Scheduling della CPU

- Sistemi multiprocessori e real time
- Metodi di valutazione
- Esempi:
  - Solaris 2
  - Windows 2000
  - Linux

Sistemi multiprocessori

- Fin qui si sono trattati i problemi di scheduling su singola CPU; se sono disponibili più unità di elaborazione, il problema diventa proporzionalmente più complesso.
- Se sono disponibili più unità di elaborazione identiche, e non sono presenti unità di I/O collegate a un bus privato di una delle CPU, si usa un'unica ready queue.
  - ☞ Ciascuna unità di elaborazione esamina la ready queue
  - ☞ Una CPU ha in carico lo scheduling di tutte le altre.
- In alcuni sistemi una CPU ha il compito di effettuare non solo lo scheduling, ma anche l'elaborazione dell'I/O e le altre attività di sistema (*multielaborazione asimmetrica)*

Sistemi di elaborazione in tempo reale

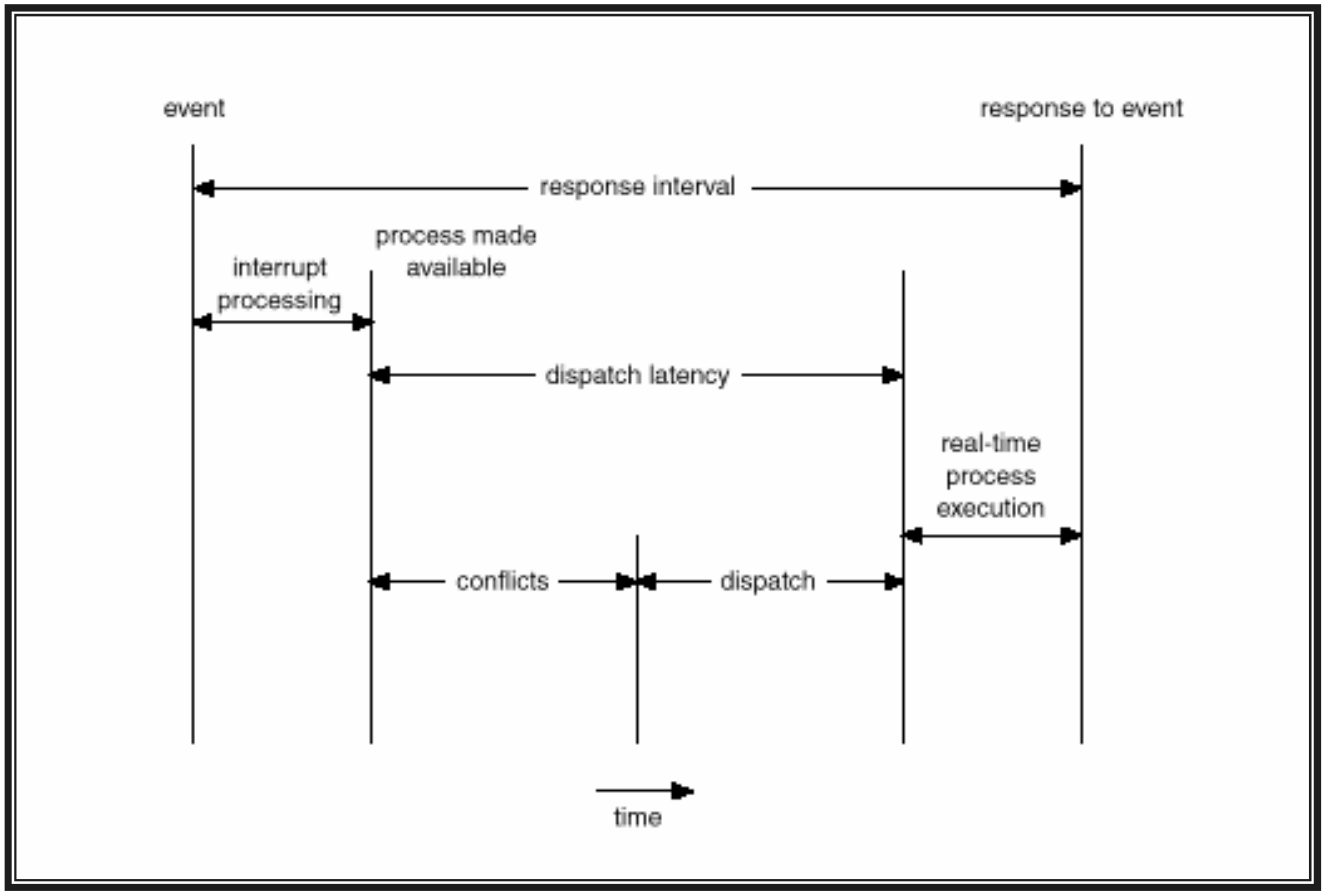
- Con il termine **tempo reale stretto** (*hard real time*) si intendono quei sistemi in grado di completare un'operazione critica in un tempo definito.
- **Prenotazione delle risorse**: un processo è accompagnato da una dichiarazione di tempo entro cui completare l'operazione; se è possibile lo scheduler accetta, altrimenti rifiuta la richiesta.
- Lo scheduler deve sapere quanto dura ciascuna operazione; questa richiesta non può essere garantita nei sistemi con memoria virtuale o con memoria secondaria.
- E' chiaro che siffatti sistemi non possono offrire tutte le funzionalità dei moderni sistemi operativi.

Sistemi di elaborazione in tempo reale

- Nelle elaborazioni in **tempo reale debole** (*soft real time*) ci si limita a richiedere che i processi critici abbiano una priorità maggiore dei processi ordinari.
- Il criterio di assegnazione delle risorse risulta **iniquo**, che ritarda l'esecuzione di alcuni processi e può provocare situazioni di attesa indefinita.
- Tali sistemi sono d'uso generale e capaci di offrire, oltre alle funzioni tradizionali, un ambiente per l'esecuzione di applicazioni multimediali e per la grafica interattiva ad alte prestazioni.
- Il sistema deve disporre di uno scheduler per priorità, in cui la priorità dei processi in tempo reale non diminuisce col passare del tempo; l'allocatore deve avere una bassa latenza.
- Per mantenere bassa la latenza, le chiamate di sistema devono essere soggette a prelazione (punti di prelazione o nucleo con possibilità di prelazione).

Latenza di Dispatch

- La **fase conflittuale** della latenza di dispatch consiste di due parti:
  - prelazione di tutti i processi correntemente in esecuzione nel nucleo,
  - rilascio delle risorse richieste dai processi ad alta priorità da parte di quei processi a priorità bassa.



Valutazione degli algoritmi

- Per effettuare la scelta di un algoritmo di scheduling in un sistema specifico bisogna utilizzare dei criteri di scelta:
  - Utilizzo della CPU
  - Produttività
  - Tempo di completamento
  - Tempo di attesa
  - Tempo di risposta
- Stabilita l'importanza relativa di questi parametri, va scelta una misura, come ad esempio:
  - rendere massimo l'utilizzo della CPU con un tempo massimo di risposta di 1s
  - rendere massima la produttività con un tempo medio di latenza lineare con il tempo di esecuzione totale.

Modelli deterministici

- La **valutazione analitica** di un algoritmo consiste nella determinazione delle prestazioni di un determinato algoritmo a partire da un determinato carico.
- Dato il carico di lavoro del sistema si valuta la prestazione mediante una formula.
- I modelli deterministici sono di tipo analitico.
- La definizione di un modello deterministico permette in maniera semplice e veloce di valutare e confrontare algoritmi differenti.
- Il risultato ottenuto è dipendente dal carico di lavoro scelto: i modelli deterministici sono troppo semplificati e per essere utili hanno bisogno di conoscenze dettagliate

Reti di code

- Se i processi variano nel tempo, non esiste un insieme di dati da usare per un modello deterministico; si possono però stimare le distribuzioni delle sequenze di operazioni di CPU e di I/O.
- Il sistema di calcolo viene descritto come una *rete di unità serventi*, ciascuna con una coda d'attesa; la CPU è un'unità servente con una ready queue, il sistema di I/O ha le sue code dei dispositivi.
- A partire da una stima della probabilità di una determinata sequenza di operazioni e degli istanti di arrivo dei processi, si può analizzare il comportamento di un sistema, rispetto a
  - la lunghezza media delle code,
  - tempo medio di attesa, …

La formula di Little

- Sia *n* la lunghezza media di una coda, detti *W* il tempo d'attesa medio nella coda e λ l'andamento medio d'arrivo dei nuovi processi, nel tempo *W* in cui un processo attende nella coda, λ x *W* nuovi processi arriveranno nella coda.
- Nell'ipotesi che il numero dei processi che lasciano la coda è uguale al numero dei processi che arrivano,

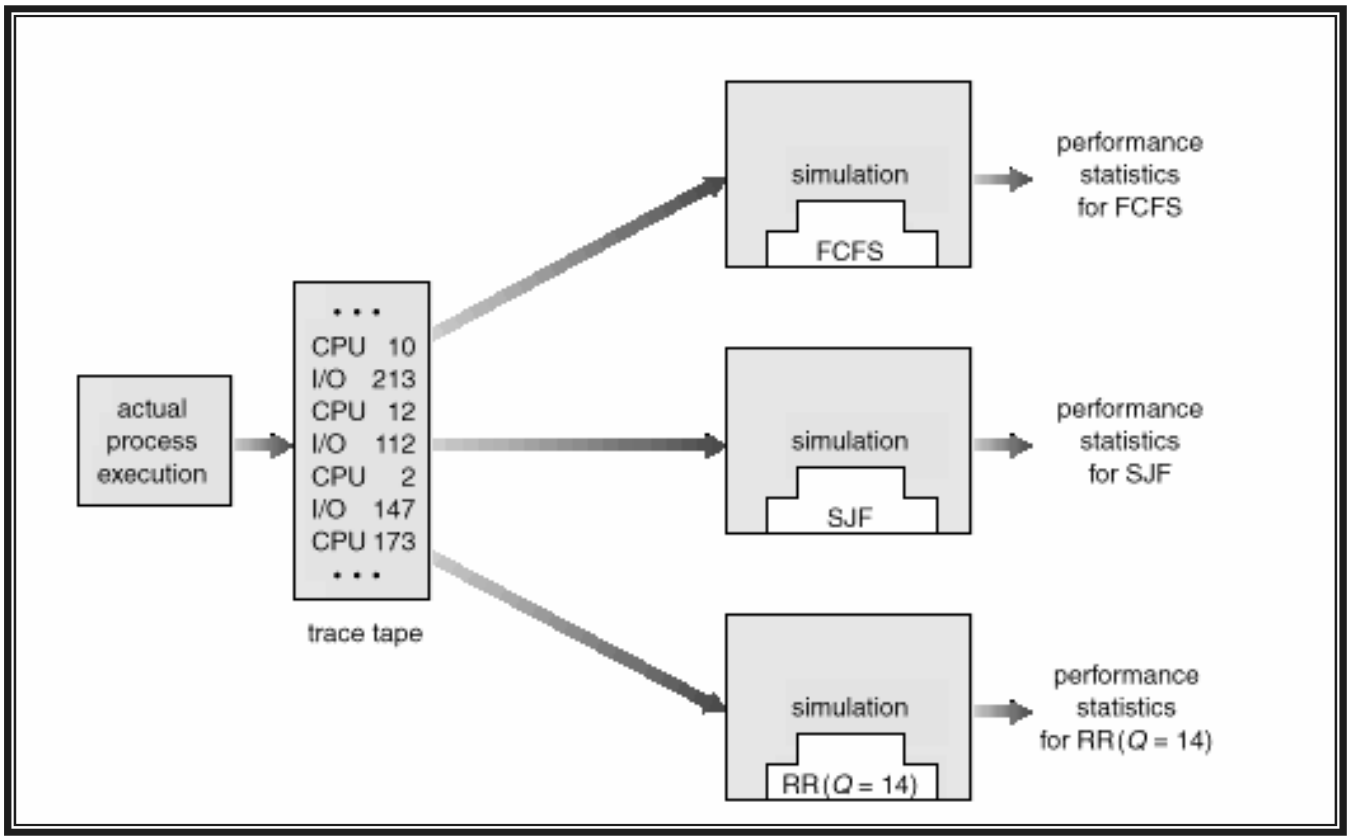
$$\boldsymbol{n = \lambda \times W}$$

