



Introduzione al Corso

Sistemi Digitali

A.A. 2024/2025

Stefano Mattoccia, Fabio Tosi

Università di Bologna

Orario delle Lezioni

Periodo di svolgimento delle lezioni: 18/09/2024 - 20/12/2024

Orario Ufficiale ([Link](#))

- **Mercoledì**, 12:30 - 15:30 (AULA 5.7) - Modulo 1
 - Inizio 12:45?
- **Venerdì**, 12:30 - 15:30 (AULA 5.6) - Modulo 2
 - Inizio 12:45?

Docenti

- **Stefano Mattoccia - Modulo 1**

- E-mail: stefano.mattoccia@unibo.it
- Web: <http://vision.deis.unibo.it/~smatt/Site/Home.html>
- Ricevimento su richiesta (via email)

- **Fabio Tosi - Modulo 2**

- E-mail: fabio.tosi5@unibo.it
- Web: <https://fabiotosi92.github.io/>
- Ricevimento su richiesta (via email)

Materiale Didattico

Informazioni, Comunicazioni, Lucidi

- Comunicazioni in **broadcast** attraverso messaggi in Virtuale
- **Per entrambi i moduli:** il materiale didattico sarà disponibile in Virtuale

Modalità Di Esame (8 CFU)

Struttura dell'Esame

1. Progetto di gruppo (max 2 persone) con discussione
2. Esame orale individuale (su tutti i temi trattati nel corso)

Dettagli

- **Progetto:**
 - Proposta dell'idea (soggetta ad approvazione)
 - Discussione (congiunta con i membri del gruppo)
 - Punteggio max: 18 punti
- **Esame Orale:**
 - Individuale per ogni membro del gruppo, nello stesso appello
 - Orale sui due moduli:
 - Modulo 1 (max 6 punti)
 - Modulo 2 (max 6 punti)

Valutazione Finale

- **Voto Totale** = Punteggio Progetto + Punteggio Orale Modulo 1 + Punteggio Orale Modulo 2
- È possibile rifiutare **una sola volta** un esito positivo

Attività Progettuale (Opzionale, 3 CFU)

Descrizione

- L'attività progettuale (soggetta ad approvazione) consiste nello sviluppo di progetti avanzati o approfondimenti relativi ai temi trattati nel corso.

Modalità di Presentazione

- Presentazione **non vincolata** all'esame
- Scelta della **modalità**:
 - **Opzione A:** Presentazione online
 - **Opzione B:** Presentazione durante l'esame in presenza

Opzioni di Svolgimento

- **Lavoro individuale:** Possibilità di svolgere l'attività da soli
- **Lavoro di gruppo:**
 - Possibilità di collaborare con lo stesso gruppo formato per l'esame
 - L'attività progettuale vale per tutti i membri del gruppo
- **Scelta del Progetto:**
 - **Opzione A:** Estensione del progetto d'esame esistente
 - **Opzione B:** Sviluppo di un nuovo progetto originale

Date degli Esami

Calendario

- **6 Appelli Totali**
 - **3 appelli** tra Gennaio e Febbraio
 - **2 appelli** tra Giugno e Luglio
 - **1 appello** a Settembre

Introduzione al Calcolo Parallelo

Il Progresso delle CPU e la Necessità di Calcolo Parallelo

1. Origini dell'Informatica

- La ricerca di **velocità nel calcolo** ha guidato lo sviluppo tecnologico fin dagli albori
- **Legge di Moore**: previsione dell'aumento esponenziale della potenza di calcolo

2. Crescita dei Dati e Complessità

- **Complessità crescente** dei problemi da risolvere (es. simulazioni climatiche, analisi genomica, etc)
- Limiti delle architetture sequenziali tradizionali sempre più evidenti

3. Transizione verso Architetture Avanzate

- Adozione di **sistemi multi-core** per superare i limiti della frequenza di clock
- **Utilizzo di GPU** per calcoli massivamente paralleli

4. Calcolo ad Alte Prestazioni (HPC)

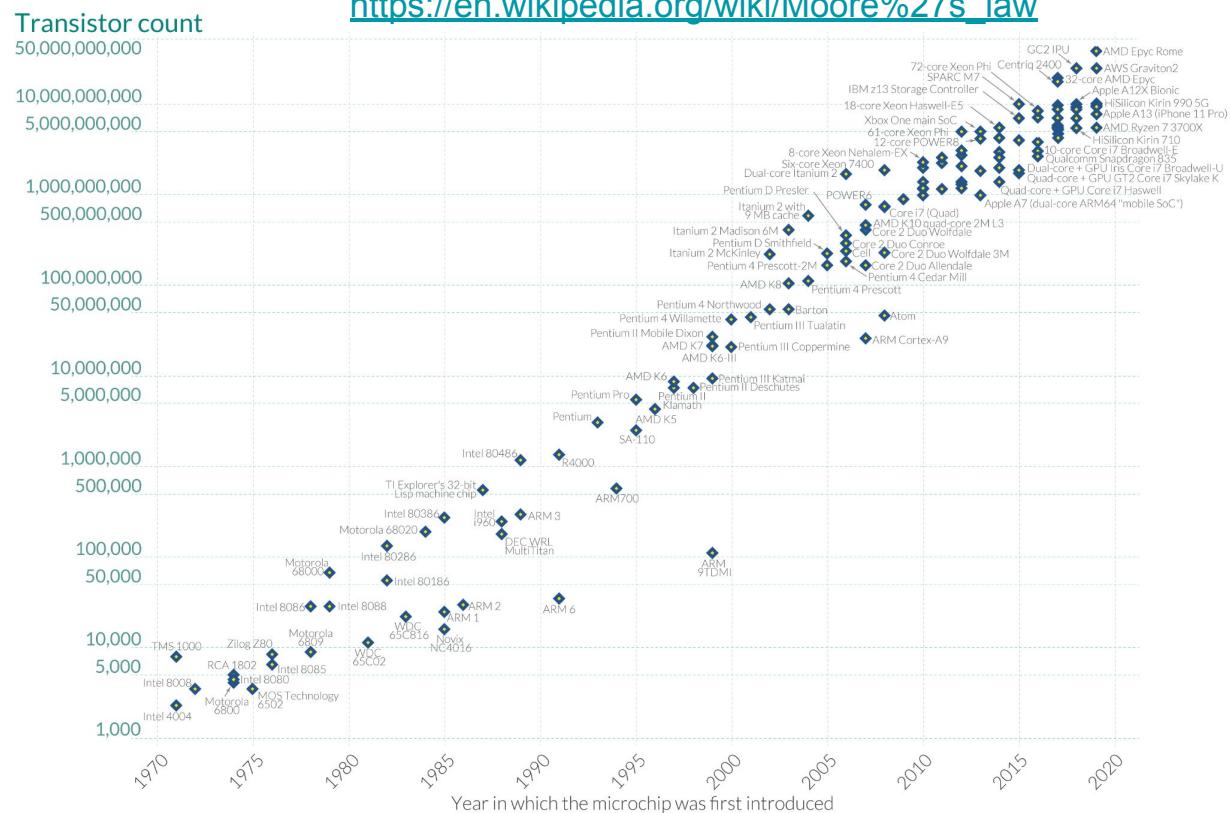
- **Supercomputer** e **cluster** per affrontare problemi scientifici complessi
- Integrazione di tecnologie diverse (CPU, GPU, FPGA) per massimizzare le prestazioni

5. Intelligenza Artificiale e Big Data

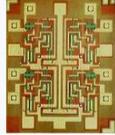
- **Machine Learning** e **Deep Learning** richiedono enormi capacità di calcolo
- Il **calcolo parallelo** diventa **imprescindibile** per addestrare modelli AI complessi

Legge di Moore (1965)

«Il numero di transistor su un circuito integrato raddoppia ogni due anni con un aumento minimo dei costi.»



Semiconductor device fabrication



MOSFET scaling (process nodes)

| | |
|--------|--------|
| 20 µm | - 1968 |
| 10 µm | - 1971 |
| 6 µm | - 1974 |
| 3 µm | - 1977 |
| 1.5 µm | - 1981 |
| 1 µm | - 1984 |
| 800 nm | - 1987 |
| 600 nm | - 1990 |
| 350 nm | - 1993 |
| 250 nm | - 1996 |
| 180 nm | - 1999 |
| 130 nm | - 2001 |
| 90 nm | - 2003 |
| 65 nm | - 2005 |
| 45 nm | - 2007 |
| 32 nm | - 2009 |
| 28 nm | - 2010 |
| 22 nm | - 2012 |
| 14 nm | - 2014 |
| 10 nm | - 2016 |
| 7 nm | - 2018 |
| 5 nm | - 2020 |
| 3 nm | - 2022 |

Future
2 nm ~ 2024

Legge di Moore (1965)

«Il numero di transistor su un circuito integrato raddoppia ogni due anni con un aumento minimo dei costi.»

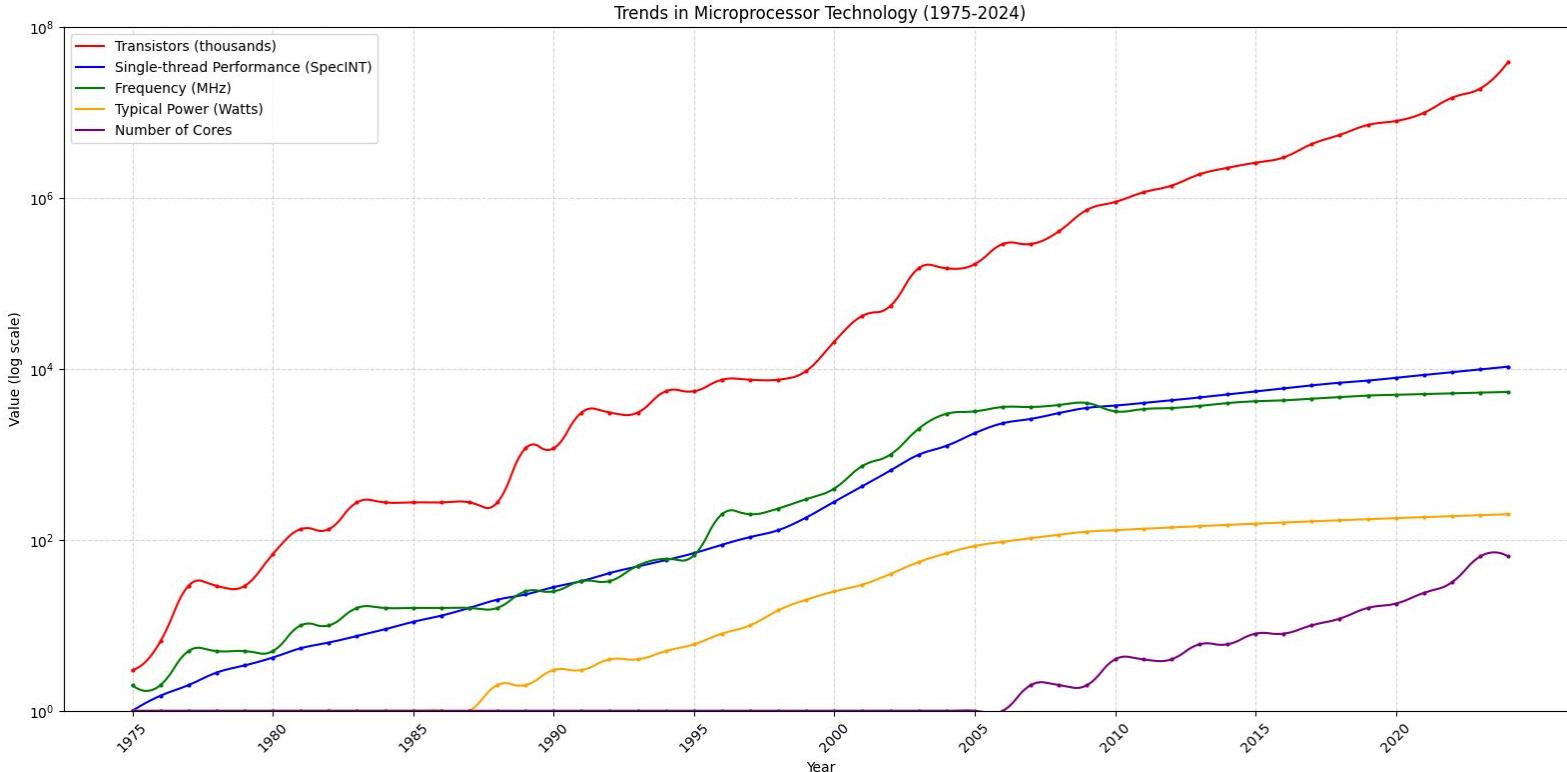


“Do not rewrite software, buy a new machine!”



L'Evoluzione dei Microprocessori (1975-2024)

- Mentre il numero di transistor e le prestazioni sono cresciuti esponenzialmente, la **frequenza di clock** e il **consumo energetico** hanno raggiunto un **plateau**, portando all'era del multi-core per continuare l'avanzamento delle prestazioni.



Limiti Strutturali delle CPU



Limite della Frequenza di Clock

Aumento da 1 MHz a 1-4 GHz in 30 anni (1000 volte), ma ora ha raggiunto un *plateau*.

Incremento: $4 \text{ GHz} / 1 \text{ MHz} = 4000$ volte



Thermal Wall

La potenza dissipata cresce con il cubo della frequenza, causando problemi di surriscaldamento.

Formula: $P \approx \alpha CLV^2 f$

Dove α = probabilità di switch, CL = capacità, $V \propto f$, quindi $P \propto f^3$



Limite Fisico della Velocità

A 4 GHz, la luce percorre una distanza limitata in un ciclo di clock.

Calcolo:

$$c \approx 3 \times 10^8 \text{ m/s}$$

$$T = 1/f = 1/(4 \times 10^9) \text{ s}$$

$$d = c \times T \approx 7.5 \text{ cm}$$



Soluzioni: Multi-core e Beyond

Adozione di architetture *multi-core* e ricerca di nuove tecnologie.

Performance teorica: $P = N \times f$

Dove N = numero di core, f = frequenza di clock

Evoluzione delle Esigenze Tecnologiche nel Tempo



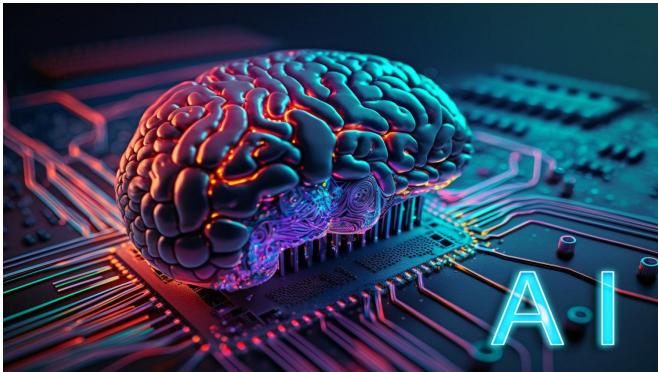
Anni '90- Internet & Multimedia



2000 - Giochi 3D



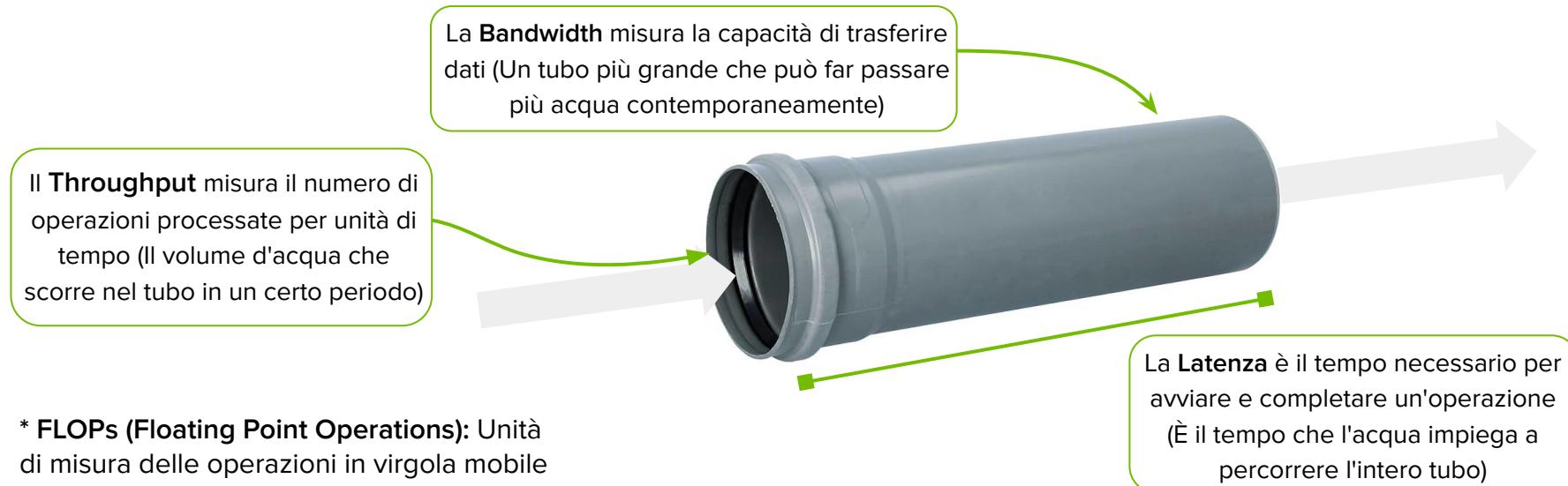
2010 - Cloud Computing



Oggi - AI e Big Data

Obiettivi degli Avanzamenti Architetturali nei Computer

- **Ridurre la Latenza (Latency)**
 - Tempo necessario per completare un'operazione (tipicamente espresso in nanosecondi o microsecondi).
- **Aumentare la Banda (Bandwidth)**
 - Quantità massima di dati che possono essere trasferiti per unità di tempo (espressa in MB/s, GB/s o Tbps).
- **Aumentare il Throughput**
 - Numero di operazioni completate per unità di tempo (espresso in MFLOPS* o GFLOPS*).



Computazione Parallelia

Negli ultimi decenni, l'interesse per la computazione parallela è cresciuto, puntando a migliorare la **velocità di calcolo**.

Cosa si intende per computazione parallela?

- La computazione parallela è una forma di calcolo in cui **molte operazioni vengono eseguite simultaneamente**.
- L'idea di base è che i problemi complessi possano essere suddivisi in **problemi più piccoli**, risolti poi contemporaneamente (Grandi porzioni di calcolo, sono lo stesso calcolo!).

Prospettiva del Programmatore

- La sfida è mappare i calcoli simultanei sulle risorse disponibili (core), risolvendo le parti del problema in parallelo facendo attenzione all'ordine delle operazioni e alla dipendenza tra i dati.

Tecnologie Coinvolte

- **Architettura dei Computer (Hardware)**: Supporta il parallelismo a livello architettonico.
- **Programmazione Parallelia (Software)**: Risolve problemi utilizzando pienamente la potenza dell'hardware.

Nota

- Per eseguire calcoli paralleli, l'hardware **deve supportare** l'esecuzione simultanea di più processi o thread o consentire l'elaborazione contemporanea di più dati.

Legge di Amdahl

La Legge di Amdahl è un principio fondamentale nel calcolo parallelo che descrive il **limite delle prestazioni** ottenibili quando si parallelizza una parte di un programma.

Formula

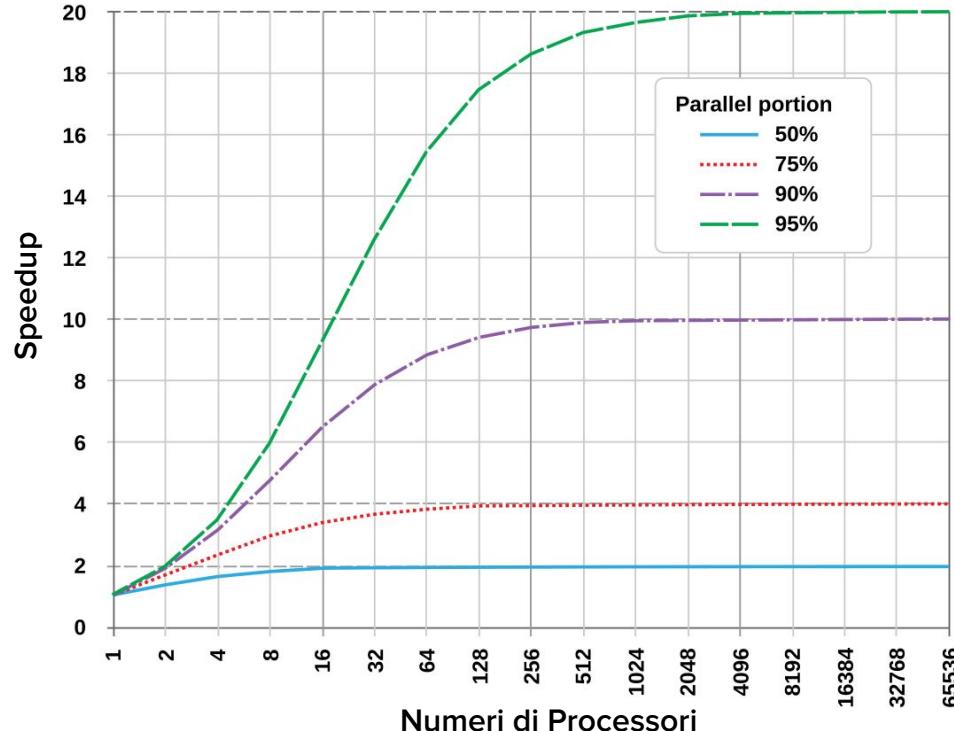
$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

Durata parte sequenziale Durata parte parallela

- S:** Accelerazione massima ottenibile.
- P:** Porzione parallelizzabile del programma
- N:** Numero di processori

Interpretazione

- Se una parte del programma non è parallelizzabile, il **miglioramento è limitato**, anche con molti processori.
- Con processori infiniti, l'accelerazione massima è $\frac{1}{1-P}$

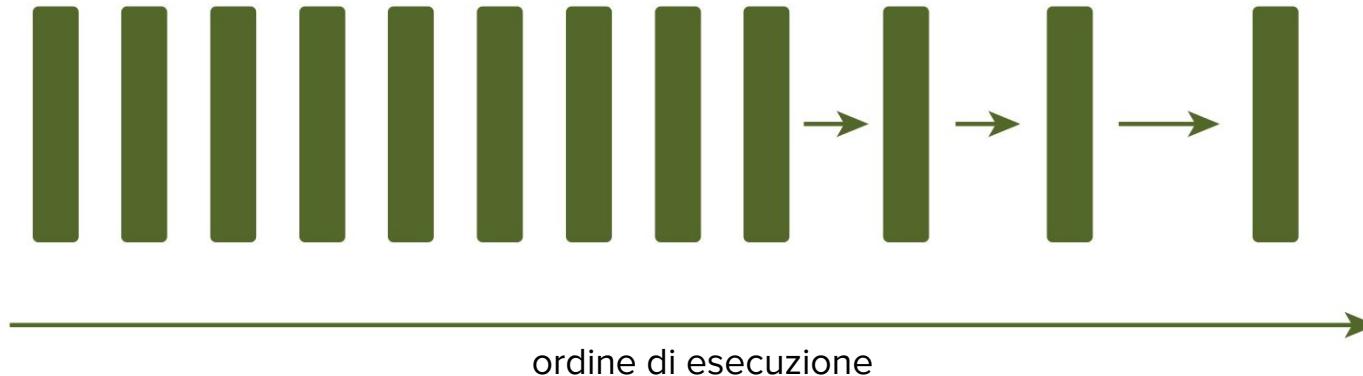


Programmazione Sequenziale vs. Parallelia

Programmazione Sequenziale

- I calcoli vengono eseguiti in un ordine fisso, uno dopo l'altro.
- Ogni istruzione dipende dal **completamento** di quella precedente.
- Rappresentazione tipica di programmi tradizionali.

Il problema è suddiviso in piccoli pezzi di calcolo



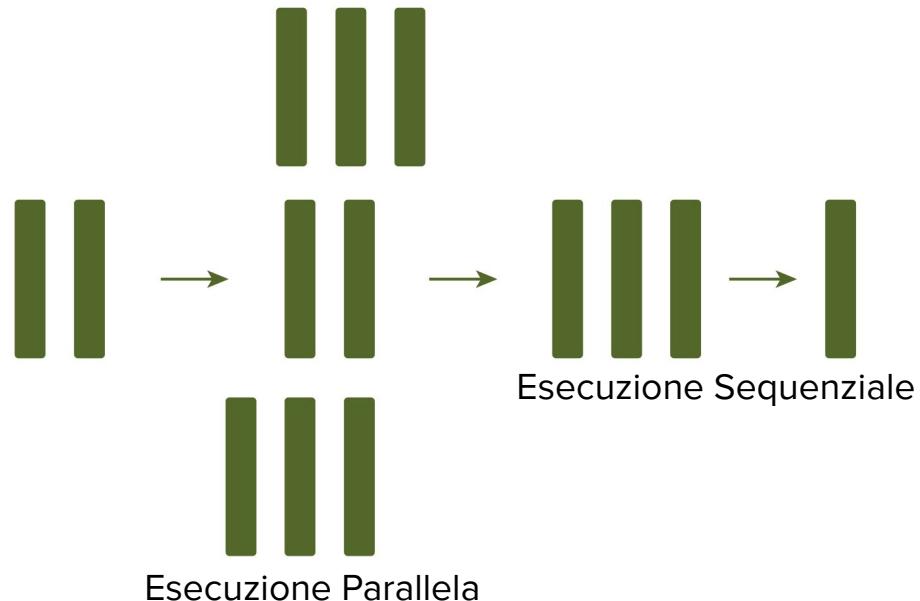
Programmazione Sequenziale vs. Parallelia

Programmazione Parallelia

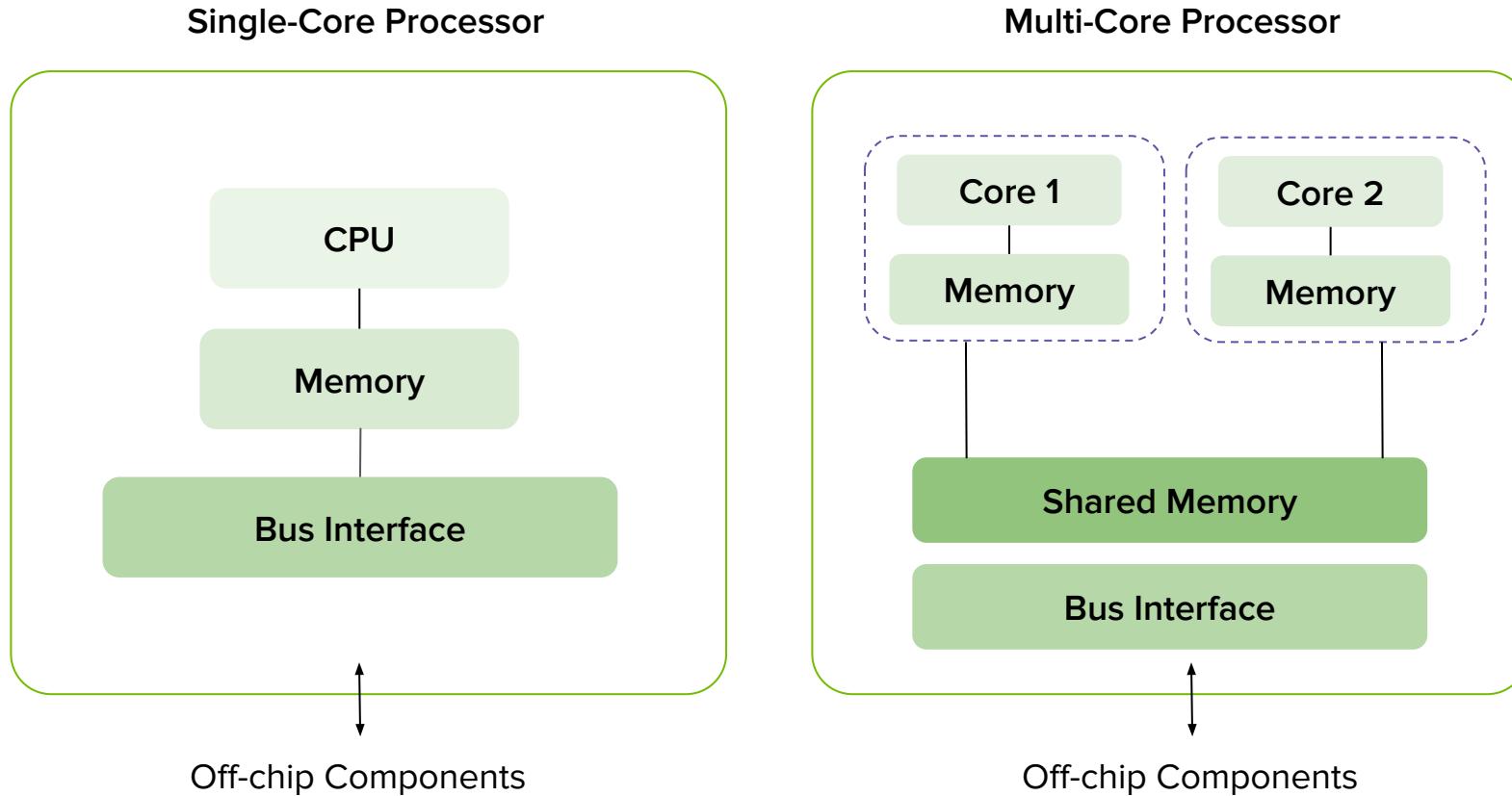
- I calcoli vengono **suddivisi in task** che possono essere eseguiti **contemporaneamente**.
- I **task indipendenti**, senza dipendenze di dati, offrono il maggiore potenziale di parallelismo.
- I programmi **parallel**i possono, e spesso lo fanno, contenere anche **parti sequenziali**.

Dipendenze tra Dati

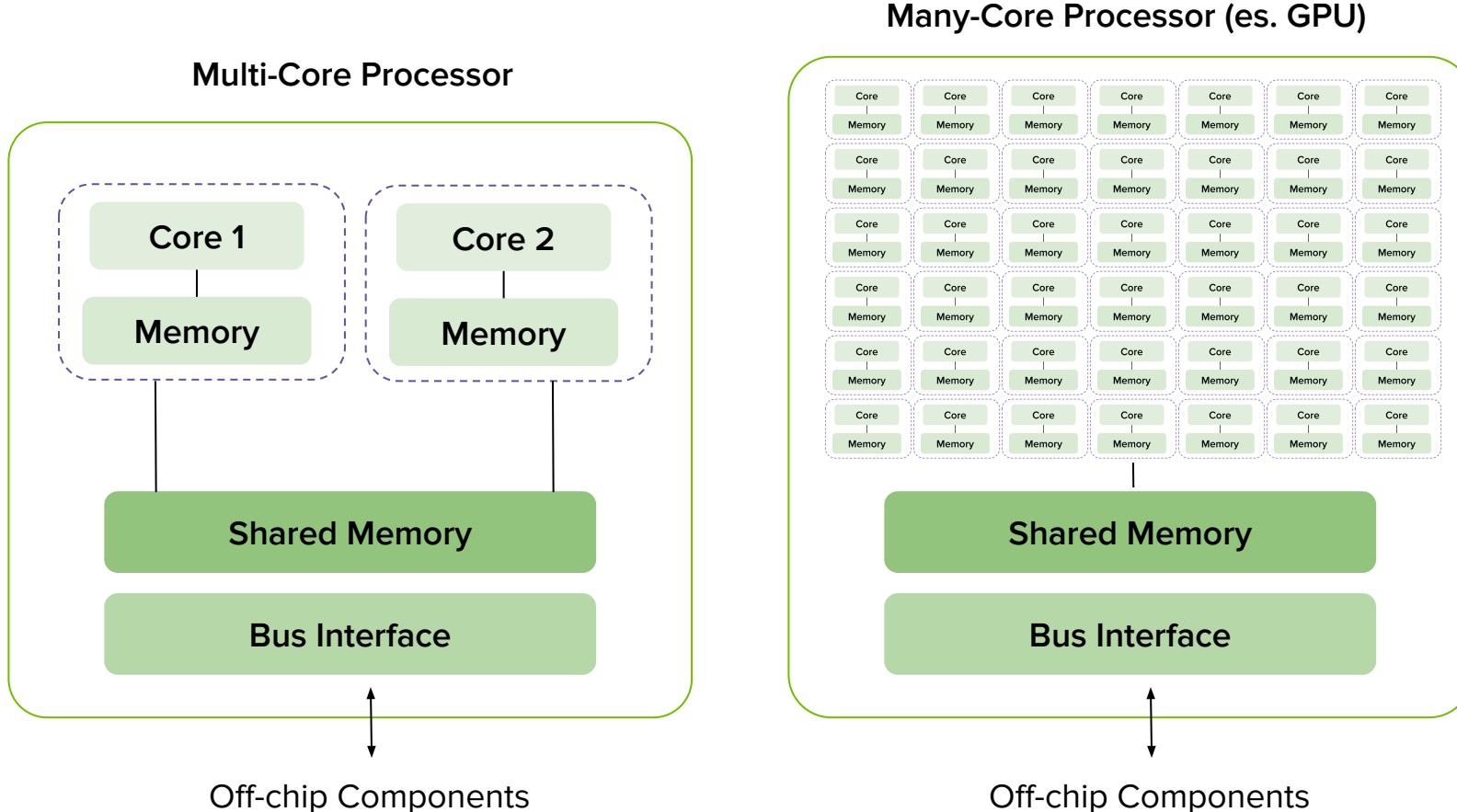
- Vincolo: ottenere gli stessi risultati della versione sequenziale.
- Una **dipendenza di dati** si verifica quando un'istruzione **richiede** i dati prodotti da un'istruzione precedente.
- Le dipendenze **limitano il parallelismo**, poiché impongono un ordine di esecuzione.
- L'**analisi delle dipendenze** è cruciale per implementare algoritmi paralleli efficienti



Single-Core vs Multi-Core Processors



Multi-Core vs Many-Core Processors



Core GPU vs Core CPU

- Nonostante i termini "multicore" e "many-core" siano usati per etichettare le architetture CPU e GPU, un core CPU è molto diverso da un core GPU.

Core CPU

- Unità di controllo **complessa** per gestire flussi di istruzioni variabili
- **Ampia cache** per ridurre la latenza di accesso alla memoria
- Unità di predizione delle **diramazioni sofisticate**
- Esecuzione **fuori ordine** per ottimizzare l'utilizzo delle risorse

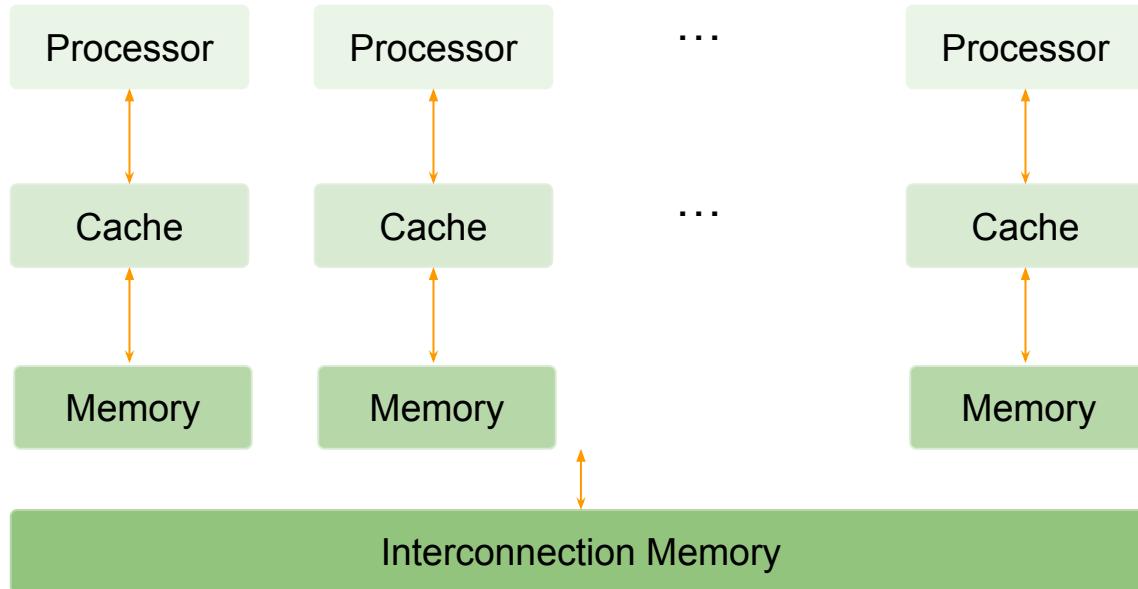
Core GPU

- Unità di controllo **semplificata** per gestire operazioni ripetitive
- **Cache più piccola**, compensata da alta larghezza di banda di memoria
- Minor enfasi sulla predizione delle diramazioni
- Esecuzione **in-order** per massimizzare il throughput

Architetture di Memoria nei Computer

Multi-nodo con Memoria Distribuita

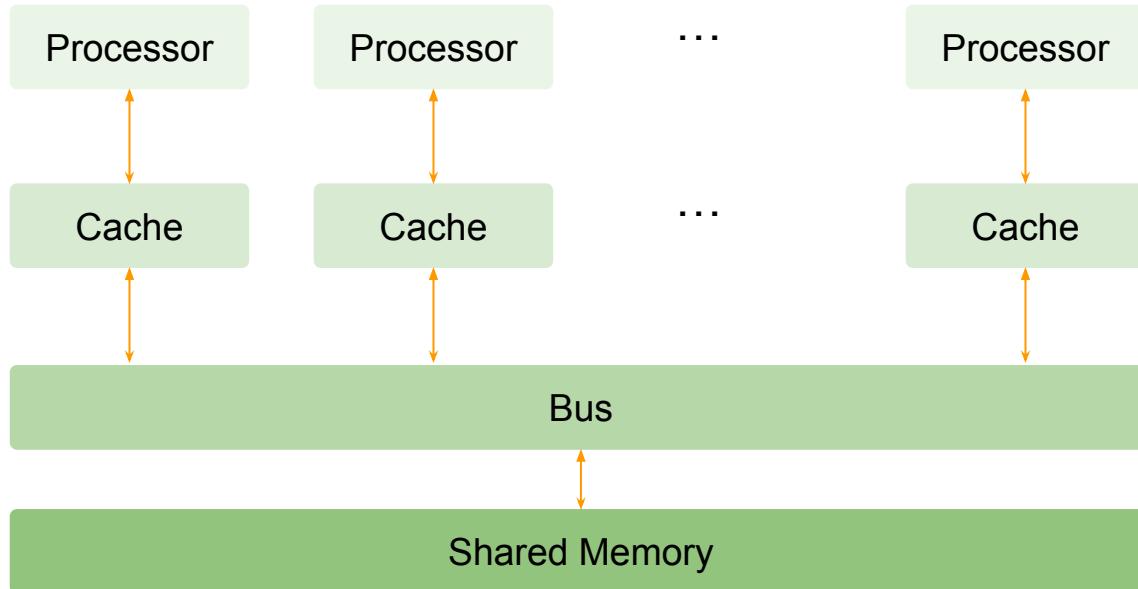
- Sistemi composti da **multi processori** connessi in rete (*cluster*).
- Ogni processore ha la propria **memoria locale**.
- La comunicazione avviene **attraverso la rete**.



Architetture di Memoria nei Computer

Multi-nodo con Memoria Condivisa

- **Dimensioni tipiche:** da dual-processor a decine o centinaia di processori
- Processori **fisicamente connessi alla stessa memoria** o condividono un link a bassa latenza (es. PCIe)
- Include sistemi multi-core e computer con più chip multi-core.



Esempi di Architetture di Calcolo

Single Core



CPU Intel Pentium 4 - 1 Core

Multi-Core (da 2 a 64 Core)

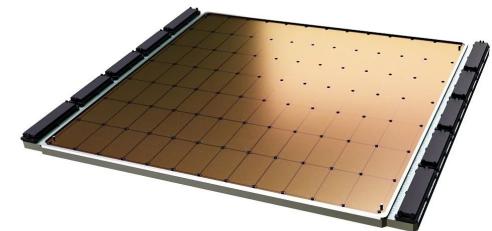


AMD Ryzen Threadripper – 64 core

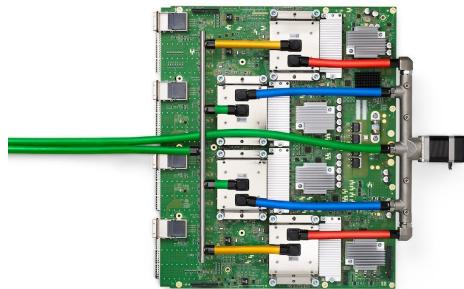


Intel Core i7-9700K – 8 core

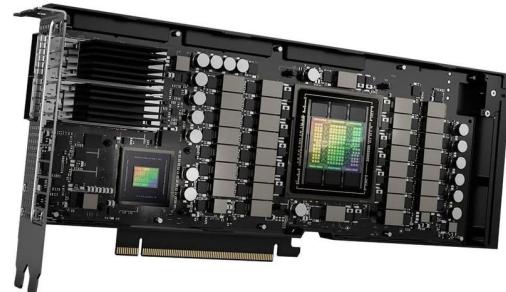
Many-Core (oltre 64 Core)



Google TPU v4 - 2048 Core



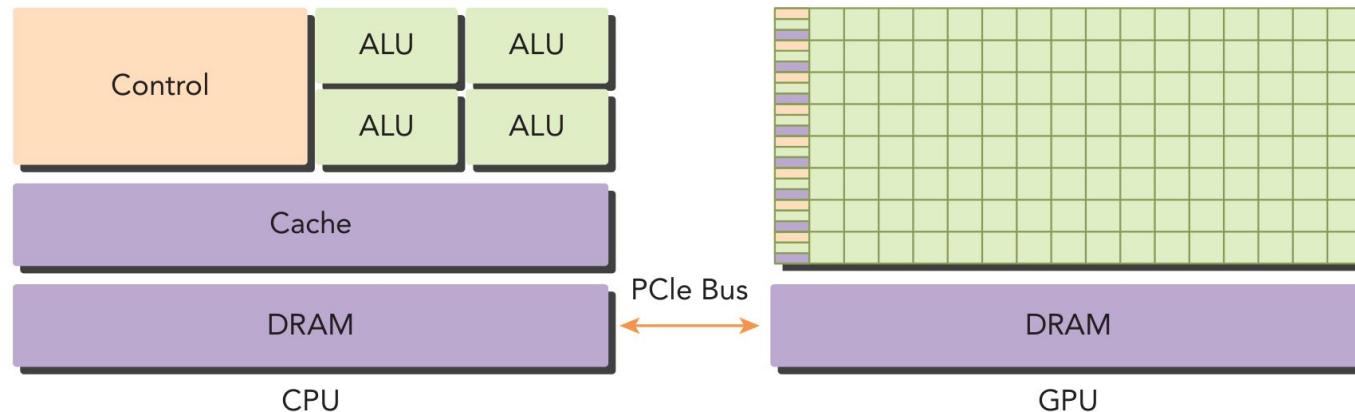
GPU NVIDIA H100 - 14.592 CUDA Core



Cerebras WSE-2 - 850.000 Core

Architetture Eterogenee

- **Cosa è una Architettura Eterogenea?** Un'architettura eterogenea è una struttura di sistema che **integra** diversi tipi di processori o core di elaborazione all'interno dello stesso computer o dispositivo.
- **Ruoli**
 - **CPU (Host)**: gestisce l'ambiente, il codice e i dati
 - **GPU (Device)**: co-processore, accelera calcoli intensivi (hardware accelerator)
- **Connessione**: GPU collegate alla CPU tramite bus **PCI-Express**
- **Struttura del Software**: Applicazioni divise in **codice host** (CPU) e **device** (GPU)



Perchè le Architetture Eterogenee?

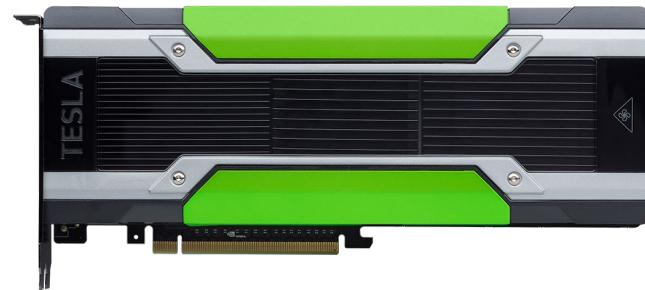
CPU Multi-Core

- Ottimizzato per Latenza:
 - Eccellente per task sequenziali
 - Meno efficiente per parallelismo massiccio



GPU Many-Core

- Ottimizzato per Throughput:
 - Eccellente per parallelismo massiccio
 - Meno efficiente per task sequenziali
- Speedup (legge di Amdahl): $S = \frac{1}{(1-P) + \frac{P}{N}}$

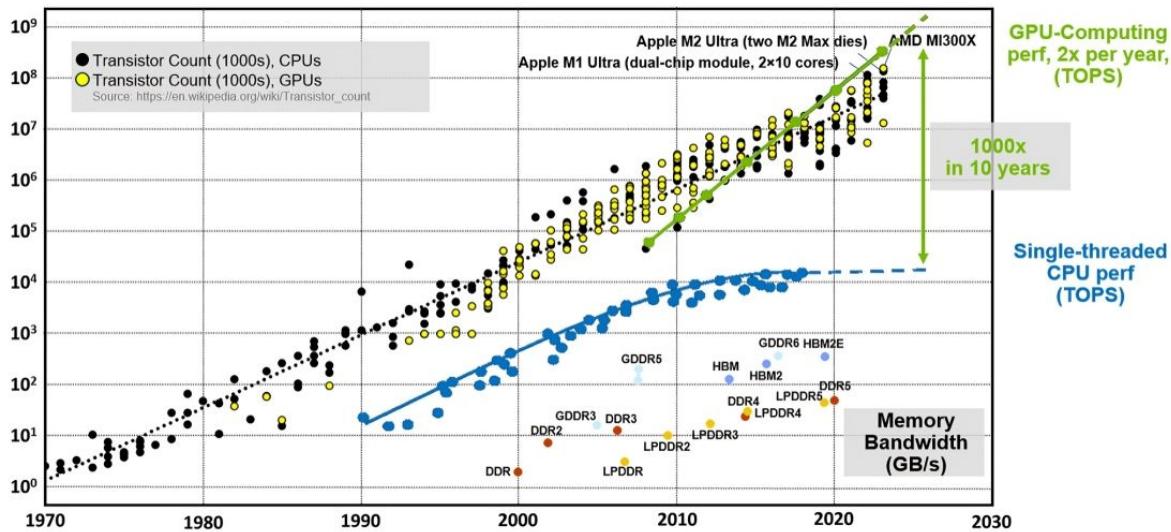


Architettura Eterogena (CPU+GPU)

- ✓ Combina i vantaggi di entrambi
- ✓ Ottimizzazione flessibile
- ✓ Migliori prestazioni complessive



CPU vs GPU: La Corsa dei Transistor e delle Prestazioni



- **Legge di Moore:** Sia le CPU che le GPU hanno beneficiato della crescita esponenziale nel numero di transistor, ma l'aumento delle prestazioni è differente.
- La potenza di calcolo delle GPU (TOPS) aumenta più rapidamente delle prestazioni CPU single-thread, evidenziando il vantaggio dell'elaborazione parallela delle GPU.
- Le prestazioni di calcolo delle GPU raddoppiano quasi ogni anno (2x per anno), portando a un incremento di 1000 volte in 10 anni. (**Legge di Huang**)

Top500 - The List

<https://www.top500.org/>

MAY 2024

| | | | SITE | COUNTRY | CORES | R _{MAX} PFLOP/S | POWER MW |
|---|-----------------|---|-----------------|---------|-----------|-----------------------------|-------------|
| 1 | Frontier | HPE Cray EX235a, AMD Opt 3rd Gen EPYC (64C 2GHz), AMD Instinct MI250X, Slingshot-11 | DOE/SC/ORNL | USA | 8,699,904 | 1,206.0 | 22.7 |
| 2 | Aurora | HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 (52C 2.4GHz), Intel Data Center GPU Max, Slingshot-11 | DOE/SC/ANL | USA | 9,264,128 | 1,012.0 | 38.7 |
| 3 | Eagle | Microsoft NDv5, Xeon Platinum 8480C (48C 2GHz), NVIDIA H100, NVIDIA Infiniband NDR | Microsoft Azure | USA | 1,123,200 | 561.2 | |
| 4 | Fugaku | Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D | RIKEN R-CCS | Japan | 7,630,848 | 442.0 | 29.9 |
| 5 | LUMI | HPE Cray EX235a, AMD Opt 3rd Gen EPYC (64C 2GHz), AMD Instinct MI250X, Slingshot-11 | EuroHPC/CSC | Finland | 2,220,288 | 379.7 | 6.01 |

Cos'è la TOP500?

- La TOP500 è una **classifica biennale** dei 500 **supercomputer (HPC)** più potenti al mondo.
- È compilata e pubblicata da esperti del settore due volte l'anno, a giugno e a novembre (dal 1993).

Scopo

- Misurare e documentare le performance dei supercomputer a livello globale.
- Fornire un **benchmark standard (High Performance Linpack)** per il confronto delle capacità di calcolo.

Leader Top500: Supercomputer Frontier

Cos'è Frontier?

- Supercomputer sviluppato da HPE e AMD per l'Oak Ridge National Laboratory (ORNL)
- Primo supercomputer a raggiungere prestazioni **exascale** (10^{18} operazioni in virgola mobile al secondo).
- Costo stimato: Circa **600** milioni di dollari.



Leader Top500: Supercomputer Frontier

Cos'è Frontier?

- Supercomputer sviluppato da HPE e AMD per l'Oak Ridge National Laboratory (ORNL)
- Primo supercomputer a raggiungere prestazioni **exascale** (10^{18} operazioni in virgola mobile al secondo).
- Costo stimato: Circa **600** milioni di dollari.



Unità di Misura Dati: Da Kilo a Exa

| UNITÀ | VALORE ESATTO | POTENZA DI 2 | POTENZA DI 1024 | APPROXIMAZIONE |
|-------|---------------------------|--------------|-----------------|-------------------|
| EXA | 1,152,921,504,606,846,976 | 2^{60} | 1024^6 | $\approx 10^{18}$ |
| PETA | 1,125,899,906,842,624 | 2^{50} | 1024^5 | $\approx 10^{15}$ |
| TERA | 1,099,511,627,776 | 2^{40} | 1024^4 | $\approx 10^{12}$ |
| GIGA | 1,073,741,824 | 2^{30} | 1024^3 | $\approx 10^9$ |
| MEGA | 1,048,576 | 2^{20} | 1024^2 | $\approx 10^6$ |
| KILO | 1,024 | 2^{10} | 1024^1 | $\approx 10^3$ |

Supercomputer Leonardo - Settimo nella TOP500

Cos'è Leonardo?

- Supercomputer avanzato sviluppato da Atos per il **CINECA**, installato a **Bologna**.
- Circa 25 PFLOPS (25×10^{15} operazioni in virgola mobile al secondo).
- Costo stimato: Circa **150** milioni di euro.

