Introduzione ai Sistemi Operativi

Corso di Laurea in Ingegneria Informatica

Università degli Studi di Napoli Federico II

Anno Accademico 2024/2025, Canale San Giovanni

Introduzione ai S.O.

- Sommario della lezione:
  - Introduzione ai Sistemi Operativi
  - Cenni Storici
  - Classificazione
- Riferimenti
  - P.Ancilotti, M.Boari, A. Ciampolini, G. Lipari, “Sistemi Operativi”, Mc-Graw-Hill (Cap.1)
  - www.ostep.org, Cap. 2, sez. 2.6

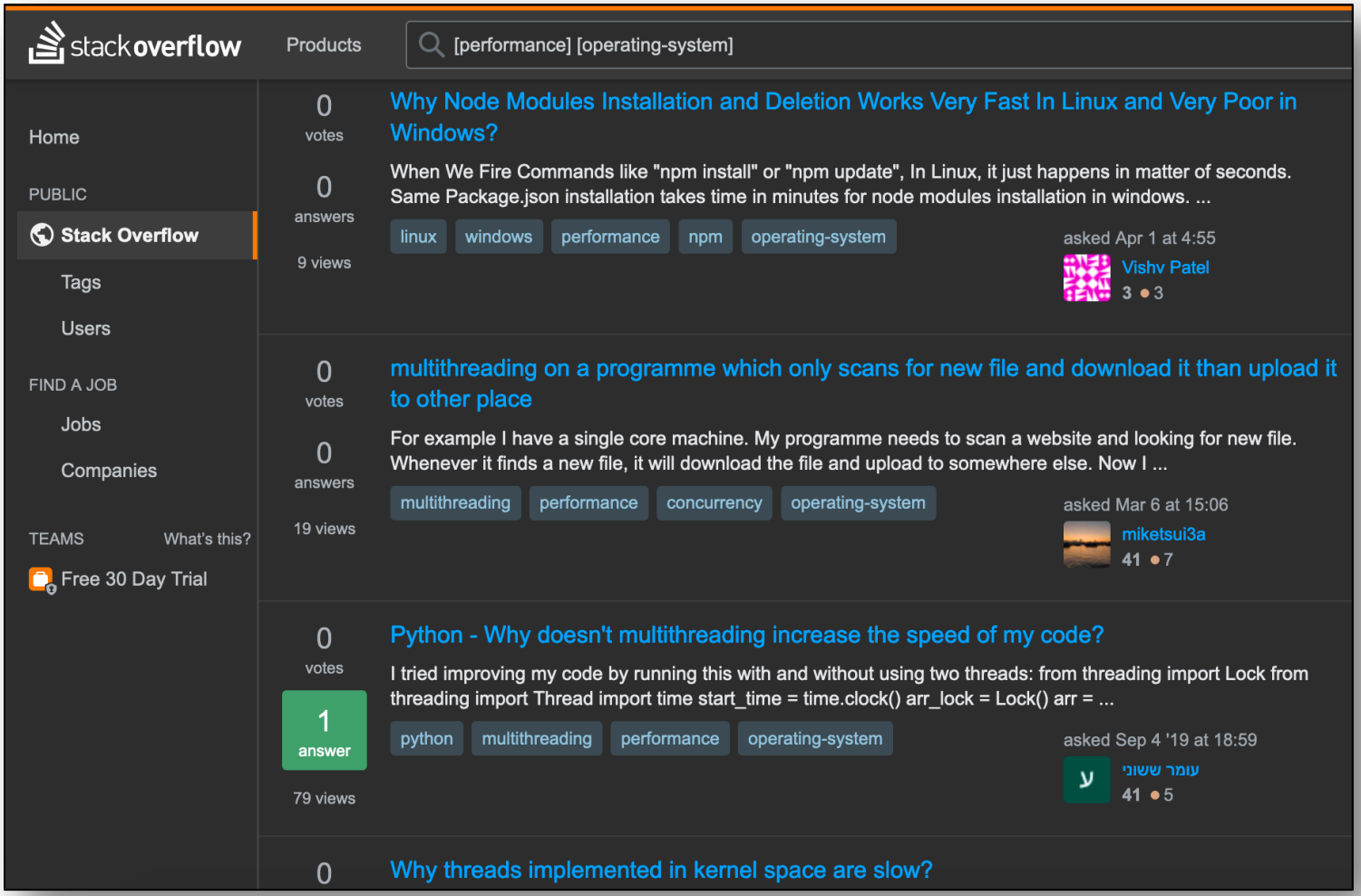
Perché studiare i SO

- Non ci sono molti sistemi operativi in corso di sviluppo
- È molto improbabile che sviluppiate un nuovo sistema operativo durante la vostra carriera professionale
- ... allora, perchè si studiano?

Perché studiare i SO

- **Capire il funzionamento dei computer**
  - I SO sono ovunque - essi hanno la funzione primaria di gestire le **risorse di calcolo** (CPU, RAM, Dischi, Rete, ...)
- **Programmare efficientemente**
  - La conoscenza della struttura interna e dei principi di funzionamento di SO è indispensabile per la **programmazione di sistema** e permette di scrivere programmi più efficienti
- **Programmare in modo concorrente**
  - La programmazione concorrente nasce storicamente con i sistemi operativi, e la si può apprendere meglio in questo contesto

Perché studiare i SO



Perché studiare i SO

- Conoscere i SO permette di **diagnosticare i problemi** di prestazioni e affidabilità
  - Perché il computer rallenta quando si riempie la memoria?
  - Perché occorre "espellere" una memoria USB ?
  - È possibile recuperare un file cancellato?
  - Perché è inutile svuotare la cache delle "applicazioni recenti" sullo smartphone?
  - ...

Perché studiare i SO

- Conoscere il funzionamento del SO è importante per comprendere e prevenire **problemi di sicurezza**
  - Buffer overflow
  - DLL injection
  - Meltdown/Spectre
  - Return-to-libc
  - Heap spraying
  - Command injection
  - Address Space Layout Randomization
  - Red zones
  - Rootkit
  - …

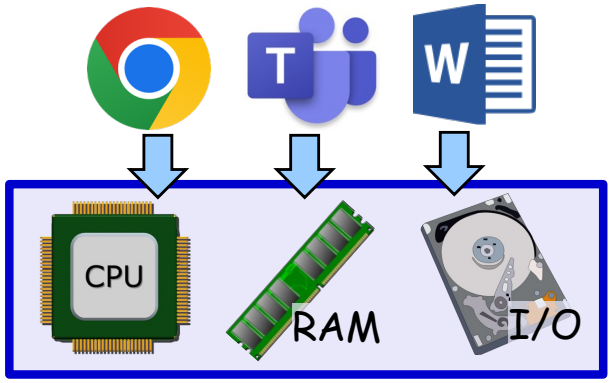
Obiettivi dei Sistemi Operativi

Un Sistema Operativo (S.O.) è un insieme di programmi che **operano sull'hardware di un calcolatore**, per:

```c
#include <stdio.h>

int main() {
  printf("Hello World\n");
  return 0;
}
```

semplificare lo sviluppo software

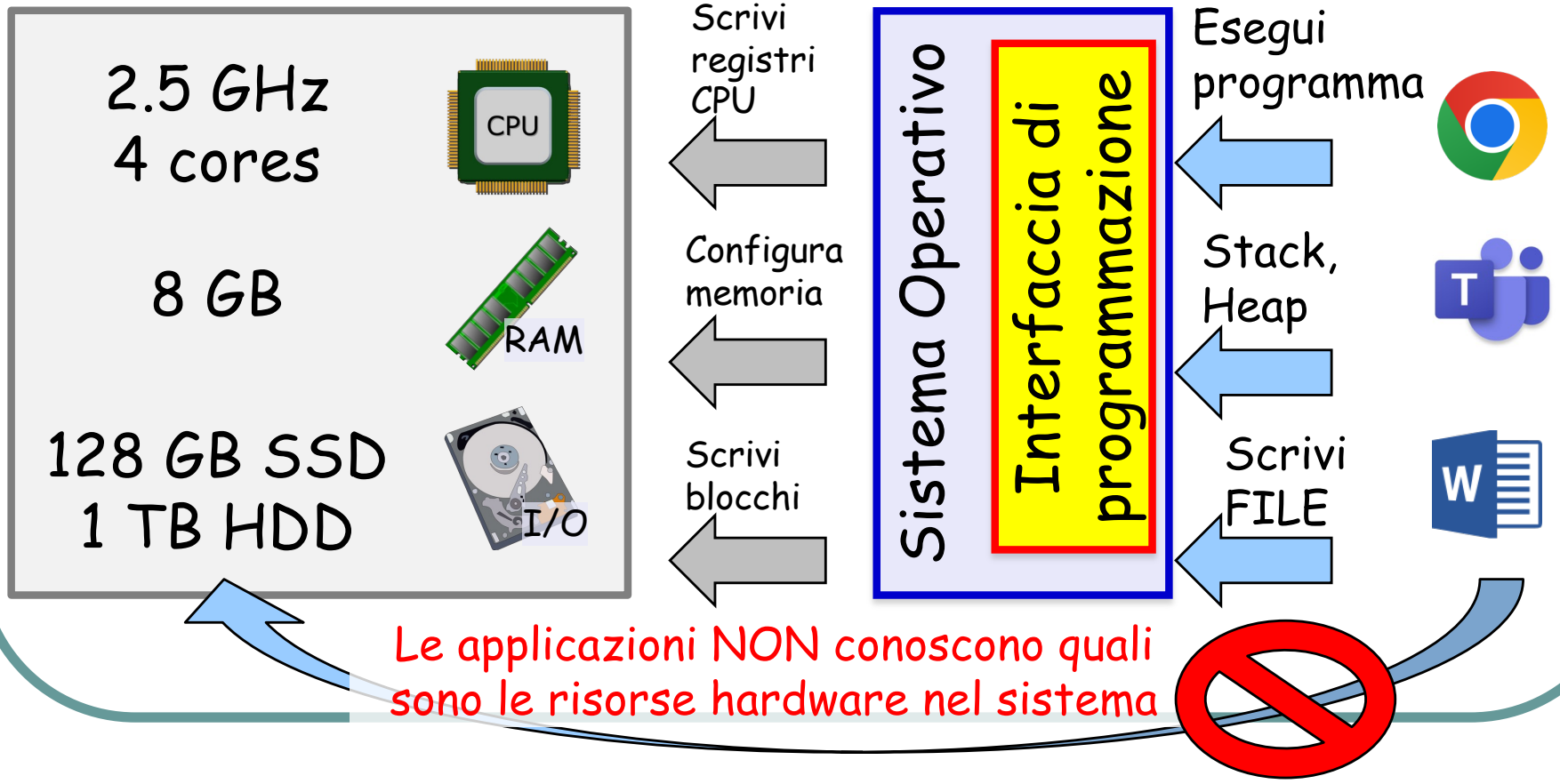


gestire efficientemente le risorse hardware

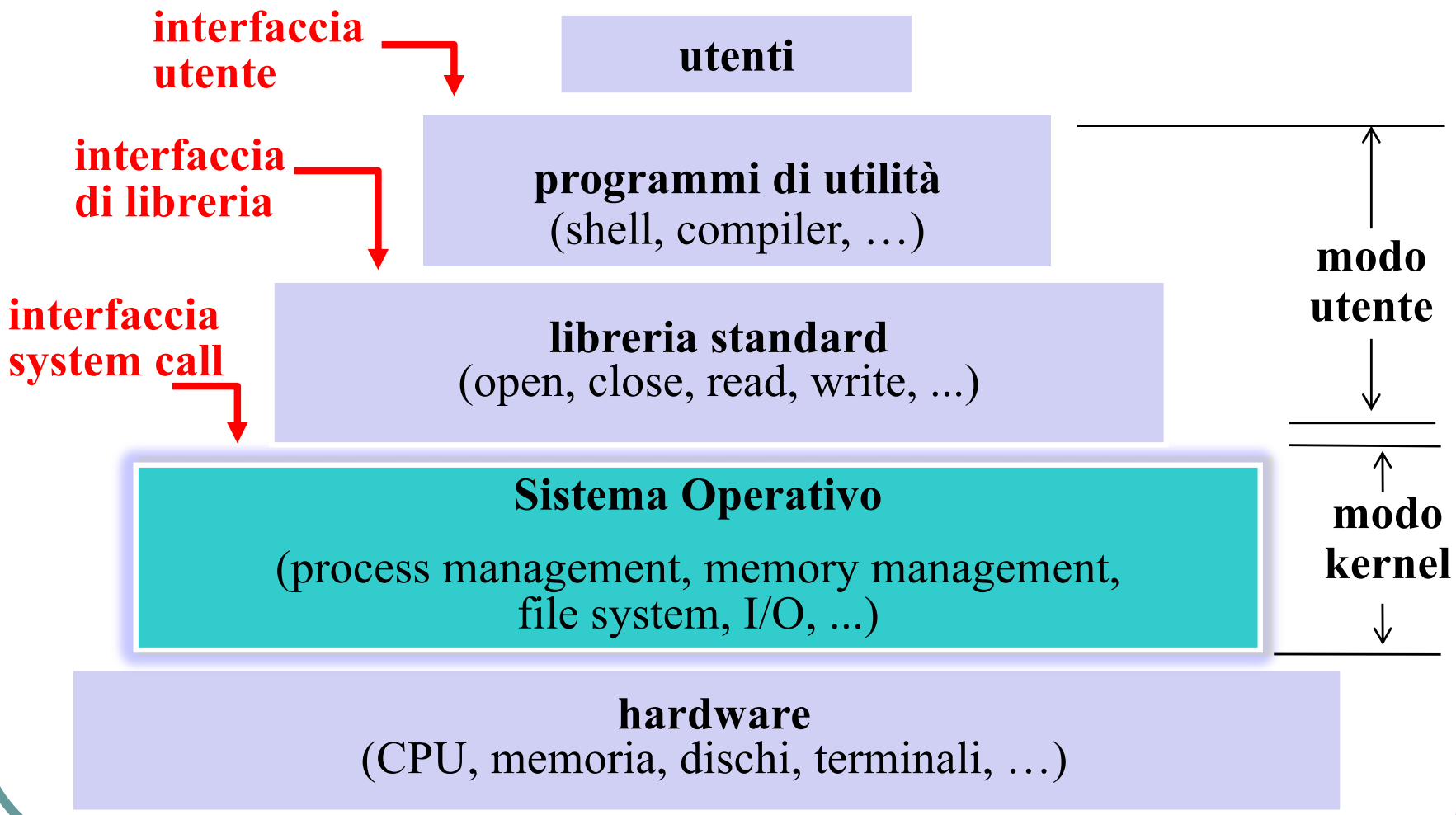
proteggere le informazioni, prevenire accessi non autorizzati

Il concetto di virtualizzazione

- Il Sistema Operativo si pone da intermediario tra l'utente (o il software) e l'hardware

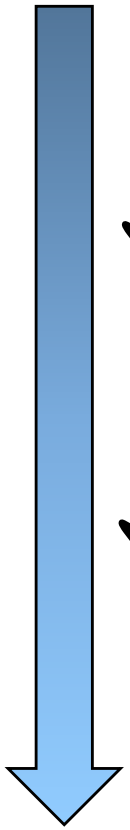


Struttura di un sistema di elaborazione



Un po' di storia …

- Sistemi "Single-User"
- Sistemi Batch
- Sistemi Multiprogrammati
- Sistemi Time Sharing
- Personal Computing
- Mobile, Embedded, e Cloud Computing



Minore costo dell'hardware

Maggiore capacità di memoria e di calcolo

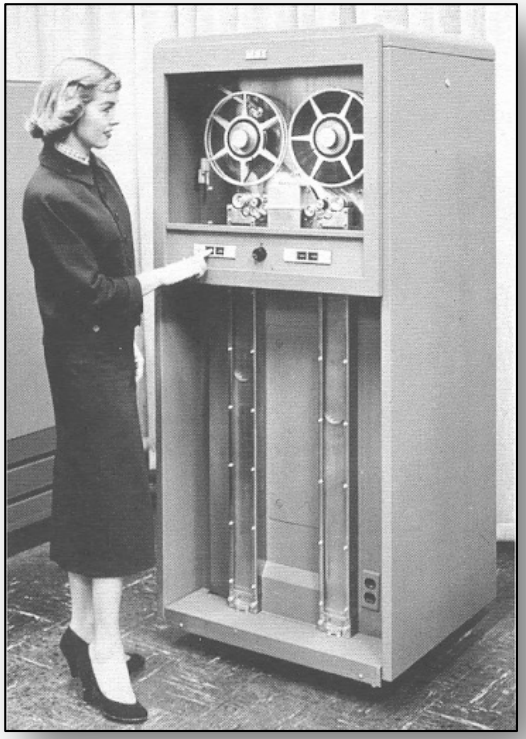
Evoluzione dei SO

… Guidata dall'evoluzione dell'hardware

	1981	2016	**Fattore migliorativo**
MIPS	1	≈200000	**200000**
€/SPECInt	≈ 100K€	< 2	**50000**
DRAM size	128KB	16 GB	**130000**
Disk Size	10MB	1 TB	**100000**
Net Bwd	9600 bps	10 Gbps	**1000000**
Address bits	16	64	**4**
User/Machine	100	<1	**100**

Single-User System

- Nessun sistema operativo!
- Il computer esegue un solo programma alla volta da un nastro fornito dall'utente
- L'utente non "interagisce" con il computer durante l'esecuzione del programma



Esempio:
IBM 701 Open Shop

Single-User System

- Problema: bassa percentuale di utilizzo di risorse hardware (costose)

$$\%Utilizzo = \frac{tempo\ di\ effettivo\ uso\ dell'hardware}{tempo\ totale}$$

Esempio (IBM 701 presso General Motors, 1983):

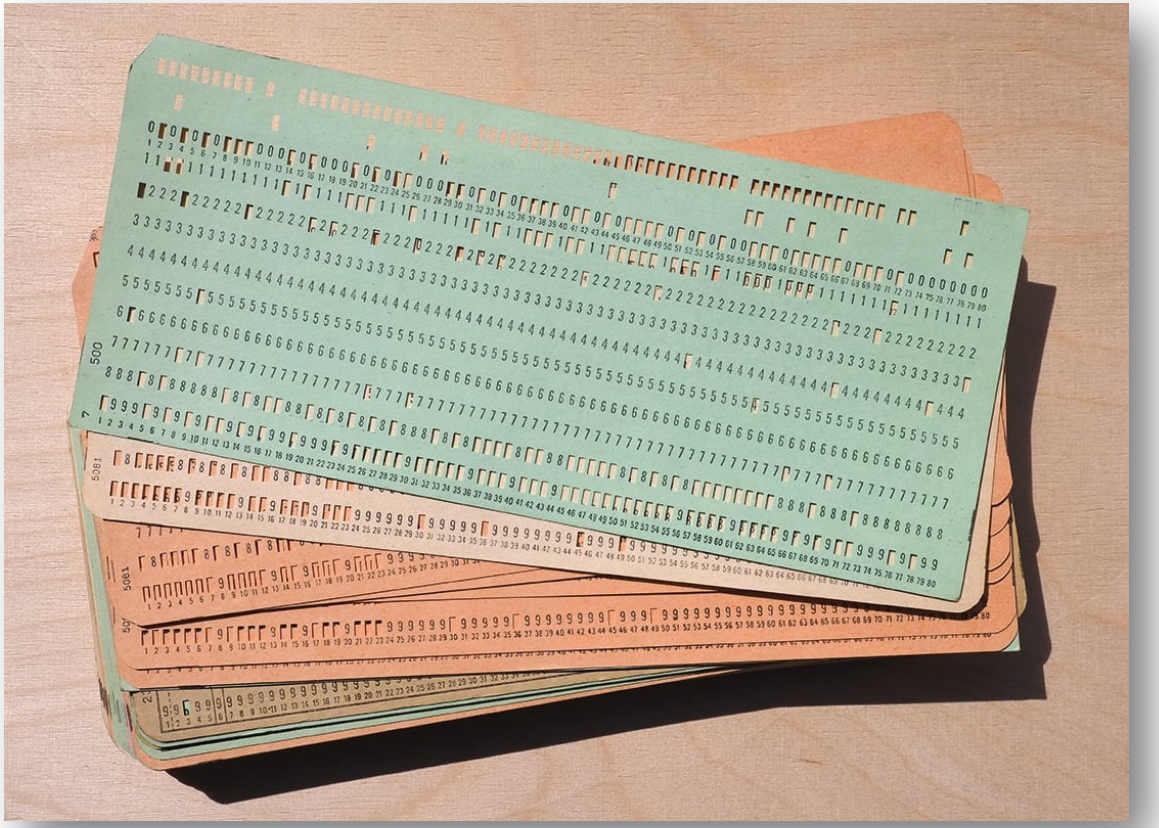
*Each user was allocated a minimum 15-minute slot, of which time he usually **spent 10 minutes in setting up the equipment** to do his computation . . . By the time he got his calculation going, he may have had only **5 minutes or less of actual computation** completed-wasting two thirds of his time slot.*

The cost of the wasted computer time was $146,000 per month--in 1954 dollars!

%Utilizzo = 33%

Batch System (sistema a lotti)

- L'utente scrive il suo programma (job) su schede perforate
- Più job di più utenti sono raggruppati (lotti, "batch")



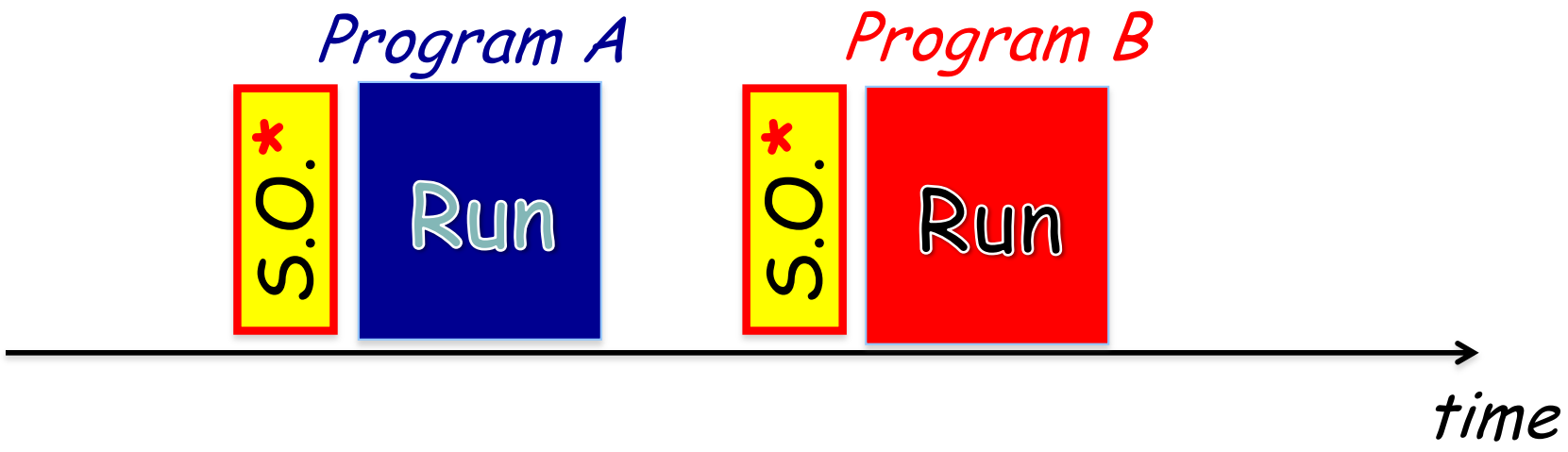
Batch System (sistema a lotti)

- I job sono caricati ed eseguiti in sequenza dal computer
- L'utente ritorna dopo ore/giorni per avere i risultati (uso non-interattivo)

Esempio:
IBM 709

Batch System monoprogrammati

Il S.O. è un semplice programma (**monitor**) residente nella memoria del computer, che lancia i job in sequenza



* Carica ed esegue il prossimo job

Batch System monoprogrammati

I primi sistemi batch erano monoprogrammati

Il programma in corso deve essere completato del tutto prima che sia eseguito il prossimo

Batch System (sistema a lotti)

Vantaggi

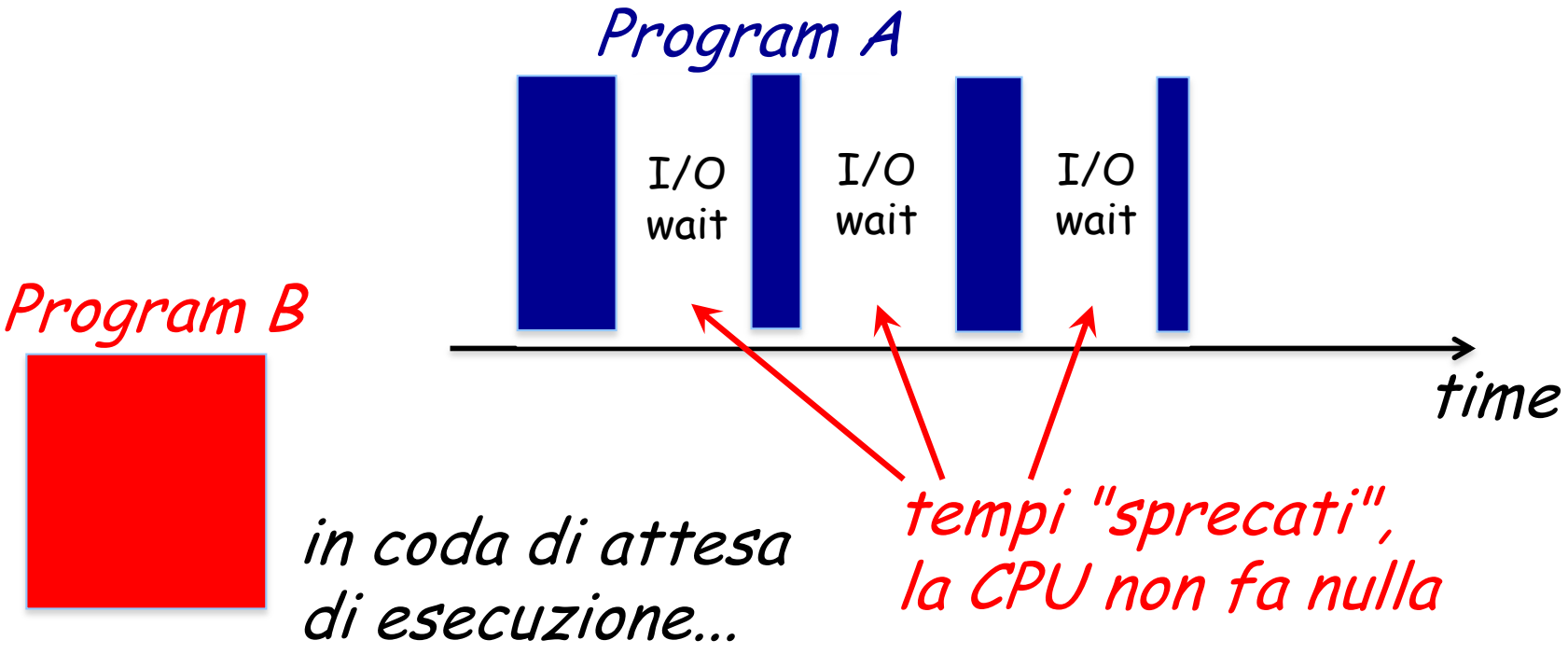
La macchina è **meglio utilizzata** grazie alla assenza di interazione con l'utente.

Svantaggi

I programmi sono eseguiti sequenzialmente. L'utente può anche attendere ore o giorni per avere i risultati del proprio lavoro.

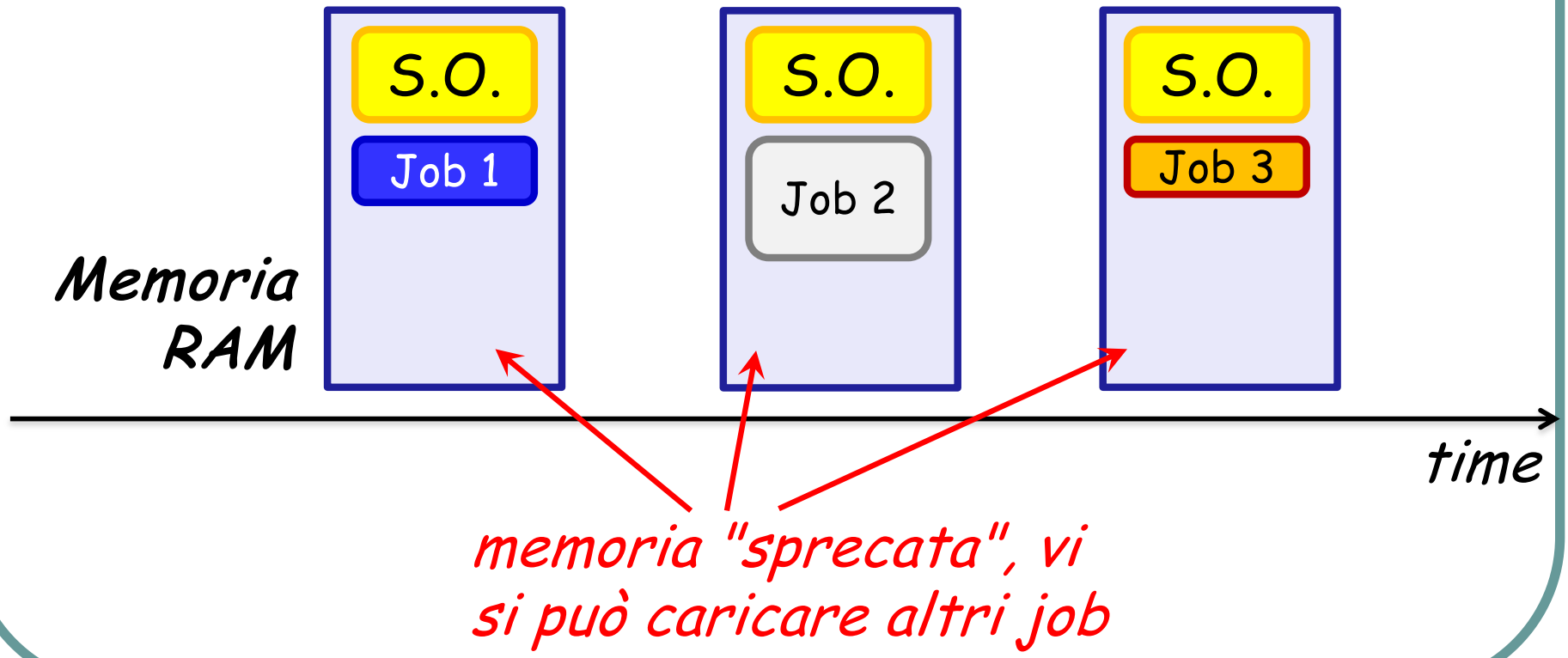
Batch System monoprogrammati

Il processore rimane **inattivo** durante l'esecuzione delle **operazioni di I/O**.



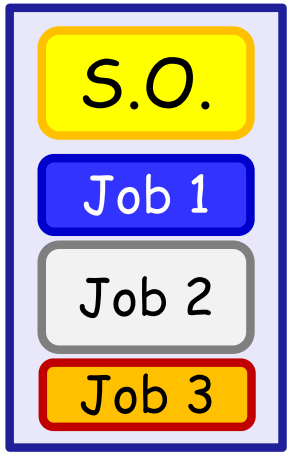
Batch System monoprogrammati

Al più un job alla volta era caricato in memoria principale



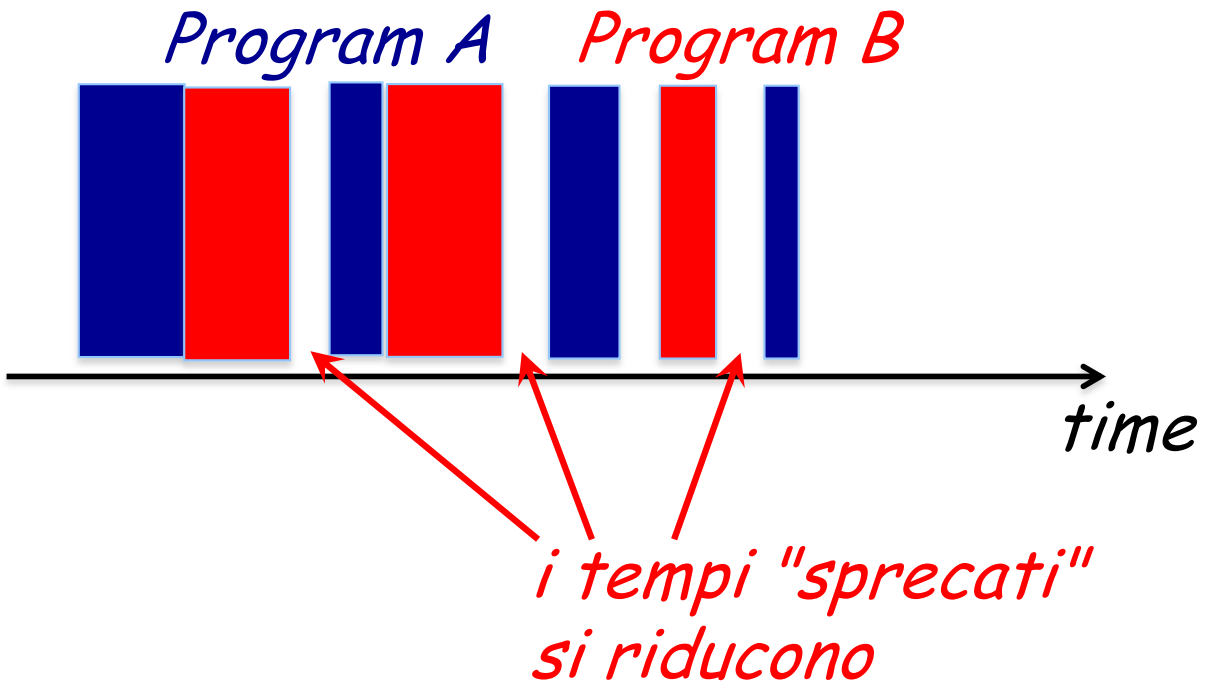
Sistemi multiprogrammati

- **Multiprogrammati = Più job** sono caricati in memoria
- La CPU esegue 1 job, gli altri rimangono in attesa
- Se il job si ferma per I/O, la CPU viene **ri-assegnata ad un altro job**



Sistemi multiprogrammati

Quando un job è in **attesa di un'operazione di I/O**, il SO assegna il processore ad un **altro job**

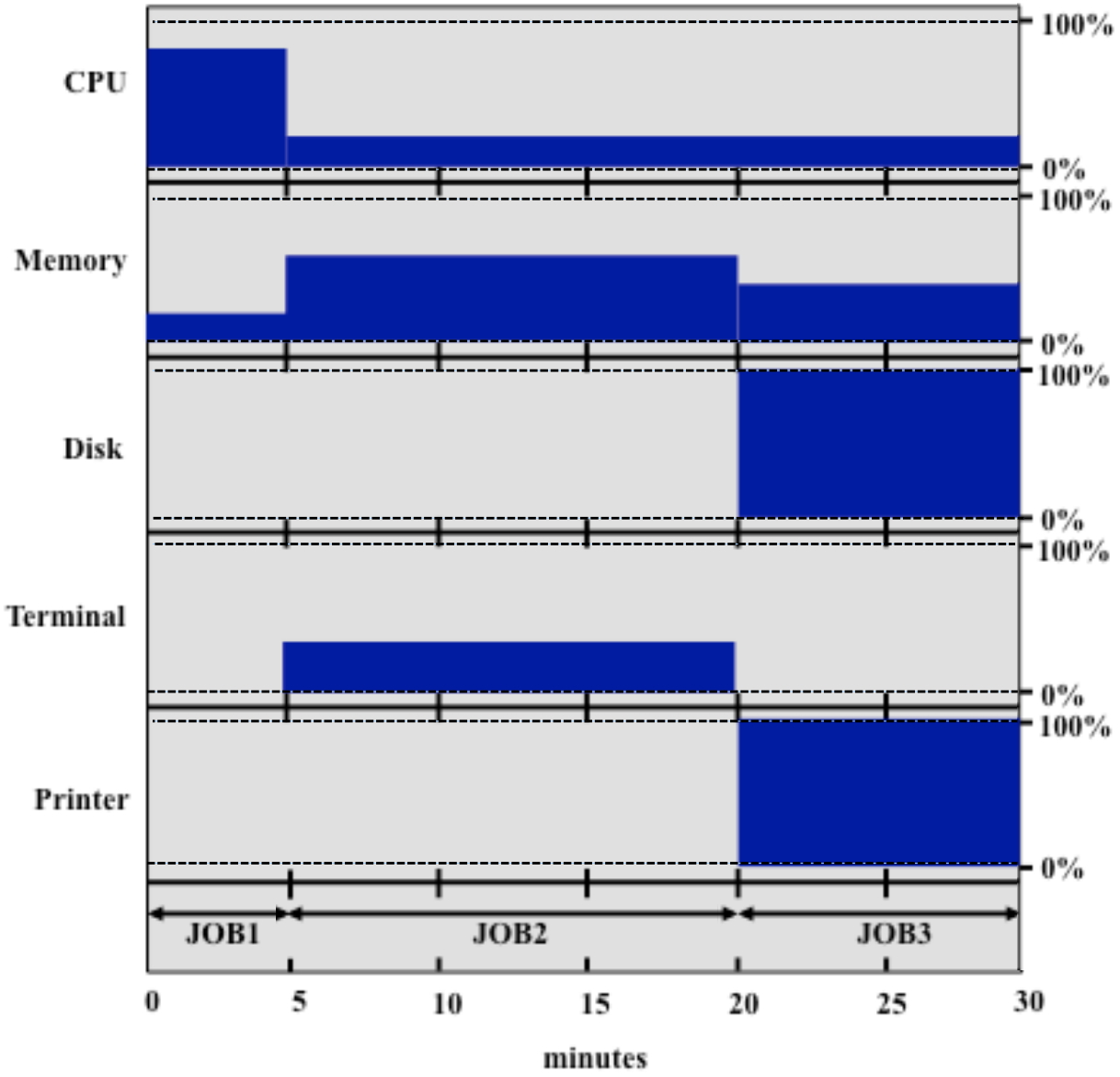


Esempio multiprogrammazione

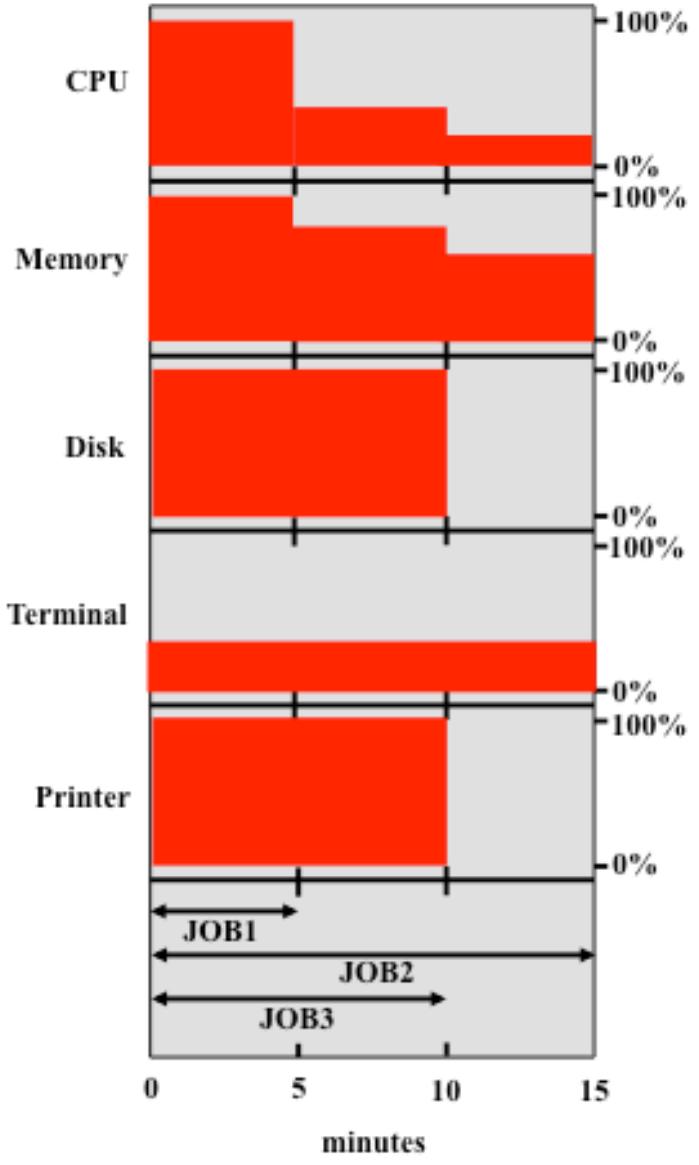
	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Esempio multiprogrammazione

MONOPROGRAMMATO



MULTIPROGRAMMATO



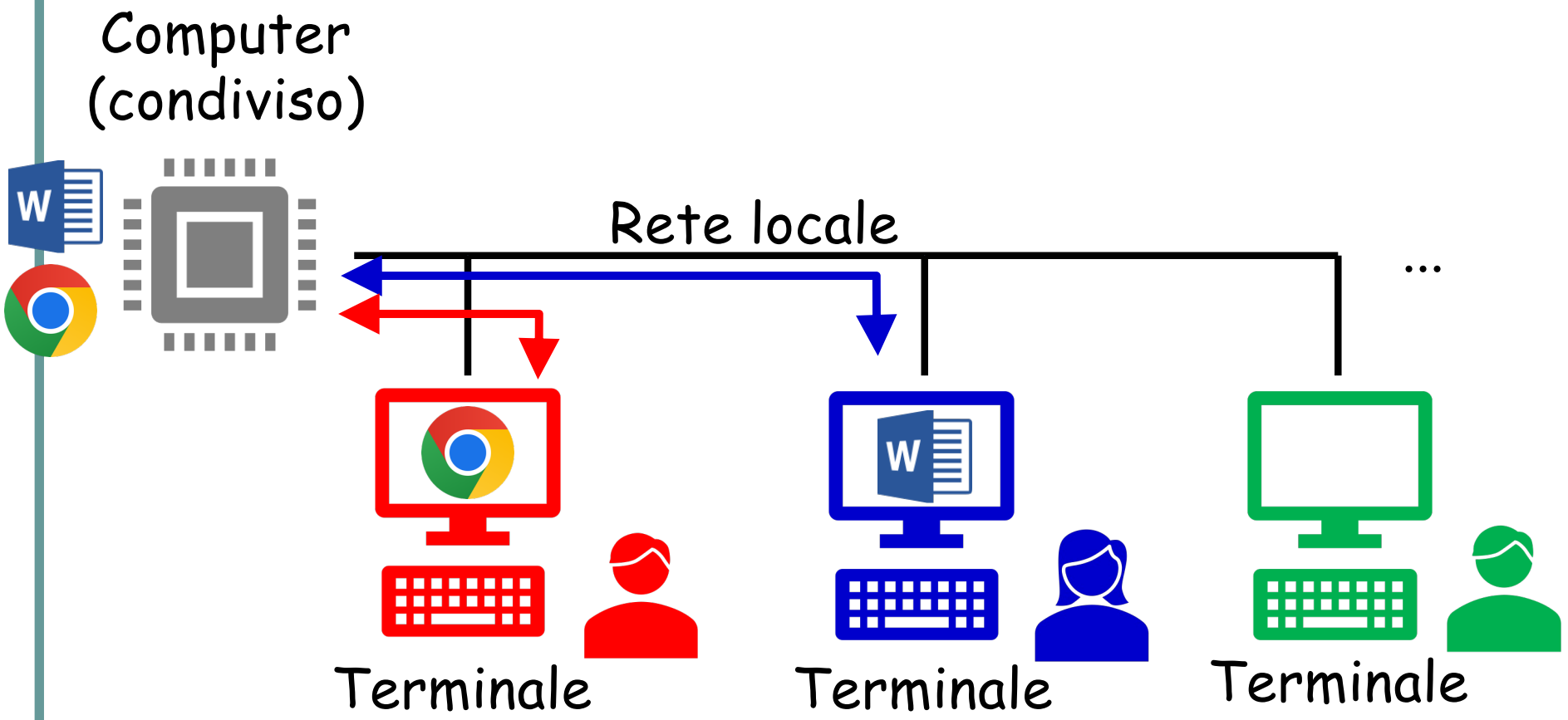
Esempio multiprogrammazione

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Time Sharing

- Il time sharing estende l'idea di multiprogrammazione
- Ma, gli utenti **non devono aspettare ore/giorni** per avere dei risultati
- Interagiscono **in tempo reale con i loro programmi**

Time Sharing



Time Sharing

Il sistema UNIX è stato uno dei primi SO time sharing

https://www.youtube.com/watch?v=tc4ROCJYbm0

Time Sharing

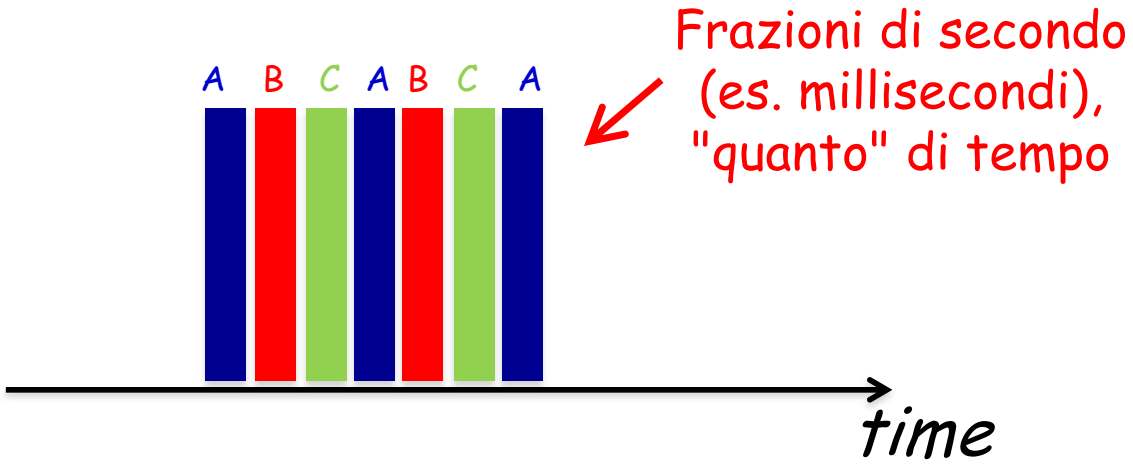
Nei sistemi **batch multiprogrammati**:

- i job si interrompono solo quando fanno I/O, oppure terminano il programma
- si alternano infrequentemente sulla CPU

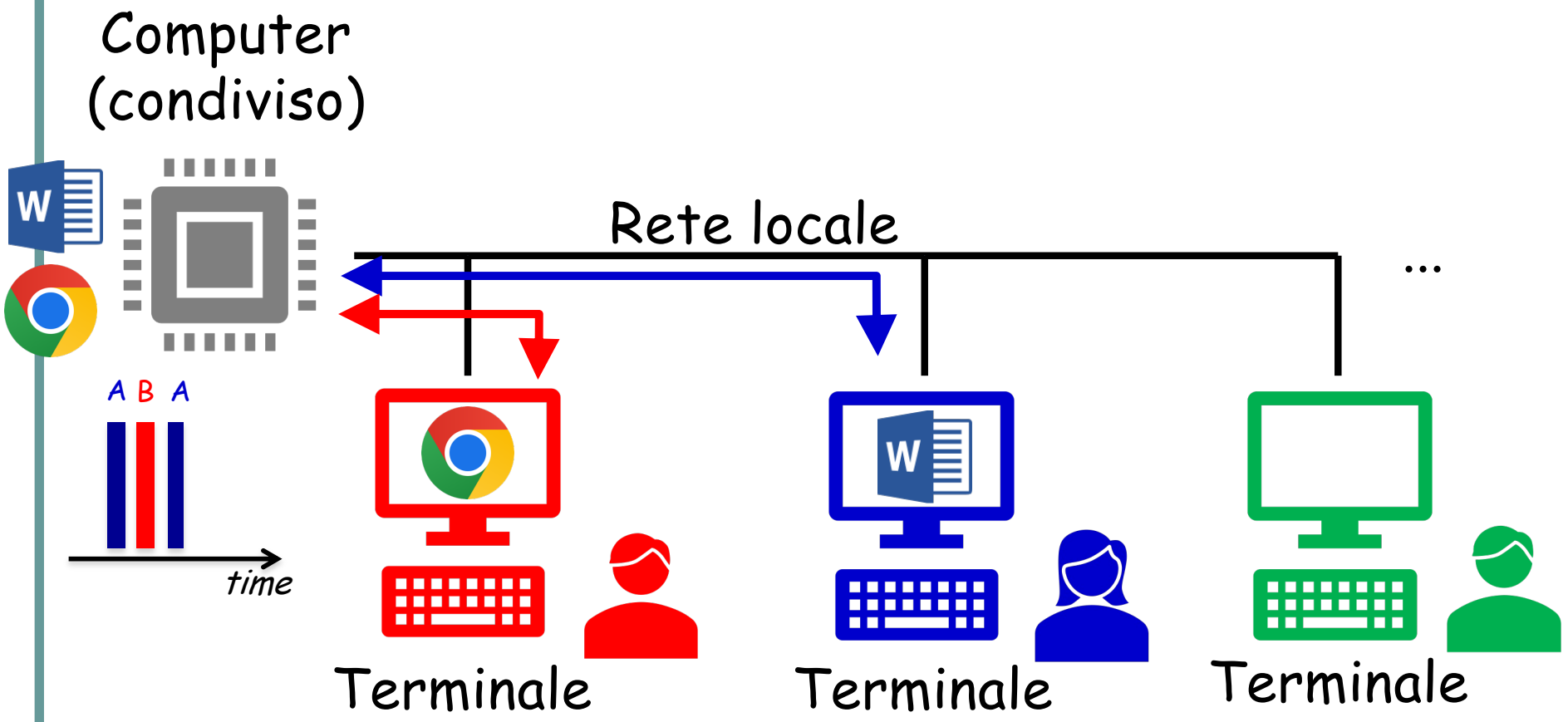
Nei sistemi **time-sharing**:

- i job sono interrotti forzatamente dal SO (**preemption**), anche se non fanno I/O
- si alternano sulla CPU molto **più frequentemente** (100+ volte al secondo!)

Time Sharing

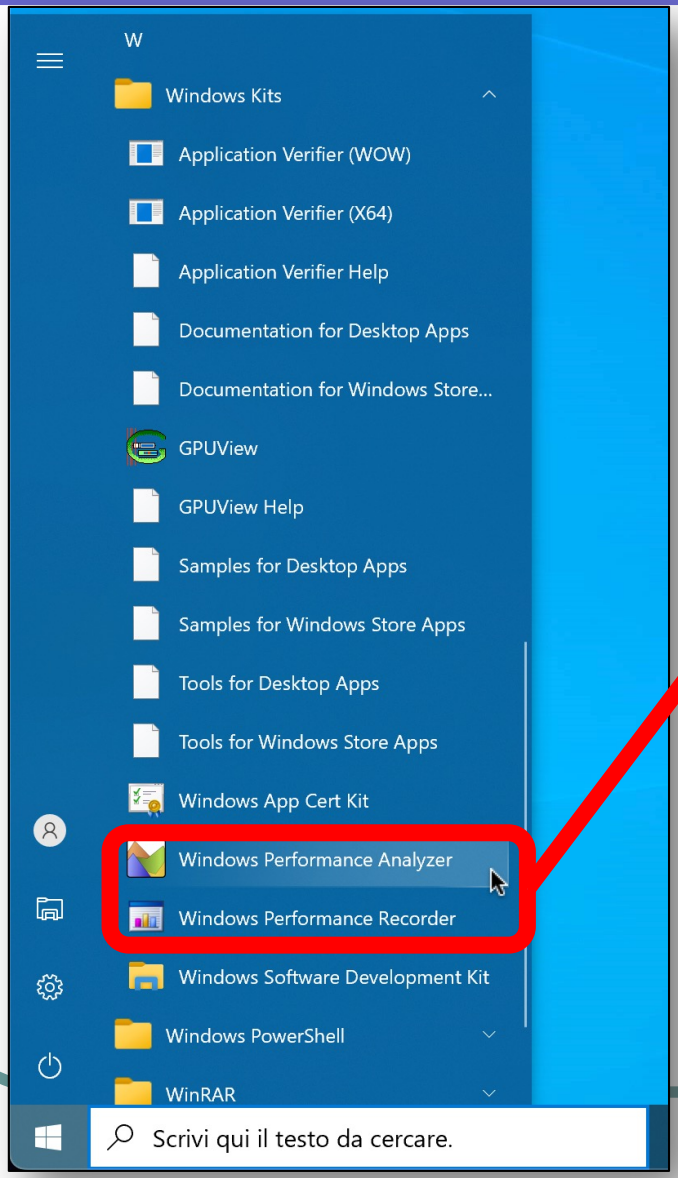


Time Sharing



I due programmi "appaiono" agli utenti come se eseguissero in contemporanea su due processori diversi

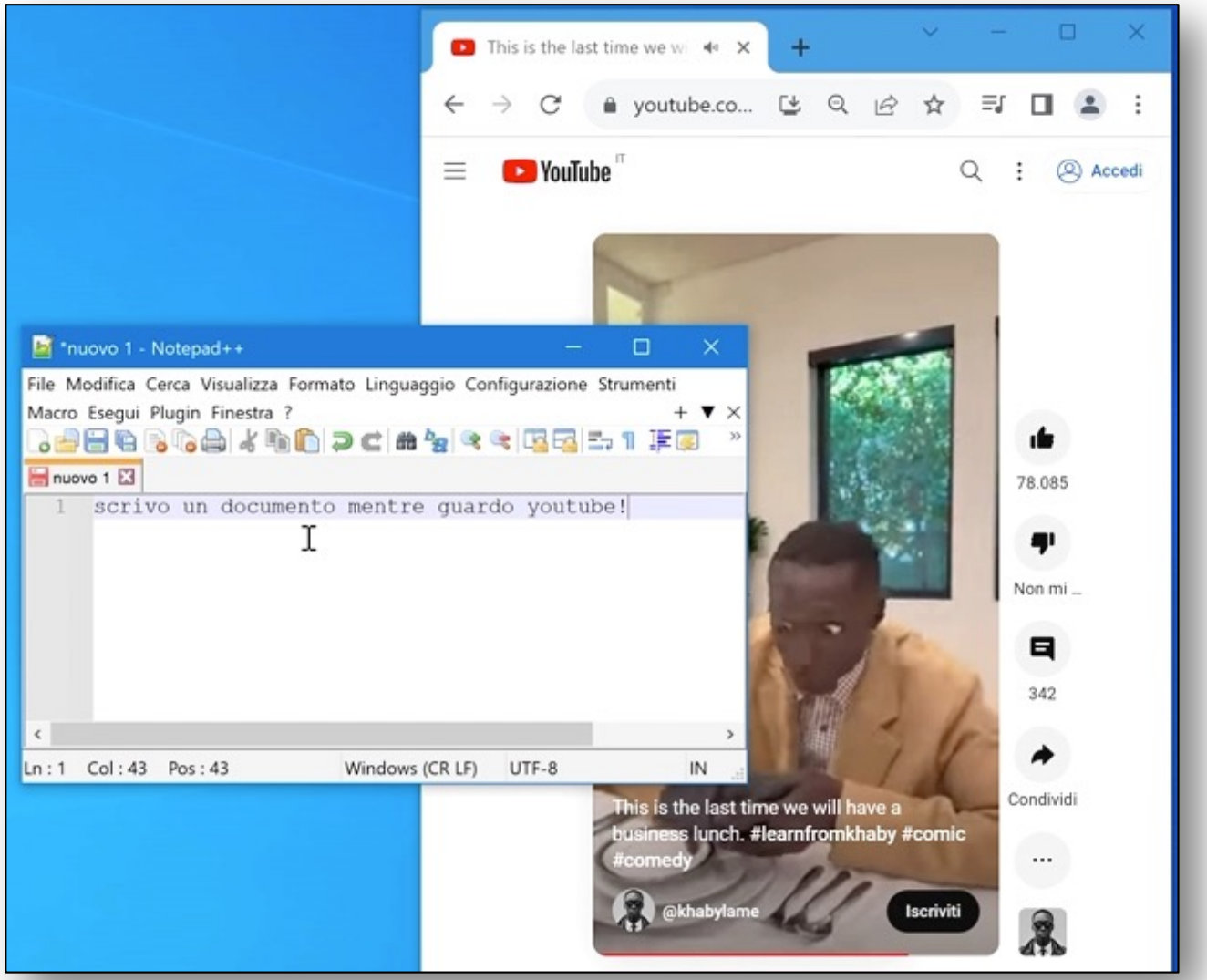
Time Sharing - Esempio



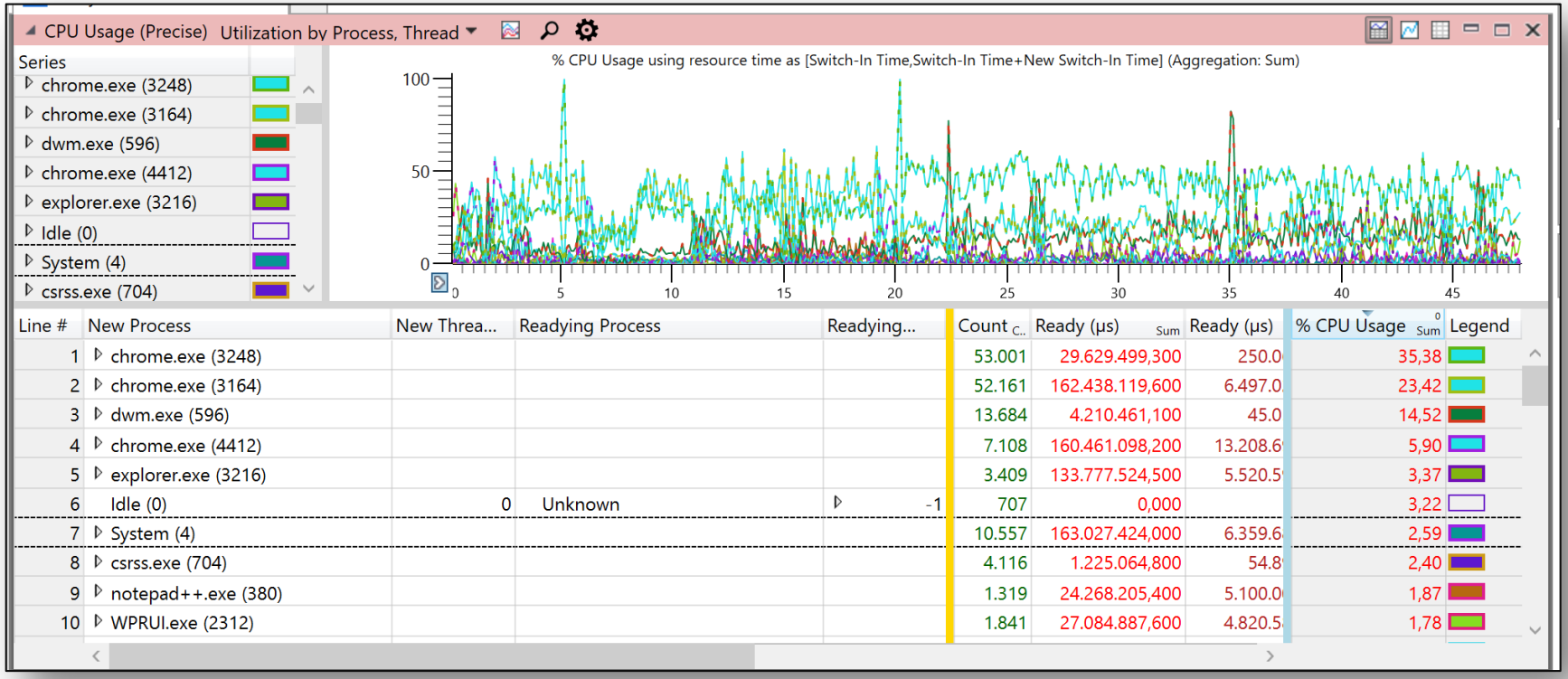
Tool di monitoraggio

- Windows Performance Recorder
- Windows Performance Analyzer

Time Sharing - Esempio



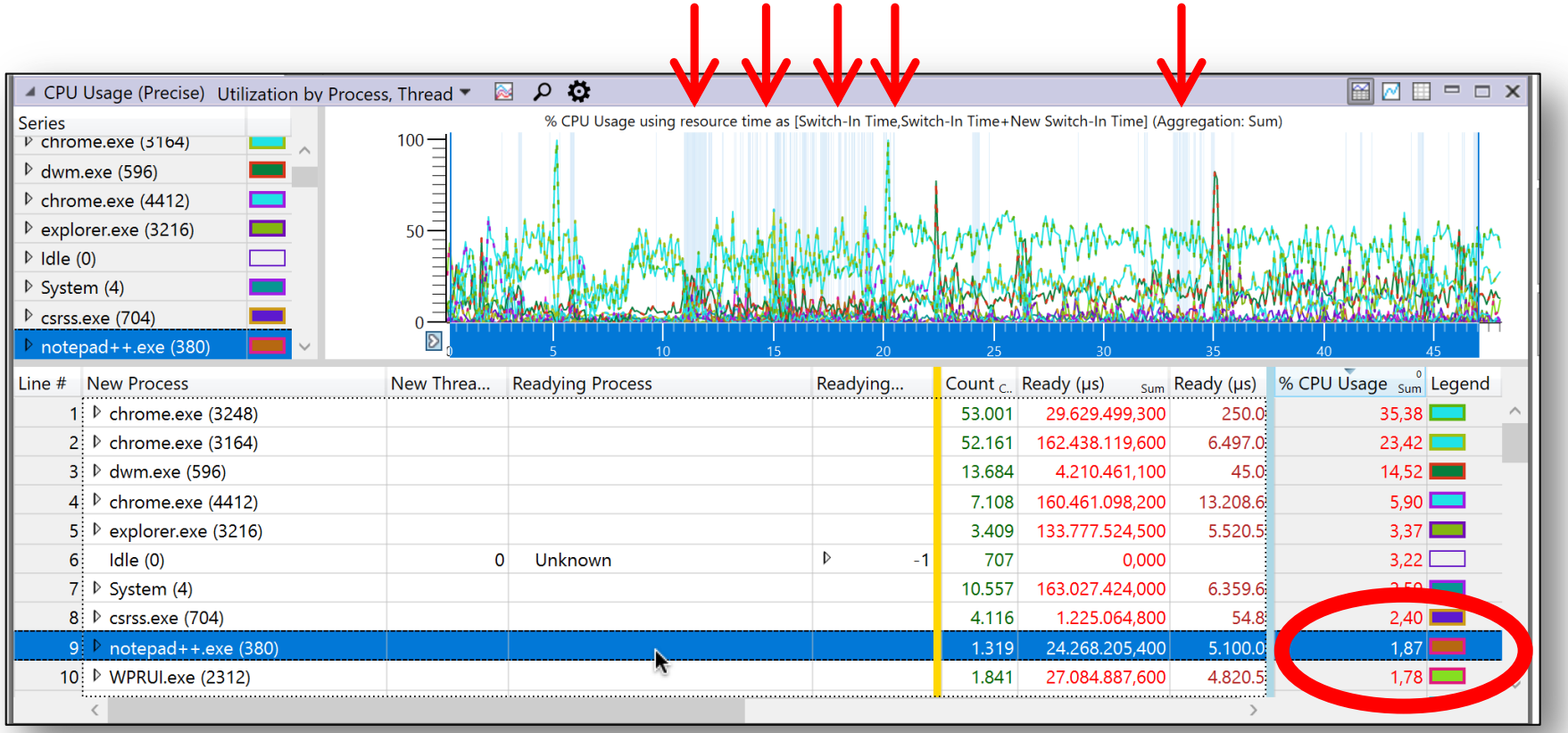
Time Sharing



Line #	New Process	New Threa...	Readying Process	Readying...	Count	Ready (µs)	Ready (µs)	% CPU Usage	Legend
1	chrome.exe (3248)				53.001	29.629.499,300	250.0	35,38	
2	chrome.exe (3164)				52.161	162.438.119,600	6.497.0	23,42	
3	dwm.exe (596)				13.684	4.210.461,100	45.0	14,52	
4	chrome.exe (4412)				7.108	160.461.098,200	13.208.6	5,90	
5	explorer.exe (3216)				3.409	133.777.524,500	5.520.5	3,37	
6	Idle (0)	0	Unknown	-1	707	0,000		3,22	
7	System (4)				10.557	163.027.424,000	6.359.6	2,59	
8	csrss.exe (704)				4.116	1.225.064,800	54.8	2,40	
9	notepad++.exe (380)				1.319	24.268.205,400	5.100.0	1,87	
10	WPRUI.exe (2312)				1.841	27.084.887,600	4.820.5	1,78	

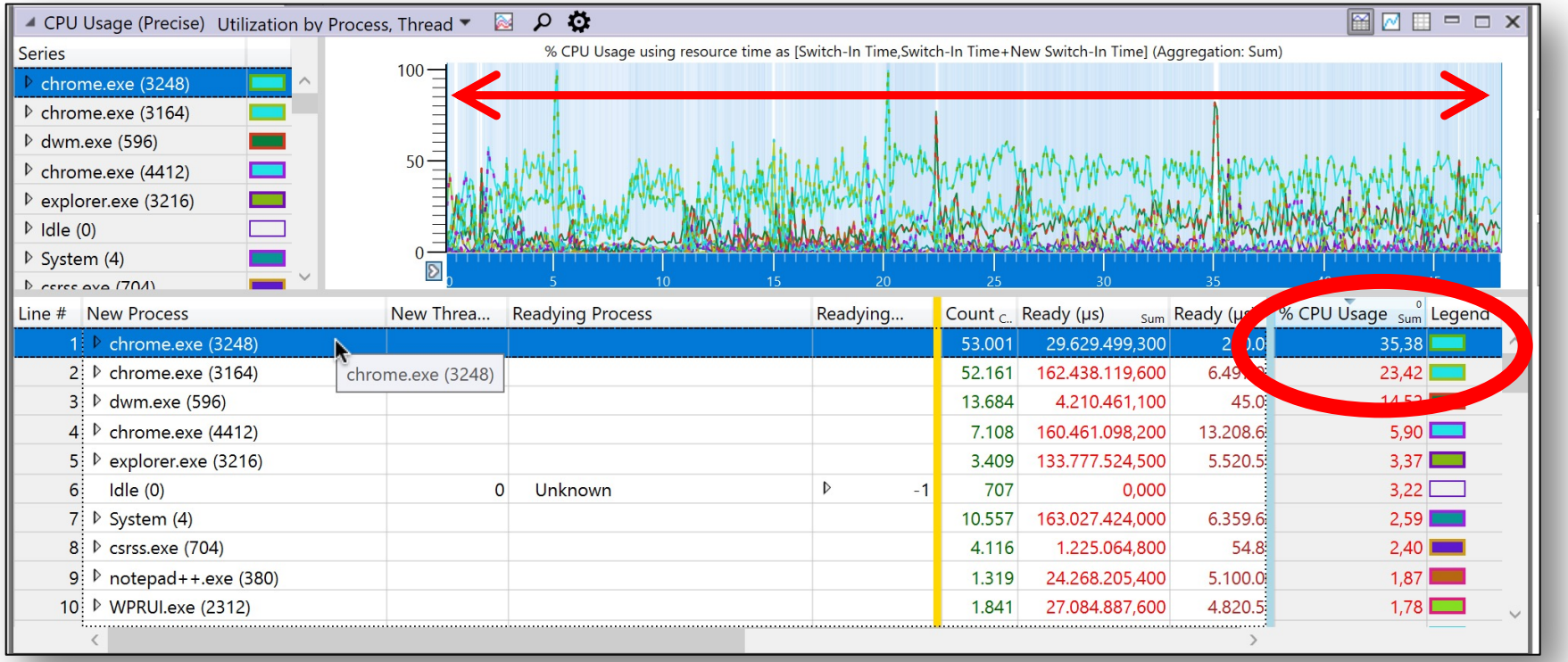
Centinaia di programmi stanno eseguendo, con solo 1 CPU!