- La formula può essere utilizzata per calcolare una variabile a partire dalle altre.

Simulazioni

- A partire dalla realizzazione di un modello del sistema di calcolo è possibile prevedere il suo comportamento.
- Il simulatore descrive le attività dei dispositivi, dei processi e dello scheduler seguendo le variazioni dello stato del sistema descritte dal modello.
- In genere l'andamento della generazione dei processi, la durata dei CPU burst, ecc. viene descritto da un generatore di numeri casuali.
- La distribuzione di probabilità degli eventi può essere determinata statisticamente o empiricamente.

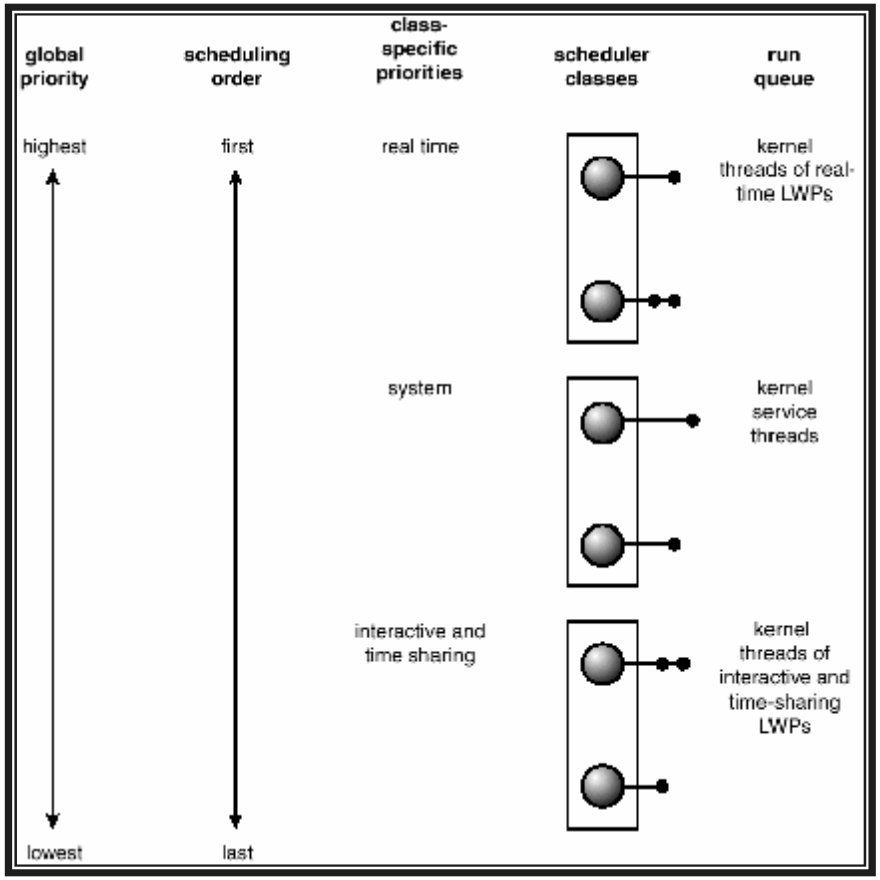
Valutazione degli scheduler della CPU con simulazione



Realizzazione

- E’ possibile utilizzare un medico empirico: per valutare un algoritmo di scheduling si può implementare e inserirlo nel sistema operativo.
- Questo metodo ha i suoi inconvenienti:
  - Costo dell’implementazione,
  - Un sistema in continua messa a punto provoca reazioni da parte degli utenti,
  - L’ambiente cambia anche in relazione al modificarsi delle condizioni di scheduling (gli utenti si adattano).

