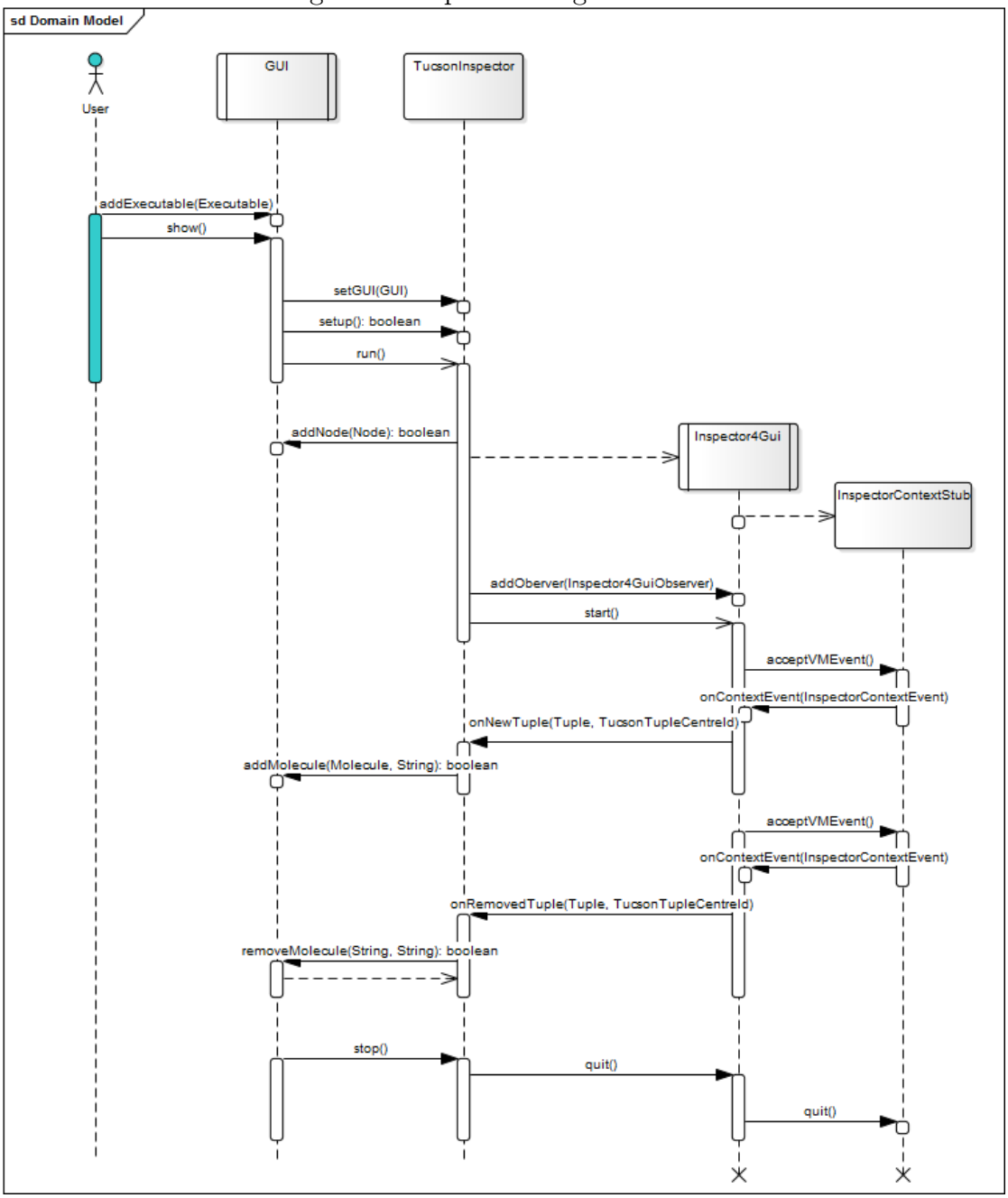
In figura 3.3 viene riportato un Sequence Diagram UML2 che rappresenta un esempio di interazione tra i vari componenti durante l'esecuzione di un *TucsonInspector*.

3.4 Configurazione

Il *TucsonInspector* istanzia inizialmente una ispezione con i *tuple center* specificati nel file *network.config*, che deve trovarsi nella stessa directory dell'eseguibile lanciato. I *tuple center* vanno specificati uno per riga e secondo la nomenclatura *TuCSoN*, ad esempio:

```
tc0 @ localhost : 20505
tc1 @ localhost : 20504
```

Figura 2: Sequence Diagram UML2



NB: se durante l'esecuzione, *TucsonInspector* rileva l'invio di una tupla a un nodo che non stia già monitorando, istanzierà un nuova ispezione con esso.