Progetto Programmazione avanzata e paradigmi
Indicizzazione documenti

Luca Guerra

July 26, 2014

1 Introduzione

Si vuole sviluppare un'applicazione in grado di indicizzare in una struttura dati situata in memoria centrale le parole contenute in diversi documenti di testo. Grazie a questa struttura dati sarà possibile determinare, in maniera veloce, quali documenti contengono le parole cercate.

2 Visione del problema

Si possono individuare i seguenti blocchi di lavoro (Task) per la risoluzione del problema in esame:

- individuazione dei file da indicizzare
- indicizzazione dei file trovati
- cercare la parola cercata nei vari file

I primi due Task possono essere realizzati in parallelo, mentre il terzo dovrà attendere il termine dei primi due (per evitare di dare una risposta incompleta).

3 Soluzione al problema

Per rendere il sistema più performante, ho deciso di sfruttare una **BlockingDeque**, questa verrà utilizzata come contenitore dei vari documenti da indicizzare. Il flusso di lavoro sarà il seguente:
Allo start dell'indicizzazione avrò un processo che naviga tutto l'albero partendo dalla root, la root sarà data in ingresso dall'utente, per realizzare questo Task sarà sufficiente un unico thread, dato la navigazione rispetto l'indicizzazione sarà molto più veloce. Questo processo aggiungerà file alla lista, i vari processi indicizzatori rimarranno in attesa di nuovi file, appena questi saranno presenti li prenderanno e inizieranno a indicizzare in una hashtable. In questa

soluzione l'hashtable sarà la nostra memoria condivisa, prendendo come assunto che questa struttura dati non diventerà mai più grande della memoria centrale dell'elaboratore. Ovviamente l'accesso a questa memoria dovrà essere mantenuto mutualmente esclusivo(garantito dalla LinkedBlockingDeque). Per indicizzare nella hashtable avremo: la parola come chiave, associata a questa avremo una lista di stringhe rappresentanti tutti nomi dei documenti che contengono la parola trovata.
Terminata questa prima parte, il sistema è pronto a rispondere alle varie query consultando la hashtable (anche questo verrà fatto da un solo thread).

4 implementazione

Il progetto è stato suddiviso in 5 classi principali:

- Gui
- ManageIndexer
- FileLoader
- Indexer
- Blackboard

4.1 Gui

La classe Gui si occupa della gestione dell'interfaccia grafica, in particolare notiamo la gestione dei Listener per lo **start**, **pause** e **stop** dell'indicizzazione. Questi, per gestire il comportamento dinamico dell'interfaccia, sfrutteranno lo SwingWorker implementato dalla classe ManageIndexer.

4.2 Blackboard

La blackboard è il contenitore per tutte le variabili statiche utilizzate per la sincronizzazione e comunicazione fra i vari processi.

4.3 ManageIndexer

La classe ManageIndexer, si occuperà di creare l'environment necessario all'indicizzazione, in particolare creerà l'executor, tutti gli indicizzatori e il loader dei file. Terminata la fase di creazione metterà in esecuzione il FileLoader attraverso l'Executor, la prima operazione del FileLoader sarà quella di segnalare l'evento al ManageIndexer, il quale potrà mettere in esecuzione gli altri thread nel pool, quando il loader segnalerà il termine del caricamento, avrà inizio il tracciamento in una progress bar dell'avanzamento dell'indicizzazione (prima non era possibile dato che non avevamo il numero totale degli elementi). Una volta che l'executor avrà concluso il suo lavoro e si sarà spento, il ManageIndexer potrà terminare e dare il via alle query.

4.4 FileLoader

Questa entità sarà il processo che si occuperà del caricamento in una LinkedBlockingDeque dei File da indicizzare, la prima operazione che esegue al run è la signal della sua partenza, in questo modo consentirà il run degli indicizzatori(avendo sfruttato una FixedThreadPool il nuomero degli indicizzatori realtmente attivi mentre sarà attivo il loader saranno sempre il numero dei processi massimo meno uno, appena il loader finirà scatterà anche l'ultimo processo). Il FileLoader aggiungerà i vari file alla lista, una volta terminato metterà in lista un File particolare che verrà identificato come di fine, seguendo il "pattern" di spegnimento poison pills. Una volta inserita la "poison pill", farà una signal per consentire al ManageIndexer di iniziare con il tracciamento dello stato di avanzamento dell'indicizzazione.

4.5 Indexer

L'indexer rappresenta il processo indicizzatore del sistema, sarà questo che assieme ad altri suoi simili prenderanno dalla queue il file, lo apriranno e inseriranno le parole trovate in una hastable. Gli indexer vivono in un while(true), la sua interruzione seguirà il pattern pison pills appena un indexer trova la poison pill la reinserisce (per indicarlo anche ad evntuali altri) e rompe il while andando a fare una wait in una CycliBarrier, una votla che tutti lgi indexer avranno fatto una wait sullla barriera questa verrà rotta e i processi potranno terminare.

5 Dinamicità grafica

Il programma deve avere un'interfaccia dinamica, che consenta, mentre è in corso l'indicizzazione, di mettere in pausa far ripartire e stoppare l'operazione. Per fare ciò non possiamo far svolgere i vari task al thread evt che si occupa dell'interfaccia ma dobbiamo sfruttare uno SwingWorker, questo realizzerà le operazione in background con un'atro thread lasciando l'interfaccia grafica attiva e disponibile per gestire la pausa ecc.. In particolare, nel momento dello start partirà lo swingWorker impementato dal manageIndexer, questo a sua volta farà partire indicizzatori e loader. Mentre l'indicizzazione elabora, è possibile premere la pausa o lo stop. La pausa richiamerà il metodo pause dello swingworker che stopperà momentaneamente il thread, ovviamente non basterà perchè dovranno essere fermati anche tutti gli indicizzatori, questi leggerenna una variabile apuse nella blackboard settata a true, mettendosi in attesa in un CountDownLatch inizializzato a 1 che avrà come compito quello di farli ripartire tutti alla pressione dello start assieme al metodo resume dello SwingWorker. Lo stop semplicemente richiama un metodo cancel dello SwingWorker il quale termina il thread il quale spegne l'executor e resetta la blackboard ai valori iniziali.

6 Poison Pills

Sfruttando la LinkedBlockingDeque è venuto naturale sfruttare il patter di spegnimento **poison pills**. Quando il FileLoader ha terminato l'inseirmento, inserirà un file identificato di chiusura, questo sarà l'ultimo file preso dagli Indexer, il file farà capire all'indicizzatore che deve spegnersi, ovviamente prima di farlo dovrà reinserire la pill per indicare anche agli altri indicizzatori la fine del loro lavoro.