Università di Roma Tor Vergata

Corso di Laurea triennale in Informatica

Sistemi operativi e reti

A.A. 2018-2019

Pietro Frasca

Lezione 2

Giovedì 4-10-2018

Sistemi a singolo processore

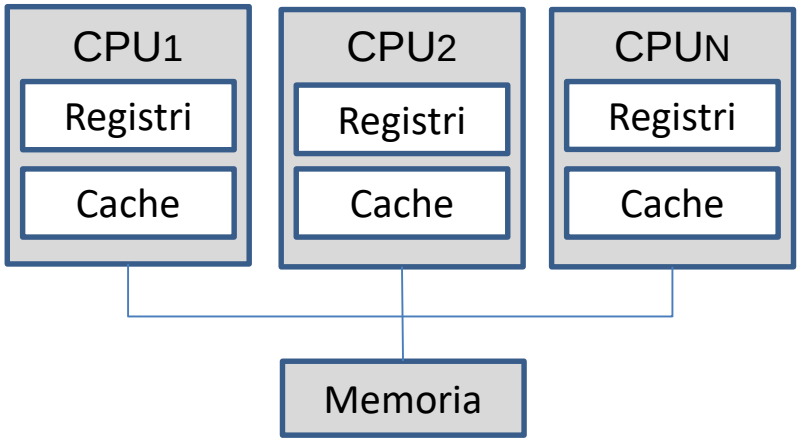
- Fino a pochi anni fa, la maggior parte dei computer era dotata di un solo processore. Un sistema monoprocessore è fornito di un processore principale, in grado di eseguire le istruzioni dei programmi e anche di altri processori per usi speciali. Questi ultimi possono essere processori per specifici dispositivi, come ad esempio le schede grafiche, i controller dei dischi, i canali DMA.
- Tutti questi processori *specializzati* eseguono un set d'istruzioni che riguardano i rispettivi dispositivi ma non eseguono istruzioni dei programmi utente.
- Ad esempio, un microprocessore per un controller di disco implementa la propria coda del disco e l'algoritmo di schedulazione. Questo funzionamento evita che il processore principale sia sovraccaricato dalle operazioni di scheduling del disco.
- L'uso di microprocessori specializzati è diffuso, ma la presenza di tali processori non rende un sistema a singolo processore un sistema multiprocessore.

Sistemi multiprocessore

- Negli ultimi anni, si sono sempre più diffusi i sistemi multiprocessore (o multicore).
- Tali sistemi hanno due o più processori, condividono i bus del computer e, generalmente la memoria e le periferiche. Le prime architetture multiprocessore sono apparse nei server ed ora, più processori si trovano nei personal computer nei tablet, smartphone e micro controller.
- I sistemi multiprocessore hanno alcuni importanti vantaggi rispetto ai sistemi monoprocessore. Un primo vantaggio è dato dall'*aumento della produttività*. Aumentando il numero di processori, più programmi possono essere eseguiti in parallelo e quindi in meno tempo. Tuttavia, l'incremento della velocità con N processori non è pari a N, ma meno di N. Infatti, la presenza di più processori implica che il sistema operativo debba eseguire varie operazioni per gestirne correttamente il funzionamento.

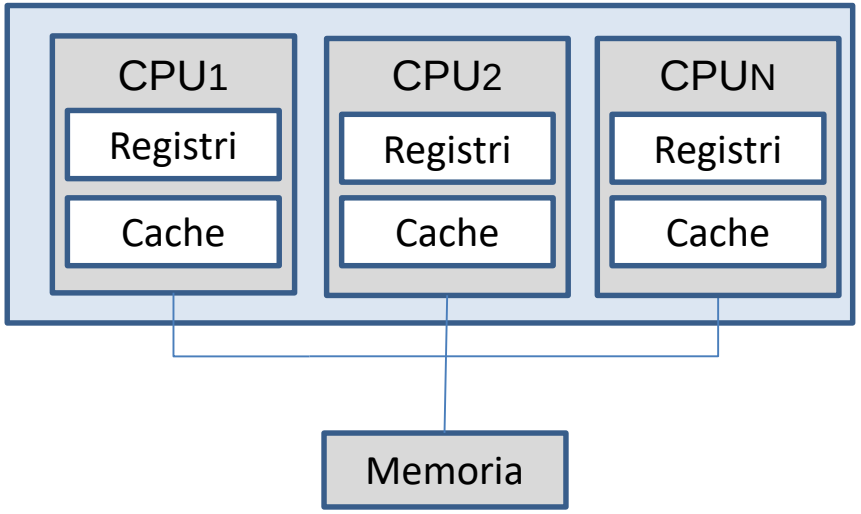
- Inoltre, i sistemi multiprocessore sono più affidabili, poiché essendo le funzioni distribuite tra diversi processori, in caso di guasto di uno di essi il sistema continua a funzionare, anche se più lentamente. Se un computer ha *N* processori e uno non funziona, allora ciascuno dei rimanenti *N-1* processori può svolgere i compiti del processore guasto. I sistemi che subiscono un guasto di uno o più componenti e continuano a funzionare sono detti ***fault tolerant***. La tolleranza agli errori richiede un meccanismo per permettere al guasto di essere rilevato, diagnosticato, e, se possibile, sistemato.
- Una maggiore affidabilità di un sistema è essenziale in molte applicazioni.
- Gli attuali sistemi multiprocessore sono di due tipi: **multiprocessori multiprocessori simmetrici (SMP) e asimmetrici (AMP).**

