Scheduling CPU

- Slide
- Indice

`24/03/22`

dal momento che in memoria può risiedere più di un processo dobbiamo scegliere a quale dare la cpu: scheduling

Scheduling dei processi

Scheduling dei processi
Scheduling: assegnazione di attività nel tempo

- attività di cpu
- processi quando esistono vengono ammessi nel sistema con scheduler a lungo termine (job sceduler)
  - determina grado di multiprogrammazione
- tra tutti questi ne deve essere scelto uno: scheduler a breve termine
  - alloca a uno cpu
- quando sono in ram hanno diversi stati e code
  1. ready queue
     1. scheduler a lungo termine
     2. n processi che puo aumentare o diminuire in base a risorse
     3. grado di multiprgrammabilità in base al n
     4. sistema dinamico
  2. 1 solo processo in cpu
     1. scheduler a breve termine
  3. n processi in attesa
- schema slide 5
- ...
- molte code
  - implementazione code slide 6
    - liste doppiamente linkate
    - puntatori a PCB

Tipi di scheduling

Tipi di scheduling

- scheduler a lungo termine
- scheduler a breve termine
- scheduler di medio termine

scheduler a lungo termine

scheduler a lungo termine (o job scheduler)

- determina quali processi devono essere portati da memoria a ready queue
- cetermina quindi grado di multiprogrammazione
- può essere lento (secondi)
- meno critico rispetto a scheduler a breve termine ma
- importante che faccia scelta giusta
  - selezione è parte critica
  - in base a selezione processi impatto su performance e uso delle risorse
- deve mettere in coda dei processi pronti una buona combinazione di processi
  - I/O bound
    - spende grand parte dell'esecuzione in operazione I/O
    - burst brevi ma frequenti
  - CPU bound
    - spende grand parte dell'esecuzione in operazione CPU
    - pochi burst ma lunghi
- bilanciamento perché sennò ho uno o l'altro sempre in attesa
  - devo massimizzare uso di tutte le risorse
- so riesce a capire le differenze tra le due in base allo storico per dedurre e fare assunzioni, assegna se è uno o l'altro e fa osservazioni e eventualmente aggiusta assunzione iniziale

- può essere assente
  - con sistemi a risorse limitate
  - politica di scheduler implementate hardcorer
    - ex politiche fifo
  - so special purplose
    - qui è lo scheduler a essere cambiato
  - num di processi limitato

scheduler a breve termine

scheduler a breve termine (o CPU scheduler)

- selezione di quale processo deve essere eseguito da cpu
- quale allocare a cpu
- deve essere molto efficiente (ns)
- invocato spesso

scheduler di medio termine

scheduler di medio termine

- rimozione forzata momentanea di un processo dalla cpu
- con aumentare ram aumentata anche complessità processi
- esigenza di avere anche piu processi in esec
- so aumentano dim di ram con backing store
  - parte di disco trattato logicamente come parte di ram
  - estensione della ram
  - virtual ram: ram fisica + backing store
- ora posso mettere processi anche in backing store
  - estendo dim ram
- dal punto di vista fisico però cpu comunica solo con ram fisica
  - quando scheduler a breve termine seleziona processo questo deve essere in cpu fisica
    - se è in backing store devo portarlo in quella fisica
    - se c'è posto
      - swap in
    - se la mem fisica è gia piena
      - swap out prima
        - togliere processo non in eseccuzione dalla ram fisica e metterlo in backing store
        - entra in gioco medium shceduler
          - che sceglie quale processo protare in backing store da ram fisica
            - cioe swap out
          - ora possono fare swap in
          - slide 11

Scheduler è un modulo
Dispatcher è un altro modulo

Dispatcher

Dispatcher

- passa controllo cpu a processo scelto da scheduler
  - cambia contesto
  - passaggio mod user
  - salto a giusta locazione nel prog per farlo ripartire
- latenza di dispatcher
  - dal momento della selezione del proc a cpu a quanto prima istruzione del proc parte

