Ma alla fine Creazione di uno scheduler EDF per Java Realtime

Si vuole creare uno scheduler che sfrutti la strategia EDF per la schedulazione di processi real time. Tale strategia prevede che vada in esecuzione il job che ha la prossima deadline più imminente. Tale strategia è molto efficace per scenari che prevedono processi periodici indipendenti gli uni dagli altri, tuttavia in scenari di sovraccarico presta il fianco a critiche legate al fatto che non essendoci una priorità esplicita tra i processi spesso il sistema mette in esecuzione job di rilevanza secondaria, facendo sforare la deadline a processi ritenuti sì più importanti dall'utente, ma con la deadline successiva.

Introduzione e modifiche alla libreria

Il sistema funzionerà con quattro livelli di priorità:

1. Il primo, quello a priorità maggiore, sarà dedicato agli handler associati ai thread
2. Il secondo sarà quello di un job quando esegue i metodi dello scheduler all'inizio ed alla fine del job
3. Il terzo sarà quello di un job durante la sua esecuzione
4. Il quarto è quello dei job pronti che sono sospesi perché è in esecuzione un job con una deadline più imminente

lo scheduler è ideato per funzionare in un ambiente monoprocessore, tuttavia nella sua progettazione si terrà conto della possibilità di estenderlo in futuro per ambienti dotati di più di una CPU.

La classe EDFScheduler

lo scheduler è un'estensione di priorityScheduler, ciò significa che è sempre in esecuzione il processo a priorità maggiore. Lo scopo dello scheduler è porre a priorità maggiore il processo con la deadline in più imminente. A tale scopo lo scheduler tiene ordinati i job che sta gestendo per deadline assoluta.

Sul ei processi pronti ad entrare in esecuzione. Contiene anche riferimento al processo attualmente in esecuzione. Fornisce inoltre i metodi per ottenere i valori numerici delle quattro priorità.

Il nucleo dello scheduler sono i due metodi statici onJobRelease e onEndJob. Questi due metodi vengono eseguiti come preambolo e come epilogo da tutti thread real - time. Per una loro spiegazione dettagliata si rimanda alla sezione modifiche a periodicThread.

EDFSchedulingParameters

ad ogni thread gestito con politiche EDF deve essere associata un'istanza della classe EDFSchedulingParameters. Questa classe, che estende la classe PriorityParameters, ha come unico campo

un riferimento temporale assoluto alla prossima deadline. I job vengono ordinati proprio in base a questo valore.

Modifiche a PeriodicThread

Dal momento che lo scheduler in Java realtime non riceve eventi nè quando c'è una nuova release nè quando il job termina la sua esecuzione e che è a tutti gli effetti un oggetto passivo, è necessario che sia la classe thread ad eseguire un preambolo ed un epilogo per ogni job.

Si è deciso di modificare il metodo run in modo da sfruttare al meglio il polimorfismo facilitando l'introduzione futura di nuovi scheduler. In pratica il metodo run non fa altro che chiamare un metodo privato che accetta come parametro lo scheduler attuale. Sfruttando il polimorfismo si creeranno tanti metodi run quanti gli scheduler supportati, più uno per la classe Scheduler Base, che non farà altro che lanciare un'eccezione per indicare che lo scheduler attuale non è supportato.

Il metodo run per PriorityScheduler resterà quello sviluppato finora.

Per quanto riguarda il metodo relativo allo scheduler EDF all'inizio di ogni job, dopo aver scritto la creazione del job sul log, eseguirà il metodo onJobRelease della classe scheduler. Questo metodo non fa altro che calcolare la prossima deadline ed inserire il thread nella coda dei processi pronti ordinato in base alla deadline; a questo punto estranei il primo processo di questa corda, che potrebbe essere anche processo corrente, e lo mette in esecuzione attribuendogli la priorità relativa al processo correntemente in esecuzione (la numero 3). In questo modo sarà sempre in esecuzione, in assenza di vincoli di sincronizzazione, il processo con la deadline più imminente. Quando il thread termina l'esecuzione esegue come epilogo il metodo onEndJob dello scheduler. Questo metodo rei imposta la priorità del thread a quella massima, in modo che alla sua prossimo istante di release non subisca preemption da nessuno ed esegua tempestivamente il preambolo. Infine, estrae il primo processo dalla coda dei processi pronti.