I NUMERI DI LEONARDO

155
RACK DI SISTEMA

4992
NODI DI CALCOLO

6
MW IN USO

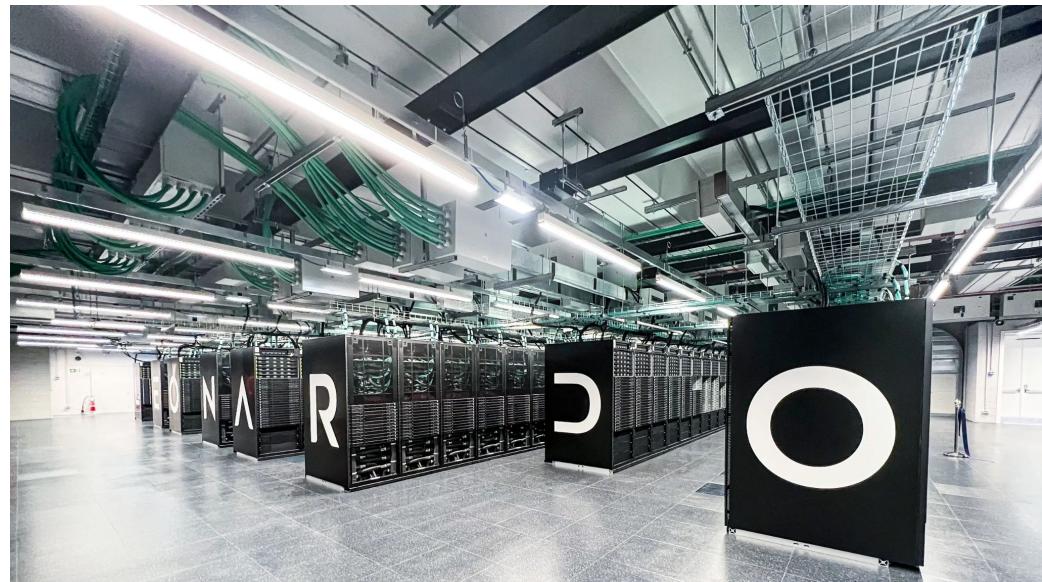
110
PB DI STORAGE

250
PETAFLOP

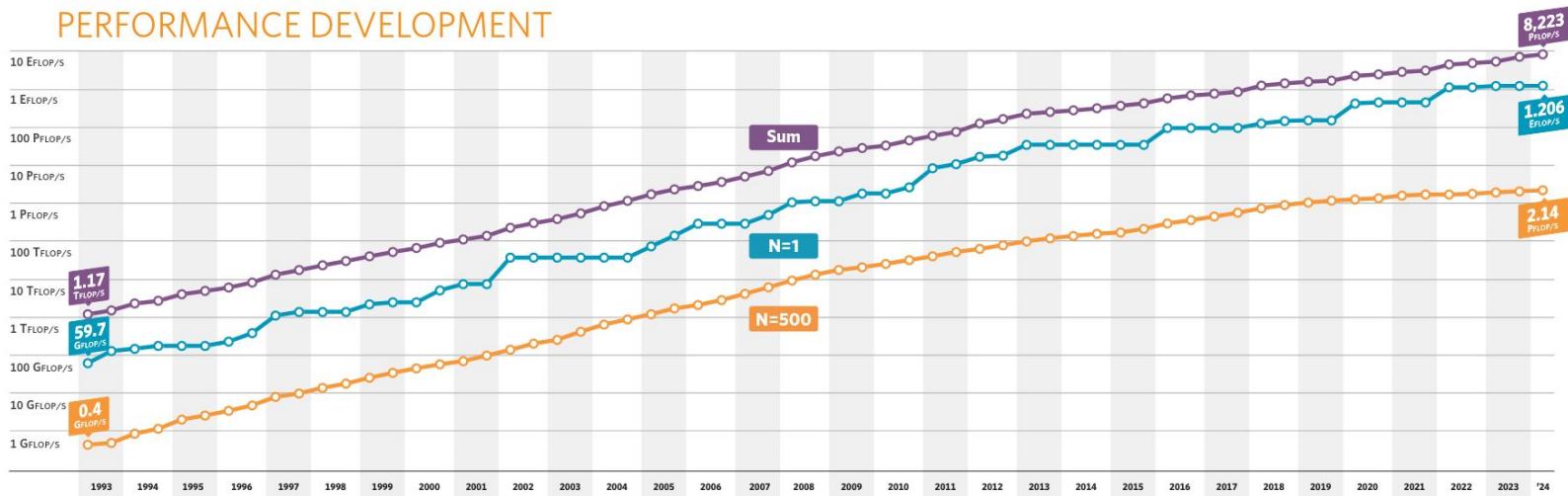
2800
TB DI RAM

600
M² CALPESTABILI

>95%
DISSIPAZIONE DEL CALORE
TRAMITE DLC



Top500 - The List



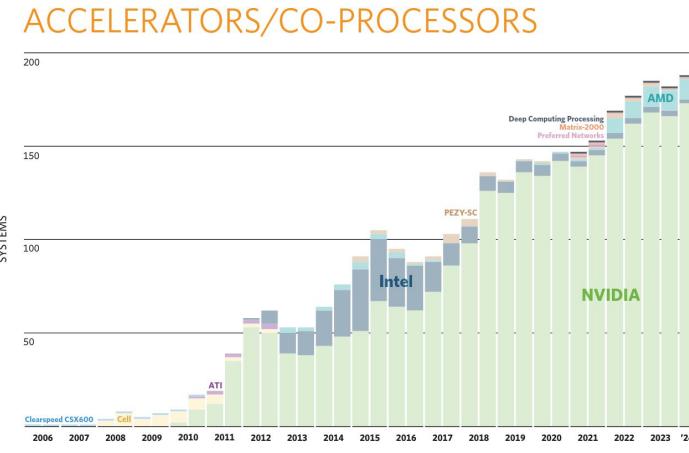
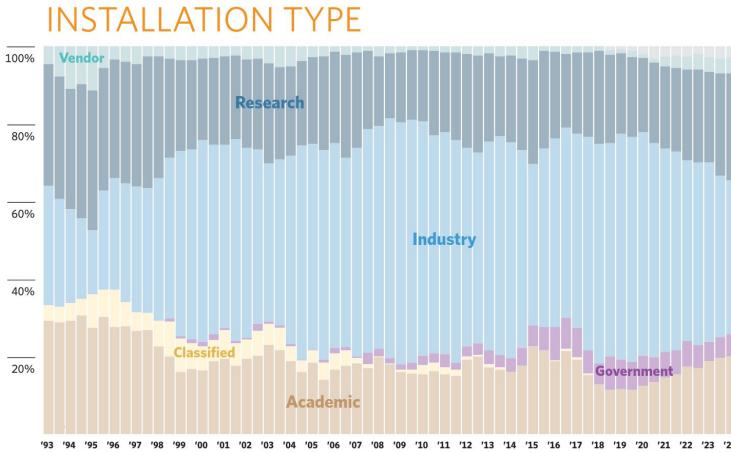
Cos'è il Performance Development?

- Monitoraggio dell'evoluzione delle performance dei supercomputer nel tempo.
- Misurato in FLOPS (Floating Point Operations Per Second).

Elementi Chiave del Grafico

- **N=1:** Performance del supercomputer più potente
- **N=500:** Performance del 500° supercomputer
- **Sum:** Somma totale delle performance di tutti i supercomputer in lista.

Top500 - The List



Installation Type

- Mostra i vari tipi di **installazione** dei supercomputer nella TOP500.
- Categorie Comuni (**Università, Centri di Ricerca, Industria, Governo**)

Tipi Comuni di Acceleratori

- **GPU (Graphics Processing Units)**: Ad esempio, NVIDIA e AMD.
- **FPGA (Field-Programmable Gate Arrays)**: Chip personalizzabili.
- **ASIC (Application-Specific Integrated Circuits)**: Chip specializzati per compiti specifici.

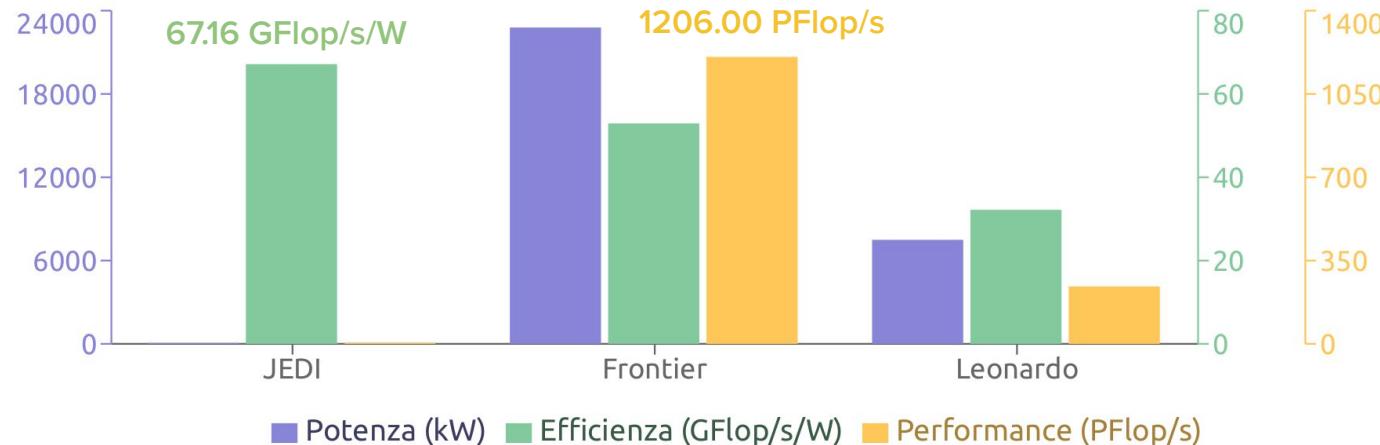
Supercomputer: Efficienza, Potenza e Prestazioni

Più Efficiente: JEDI (Rank 189)

- Efficienza: 67.16 GFlop/s/W
- Potenza: 0.07 MW
- Performance: 4.50 PFlop/s
- Equivalente a 55 case USA*

Più Performante: Frontier (Rank 1)

- Efficienza: 52.93 GFlop/s/W
- Potenza: 22.79 MW
- Performance: 1206.00 PFlop/s
- Equivalente a 18,677 case USA*