Modello astratto del sistema

Modello astratto del sistema

- modello di esecuzione di un processo è alternanza tra cpu burst e i/o burst
- modello a cicli di burst

Distribuzione dei CPU burst

Distribuzione dei CPU burst

- statisticamente distribuzione esponenziale
  - pochi burst molto lunghi
  - molti burst brevi e invocati molto di frequente
- alg di scheduling con burst lungi di base non avranno grandi performance
- infatti tendono a fare prec a burst brevi

Prelazione

Prelazione

- algoritmi non-preemptive
  - quando cpu allocato a processo:
  - la usa quando li serve per finire cpu burst
    - (non per completare esec)
  - non puo succedere che proc torna dirett in ready queue
- algoritmi preemptive
  - rilascio
  - scheduler puo forzare proc che sta usando cpu a lasciare cpu prima che finisca il suo cpu burst
  - proc puo tornare dirett nella ready queue
- slide 17
- slide 18

Valutazione algoritmi di scheduling

Valutazione algoritmi di scheduling

- varie metriche con obiettivi che possono anche essere in conflitto tra loro
  - trovare sol di compromesso o scelta obiettivi prioritari

obiettivi:

- utilizzo cpu
  - maggior è migliore è l'alg
- throughput
  - numero di proc completati per unita di tempo
  - maggior è migliore è l'alg
- tempo di attesa tw
  - tempo speso da un proc in code di attesa
  - minore è migliore è l'alg
- tempo di completamento (turnaround, tt)
  - tempo per eseguire proc da quando viene creato a quando termina
  - è il tempo di completamento medio per un insieme di processi
  - tempo di esec + tempo di attesa
- tempo di risposta (response time, tr)
  - tempo medio di risposta
  - tempo da quando creo processo a quando ottengo prima risposta
    - cioe ottengo la cpu e ho primo feedback
  - ex per sistemi interattivi è importante che sia basso
  - spesso non sono ottimali su tempo di completamento o throgput

`25/03/22`

Criteri di ottimizzazione

Criteri di ottimizzazione

- massimizzare
  - uso cpu
  - througput
- minimizzare
  - tempo di attesa
  - tempo di completamento
  - tempo di risposta

Agoritmi di Scheduling

ALGORITMI DI SCHEDULING

- First-Come, First-Serve (FCFS)
- Shortest-Job-First (SJF)
- Higher Response Ratio Next (HRRN)
- Round Robin (RR)

First-Come, First-Serve (FCFS)

First-Come, First-Serve (FCFS)

- facile da implementare
- ma non è la scelta implementativa usata di solito
- coda processi pronti gestita in FIFO
  - semplice
- non preemptive
- questo algoritmo ha un problema
  - slide 25
  - c'è un ordine di arrivo diverso
  - calcoli..
  - ora il tempo medio è = 1
- quindi è troppo legato a arrivo di processi (che è imprevedibile)
  - quindi performance sono troppo variabili
  - se il primo che arriva a cpu grande allora ho "effetto convoglio"
    - tutti i proc dopo sono ritardati da questo anche se hanno cpu burst piccolo
- funzionamento dell'alg: Esempio FCFS

Esempio FCFS

Esempio FCFS
esempio slide 24

- tempo di risposta
  - t1: 0
  - t2: entra a 24 e 2 tempo di arrivo -> 24 - 2 = 22
  - t3: entra a 27 e 4 tempo di arrivo -> 27 - 4 = 23
- tempo di attesa (tempo speso della ready queue)
  - in caso di alg non preemptive è uguale al tempo di risposta
  - quindi stessi dati
- tempo di turnaround
  - t1: finisce a 24 entra a 0 -> 24 - 0 = 24
  - t2: finisce a 27 entra a 2 -> 27 - 2 = 25
  - t3: finisce a 30 entra a 4 -> 30 - 4 = 26
- tempo di attesa media: (0+22+23)/3 = 15
- oss:
  - ovviamente è una semplificazione dell'alg (non è un sistema reale)
  - non è importante unità di misura (è una valutazione analitica)

Shortest-Job-First (SJF)

Shortest-Job-First (SJF)

- sceglie sempre processo di attesa che cpu burst piu breve
  - buona strategia (vedi ...)
- puo essere sia preemtive che non
  - nel caso di preemptive però
    - se arriva nuovo processo lo scheduler deve fare valutazione:
    - se proc che entra ha cpu burst piu piccolo del tempo che rimande del cpu burst di quello che è dentro (cpu burst che era entrato che però intanto ne ha consumato un po)
    - allora deve essere rimosso quello che c'era e sostiuito
      - in questo caso l'algoritmo si chiama Shortest-RemainingTime-First (SRTF)
- è algoritmo ottimo:

    - è quello che assicura minimo tempo medio di attesa
- funzionamento: Esempio SJF
- Come fa scheduler a sapere che deve fare la selezione
- Come stabilire qual è il cpu burst
- Problema:
    - se arrivano sempre processo con cpu burst piu brevi di uno che è in ready queue che burst piu lungo
        - questo non verrà mai eseguito
    - starvation
    - problema di tutti gli alg di scheduling a priorità

Esempio SJF

Esempio SJF
caso non preemptive: esempio slide 29

- guardo solo cpu burst
- se è lo stesso, il secondo criterio puo essere tempo di attesa
    - scelgo quello che è entrato prima
- importante scegliere qual è il secondo criterio (specificarlo all'esame)
- poi calcolo tempi di attesa e completamento sono gli stessi

caso preemptive: esempio slide 30

- funzionamento
    - quando arriva p2, p1 ha remaining time di 5
        - confronto con p2 (4 < 5) -> entra p2
    - poi arriva p3
        - confronto con p2 (e p1) (1 < 2 and 5) -> entra p3
    - poi arriva p4, p3 ha finito
        - confronto con p1 e p2
        - (2 < 4 and 5) -> entra p2
    - p2 finice, confronto p1 e p4 (4 < 5) -> entra p4
    - finisce p4, finisce p5
- tempo di risposta
    - p1 : 0
    - p2: entra a 2 e ha tempo di arrivo 2 -> 2 - 2 = 0
    - p3: entra a 4 e ha tempo di arrivo 4 -> 4 - 4 = 0
    - p4: entra a 7 e ha tempo di arrivo 5 -> 7 - 5 = 2
- tempo di attesa
    - p1: entra con ta = 0, finisce a 2 e poi entra a 11 -> 11 - 2 = 9
    - p2: entra con ta = 2, finisce a 4 e poi entra a 5 -> 5 - 4 = 1
    - p3: entra con ta = 4, prende a 4 -> 0
    - p4: entra con ta = 5, prende a 7 -> 7 - 5 = 2
- tempo di completamento
    - basta fare tempo di attesa + cpu burst
- semplificazioni
    - non viene tenuto conto del content switch
    - del dispatcher ...
    - eventuale swap in
    - non sto considerando tutto questo overhead

Come fa scheduler a sapere che deve fare la selezione

Come fa scheduler a sapere che deve fare la selezione?

- quando proc viene messo in ready queue c'è un interrupt che viene catturato da so
- e vede che deve essere servita da scheduler

Come stabilire qual è il cpu burst

Come stabilire qual è il cpu burst?

- si fa una stima
- usa le lunghezze dei burst precedenti proiezione di quelli futuri
    - con funzioni matematiche
    - stima esponenziale
        - guardo quello che è successo in passato e stimo che in futuro il comportamento sia simile

        - in più controllo l'ultimo burst che ho osservato
            - per capire se la storia sta cambiando o è la stessa
    - formula slide 31
        - tau n + 1 è il valore che devo stimare
        - formula: (ultima riga)
        - piu la storia è lontana meno è significativa
        - alfa dice che peso dare al burst che ho appena osservato e che peso dare alle vecchie misure
        - casi limiti:
            - alfa = 0: solo la storia vecchia e non quello appena misurato
            - alfa = 1: elimino storia vecchia e uso solo quello di recente
    - slide 32
    - slide 33