- Quest'ultimi utilizzano il ***multiprocessing asimmetrico (AMP)***, in cui a ogni processore è assegnato un compito specifico. Questi sistemi seguono il modello ***master-worker*** in cui, un processore ***master*** controlla il sistema e assegna i compiti ai processori worker, i quali eseguono attività predefinite.
- I sistemi più diffusi usano il ***symmetric multiprocessing (SMP)***, che prevede che ogni processore esegua tutte le funzioni del sistema operativo. SMP significa che tutti i processori sono pari; non esiste alcun rapporto gerarchico tra i processori



- Ogni processore ha il proprio insieme di registri e una cache privata. Tuttavia, tutti i processori condividono la memoria fisica.
- Il vantaggio di questo modello è che molti processi possono essere eseguiti in parallelo, allo stesso tempo.
- Praticamente tutti i moderni sistemi operativi, tra cui Windows, Linux e Mac OS X forniscono il supporto per SMP.
- I sistemi SMP possono avere due modelli di accesso alla memoria: ***accesso alla memoria uniforme (UMA)*** e *accesso* ***alla memoria non uniforme (NUMA)***. UMA è una tecnica in cui l'accesso a qualsiasi RAM da qualsiasi CPU richiede la stessa quantità di tempo. Con NUMA, alcune parti della memoria possono richiedere più tempo di accesso rispetto ad altre parti, creando una riduzione delle prestazioni.
- Una recente tendenza nella progettazione della CPU è di includere più core di elaborazione su un unico chip. Tali sistemi multiprocessore sono definiti ***multicore***. Possono essere più efficienti di multipli chip separati, perché la comunicazione on-chip è più veloce della comunicazione tra chip separati.

- Inoltre, un chip con più core utilizza molta meno energia rispetto a più chip single-core.



Cluster

- Un altro tipo di sistema multiprocessore è il ***cluster***. I sistemi cluster differiscono dai sistemi multiprocessore in quanto sono composti da due o più computer, detti ***nodi***, connessi tra loro.
- I computer del cluster condividono la memoria secondaria e sono collegati tramite una LAN o, più strettamente, con un bus di interconnessione veloce.
- Il cluster è di solito utilizzato per fornire un alta affidabilità di servizio, cioè, il servizio continua a funzionare anche se uno o più computer del cluster si guastano. Un software appropriato è eseguito sui nodi del cluster. Se un nodo si guasta, un altro nodo può eseguire le applicazioni che erano in esecuzione sulla macchina guasta
- Un cluster può essere strutturato in modo asimmetrico o simmetrico.
- Nel cluster asimmetrico, una macchina è in modalità ***hot-standby***, mentre le altre eseguono le applicazioni. La macchina host hot-standby non fa altro che monitorare i nodi attivi.

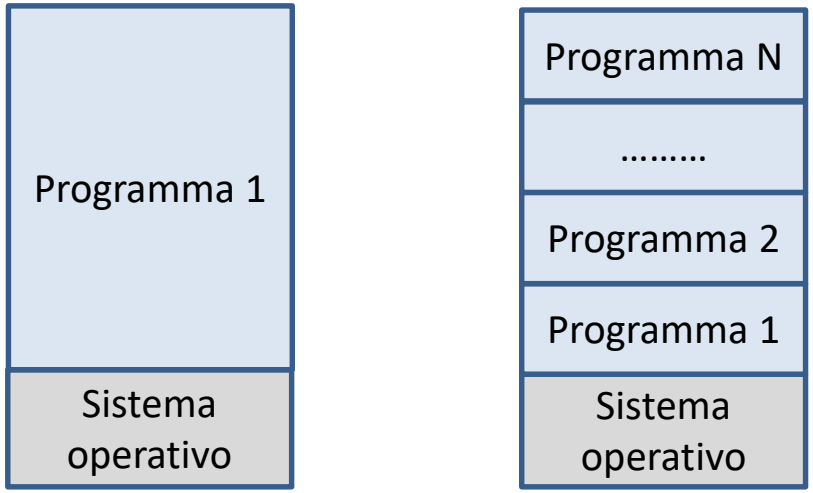
- Se un nodo si guasta, il nodo hot-standby diventa un server attivo. Nel cluster simmetrico, due o più nodi eseguono applicazioni e si controllano a vicenda. Questa architettura è più efficiente, in quanto utilizza tutto l'hardware disponibile.
- Con cluster costituiti da molti computer collegati tramite una rete, è possibile ottenere ambienti di calcolo ad alte prestazioni.
- Altre forme di cluster sono i cluster paralleli e il cluster su *rete geografica (WAN, Wide Area Network)*.

Struttura del sistema operativo

- I sistemi operativi variano molto nella loro struttura e composizione, in quanto sono realizzati seguendo obiettivi di progetto differenti. Essi hanno, tuttavia, molti punti in comune.
- Una delle caratteristiche più importanti che condividono i sistemi operativi è la ***multiprogrammazione (multitasking)*.**

Sistemi operativi batch multiprogrammati

- La multiprogrammazione, introdotta con i sistemi batch, nei mainframe negli anni ’60, si basa sull’evidenza che un solo programma in esecuzione non può tenere la CPU e i dispositivi di I/O occupati allo stesso tempo, poiché tipicamente esso alterna istruzioni di computazione a istruzioni di ingresso/uscita.
- La multiprogrammazione aumenta l’utilizzo della CPU organizzando l’esecuzione dei programmi (job) in modo che la CPU abbia sempre (o quasi sempre), istruzioni da eseguire.
- Con la multiprogrammazione, il sistema operativo mantiene diversi programmi in memoria contemporaneamente.

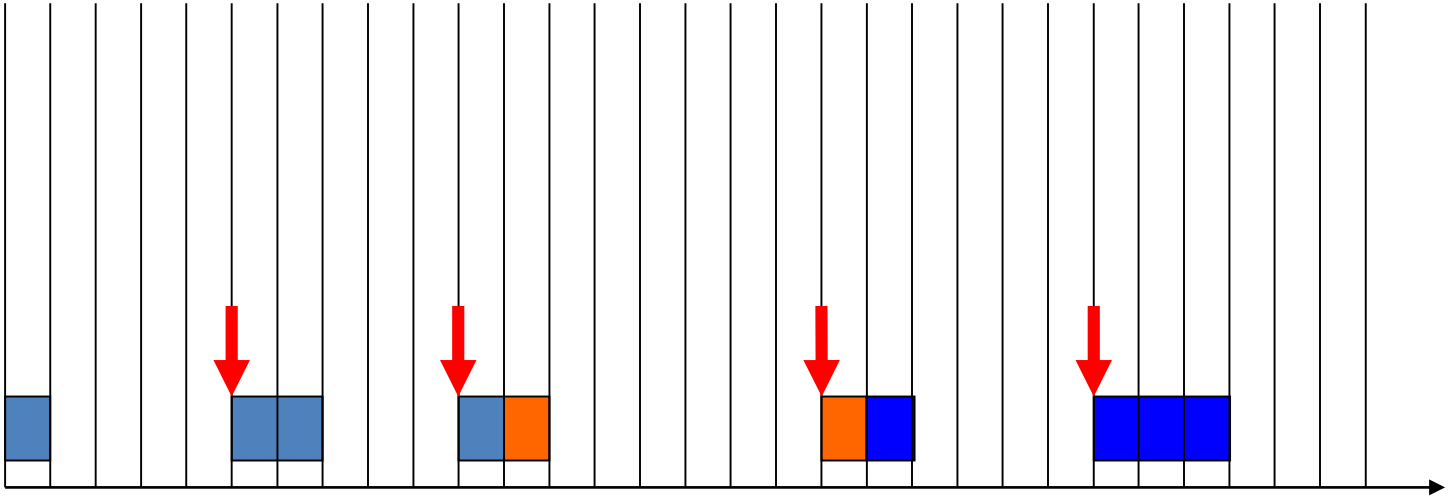


Struttura della memoria in un sistema batch monotasking e multitasking

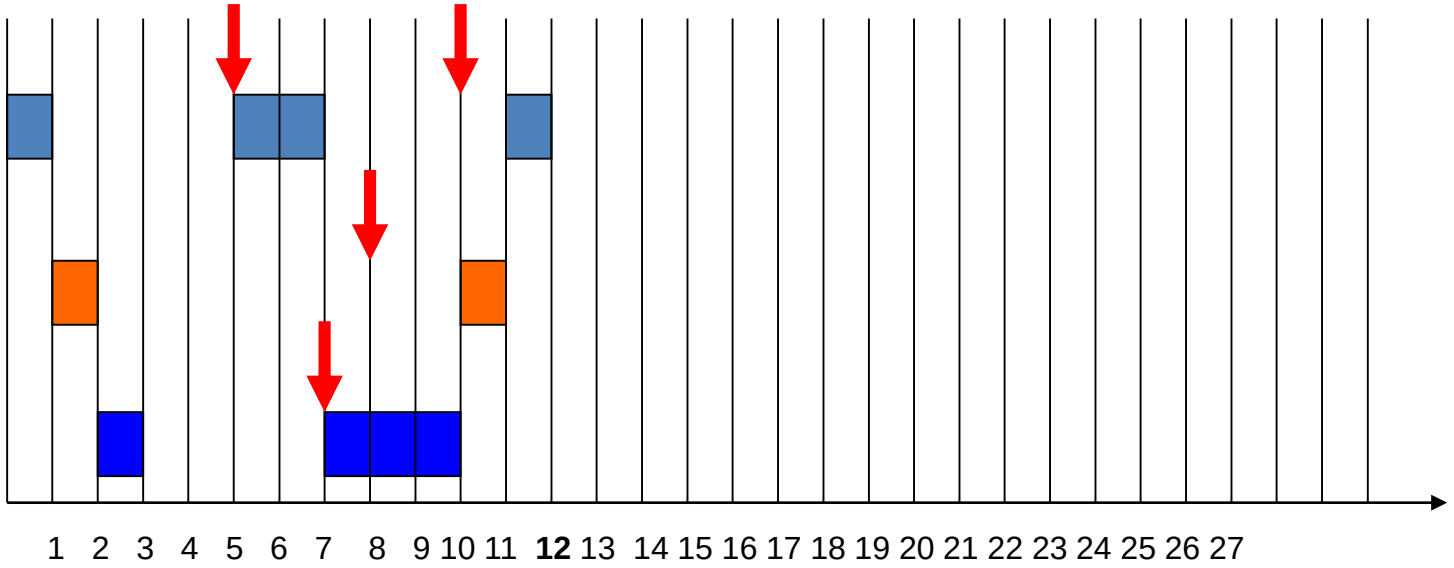
- Il sistema operativo seleziona e inizia ad eseguire uno dei programmi in memoria. In un certo istante, il programma in esecuzione potrebbe eseguire un'operazione di I/O e trovarsi quindi in attesa del suo completamento.
- In questa situazione, in un sistema non multiprogrammato, la CPU sarebbe inattiva. In un sistema multiprogrammato, invece, il sistema operativo passa semplicemente ad eseguire un altro programma.

- Quando anche questo programma deve attendere qualche risorsa, la CPU passa ad un altro programma, e così via. Quando i programmi terminano l'attesa, possono essere di nuovo selezionati dal sistema operativo per tornare di nuovo in esecuzione. Finché è presente almeno un programma da eseguire, la CPU non è mai inattiva.
- Con la multiprogrammazione è possibile aumentare nettamente l'efficienza d'uso della CPU, rispetto ai sistemi mono programmati, fino a raggiungere valori superiori all'80%.
- Per esempio, consideriamo tre processi P1, P2 e P3 e calcoliamo l'efficienza d'uso per i casi in cui siano eseguiti in un sistema monotasking e in un sistema multitasking.
- Nel primo caso, i programmi sono eseguiti sequenzialmente, uno dopo l'altro, come mostrato in figura. Il tempo d'uso di CPU è pari a 10 unità di tempo, mentre il tempo totale è pari a 27 unità. L'efficienza d'uso della CPU è quindi data da 10/27 = 0,37 (37%).
- Con la multiprogrammazione il tempo totale di esecuzione dei tre programmi scende a 12 unità di tempo e quindi l'efficienza d'uso è pari a10 / 12 = 0,83 (83%).

Esecuzione monotasking



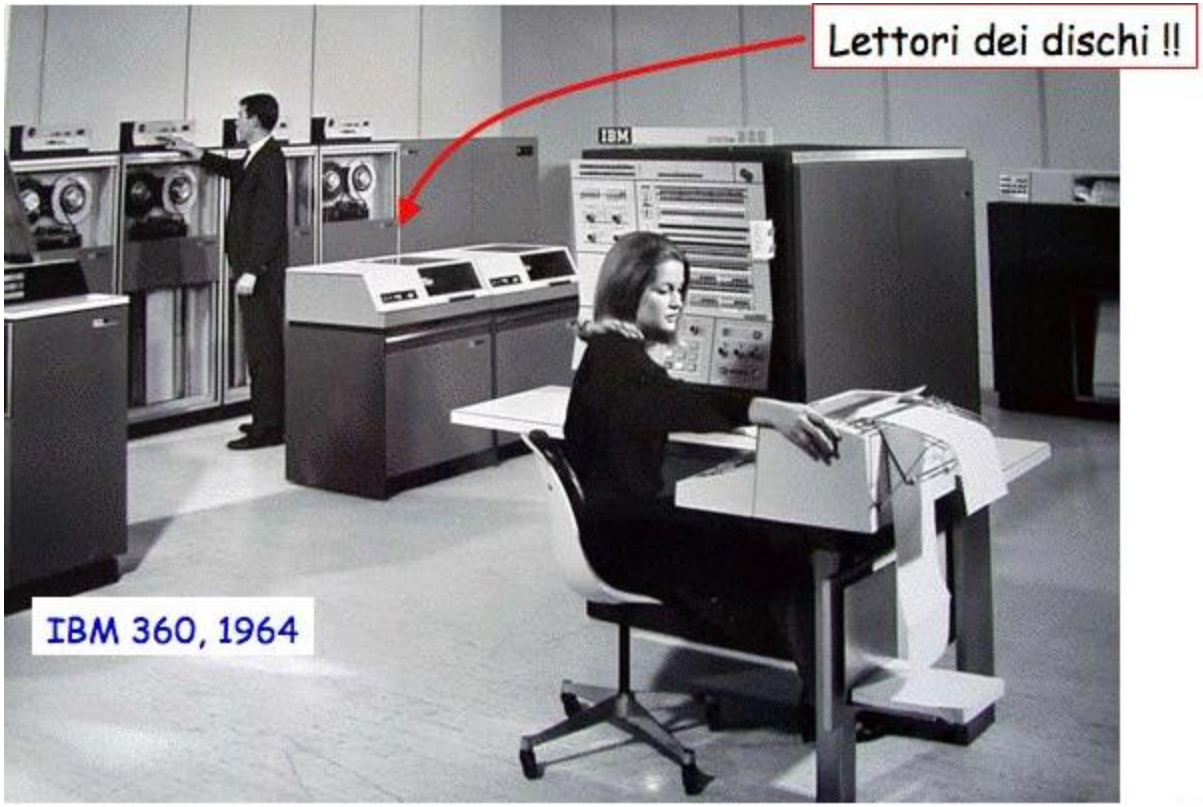
Esecuzione in multitasking



- A prima vista, l'efficienza d'uso della CPU potrebbe essere migliorata, aumentando il ***grado di multiprogrammazione*** cioè il numero di programmi caricati in memoria allo stesso tempo. Tuttavia bisogna considerare anche il tempo necessario per commutare la CPU da un programma ad un altro. Questo meccanismo, detto ***cambio di contesto*****,** porta ad un aumento dell'***overhead*** di sistema. Infatti, quando la CPU è assegnata ad un altro programma, il SO deve eseguire varie operazione tra cui il salvataggio dello ***stato della CPU*** (il valore di tutti i suoi registri).
- Per la scelta dei programmi da caricare in memoria, una strategia spesso usata è preferire un insieme di programmi con caratteristiche diverse, ad esempio alcuni programmi che eseguono molte operazioni di computazione e poche operazione di I/O (***CPU-bound***) e altri con poche istruzioni di calcolo e molte operazioni di I/O (***I/O-bound***) in modo bilanciato e quindi ottimizzare tutte le risorse di macchina.

- Inoltre quando avviene un cambio di contesto, a causa di operazioni di I/O, è necessario scegliere quale sarà il prossimo programma a cui assegnare la CPU. Un criterio spesso utilizzato nei sistemi batch è di assegnarla in base al programma che da più tempo attende di essere eseguito (criterio FIFO). Un altro criterio, è SJF (Short Job First) che assegna la CPU al programma in coda che ha una durata d'esecuzione più breve.
- Con più programmi in esecuzione il SO si deve anche occupare della **gestione delle risorse** per evitare conflitti tra i programmi.
- Se le risorse non fossero gestite dal SO, un programma in esecuzione potrebbe usarle quando queste sono già state assegnate ad un altro programma.
- Consideriamo, ad esempio, il caso in cui più programmi richiedano allo stesso tempo l'uso di una stampante. Senza l'intervento del sistema operativo, si avrebbero delle stampe con contenuto misto relativo ai vari programmi. E' necessario quindi che il sistema operativo implementi degli algoritmi di assegnazione delle varie risorse ai programmi che le richiedono.

- I sistemi batch multiprogrammati sono stati i primi SO complessi funzionanti sui grossi mainframe. Molto famosi sono stati i sistemi dell'IBM della serie 360 e 370.
- I sistemi batch sono oggi ancora molto usati in ambiente scientifico, aziendale, finanziario per l'elaborazione di buste paga, gestione dei conto correnti, gestione assicurazioni e altre attività che richiedono una grande potenza elaborativa su grandissime quantità di dati.
- I sistemi batch multiprogrammati forniscono un ambiente in cui le varie risorse di sistema (per esempio, CPU, memoria e periferiche) sono utilizzate in modo efficace, ma non prevedono l'interazione dell'utente con il computer.


