Il trasferimento asincrono di controllo in Java real – time

Con il termine trasferimento asincrono di controllo in Java real - time si intende la possibilità di interrompere in modo asincrono un thread e di fargli eseguire dell'altro codice. Per ottenere questa importante funzionalità esistono sostanzialmente due approcci, uno basato sulla dichiarazione di metodi interrompibili, uno basato sull'esecuzione di oggetti interrompibili.

Il primo approccio si basa sul poter dichiarare certi metodi come interrompibili. Se si chiama il metodo interrupt di un thread real time mentre questo è in esecuzione verrà lanciata una AsyncronouslyInterruptedException che il sistema deve raccogliere e, nel relativo catch, si può eseguire del codice per rispondere all'interruzione aggiunta dall'esterno. Il comportamento è quindi sostanzialmente diverso da quanto accadeva con i normali thread nei quali bisognava verificare con una strategia a polling se il thread era stato interrotto meno.

Per poter gestire questa funzionalità occorre specificare i metodi che supportano un'interruzione asincrona e quelli che non lo fanno. I primi metodi vengono chiamati metodi interrompibili asincronicamente (AIMetods); tali metodi si differenziano dagli altri per il fatto dichiarano una AsyncronouslyInterruptedException nella loro throw list. I secondi metodi, quelli ordinari e le sezioni di codice synchronized, vengono definite ATCDeferred in quanto se viene invocato il metodo interrupt mentre un thread sta eseguendo questa regione di codice l'eccezione verrà inviata solamente una volta che il thread entrerà in una sezione interrompibile. Visto che questo approccio è a tutti gli effetti conforme con la gestione di un'eccezione non è possibile rientrare all'interno del catch nel quale l'eccezione è stata generata. Ciò comporta che, una volta gestita l'interruzione, non è possibile riprendere l'esecuzione dove la si era lasciata.

Il secondo approccio si basa sulla classe AsyncronouslyInterruptedException e sull'interfaccia Interrumpible.

L'interfaccia Interrumpible contiene due metodi: il metodo run che contiene la il codice dell'esecuzione "normale" ed il metodo interruptAction che viene eseguito quando c'è l'interruzione. Tutto è gestito dalla classe AsyncronouslyInterruptedException: l'oggetto interrompibile viene passato come parametro al metodo doInterruptible che non fa altro che eseguire il metodo run dell'oggetto passatogli. Se si vuole interrompere l'esecuzione del thread che sta eseguendo questo metodo si può chiamare dall'esterno il metodo fire di AsyncronouslyInterruptedException. Questo causa l'interruzione dell'esecuzione del metodo run e fa eseguire al thread interrotto il metodo interruptAction dell'oggetto interrompiblie. Anche questo approccio ha il difetto di non permettere il ritorno nel punto del metodo run nel quale si è verificata l'interruzione.

La politica Stop

ho provato a sfruttare i meccanismi del trasferimento asincrono di controllo per creare una nuova politica di gestione dei deadline miss. Questa politica prevede che quando un thread viola la deadline l'esecuzione del job viene immediatamente interrotta.

Modifiche al modulo di busy wait

Per poter creare questa politica è stato necessario modificare il modulo di busy wait in modo da creare una busy wait interrompibile, creare un thread peridico interrompibile e un nuovo gestore di deadline miss, StopPolicyMissHandler.

Per quanto riguarda le modifiche al modulo di busy wait si è introdotto un nuovo metodo:doInterumpibleJobFor(log millis). Questo metodo è identico a doJobFor, con la differenza che, quando viene eseguito può essere interrotto asincronicamente. Ovviamente non è possibile richiamare al suo interno il metodo doJobfor dal momento che se il metodo interrupt viene chiamato mentre si sta eseguendo del codice ATC-deferred (ossia un metodo syncronized o che non dichiara di poter lanciare una AsynchronouslyInterruptedException) l'eccezione viene lanciata solo quando si esce dal metodo e si torna in un metodo interrompibile.

La classe Interrumpible PeriodicThread

La classe InterrumpiblePeriodicThread estendePeriodicThread e che modella un thread periodico la cui esecuzione può essere interrotta. Ciò è fatto eseguendo, nel metodo do job, una busy wait interrompibile al posto di una busy wait “normale”. L'azione da esegure nel caso di interruzione, inserita nella catch della AsynchronouslyInterruptedException, si limita a fare la clear dell'eccezione (necessaria per evitare che questa venga ulteriormente propagata) e a scrivere sul log che il job è stato interrotto.

La classe StopPolicyHandler

La classe stopPolicyHandler modella la politica di gestione dei deadline miss secondo la quale, quando un job viola la sua deadline, questo debba essere interrotto. Perchè ciò sia possibile occorre che il thread controllato possa ricevere un'interruzione dall'handler quando quest'ultimo viene invocato dal sistema in occasione del deadline miss. Occorre, quindi che il thread controllato da questo tipo di handler sia un InterrumpiblePeriodicThread. A tale scopo si sono modificati sia il costruttore, sia la prioprietà controlledThread al fine di supportare solo un InterrumpiblePeriodicThread e non il generico PeriodicThread.

Il metodo handleAsyncEvent, oltre a registrare sul log la violazione della deadline, interrompe il lavoro del thread invocando metodo interrupt del thread controllato, che sta eseguendo una busyWait Interrompibile. Questo causa l'interruzione del job. Occorre, infine, rischedulare il thread affinchè un eventuale nuovo job possa partire.