Process scheduler
Linux Kernel

Sistemi Operativi

A.A. 2020-2021

Process scheduling (o task scheduling)

- Se ci sono **processi eseguibili** in un sistema, almeno un processo deve essere sempre in esecuzione. Se ci sono più processi eseguibili rispetto al numero di processori in un sistema, non tutti i processi possono essere sempre in esecuzione.

- Pertanto, alcuni processi devono essere temporaneamente arrestati o sospesi, in modo che altri possano essere eseguiti nuovamente. Lo schedulatore dei processi decide quale processo nella coda verrà eseguito successivamente.

Sistema operativo multitasking

- Linux, come tutte le altre varianti di Unix, è un sistema operativo **multitasking**: ciò significa che è possibile eseguire più task (o processi) contemporaneamente.

- Linux fornisce un cosiddetto multitasking preventivo (preemptive), in cui lo schedulatore decide quando un processo viene sospeso. Questa sospensione forzata si chiama «preemption».
- Tutta la famiglia di OS Unix ha fornito il multitasking preventivo sin dall'inizio del suo sviluppo.

Virtualizzazione di CPU e memoria

I sistemi operativi più moderni sono progettati per cercare di ottenere **prestazioni ottimali** dalle risorse hardware.

Questo risultato è ottenuto principalmente dalla **virtualizzazione** delle due principali risorse hardware: CPU e memoria (che sono risorse finite).

Virtualizzazione della CPU

- I moderni sistemi operativi forniscono un ambiente multitasking che assegna ad ogni processo la propria CPU virtuale
- Il processo non è «consapevole» del fatto che non ha un uso esclusivo della CPU
- La **virtualizzazione della CPU** si ottiene "condividendo" la CPU tra più processi, ovvero ogni processo in esecuzione riceve una piccola frazione della CPU a intervalli regolari.

https://www.linuxjournal.com/node/10267

Kernel Linux - schedulazione dei processi

Lo schedulatore dei processi (**process scheduler**) è il **sottosistema del kernel** che suddivide la risorsa finita del tempo del processore tra i processi di un sistema

https://elixir.bootlin.com/linux/latest/source/kernel/sched

Quale processo eseguire?

lo schedulatore dei processi seleziona quale processo eseguire successivamente (al processo corrente)

Quando eseguire?

Lo schedulatore dei processi decide **quali** processi possono essere eseguiti e **quando** (massimizzando l'utilizzo del processore)

Esecuzione contemporanea (multitasking)

Il sistema operativo fornisce l'illusione che più processi vengano eseguiti contemporaneamente, senza interruzioni e senza ritardi

Preemptive process scheduler

Lo schedulatore dei processi del Kernel Linux è di tipo «preemptive» (preventivo o anticipante) ovvero:

lo schedulatore dei processi decide quando interrompere un processo e riprendere un processo diverso

l'azione di sospendere un processo in corso al posto di un altro si chiama «**preemption**»

timeslice

timeslice del processo:

un breve intervallo di tempo durante il quale il processore esegue senza interruzioni un particolare processo (prima di passare a un altro processo)

lo scheduler assegna al processo una "porzione" di tempo (timeslice) del processore

timeslice: impatto sul sistema

Se la timeslice è troppo grande, il singolo processo deve attendere a lungo prima di poter riprendere l'esecuzione, rovinando l'illusione dell'esecuzione simultanea di processi (l'utente percepisce ritardo nell'uso dei processi)

se la timeslice è troppo piccola, una parte significativa del tempo di CPU viene sprecata per passare da un processo all'altro (context switch o process switch)

Classificazione dei processi

- I processi possono essere classificati come legati all'Ingresso/Uscita (**I/O bound**) o legati al processore (**processor bound**)
- Con I/O si intendono dispositivi di input / output, come tastiere, mouse o dischi ottici e fissi
- I processi di tipo **«I/O bound»** impiegano la maggior parte del tempo a inviare e attendere richieste di I/O. Di conseguenza, tali processi sono eseguibili solo per brevi periodi, perché si bloccano in attesa di I/O. Questi processi vengono eseguiti spesso e brevemente.