Lettori dei dischi !!
IBM
IBM 360, 1964

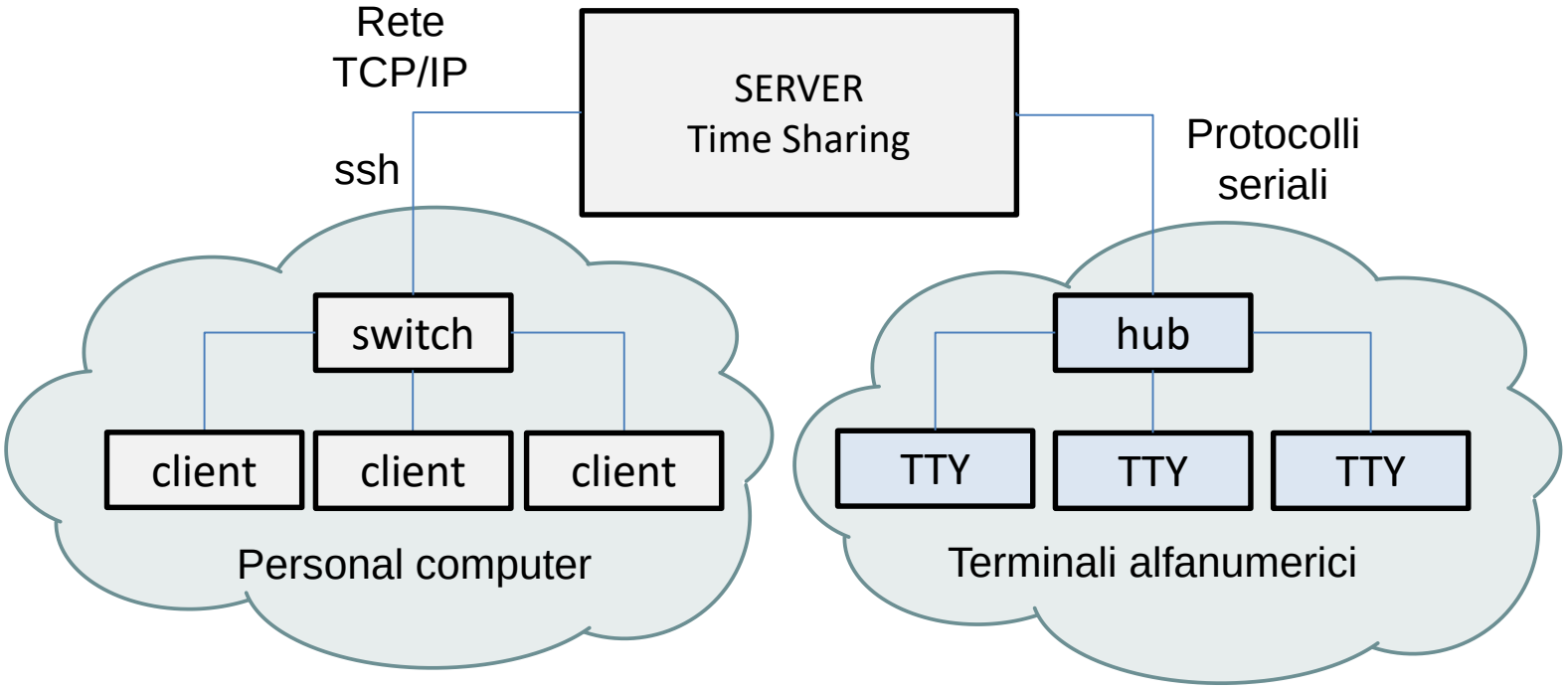
Moderno sistema batch

Sistemi operativi time sharing

- I sistemi ***time sharing*** *(a condivisione di tempo)* utilizzano una logica estensione della multiprogrammazione. Un sistema operativo a condivisione di tempo consente a molti utenti di condividere allo stesso tempo il computer.
- I primi sistemi time sharing risalgono agli anni ’70 con l’introduzione dei minicalcolatori, molto più piccoli ed economici rispetto ai mainframe. Unix e Linux sono a partizione di tempo e multiprogrammati.
- In questi sistemi, la CPU esegue più programmi passando dall’uno all’altro con una frequenza elevatissima in modo da consentire agli utenti di interagire con i propri programmi quando sono in esecuzione.
- Poiché la CPU è commutata rapidamente da un programma all’altro ciascun utente ha l’impressione che l’intero sistema di elaborazione sia dedicato al suo utilizzo, anche se è condiviso tra molti utenti.
- Il time sharing è un sistema interattivo che consente all’utente di comunicare direttamente col sistema operativo eseguendo *comandi* o programmi, in modo da avere risultati immediati.

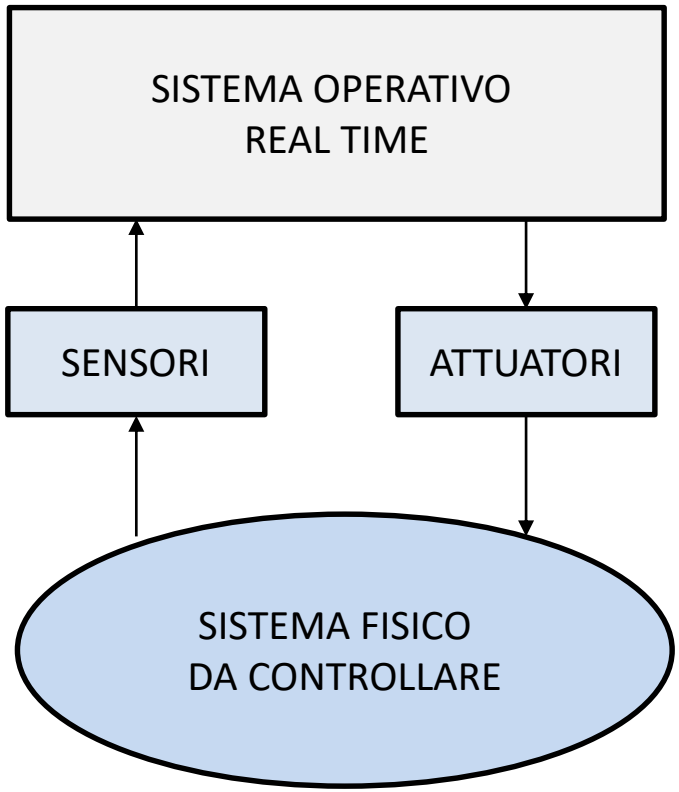
- Pertanto, il tempo di risposta deve essere breve, tipicamente meno di un secondo.
- L'obiettivo principale per i sistemi time-sharing è minimizzare il tempo di risposta dei programmi e quindi minimizzare il tempo di attesa medio da parte degli utenti per ottenere dal programma una risposta alle richieste effettuate.
- Un utente si collega ad un sistema time-sharing utilizzando un terminale o un computer. Nel computer deve essere installata un'applicazione di rete che utilizza un protocollo di comunicazione per la connessione remota, come ad esempio telnet, rsh, rlogin e ssh (security shell). Quest'ultimo è l'unico che garantisce sicurezza nella connessione.
- Per connettersi, un utente deve avere un account (utenza) sul sistema. L'utente inizia una sessione di lavoro inserendo uno username e una password (login). Una volta connesso comunica con il SO mediante una shell o con una GUI (Graphics User Interface).
- La shell è un'interfaccia a riga di comando con la quale l'utente può digitare i comandi per la gestione e il controllo del SO o avviare

- applicazioni. Al termine del lavoro, l'utente esegue l'operazione di *logout* per disconnettersi dal sistema.
- La politica di scheduling della CPU usata nei sistemi a partizione di tempo è di eseguire i vari programmi assegnando ad essi un ***quanto di tempo (time slice)*** di CPU dell'ordine di alcune decine di millisecondi.



