



UNIVERSITÀ
DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE ED INFORMATICHE
Corso di Laurea in Informatica

Sistemi per il calcolo ad alte prestazioni

Programmazione parallela e HPC - a.a. 2023/2024
Roberto Alfieri

Programmazione Parallela e HPC: sommario

PARTE 1 - INTRODUZIONE

PARTE 2 – SISTEMI PER IL CALCOLO AD ALTE PRESTAZIONI

PARTE 3 – PERFORMANCE DELL'HARDWARE

PARTE 4 – PROGETTAZIONE DI PROGRAMMI PARALLELI

PARTE 5 – PROGRAMMAZIONE A MEMORIA CONDIVISA CON OPENMP

PARTE 6 – PROGRAMMAZIONE A MEMORIA DISTRIBUITA COM MPI

PARTE 7 – PROGRAMMAZIONE GPU CON CUDA

Tassonomia di FLYNN

Riferimenti :

- <https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial##Concepts>
- HPC for Science and Engineering – Slide da 1 a 33

Ci sono differenti modi per classificare i sistemi per il calcolo parallelo.

Uno dei più usati è stato introdotto da M.J. Flynn nel 1966.

Questa tassonomia è basata su una tabella a 2 dimensioni indipendenti:

Il flusso di **Istruzioni** e il flusso di **Dati**, che possono essere **Single** o **Multiple**.

Gli elementi della tabella hanno a loro volta sottocategorie:

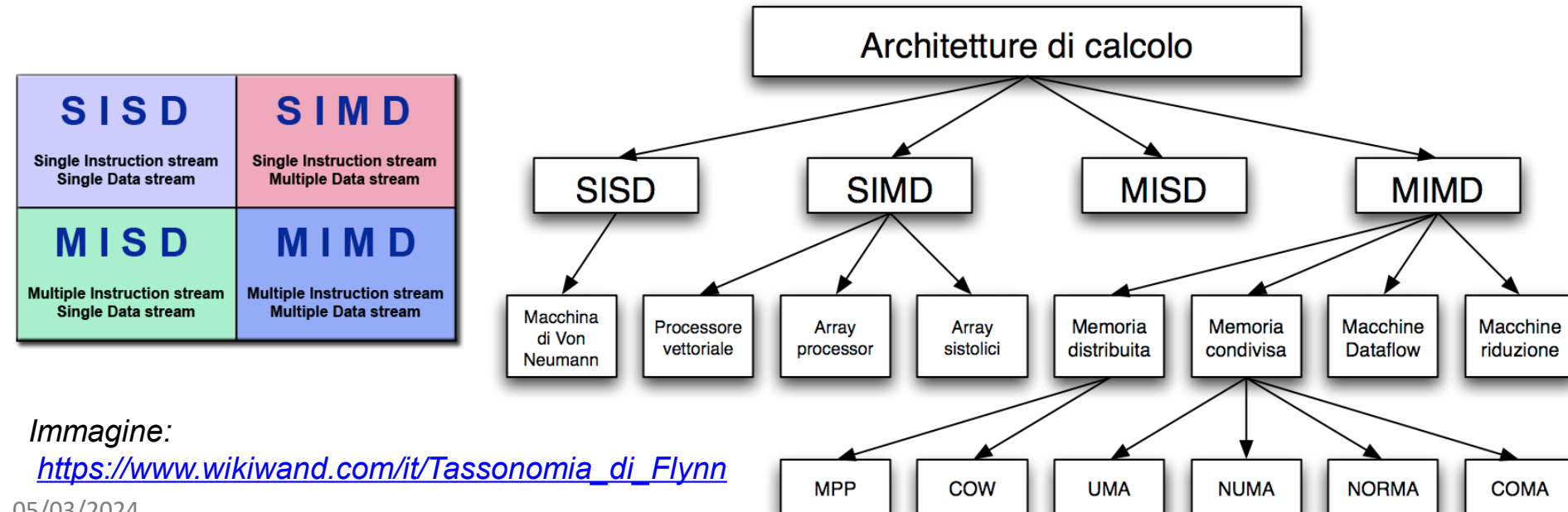


Immagine:

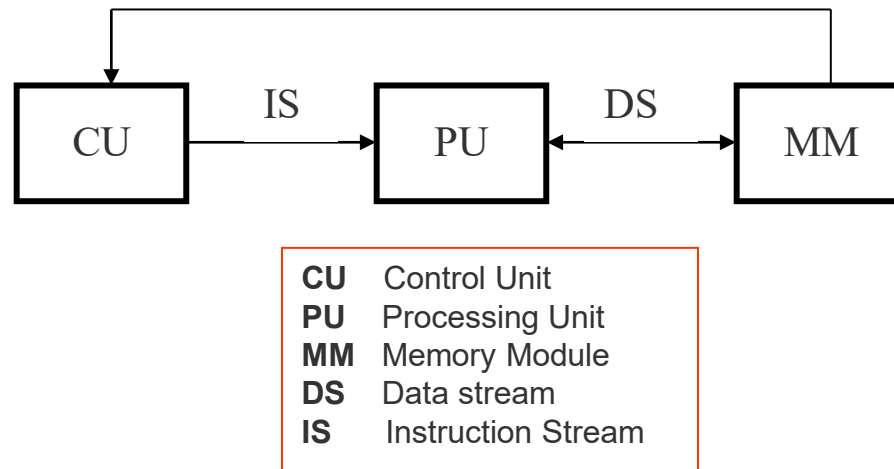
https://www.wikiwand.com/it/Tassonomia_di_Flynn

SISD

Un solo flusso di istruzioni eseguito dalla CPU

Un solo flusso di dati

Sono i sistemi seriali (classica architettura di Von Neumann)



SIMD

Un solo flusso di istruzioni eseguito dalla CPU

Più flussi di dati elaborati contemporaneamente

Una singola istruzione opera simultaneamente su più dati.

I sistemi SIMD hanno una sola unità di controllo

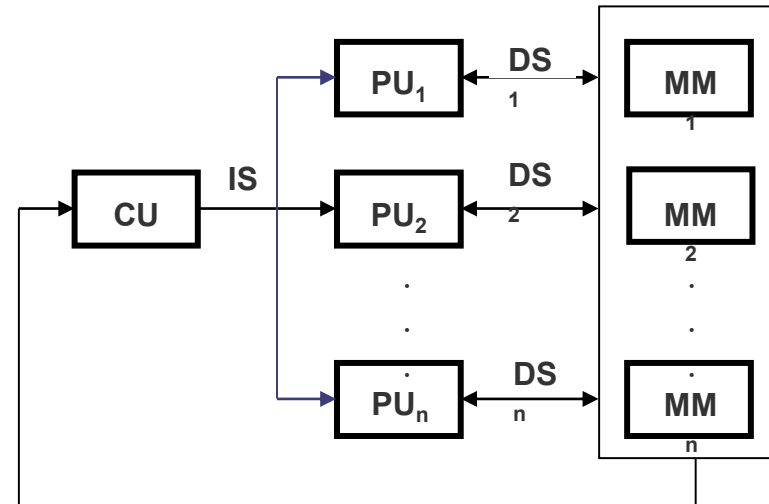
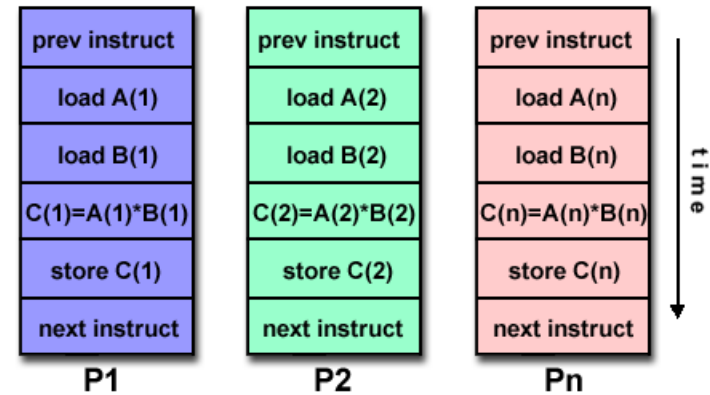
Principali tipologie SIMD:

PROCESSORI VETTORIALI

- Array di elementi di elaborazione che condividono l'unità di controllo
- Istruzioni distribuite in parallelo a tutte le PU
- Ogni PU ha la propria memoria
- Necessaria una rete di comunicazione per i dati

ISTRUZIONI VETTORIALI

- Parallelismo realizzato all'interno del processore
- La **memoria è condivisa**
- Fattore critico: banda di memoria offerta alle Unità Funzionali



CU	Control Unit
PU	Processing Unit
MM	Memory Module
DS	Data stream
IS	Instruction Stream

Sistemi Vettoriali: Installazioni

[CRAY-1](#) Fine anni 70 (Cray-1A, Cray-1S, Cray-1M)

Fu il primo progetto Cray con impiego di circuiti integrati

Pesava 5.5 tonnellate.

Installazioni: [CRAY-1M CINECA](#)

Prestazioni: fino a 250 Mflops - 115 KW.

8 registri vettoriali contenenti ognuno 64 elementi a 64 bit

[Connection Machine](#) 2 (CM2). Anni 80

Conessioni a Ipercubo

Fino a 65000 processori single-bit SIMD

Installazioni: Univ. Parma (vedi CM2@UniPR in materiale didattico)

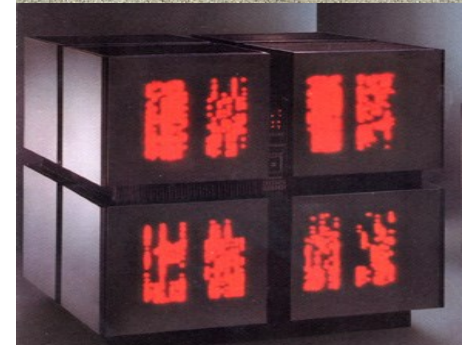
Progetti APE (INFN)

APE (84-88), APE100 (89-93), APE1000 (94-99), APEnext (2000-2005)

<http://apegate.roma1.infn.it/>

Processori SIMD, connessioni 3D toroidali

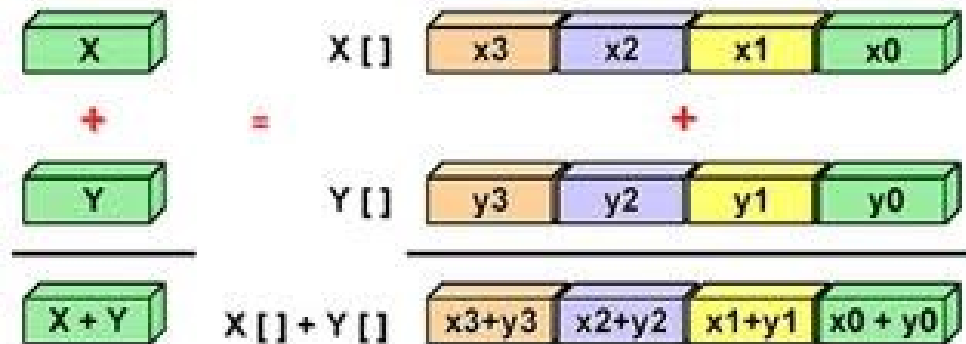
Installazioni: INFN-Roma



ISTRUZIONI VETTORIALI

I processori moderni supportano un set di **istruzioni vettoriali** (o istruzioni SIMD) che si aggiunge al set di istruzioni di istruzioni scalari. Le istruzioni vettoriali specificano una particolare operazione che deve essere eseguita su un determinato insieme di operandi detto **vettore**.

Le unità funzionali che eseguono istruzioni vettoriali sfruttano il pipelining per eseguire la stessa operazione su tutte le coppie di operandi.

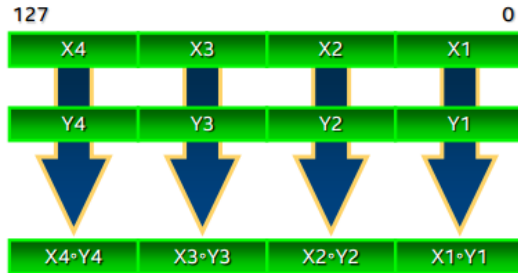


Attualmente istruzioni SIMD sono incluse in quasi tutti i microprocessori, tra cui:

Intel: MMX, SSE, SSE2, SSE3, SSE4, AVX, AVX2, AVX512 AMD: 3DNow!

SIMD nei processori INTEL

1999



SSE

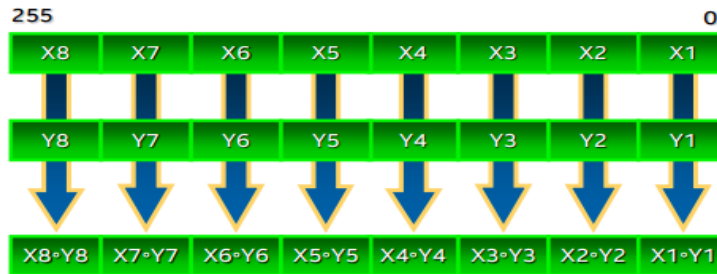
Vector size: **128 bit**

Data types:

- 8, 16, 32, 64 bit integer
- 32 and 64 bit float

VL: 2, 4, 8, 16

2008



AVX

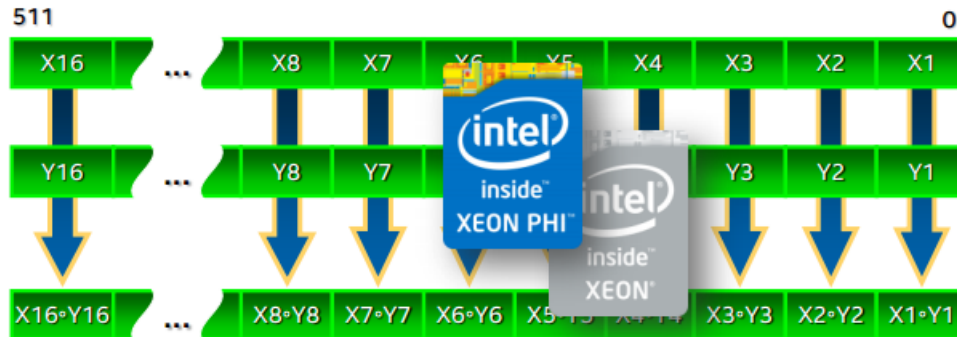
Vector size: **256 bit**

Data types:

- 8, 16, 32, 64 bit integer
- 32 and 64 bit float

VL: 4, 8, 16, 32

2013



Intel® AVX-512 & Intel® MIC Architecture

Vector size: **512 bit**

Data types:

- 8, 16, 32, 64 bit integer
- 32 and 64 bit float

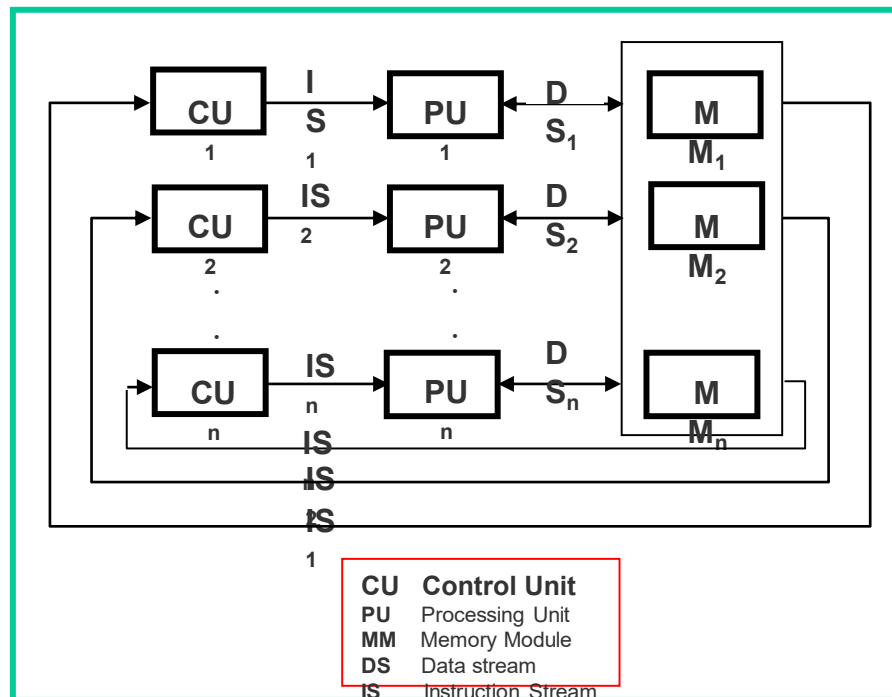
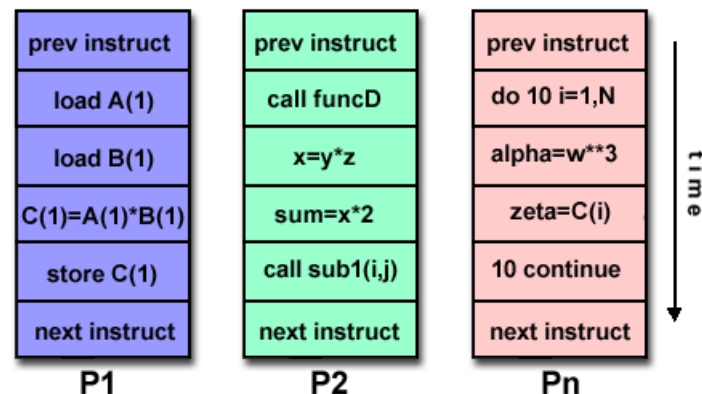
VL: 8, 16, 32, 64

MIMD

Ogni processore può eseguire un differente flusso di istruzioni. Ogni flusso di istruzioni lavora su un differente flusso di dati.

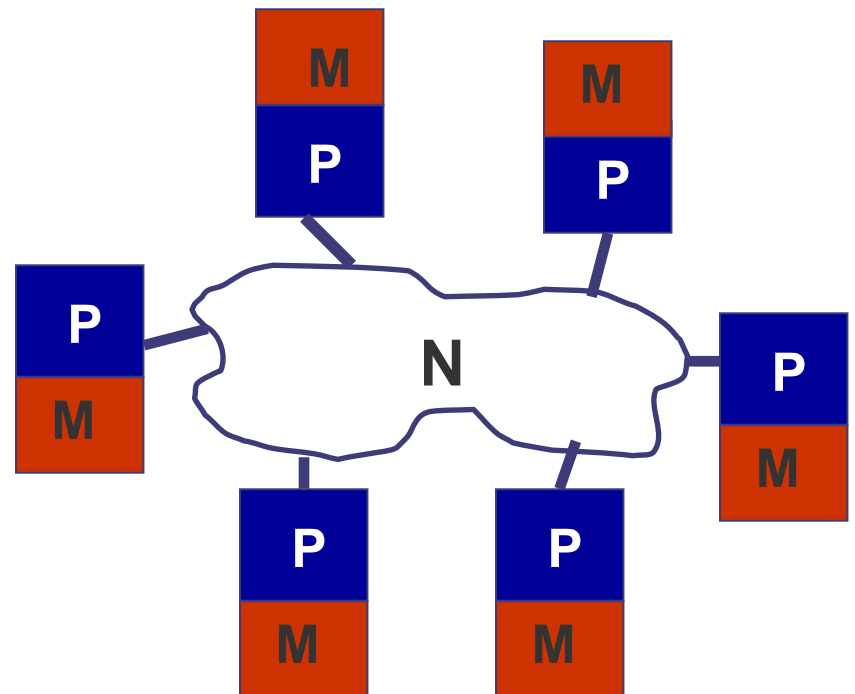
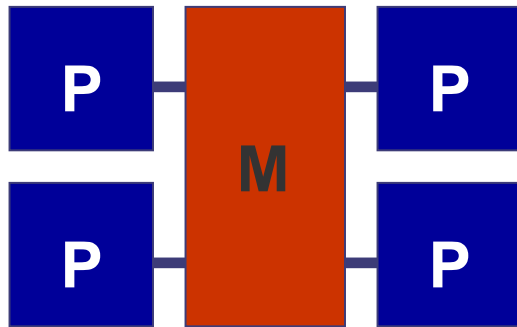
Molte architetture MIMD includono SIMD come caso particolare.

I più moderni sistemi di calcolo parallelo ricadono in questa categoria.



Limiti della tassonomia di Flynn

Classificazione non consente di esprimere caratteristiche come la distinzione tra architettura a memoria distribuita e architettura a memoria condivisa

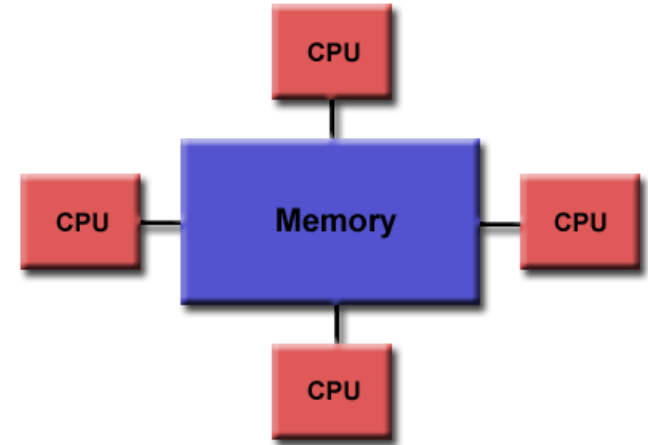


Sistemi a memoria condivisa UMA

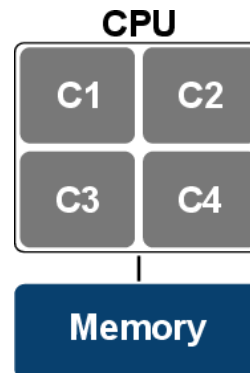
Caratteristica principale: tutti i processori accedono alla memoria come spazio di indirizzamento globale.