Time Sharing



Notepad++ esegue sporadicamente
(le sottili linee azzurre verticali)

Time Sharing

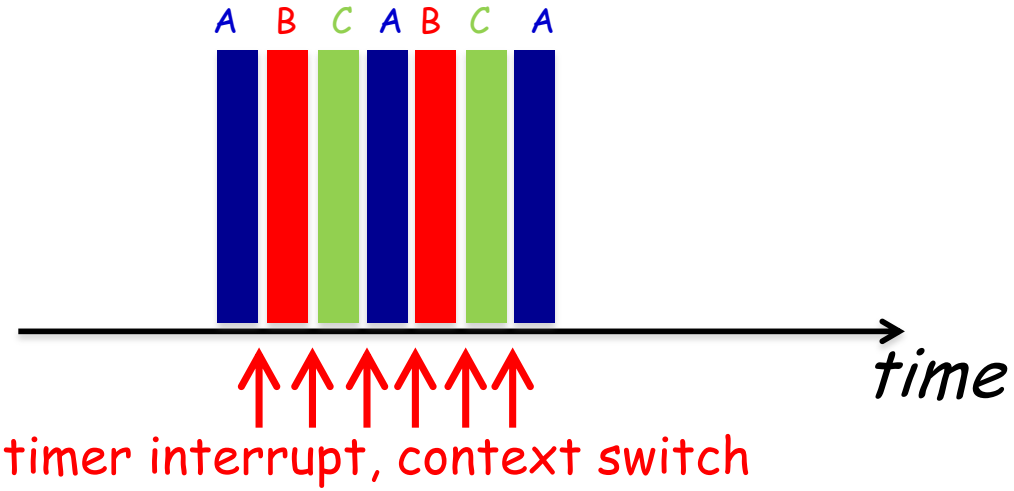


Chrome utilizza maggiormente la CPU

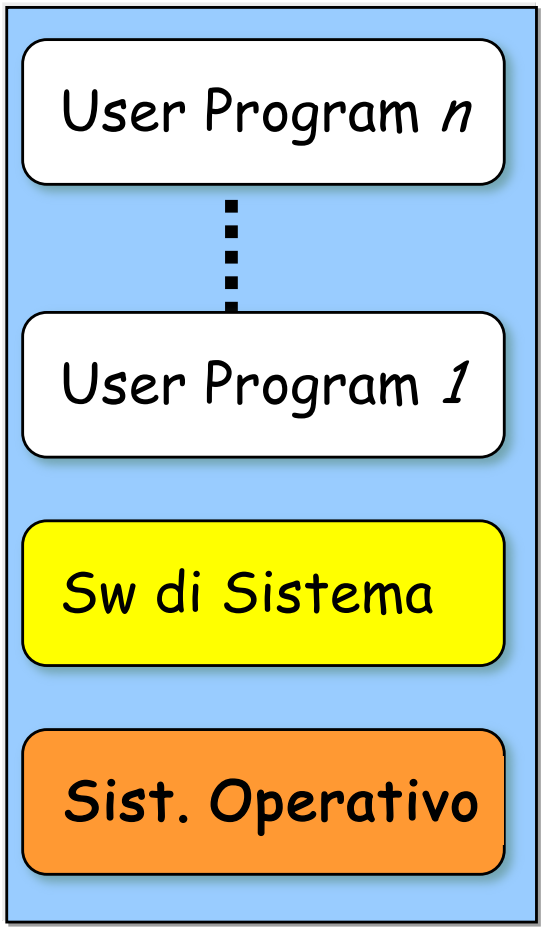
(le sottili linee azzurre verticali)

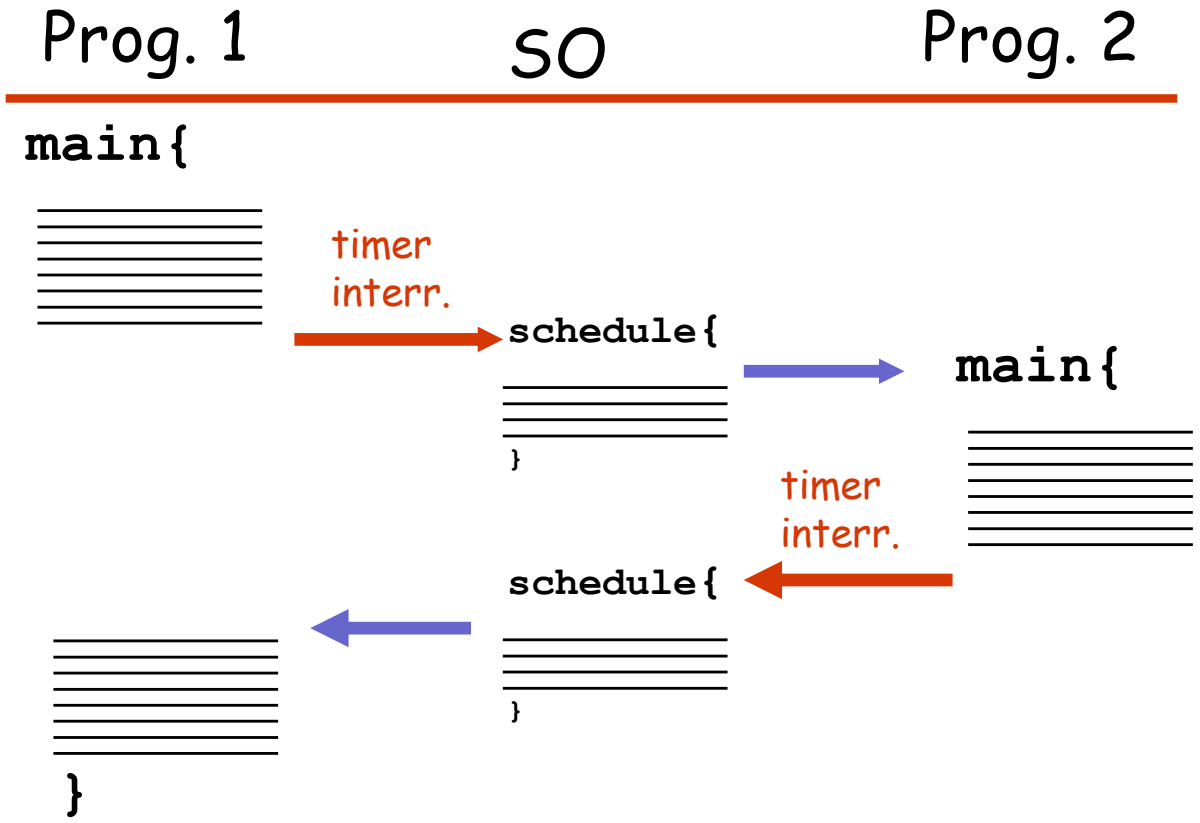
Time Sharing

- In che modo si realizza?
  - Un **timer** innesca periodicamente un **interrupt**
  - Allo scadere, il SO **rimuove dalla CPU** il **programma corrente**, e ne pone in esecuzione un altro ("context switch")



Time Sharing





Il tempo di esecuzione del SO rappresenta un "overhead" di sistema

Time Sharing

- Sistemi **time sharing**:
  - **Vantaggio**: L'utente opera come se avesse un sistema dedicato, in modo **interattivo**
  - **Svantaggio**: **Overhead** introdotto dal S.O

Esempio: Ipotizzando che
- ogni **quanto di tempo dura 10 ms**
- ogni **context switch dura 1 ms (un caso estremo!)**

viene **"sprecato" circa il 10%** del tempo di CPU per la gestione del SO (invece che fare altro lavoro utile)

Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	Time Sharing
Obiettivo principale	Massimizzare l'**uso delle risorse**	Minimizzare i **tempi di risposta**
Modalità di utilizzo	Comandi di controllo forniti insieme al job al momento del **caricamento**	Comandi forniti tramite **interfaccia interattiva** (grafica oppure console testuale)

Personal Computing

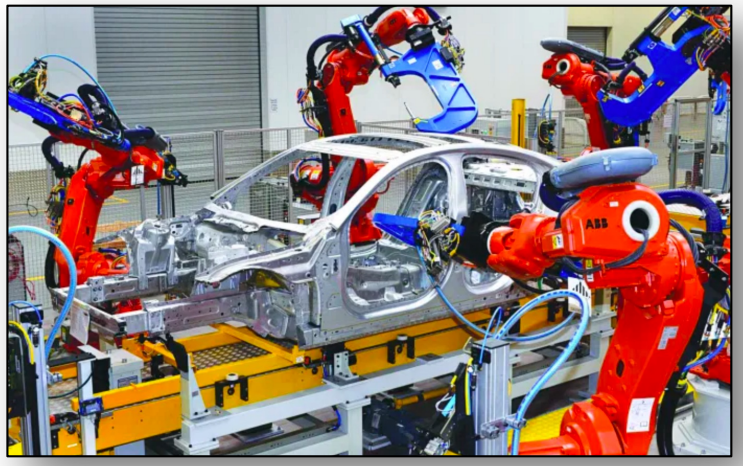
- Personal Computer (PC)
  - Primi anni '80
  - Tipicamente single-user
  - Enfasi sulle interfacce utente e interattività

- Evoluzione
  - **Inizialmente**: SO mono-utente e mono-task (es. MS-DOS)
  - **Oggi**: Sistemi **multiutente e time-sharing** (Windows, MacOS)

Embedded computing

- Applicazioni **embedded** (es., sistemi industriali, automotive, Internet of Things, ...)

- SO di tipo **real-time**
  - Applicazioni che lavorano con sistemi fisici
  - Il corretto funzionamento dei sistemi è legato al tempismo dei comandi dati dalla applicazione