Classificazione dei processi

- I processi di tipo «**processor bound**» usano il loro tempo di CPU per eseguire istruzioni o algoritmi e di solito vengono eseguiti fino a quando non vengono anticipati (preempted) dallo scheduler.
- Questi processi non si bloccano in attesa di richieste di I/O e possono essere eseguiti con minore frequenza ma per intervalli di tempo più lunghi.

Classificazione dei processi (esempi)

- La maggior parte delle applicazioni di interfaccia utente grafica (GUI), ad esempio, sono «**I/O bound**», anche se non leggono o scrivono mai sul disco, perché trascorrono la maggior parte del loro tempo ad aspettare l'interazione dell'utente tramite tastiera e mouse.

- Matlab, programmi che fanno calcoli scientifici sono «**processor bound**» e non hanno bisogno di mettersi in attesa dei risultati a richieste di operazioni di I/O

Classificazione dei processi (esempi)

- Ci sono processi che possono mostrare entrambi i comportamenti contemporaneamente: il server X Window (sistema a finestre), ad esempio, è sia processor bound che I/O bound.

- Word processor: di solito in attesa di I/O (quindi I/O bound) ma con periodi di «processor bound» quando si attiva il controllo ortografico

Obiettivi del process scheduler

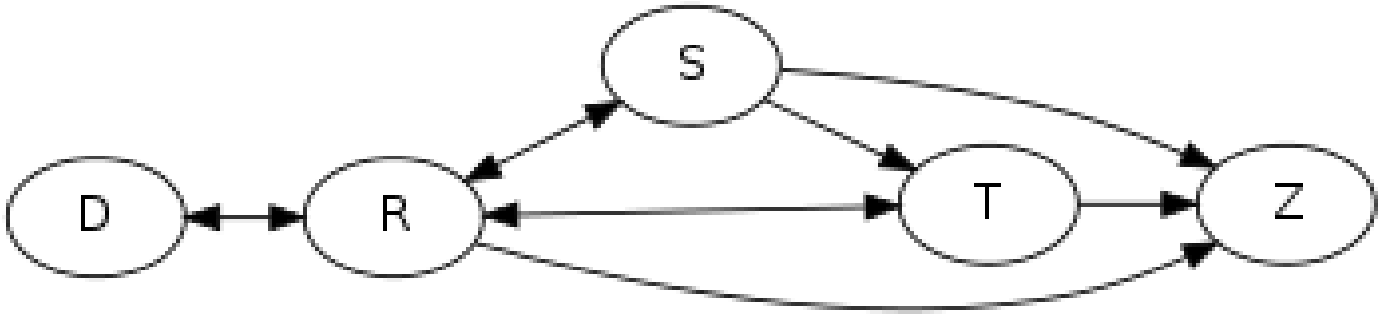
La politica dello schedulatore di processi in un OS deve tentare di soddisfare due obiettivi contrastanti:

- tempi di risposta del processo rapidi (bassa latenza)
- utilizzo massimo del sistema (throughput elevato, «sprechi» minimi)

Linux Kernel - process scheduler

Dalla versione 2.6.23 di Linux Kernel, viene utilizzato lo scheduler chiamato **Completely Fair Scheduler** (**CFS**)

CFS usa un algoritmo chiamato «weighted fair queueing»



A Linux process can be in one of the following states:

D uninterruptible sleep (device drivers, waiting for IO) cannot be woken up by signals

I Idle kernel thread

R running or runnable (on run queue)

S interruptible sleep (waiting for an event to complete), can be woken up by signals

T stopped by job control signal (ctrl-Z or SIGSTOP signal)

t stopped by debugger during the tracing

X dead (should never be seen)

Z defunct ("zombie") process, terminated but not reaped by its parent

Riferimenti

- http://www.ece.ubc.ca/~sasha/papers/eurosys16-final29.pdf
- https://doc.opensuse.org/documentation/leap/archive/42.1/tuning/html/book.sle.tuning/cha.tuning.taskscheduler.html
- https://lwn.net/Articles/254711/
- https://www.linuxjournal.com/node/10267
- https://developer.ibm.com/tutorials/l-completely-fair-scheduler/
- https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt