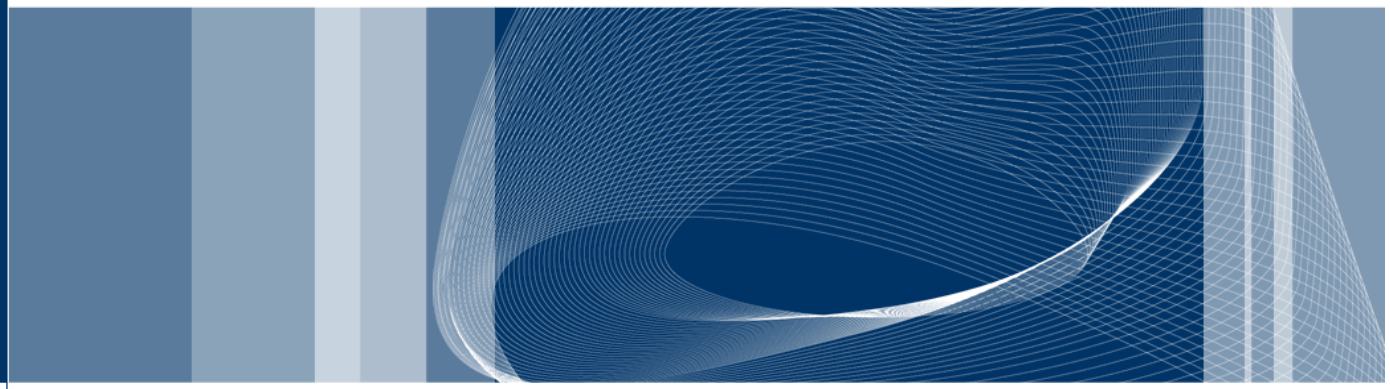
POLITECNICO DI MILANO



Introduzione al Calcolo Parallelo

Algoritmi e Calcolo Parallelo

Daniele Loiacono

Riferimenti

- Questo materiale deriva dalle slide del prof. Lanzi per il corso di Informatica B, A.A. 2009/2010

- Il materiale presente in queste slide è preso dai seguenti tutorial:
  - Introduction to Parallel Computing
    Blaise Barney, Lawrence Livermore National Laboratory
    https://computing.llnl.gov/tutorials/parallel_comp/
  - Anche pubblicato come Dr.Dobb's "Go Parallel"
    Introduction to Parallel Computing: Part 2
    Blaise Barney, Lawrence Livermore National Laboratory

Calcolo Sequenziale

- Tradizionalmente i programmi sono stati scritti per un modello di computazione sequenziale (Von Neuman)
- Per esere eseguiti su un computer con una singola CPU
- Un problema viene spezzato in sequenze (discrete) di istruzioni che sono eseguite in sequenza (una dopo l'altra)
- In un dato istante di tempo solo un'istruzione è in esecuzione

Che Cos'è il Calcolo Parallelo?

- Con il termine calcolo parallelo si intende l'uso di più unità di computazione (più CPU) per risolvere problemi
  - Un problema viene decomposto in parti che posso essere risolte in maniera concorrente (in parallelo)
  - Ogni parte è spezzata in sequenze di istruzioni da eseguire su una singola CPU
  - Le istruzioni di ognuna delle parti sono eseguite simultaneamente su CPU differenti.
- Perchè il calcolo parallelo?
  - Risparmiare tempo e denaro
  - Risolvere problemi molto (troppo) difficili
  - Dare la possibilità di fare più cose contemporaneamente
  - Utilizzare risorse distribuite su tutto il pianeta (e.g., SETI@HOME, Folding@home)
  - Limiti del modello sequenziale per i calcolatori

Quali Assunzioni?

- Risorse di Calcolo
  - Un singolo calcolatore con più processori
  - Un numero arbitrario di calcolatori connessi in rete
  - Una combinazione dei due

- Problema da Risolvere
  - Può essere suddiviso in parti che possono essere risolte contemporaneamente, in parallelo
  - Esecuzioni di più istruzioni in un certo istante di tempo
  - Se risolto con più CPU in parallelo richiede meno tempo rispetto a risolverlo in maniera sequenziale

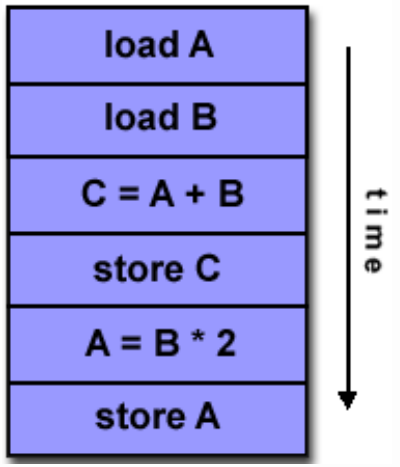
Architetture

Classificazione di Flynn

S I S D Single Instruction Single Data	S I M D Single Instruction Multiple Data
M I S D Multiple Instructions Single Data	M I M D Multiple Instructions Multiple Data

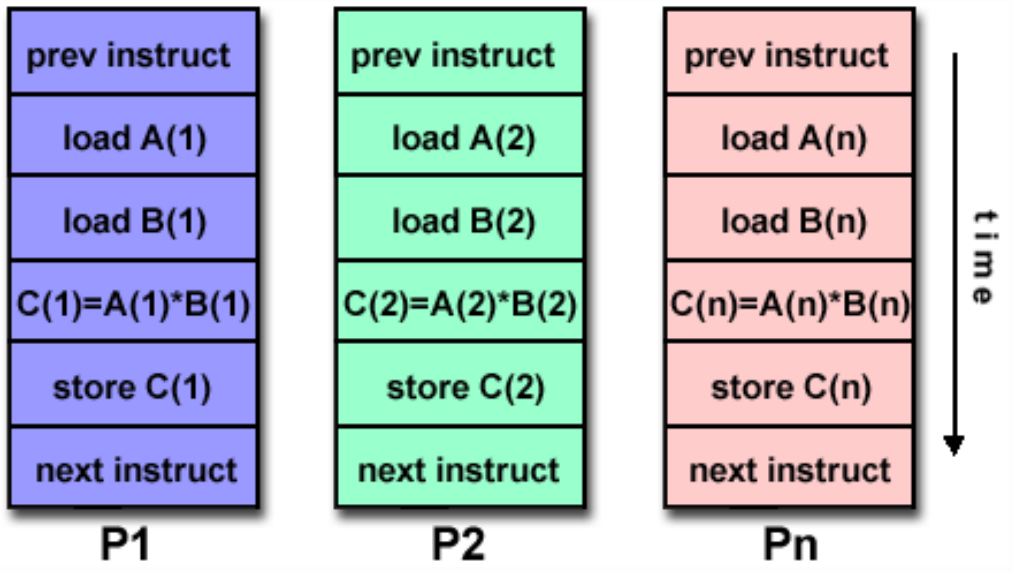
Single Instruction, Single Data (SISD)

- Singolo calcolatore sequenziale (non-parallelo)
- Single instruction
  - La CPU gestisce un solo stream di istruzioni
- Single data
  - Un solo data stream è utilizzato come input
- L'esecuzione è deterministica
- Individua la tipologia di calcolatori più comune e più vecchia
- Esempi: i primi mainframe, minicomputer, workstation; la maggioranza di tutti i PC moderni single-core



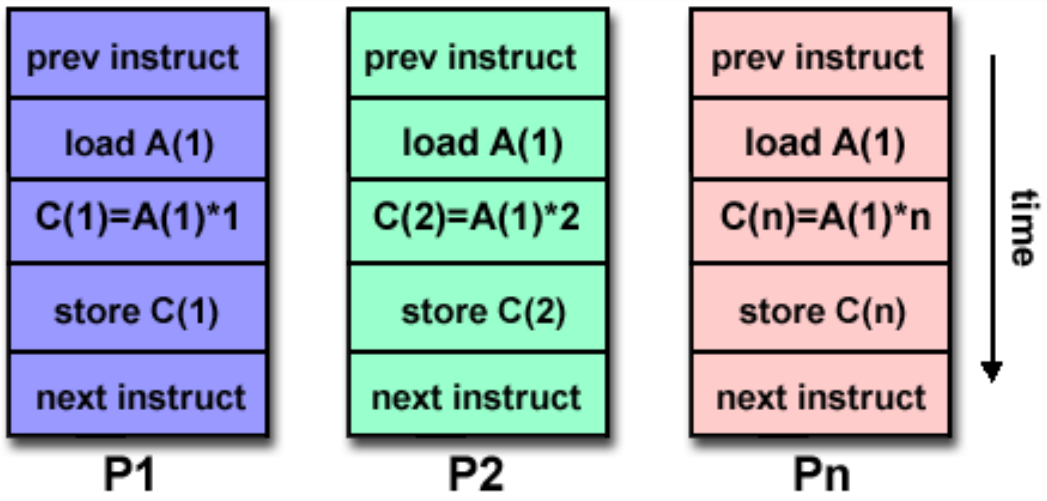
Single Instruction, Multiple Data (SIMD)

- Single instruction: tutte le unità eseguiscono la stessa istruzione allo stesso istante di tempo (ciclo di clock)
- Multiple data: ogni unità opera su un elemeno differente
- Esecuzione sincrona (lockstep) e deterministica
- Esempi
  - Processor Arrays
  - Vector Pipelines
- La maggior parte dei calcolatori moderni che utilizzano graphics processor units (GPUs) seguono un modello SIMD



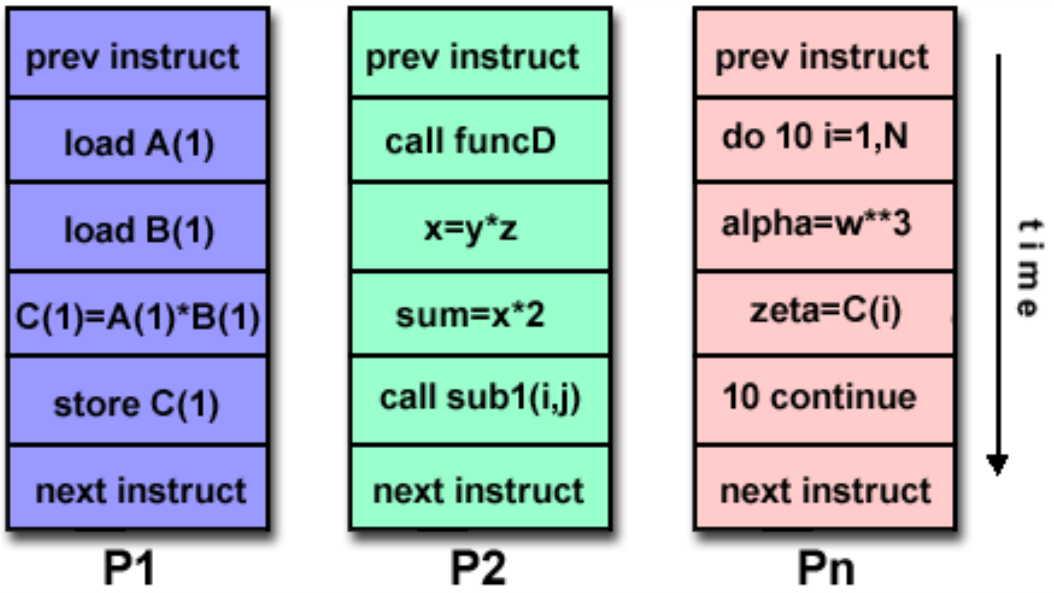
Multiple Instruction, Single Data (MISD)

- Singolo stream di dati che alimenta più unità di elaborazione
- Ogni unità opera sui dati indipendentemente con stream di istruzioni differenti
- Pochisssimi esempi di questo tipo di calcolatori
- Filtraggio multiplo su un singolo segnale
- Crack di un singolo segnale crittografato utilizzando più algoritmi



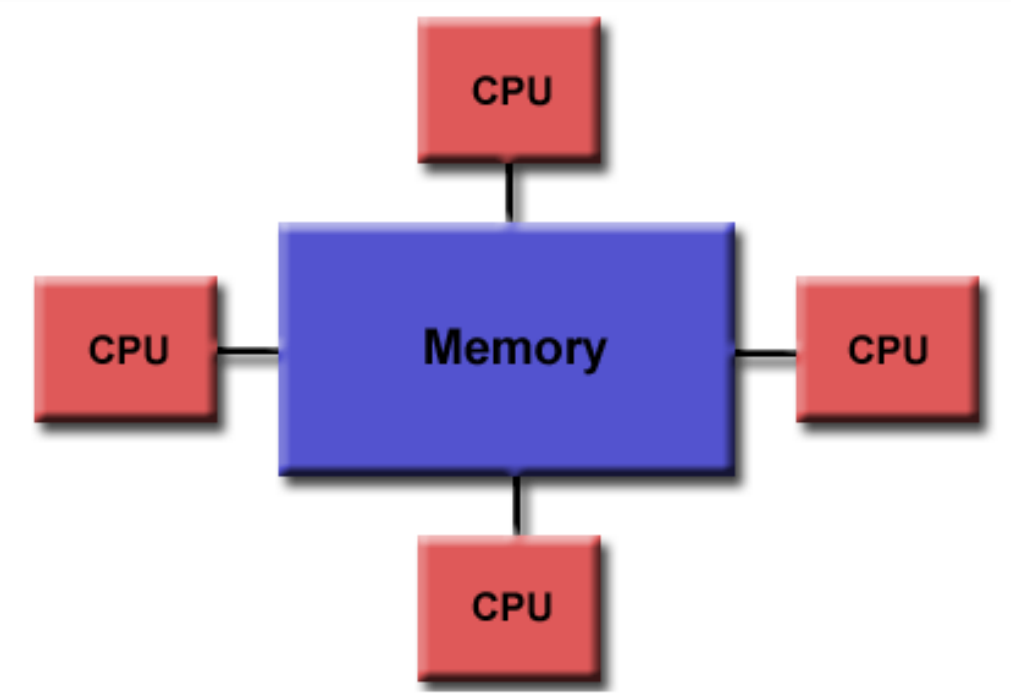
Multiple Instruction, Multiple Data (MIMD)

- Attualmente il modello più diffuso. La maggior parte dei calcolatori paralleli rientrano in questa categoria
- Multiple Instruction: ogni processore esegue un instruction stream differente
- Multiple Data: ogni processore lavora su un diverso data stream differente.
- L'esecuzione può essere sincrona o asincrona, deterministica o non-deterministica
- Esempi: maggior parte dei supercomputer, cluster, grid, ecc.



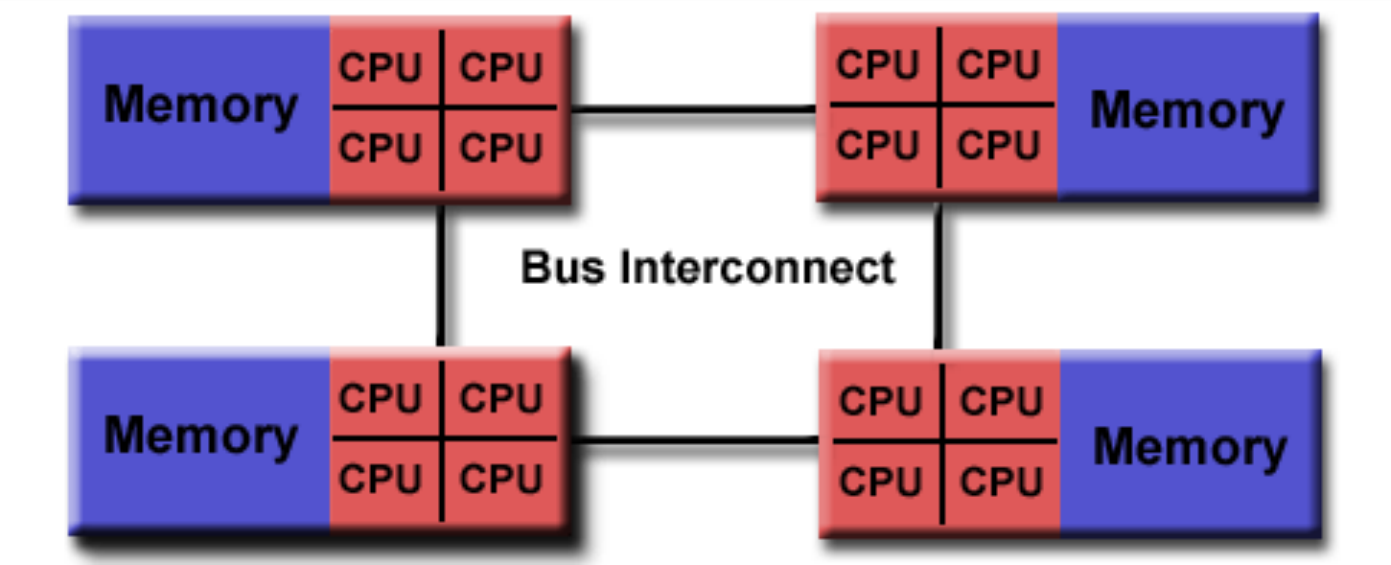
Modelli di memoria

Shared Memory Uniform Access



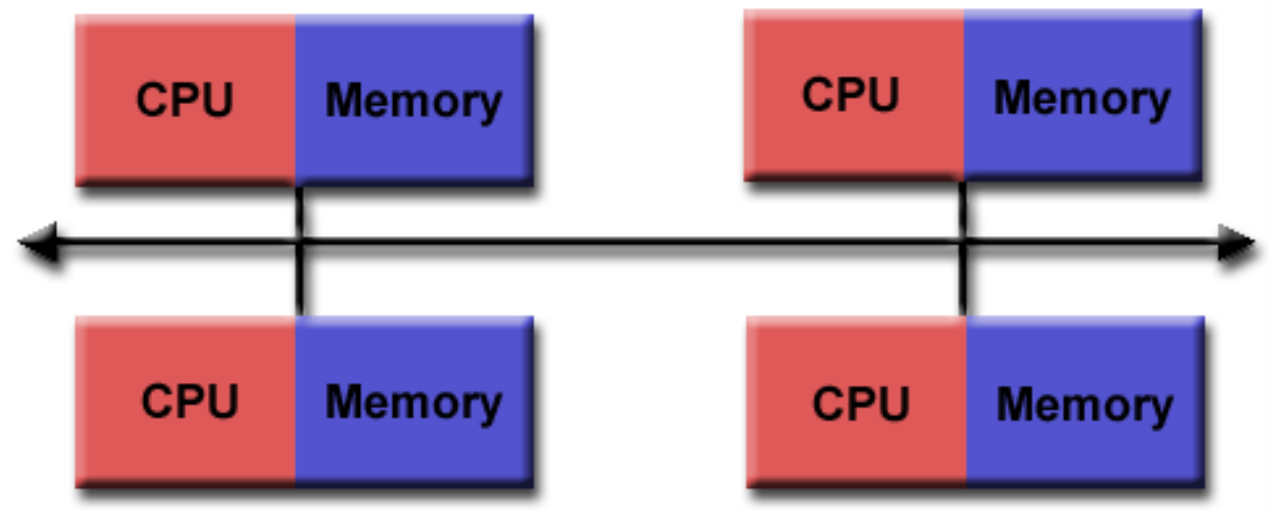
- Semplicità di gestione
- Prestazioni
- Scalabilità
- Costi

Shared Memory Non-Uniform Access



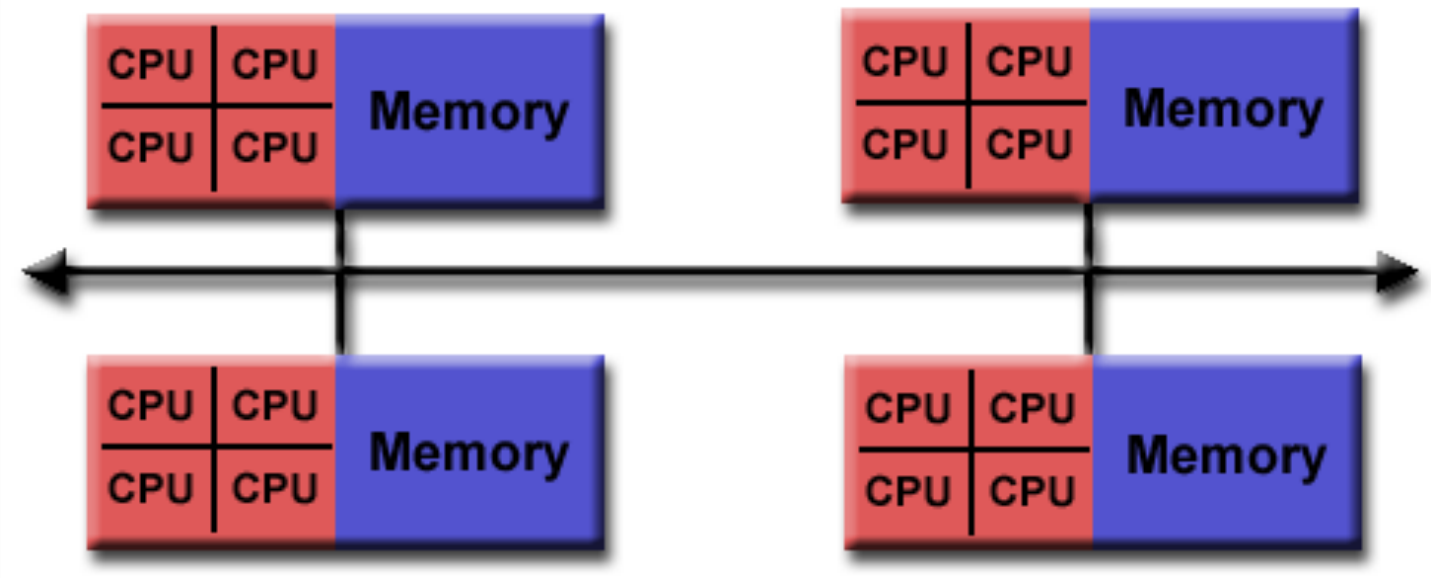
- Rispetto ad uniform-acces
  - Più scalabile
  - Meno costoso
  - Gestione più complessa (corenza)
  - Meno efficiente

Distributed Memory



- Poco costoso
- Scalabilità
- Difficoltà di gestione
- Prestazioni inferiori (costi comunicazioni)

Hybrid Distributed-Shared Memory



- Vantaggi e svantaggi dei modelli shared e distributed