Algoritmi di scheduling con priorità

Algoritmi di scheduling con priorità

- associata priorità a processi
    - in Shortest-Job-First priorità è inverso del cpu burst
- generalmente cpu allocata a priorità piu alta
- calcolo priorita dipende
- possono essere preemtive e non
- politiche assegnamento priorità
    - calcolo in base a caratt interne al so
        - legate a funz so
        - ex requisiti di memoria, numero file aperti, cpu burst...
    - in base a fattori esterni a so
        - soldi pagati per uso pc
        - differenza di tipo utente
        - non tecniche
- esempio slide 36
    - a 0 ho p2 e p4
        - guardo priorità -> p2
    - a 1 ho p4 (da prima), p1, p5
        - guardo priorità -> p5
    - poi gli altri cosi ...
- tutti gli alg di scheduling basati su priprità soffrono la starvation
    - so deve evitarlo
    - ponendo dei correttivi per evitarlo
        - basato sul tempo ad esempio: aging
            - tengo conto dell'età del processo

Higher Response Ratio Next (HRRN)

Higher Response Ratio Next (HRRN)

- algoritmo a priorità con meccanismo di aging
    - no starvation
- calcolo priorità adattivo, dinamico e sofisticato
- non preemtive
- priorità = (tempo di attesa + tempo burst) / tempo di burst = 1 + tempo attesa / tempo di burst
    - maggiore per priorità più alte
- privilegi proc con cpu burst piccoli o da tanto di attesa
- come viene ricacolata la priorità: 2 possibilità
    - quando il processo termina (cioe lascia cpu) ricalcolo cioè al termine di ogni processo
    - quando il processo termina (cioe lascia cpu) ricalcolo solo se sono entrati nuovi processi nel frattempo
        - altrimenti rimane la stessa
    - quelli che ricalcolo sempre sono quelli con temi di attesa piu lunghi
- se la ricalcolo sempre do più peso all'attesa
- altrimenti do più peso al burst
- esempio slide 40
    - specificare quale delle 2 opzioni si usa per il ricalcolo
    - nell'esempio priorità ricalcolata sempre
    - ...

Round Robin (RR)

Round Robin (RR)

- preemptive
- comportamento stabile
- ottimo per sistemi interattivi
- tempo di cpu distribuito in interavalli: quanti
- scheduler assegna cpu per quato: tempo predefinito
  - indipendendentemente da cpu burst
- proc puo usare cpu per un tempo max di un quanto
- puo essere rilasciata prima se proc non ne ha bisogno (quando va in attesa)
- coda dei proc posti è gestita in fifo (circolare)
- quanto ha valori che variano da 10 a 100 millisecondi
- è deterministico
  - se so dimensione dei proc pronti so quanto devo aspettare
- scelta del quanto è critica e determina performance
  - se quanto è troppo grande diventa un FCFS
  - se quanto troppo piccolo problema di overhead context swtch
- deve fare si che 80% dei burs cpu siano minori del quanto
  - aggiustato in modo addattativo
- è come FCFS con prelazione
- prestazioni
  - tempi di turnaround >= di SJF
  - tempi di risposta < di SJF
- esempio slide 43
- slide 44
  - relazione quanto - content switch
- slide 45
  - relazione quanto - turnaround
    - turnaround non descresce all'aumentare del quanto

`31/03/22`

In realtà non viene usato nessuno di questi algoritmo così specificato
Vengono implementate soluzioni più complesse

Code multilivello

Code multilivello
Coda dei processi pronti gestita come code a multilivello (non come coda unica)

- riesco a differenziare processi
  - coda per processi in foreground (primi da servire, interattivi)
  - coda per processi in background (batch)
- posso avere diversi alg di scheduling per code diverse
- meccanismo più complesso con code
  - si aggiunge problema di scelta di coda
  - necessario scheduling di code
    - a priorità fissa
      - ma se servo sempre prima i foreground potrei avere starvation e non servire mai background
      - problema di starvation
    - a time slice
      - ogni coda ottiene tempo di cpu che può usare per scheduling suoi processi
      - passa da proc foreground a proc di background dopo il tempo
- slide 48