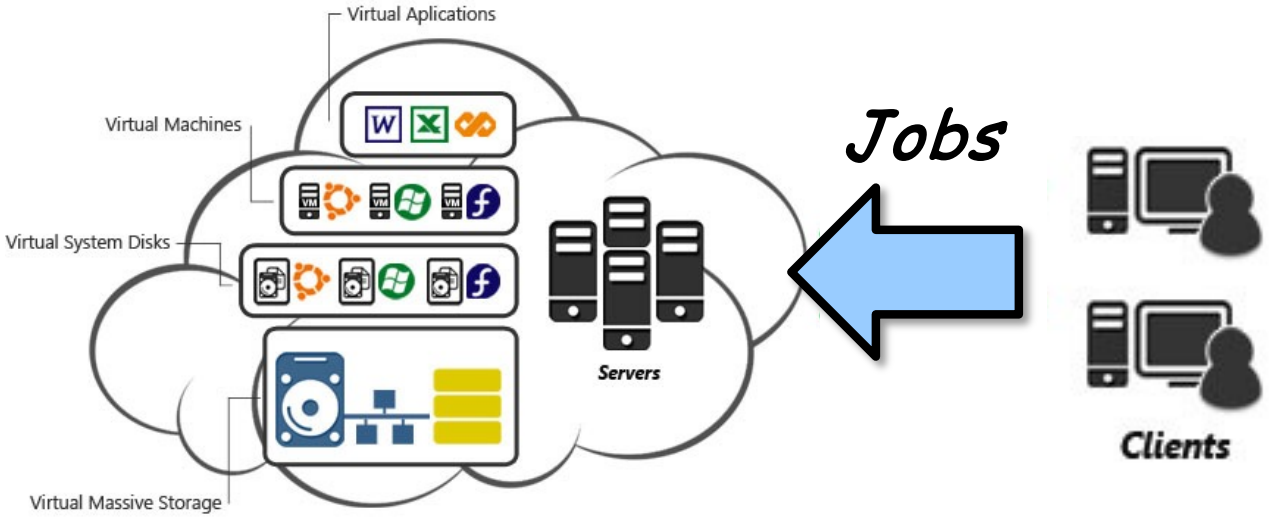


Mobile computing

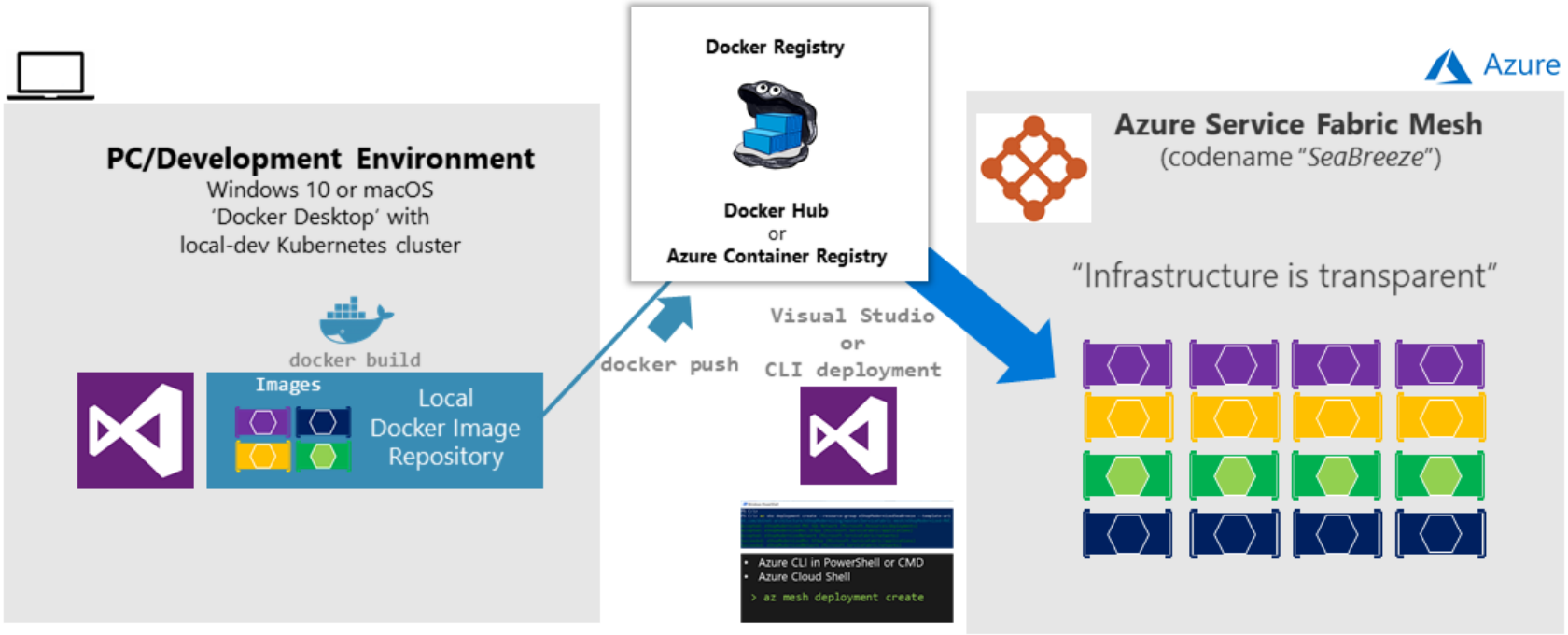
- I **mobile device** pervadono ormai la nostra società (smartphones, wearables, ...)
  - Social networking
  - Digital assistants
  - Electronic payments
  - GPS navigation, fitness, ...
- I **SO mobile** sono progettati per:
  - Sicurezza e privacy
  - Efficienza energetica
  - Multimedia

Distributed, cloud, edge computing

- **Distributed computing**: ulteriore evoluzione nella virtualizzazione delle risorse
  - Il sistema è formato da **gruppi (cluster) di macchine**
  - Un **orchestratore** distribuisce automaticamente nel cluster le applicazioni degli utenti



Distributed, cloud, edge computing

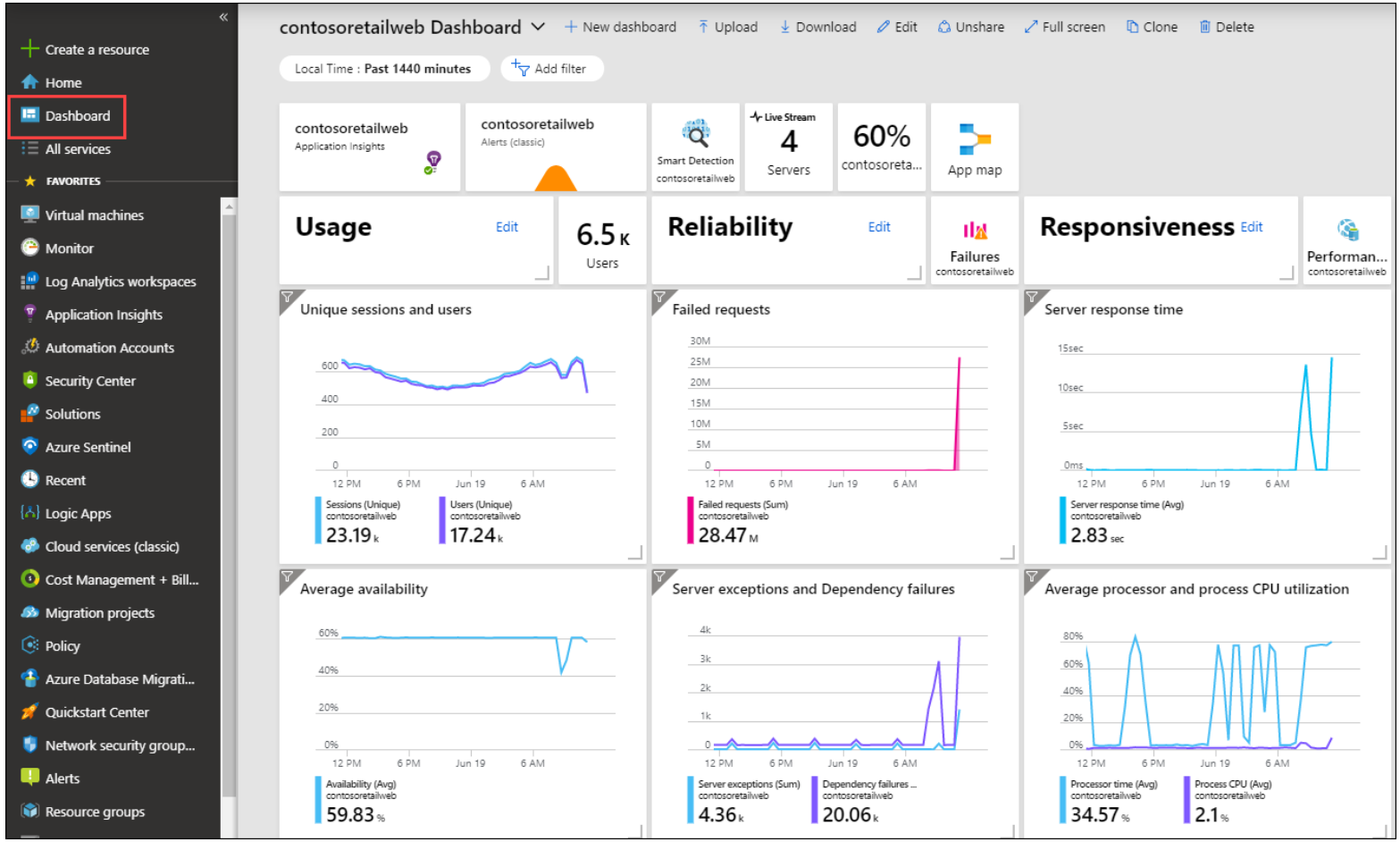


https://learn.microsoft.com/it-it/dotnet/architecture/containerized-lifecycle/design-develop-containerized-apps/orchestrate-high-scalability-availability

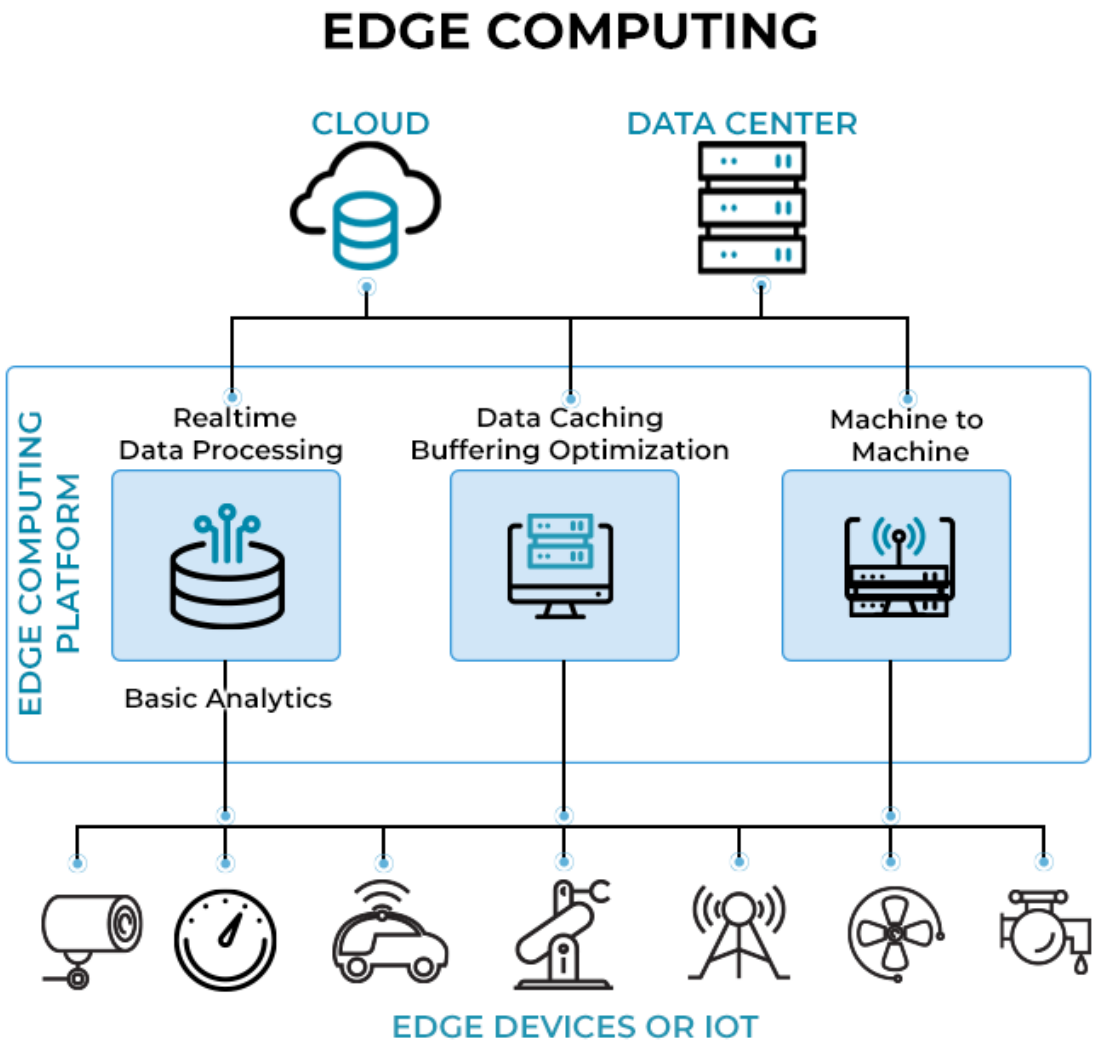
Distributed, cloud, edge computing

- Nel **cloud computing**, i provider mettono a disposizione i loro data center "su domanda"
- L'utente paga solo per le risorse usate (pay-per-use)

Distributed, cloud, edge computing



Distributed, cloud, edge computing



Quiz

1. Quali di queste affermazioni sono vere per un sistema "time sharing" (selezionare più di una):

- [ ] riduce il tempo di caricamento dei programmi
- [ ] minimizza il tempo di risposta
- [ ] raggruppa i programmi in lotti
- [ ] il SO alterna frequentemente le applicazioni
- [ ] ha un "overhead" dovuto al SO

https://forms.office.com/r/jaJtcYkVrY