Sistemi multiprocessore

7-04-2021

Caratteristiche architetturali

- unità di calcolo (CPU/CPU-core) multiple che condividono una memoria principale comune
- i processori sono controllati da un unico sistema operativo
  - ↳ tightly coupled system (sistema strettamente accoppiato)

Abbiamo un unico sistema operativo che deve governare tutto ciò che succede sulle multiple CPU della nostra architettura.
L'assegnazione dei processi ai processori, in passato, non era un problema, perché il processore era unico: Sapevamo a quale processore assegnare il processo. Questo ha portato a delle problematiche.

Problematiche

- assegnazione dei processi ai processori
- uso (o non) di politiche classiche di multiprogrammazione sui singoli processori
- selezione dell'entità schedulabile da mandare in esecuzione

Assegnazione di processi/thread alle unità di calcolo

L'assegnazione statica implica dire che, quando abbiamo un processo che viene creato, noi decidiamo che questo processo può andare in esercizio su una specifica CPU: Abbiamo un problema della possibilità di sottoutilizzo dei processori.

→ *È possibile che i processi che noi avevamo assegnato originariamente alla CPU1, vengono ad essere tutti completati, mentre ad esempio sulla CPU0 ci saranno processi che dureranno tanto: questo è un classico esempio di disequilibrio delle CPU.*
Per eseguire i processi della CPU0 sfruttiamo solo quella CPU mentre magari la CPU1 è libera e non la usiamo.

L'assegnazione dinamica è l'approccio che abbiamo tipicamente nei sistemi moderni. Esso ha un overhead superiore dovuto al fatto che quando noi abbiamo deciso di assegnare originariamente un processo P alla CPU0, possiamo decidere dinamicamente nel tempo di assegnare questo processo alla CPU1 più avanti nel tempo.

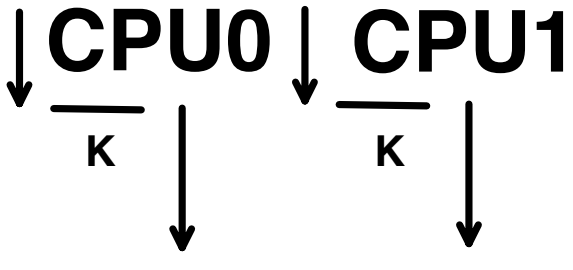
→ **Overhead, in informatica, è utilizzato per definire le risorse accessorie, richieste in sovrappiù rispetto a quelle strettamente necessarie per ottenere un determinato scopo in seguito all'introduzione di un metodo o di un processo più evoluto o più generale.**

In un approccio MASTER-SLAVE abbiamo che il sistema operativo e quindi anche il kernel, viene ad essere eseguito solo su una specifica CPU, quindi su una specifica unità di calcolo, e ogni volta che un'applicazione è in esercizio su un altra CPU e vuole eseguire una system call, ci deve essere una richiesta esplicita a strutture del kernel da parte delle altre CPU.
Se abbiamo una CPU0 e una CPU1, magari soltanto la CPU0 ha la possibilità di toccare le strutture dati del kernel, ma se sulla CPU1 è in esercizio un'applicazione che ha necessità di andare ad eseguire una sys_call, la CPU1 deve in qualche modo mandare una richiesta verso la CPU0 di poter eseguire specifiche attività di livello kernel.

→ **Il kernel è eseguito dalla CPU0, quindi da un'unica CPU, però c'è un problema di soddisfare le richieste di CPU qualora ve ne siano molte che effettuino appunto questa richiesta alla CPU0. Problema di scalabilità.**

Nell'approccio peer, che è un approccio che viene eseguito attualmente, il sistema operativo viene eseguito su tutte le unità di calcolo.

→ **Concorrentemente più flussi d'esecuzione, allo stesso istante di tempo, sono in esercizio all'interno delle strutture del kernel. Questo ha una complessità di progetto nettamente superiore.**



Sui sistemi operativi moderni ci siamo mossi con un approccio di assegnazione dei processi ai core di tipo Dinamico, e un approccio peer per la possibilità di eseguire software del sistema operativo e quindi anche del kernel all'interno della nostra architettura.