Code multilivello con feedback (adattative)

Code multilivello con feedback (adattative)

- in implementazione reale non ho mai code multilivello statiche
  - anche la priorità è adattativa e dinamica (non statica)
  - proc può spostarsi da una coda all'altra in base al comportamento
  - anche per aging
  - o proc che scalano a code di priorità (aumentando o diminuendo)
- paramentri e scelte

- numero di code gestire
- alg. per ogni coda
- criteri per spostarsi a code
- criteri per scegliere coda a cui mandare nuovo proc ammesso

- esempio slide 51 - 52 <-
  - gestisco meglio cpu burst piccoli (che sono molti) rispetto a quelli lunghi (rari)
  - adattamento dinamico
- prob di fair share

Scheduling Fair Share

Scheduling Fair Share

- potrei avere scheduling non molto fair in termini di utenti
- allocazione si sbilancia a programmi con molti processi e thread rispetto a programmi semplici con pochi processi
- con fair share scheduling si risolve
  - si fa divisione tra gruppi di processi e non tra tutti i processi
  - divisione cpu tra utenti

Contesto reale scheduling cpu

Contesto reale scheduling cpu
esempio: CPU scheduling in Solaris

- slide 54
- obiettivo: minimizzare complessità scheduling
- usa Round Robin (RR)
- priorità con aging
  - priorità = priorità di base + priorità corrente
    - numero negativo = priorità pi alta
  - ricalcolata
  - priorità di base
    - tra -20 e +20
    - -20 è il max,+20 il min
  - priorità corrente = 0,1 *cpu (5* n)
    - cpu(t) : uso cpu negli ultimi t sec
    - n : numero medio di proc pronti nell'ultimo sec
- se ho usato cpu la priorità aumenta quindi sono penalizzato
- chi non usa cpu è favorito (non viene implementata priorità corrente)

Valutazione degli algoritmi di scheduling cpu

Valutazione degli algoritmi di scheduling cpu

- Modello deterministico
- Modello a reti di code
- Simulazione algoritmo
- Implementazione algoritmo

Modello deterministico

Modello deterministico

- come abbiamo fatto noi
- si descrive l'alg e si calcola assumendo un calcolo di lavoro preciso
  - che puo anche non essere realistico
  - prestazione su quel carico specifico
- usando stesso carico di lavoro
  - capire come funzione
  - e confronto con altri alg
- analitico
- usato per rappresentare alg
- problemi
  - nella realtà verificare se distribuzioni di carico sono realistiche

    - io faccio valutazione su un carico, se lo cambio come si comporta l'alg ?
- non utile per confrontaro

Modello a reti di code

Modello a reti di code

- ho distribuzione statische dei cpu burst
    - non ho solo un caso
- distribuzione
    - piccoli e frequenti cpu burst
- quindi ora riesco a capire come sistema si comporta nel caso medio
- riesco anche a confrontare algoritmi tra di loro nel caso generale
- distribuzioni teoriche fatti da modelli statistici

Simulazione algoritmo

Simulazione algoritmo

- passo successivo del Modello a reti di code
- usare dati realistici e non modelli teorici
    - colleziono dati su carichi di lavoro veri
    - e gli uso per simulare comportamento dell'alg
- valutazioni reali e precisa
- se ho valutazioni che mi soddisfano passo a Implementazione algoritmo

Implementazione algoritmo

Implementazione algoritmo

- unico modo sicuro di valutare alg di scheduling
- test su come si comporta prima di inserirlo in un so vero

DA SLIDE 60 ESERCITAZIONE !!
Esercizi <-

- se sono non preemtive
    - tr = ta
- poi sempre tt = ta + cpu burst
- sempre specificare le scelte fatte
    - ex con rr a quanto = 1 possibile che si debba scegliere tra due processi in coda dei processi pronti -> ex specificare che si usa il primo proc che è arrivato
-