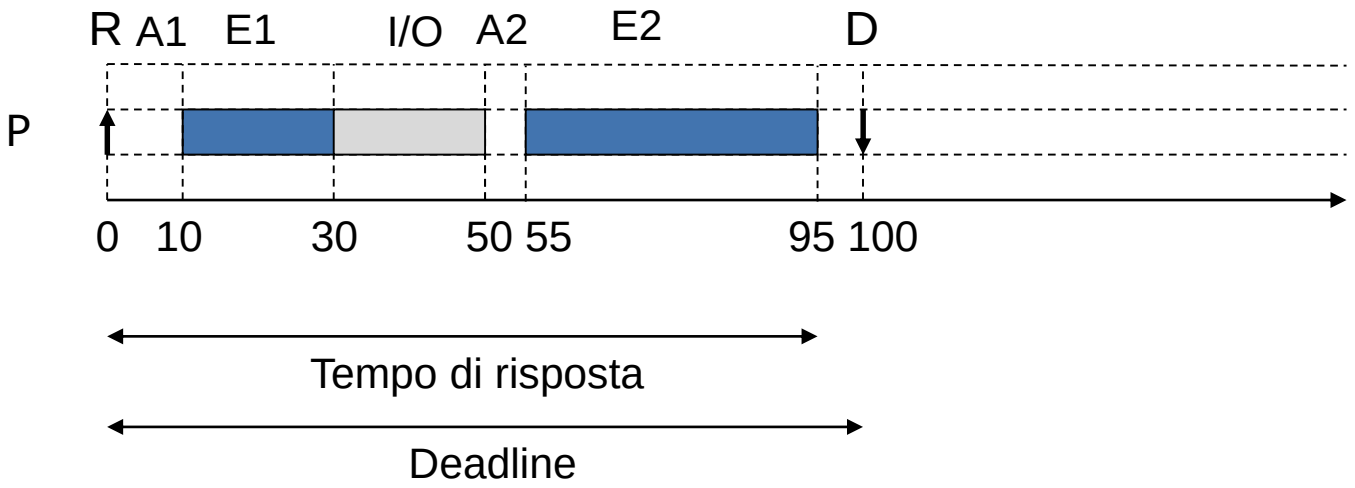
Sistemi operativi real time

- I *SO real time (tempo reale)* costituiscono un'altra classe di sistemi operativi che usa la multiprogrammazione.
- Il calcolatore è utilizzato per la gestione e il controllo di un sistema fisico, detto ***ambiente operativo***, costituito ad esempio da impianti industriali, centrali elettriche, robot, etc.



- Caratteristica fondamentale dei SO realtime è che ogni task deve essere eseguito entro un intervallo di ***tempo definito (deadline)*** imposto dall'applicazione (da microsecondi a millisecondi).
- In altri termini, la validità dei risultati ottenuti da un programma non dipende solo dalla correttezza del programma, ma anche dall'intervallo di tempo entro il quale i risultati sono prodotti.
- Ad esempio, supponiamo che un'applicazione real time, quando avviata, esegua dapprima un cpu burst di 20 ms, quindi I/O burst per altri 20 ms e infine di nuovo cpu burst per 40 ms. Supponiamo, inoltre, che tale programma sia stato progettato affinché esegua il suo task entro 100 ms (deadline).
- Molto probabilmente il sistema operativo non assegnerà la CPU immediatamente all'applicazione, poiché in un sistema multitasking ci sono altri programmi che competono per l'uso della CPU.
- Nella figura seguente è mostrato uno scenario in cui l'esecuzione dell'applicazione è stata completata entro la scadenza stabilita.

- istante di richiesta **R** = 0 ms;
- tempo di esecuzione **E** = E1 + E2 = 20 + 40 = 60 ms
- tempo I/O = 20 ms
- deadline **D** = 100
- tempo di attesa = A1 + A2 = 10 + 5 = 15 ms
- tempo di risposta = 90 ms



- Nei SO real-time generalmente i task hanno diverse criticità e quindi hanno diverse priorità.
- Le priorità possono essere assegnate in modo statico (**priorità statiche**) o calcolate dinamicamente in base alle caratteristiche dei singoli task (**priorità dinamiche**).
- I SO real-time sono classificati in ***hard real-time e soft real-time.*** In questi ultimi, a differenza dei primi, una deadline non rispettata, non danneggia il funzionamento dell'ambiente operativo, ma ne abbassa le prestazioni e quindi la ***qualità del servizio*** *(QoS Quality of Service).*

Sistemi embedded Real-Time

- I computer embedded sono la forma più diffusa di computer.
- Questi dispositivi, spesso chiamati ***micro computer*** o ***micro controller***, sono usati in molti ambienti che comprendono automobili, aeroplani, robot industriali, sistemi di controllo per la domotica etc.

- Con questi sistemi, ad esempio, le case possono essere informatizzate e controllate, in modo da gestirne il riscaldamento, l'illuminazione, i sistemi di allarme e altro.
- Essi hanno compiti molto specifici e quindi spesso i sistemi operativi forniscono funzionalità limitate.
- Di solito, l'interfaccia utente è molto semplice o a riga di comando.
- In alcuni di essi sono installati sistemi operativi d'uso generale come Linux o Windows, in cui alcune parti del sistema, come ad esempio lo scheduler, sono modificate opportunamente per far girare applicazioni real-time.
- L'uso di sistemi embedded continua ad espandersi, sia come unità *stand alone* che come nodi connessi alla rete Internet e gestiti via web.
- L'accesso al Web può consentire a un utente di monitorare e controllare *da remoto* i dispositivi e lo stato dell'abitazione.

Sistemi operativi per personal computer

- I primi SO per personal computer erano semplici mono programmati e monoutente.
- Con l'aumento delle prestazione dei microprocessori e delle dimensioni della memoria, la tecnica della multiprogrammazione è stata implementata anche nei SO per PC.
- Tutti i sistemi operativi moderni per PC sono multitasking: windows XP/7/8/10 (Microsoft), MacOs X di Apple e Linux che è un anche un sistema time-sharing.