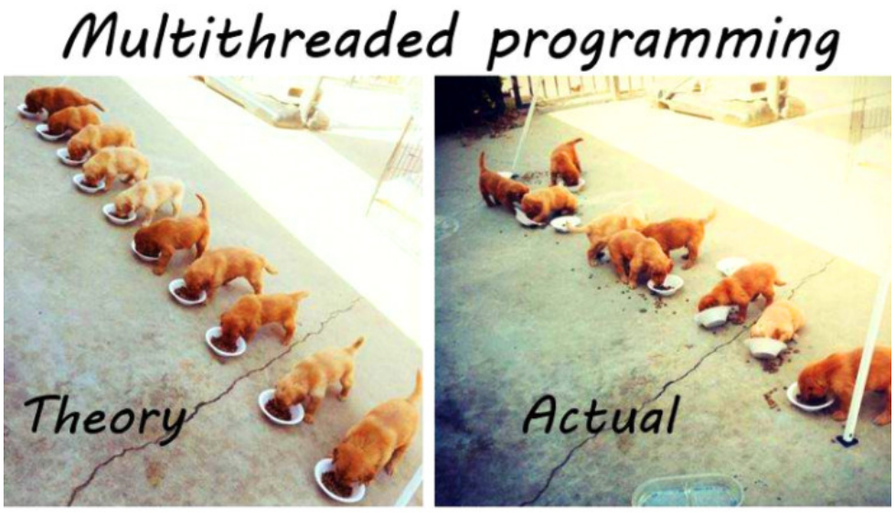
Parentesi sul multithreading

linkedin.com/pulse/parentesi-sul-multithreading-roberto-a-foglietta



Published on April 17, 2016

Introduzione

Piccola parentesi sul multi tasking e multi threading a livello divulgativo.

Multi-Tasking

Su una CPU a singolo core il multi-tasking è simulato mediante l'alternarsi dei processi e la CPU è una delle risorse per la quale sono in concorrenza i processi. Le CPU più evolute realizzano lo SMP (Symmetric multiprocessing) quindi i processi possono girare in parallelo ma devono continuare a contendersi le altre risorse quali memoria, I/O e linee di interrupt. I processi comunicano fra loro diverse modalità ma sempre ben definite, in generale si parla di IPC (Inter-Process Communication).

In un sistema operativo multi-tasking l'alternarsi dei processi è gestito dallo scheduler che è una routing del kernel. I moderni sistemi operativi sono costituiti da almeno due parti ben separate: il kernel e lo spazio applicazioni (user space). All'interno del kernel esistono dei meccanismi per gestire il multi-tasking fra questi i più importanti sono i semafori. Questi permettono ai processi di avere accesso alle risorse condivise in maniera che siano rispettati i criteri di atomicità, consistenza, isolamento e persistenza (ACID).

I processi sono separati fra loro dal kernel e tale separazione è garantita, a partire dai processori Intel 286, attraverso l'allocazione di uno spazio di memoria virtuale protetto. Ogni processo può avere al suo interno una moltitudine di thread ognuno dei quali condivide le risorse allocate al processo.

Multi-Threading

All'interno di un processo, l'alternanza dei thread e la gestione della loro concorrenza nell'accedere alle risorse condivise del processo viene implementata attraverso i semafori. Nello spazio utente non vi sono altri meccanismi di protezione se non quelli forniti dai semafori e dalle funzioni relative offerte dalle librerie specifiche per la gestione del multi-thread. I semafori nello spazio utente fanno affidamento sulle controparti nello spazio kernel così come quelli nello spazio kernel si affidano a quelli forniti dall'architettura della CPU. Il semaforo, per sua natura e funzione, esiste in uno spazio privilegiato rispetto al contesto in cui è definito.

Semafori

Se il thread A e il thread B si alternano allora potrebbe capitare che il thread A venga interrotto mentre stia scrivendo FRA[*GOLA*] dall'altro che si accinga a scrivere BANANA e poi rientri il primo thread che completi la sua scrittura ottenendo BANA|*GOLA*. Si tratta di una condizione

imprevedibile perché la prima scrittura può essere interrotta in qualsiasi momento e questo esempio, esteso in generale, comporterebbe potenzialmente il caos.

Per poter effettuare delle operazioni lunghe, ovvero diverse istruzioni Assembler, come se fosse una sequenza atomica (o si scrive FRAGOLA oppure BANANA ma non un misto frai due) i thread debbono sincronizzarsi e mutualmente escludersi attraverso i semafori. Per questo scopo non possono essere usate variabili comuni perché esse sarebbero soggette allo stesso problema della stringa FRUTTA, non sono atomiche e anche qualora lo fossero con una scelta appropriata di tipo variabile e opzioni compilatore, non sarebbe atomica il loro utilizzo per la scelta di una ramo di codice.

Quindi, un semaforo è molto di più che un valore binario atomico è la possibilità di decidere quale ramo di codice eseguire in funzione del valore interno di un semaforo. In generale si parla di istruzione test-and-set atomica.

Librerie

L'implementazione dei semafori e delle funzioni di multi-threading hanno interfacce di programmazione (API) cangianti da sistema operativo e fra le varie librerie e implementazioni disponibili. Quindi un codice applicativo multi-threading potrebbe non compilare su altri sistemi operativi a meno che non sia presente la stessa libreria oppure la stessa API esportata da una libreria che si occupi di traslitterare le API nel codice in quelle della libreria nativa.

Wrapping layer

Quindi a volte si utilizzano delle librerie di wrapping che a fronte di un set di chiamate forniscano sempre le stesse funzionalità o per lo meno un sottoinsieme di funzionalità disponibile in tutti i sistemi operativi in cui il software si suppone possa funzionare. Non è sempre una buona idea per diverse ragioni prima fra le quali il presupposto che sia più veloce partire da un progetto già validato per un altro sistema operativo piuttosto che avvantaggiarsi della nuova architettura e della nuova opportunità.

Ovviamente ogni sistema operativo multi-threading offre delle primitive. Ad esempio le GNU libpthread per Linux oppure le analoghe per Solaris oppure per VxWorks. C'è però una sensibile differenza fra alcuni sistemi operativi perché mentre GNU/Linux può essere considerato nativamente multi-task e multi-thread, altri sistemi operativi specificatamente orientati al mondo embedded, invece non hanno mai avuto bisogno della separazione offerta dalla memoria virtuale e quindi non hanno nemmeno sviluppato una sostanziale differenza fra thread e processi.

Perciò delle librerie che unificassero l'accesso alle primitive su diversi sistemi operativi sarebbero particolarmente delicate, non solo perché potrebbero contenere degli errori e fornire delle false convinzioni agli sviluppatori, ma anche perché dovrebbero creare un'uniformità di comportamento in presenza di sensibili differenze di architettura.

Verifiche

I sistemi automatici e manuali di verifica del software prevedono diversi approcci. L'analisi statica, l'analisi dinamica, l'analisi dei rami, la verifica del rilascio dei semafori presi, inizializzazione variabili, gestione dei puntatori, l'ordine con cui vengono presi e rilasciati semafori, unit test, system test, integration test, field test, etc.

Ovviamente questi sistemi daranno un responso incompleto qualora si concentrino solo sull'applicativo e tralascino di verificare anche il codice delle librerie sviluppate internamente. Ma anche qualora fossero eseguiti esaustivamente su tutto il codice interno al progetto essi hanno comunque dei limiti intrinseci e non sono in grado di evidenziare alcune tipologie di problemi come descritto in questo altro post l'illusione del controllo.

Conclusione

Un'immagine vale più di mille parole!