Scheduling in Solaris 2



- Il sistema operativo Solaris 2 adotta uno scheduling dei processi basato su priorità con quattro classi:
  - *real time,*
  - *system,*
  - *time sharing,*
  - *interactive.*
- Ogni classe ha il suo criterio di scheduling (code multiple con retroazione

Scheduling in Solaris 2

- Un processo inizia la sua esecuzione con un LWP e la classe di scheduling predefinita à la *time sharing*.
- Le priorità dei processi si modificano dinamicamente, assicurando un tempo di risposta basso per i processi interattivi e una buona produttività per i processi CPU bound.
- La classe *interactive* assegna priorità più alte a quei processi interattivi associati a interfacce a finestre.
- I processi del nucleo appartengono alla classe *system:* un thread di questa classe è eseguito fino a quando si blocca oppure è sottoposto a prelazione.
- I thread nella classe *real time* ottengono priorità più alte fra tutte le classi e vengono eseguiti prima dei thread appartenenti alle altre classi.

Scheduling in Windows 2000

- L'algoritmo di scheduling dei thread in Windows 2000 è basato su priorità e prelazione: vengono sempre eseguiti i thread a priorità più alta.
- Un thread viene eseguito fino a quando non è sottoposto a prelazione da un thread a priorità più alta oppure termina, esaurisce il suo quanto di tempo, o esegue una chiamata di sistema bloccante.
- I thread real time sottopongono a prelazione tutti gli altri thread.
- Lo schema delle priorità ha 32 livelli e due livelli:
  - *variable*: thread con priorità 1-15
  - *real-time*: thread con priorità 16-31

Priorità in Windows 2000

- Le priorità numeriche del nucleo del sistema operativo sono in relazione con quelle dell'API Win 32.
- Tutte le classi, eccetto quella real time, hanno priorità variabile; ciascuna classe ha priorità relativa.
- Ogni thread ha una priorità di base nella classe di appartenenza e il valore predefinito è quello della priorità relativa NORMAL

Priorità API Win 32

Priorità relativa

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Scheduling in Linux

- Il sistema operativo Linux ha due diversi algoritmi di scheduling: uno a partizione del tempo di elaborazione, l'altro con priorità per le elaborazioni in tempo reale.
- Il Linux permette la prelazione dei soli processi eseguiti in modo utente; un processo del nucleo non si può sottoporre a prelazione, neanche da parte di processi real time.
- Nello scheduling a partizione del tempo, viene usato un algoritmo a priorità **basato sui crediti**; ad ogni interruzione del temporizzatore il processo perde un credito.
- Quando tutti i crediti sono esauriti, questi vengono assegnati a tutti i processi nel sistema con la seguente formula:

  crediti=crediti/2 + priorità

- In questo modo si dà priorità ai processi interattivi o con prevalenza di I/O.

Scheduling real time di Linux

- Lo scheduling real time di Linux è basato sulle due classi definite in POSIX.1b: FCFS e RR.
- Le priorità relative dei processi di elaborazione in tempo reale sono assicurate, ma il nucleo non fornisce garanzia sui tempi di attesa nella coda dei processi pronti (*soft real time*).
- Se un segnale d'interruzione dovesse rendere eseguibile un processo real time mentre il nucleo è impegnato nell'esecuzione di una chiamata di sistema, il processo attende.

Sommario

- Lo scheduling della CPU consiste nella scelta di un processo nella coda dei processi pronti; l'assegnazione alla CPU è eseguita dal dispatcher.
- Esistono differenti algoritmi di scheduling con proprietà diverse.
- Per la scelta di un algoritmo è necessario utilizzare delle tecniche di valutazione: metodi analitici, simulazione, le reti di code,…
- I sistemi operativi che gestiscono i thread a livello del nucleo, si occupano dello scheduling dei thread, non dei processi