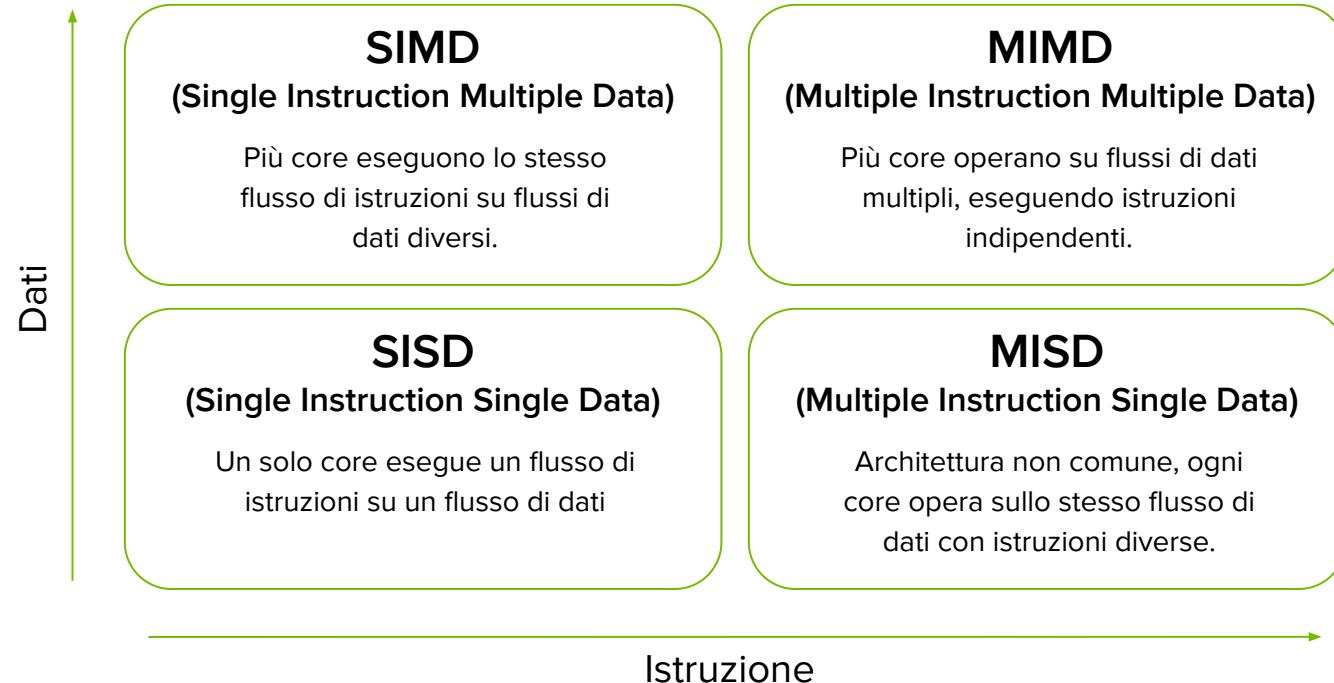
Confronto Chiave: JEDI è 1.27 volte più efficiente di Frontier, ma Frontier offre prestazioni 268 volte superiori.

*Secondo l'U.S. Energy Information Administration (EIA), una casa americana media consuma circa 10,715 kWh all'anno, ovvero 1.22 kW di potenza media continua.

Tassonomia di Flynn

Classificazione delle Architetture

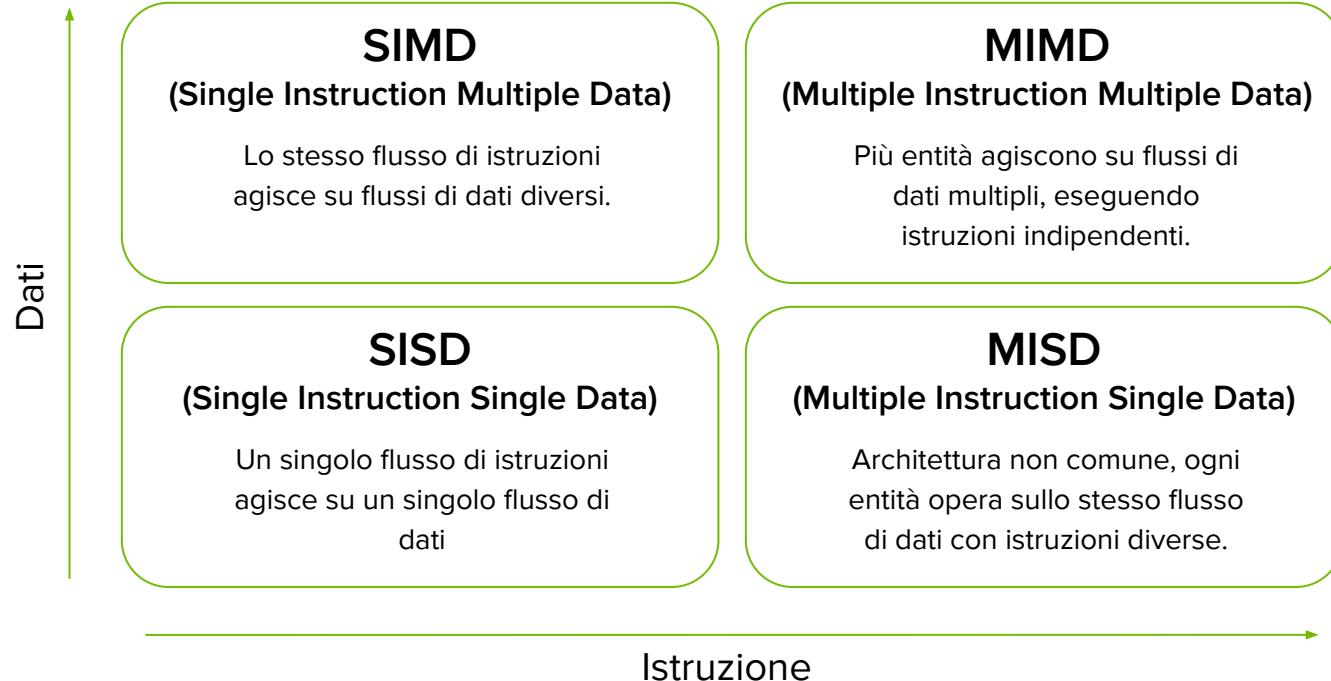
- La Tassonomia di Flynn è un sistema ampiamente utilizzato per classificare le architetture dei computer in base al flusso di istruzioni e dati



Tassonomia di Flynn

Classificazione delle Architetture

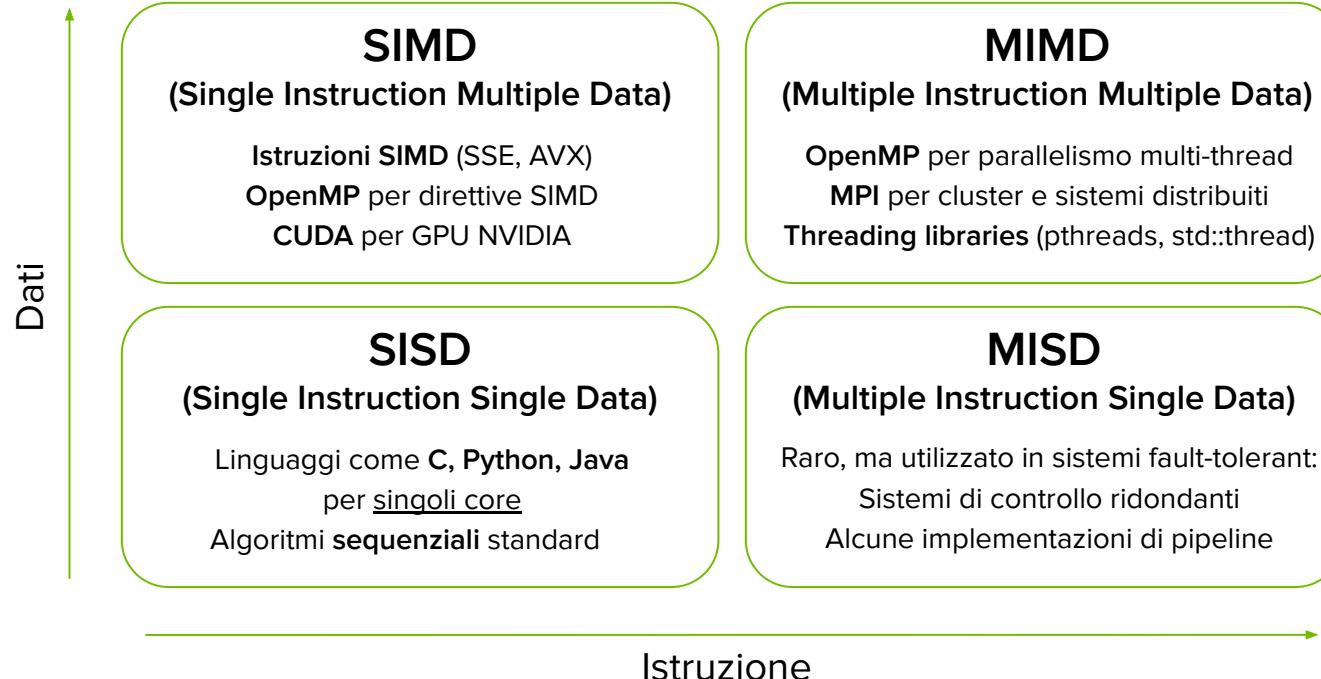
- La Tassonomia di Flynn è un sistema ampiamente utilizzato per classificare le architetture dei computer in base al flusso di istruzioni e dati attraverso i core



Modelli di Programmazione per le Architetture di Flynn

Modelli di Programmazione

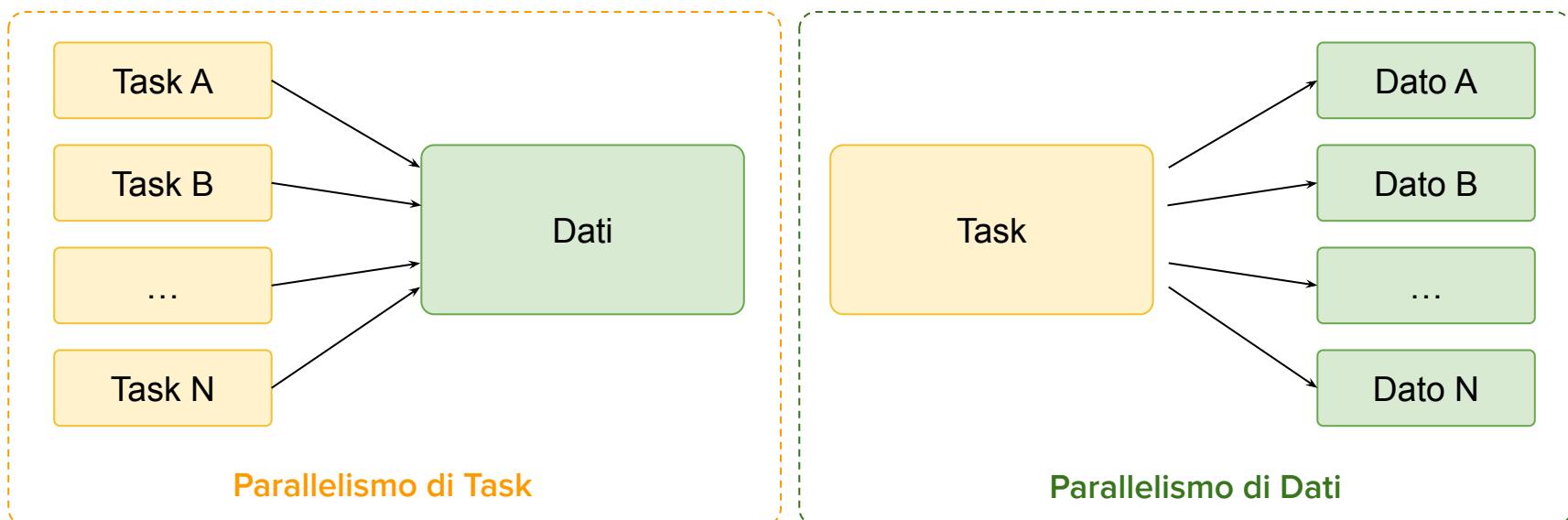
- La Tassonomia di Flynn si traduce in pratica attraverso **specifici modelli e strumenti di programmazione**, ognuno ottimizzato per sfruttare le caratteristiche delle diverse architetture.



Concetti Fondamentali del Parallelismo

Programmi di Calcolo

- Un'istruzione è una coppia (**TASK**, **DATI**)
- Ci sono due modi puri per eseguire le istruzioni in parallelo:
 - **Parallelismo di Task:** Esegue task diversi sugli stessi dati (Distribuisce Task su Core multipli)
 - **Parallelismo di Dati:** Esegue lo stesso task su porzioni diverse di dati (Distribuisce Dati su Core multipli)



Approcci Tecnici al Parallelismo

Parallelismo a Livello di Dati (Data-Level Parallelism, DLP):

- Esegue la **stessa operazione** su più elementi di dati contemporaneamente.
 - **SIMD**: Esecuzione della stessa istruzione su set diversi di dati.
 - **Vettorizzazione**: Ottimizza le operazioni su array di dati per l'esecuzione parallela.
 - **GPU Computing**: Utilizza le GPU per elaborare grandi dataset in parallelo (es. CUDA, OpenCL).

Parallelismo a Livello di Istruzione (Instruction-Level Parallelism, ILP)

- **Parallelizzazione delle istruzioni** all'interno di un singolo thread.
 - **Pipelining**: Sovrapposizione di fasi di esecuzione di istruzioni diverse.
 - **Superscalarità**: Esecuzione simultanea di più istruzioni indipendenti nello stesso ciclo di clock.
 - **Out-of-Order Execution**: Il processore esegue le istruzioni indipendenti prima, ottimizzando le risorse.

Parallelismo a Livello di Thread (Thread-Level Parallelism, TLP)

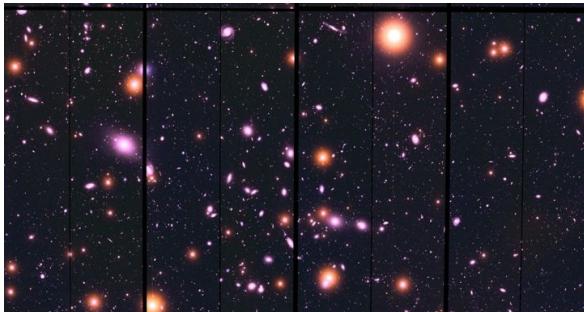
- **Esecuzione simultanea di più thread**.
 - **Multithreading**: Esegue più thread concorrentemente su uno o più core del processore.
 - **Multiprocessing**: Distribuisce l'elaborazione su più processori o core fisici.
 - **Hyper-Threading**: Simula core aggiuntivi per eseguire più thread su un singolo core fisico.

Performance: Terminologia

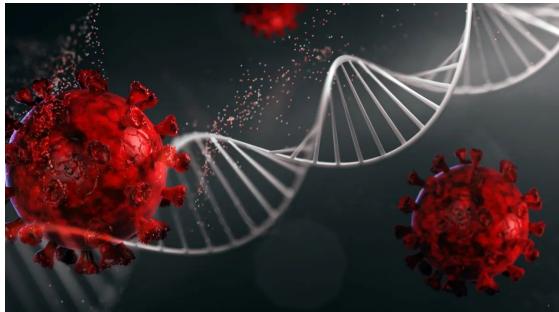
| Termine | Descrizione |
|-------------------------------|---|
| Clock Rate | La frequenza di funzionamento del processore, misurata in Hertz (Hz) o suoi multipli (MHz, GHz). |
| Computer Cycle | Un singolo impulso di clock durante il quale viene eseguita un'operazione di base del processore. |
| Instructions per second (IPS) | Il numero di istruzioni che un processore può eseguire in un secondo, spesso indicato come IPS. |
| FLOPs | Floating-Point Operations Per Second, una misura delle prestazioni di un sistema di calcolo per operazioni in virgola mobile. |
| Peak Performance | La massima velocità teorica di esecuzione di un sistema di calcolo, spesso espressa in FLOPS |
| Speedup | Il rapporto tra il tempo di esecuzione di un programma sequenziale e il tempo di esecuzione dello stesso programma su un sistema parallelo. |
| Benchmark | Un test standardizzato utilizzato per misurare e confrontare le prestazioni di diversi sistemi di calcolo. |

Calcolo Parallelo: Alcune Applicazioni Odierne

- Il calcolo parallelo potenzia la risoluzione di problemi complessi attraverso l'elaborazione simultanea, rivoluzionando **molteplici settori**



Elaborazione Dati Astronomici



Analisi del genoma



Crittografia



Reti Neurali



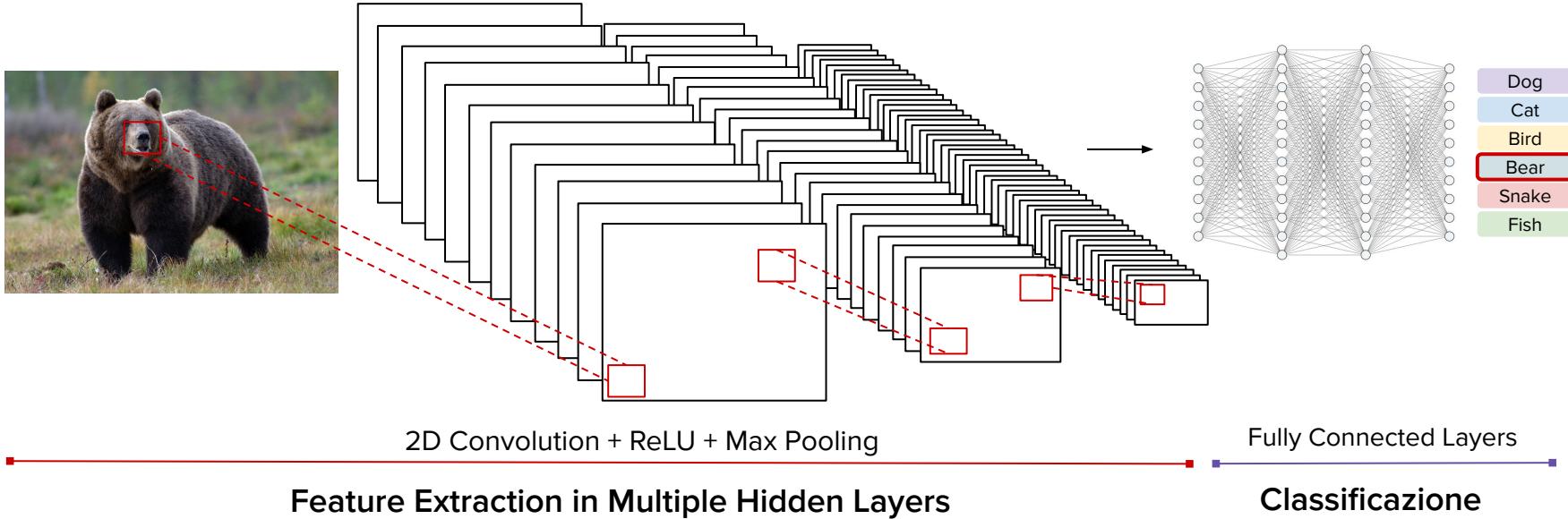
Videogiochi



Deep Vision

Elaborazione delle Immagini con GPU e CNN

- Le reti neurali convoluzionali (CNN) eseguono milioni di operazioni su matrici (**convoluzioni 2D**) per estrarre caratteristiche (dette **feature**) dalle immagini.
- Ogni operazione su pixel e feature può essere eseguita in **parallelo**, accelerando il calcolo.
- Le GPU processano queste operazioni simultaneamente, permettendo applicazioni come il riconoscimento di immagini, ricostruzione 3D, segmentazione semantica e molto altro.



Panoramica del Modulo 1 - CPU Oriented e FPGA

- **Elaborazione parallela con istruzioni SIMD**
 - Scrivere codice per elaborazione parallela di dati per CPU con paradigma SIMD
- **Elaborazione parallela con CPU multicore**
 - Scrivere codice per distribuire il carico computazionale su multipli thread/Core
- **Field Programmable Gate Array (FPGA)**
 - Apprendere le caratteristiche salienti dei dispositivi FPGA (Field Programmable Gate Array)

Riferimenti Bibliografici

Testi Generali

- Pacheco, P. S (2011). **An Introduction to Parallel Programming**. Morgan-Kaufmann (1+ edizione)
- Chapman, B (2007). **Using OpenMP: Portable Shared Memory Parallel Programming**. (MIT Press)

Risorse Online

- Intel SIMD intrinsics:
<https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html>
- ARM SIMD intrinsics
<https://developer.arm.com/documentation/100076/0100/Instruction-Set-Overview/Overview-of-the-Arm-Architecture/Advanced-SIMD?lang=en>
- OpenCV universal intrinsics
https://docs.opencv.org/4.9.0/df/d91/group__core__hal__intrin.html

Panoramica del Modulo 2 - GPU Oriented

➤ Modello di Programmazione CUDA

- Scrivere codice parallelo per GPU con CUDA C/C++, organizzandolo in thread, blocchi e griglie.

➤ Modello di Esecuzione CUDA

- Come la GPU gestisce ed esegue il codice CUDA

➤ Organizzazione delle Memorie in CUDA

- Tipi di memoria disponibili in CUDA e il loro utilizzo efficiente.

➤ Stream e Concorrenza in CUDA

- Utilizzare gli stream per sovrapporre calcolo e trasferimento dati.

➤ Librerie e Applicazioni in CUDA

- Librerie CUDA ottimizzate per diversi domini e come usarle per problemi reali.

Riferimenti Bibliografici

Testi Generali

- Cheng, J., Grossman, M., McKercher, T. (2014). **Professional CUDA C Programming**. Wrox Pr Inc. (1^a edizione)
- Kirk, D. B., Hwu, W. W. (2013). **Programming Massively Parallel Processors**. Morgan Kaufmann (3^a edizione)

NVIDIA Docs

- CUDA Programming:
<http://docs.nvidia.com/cuda/cuda-c-programming-guide/>
- CUDA C Best Practices Guide
<http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/>

Risorse Online

- Corso GPU Computing (Prof. G. Grossi): Dipartimento di Informatica, Università degli Studi di Milano
 - <http://gpu.di.unimi.it/lezioni.html>