Modifiche alla memoria da una CPU sono visibili da tutti gli altri.

Esistono 2 sotto-categorie principali: UMA e NUMA



Uniform Memory Access (UMA) model \Leftrightarrow SMP: Symmetric Multi Processors
L'accesso alla memoria è uniforme: i processori presentano **lo stesso tempo di accesso per tutte le parole di memoria**



Sistemi a memoria condivisa NUMA

Ogni processore ha una propria memoria *locale*

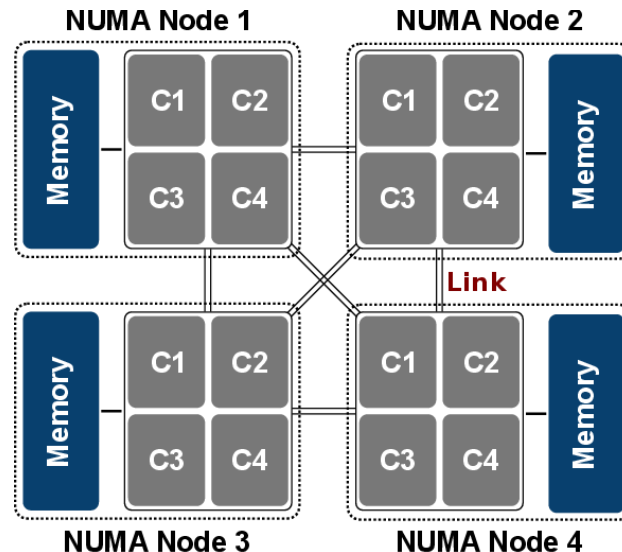
L'insieme delle memorie locali forma uno spazio di indirizzi globale, accessibile da tutti i processori.

Supporto hw per far sì che ogni processore possa indirizzare la memoria di tutti i processori: un processore ha accesso diretto alla memoria degli altri

Nel processore il tempo di **accesso alla memoria non è uniforme**:

- ▶ l'accesso è più veloce se il processore accede alla propria memoria locale;
- ▶ quando si accede alla memoria dei processori remoti si ha un *delay* dovuto alla rete interna di interconnessione.

I moderni sistemi multisocket sono NUMA.

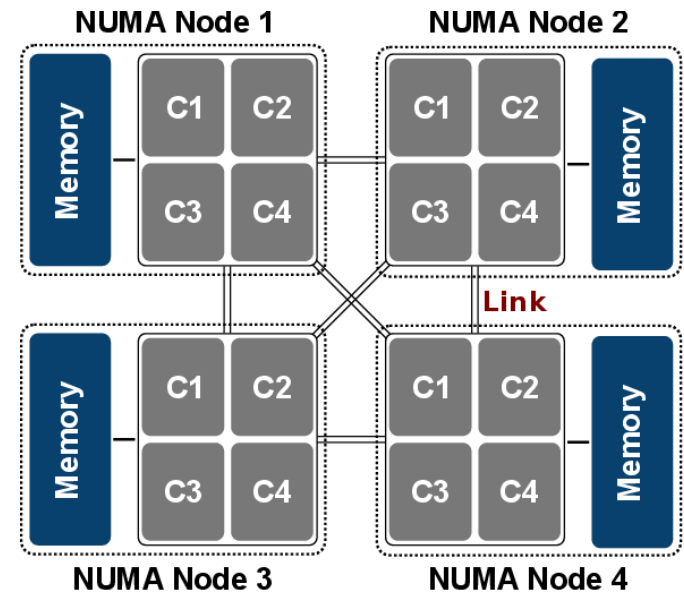


Il comando numactl

In ambiente linux il comando numactl consente di visualizzare l'architettura e gestire l'esecuzione di un programma in architettura NUMA.

Esempio sul nodo quad-socket wn21 del cluster HPC di UniPR (4x18 cores):

```
> ssh wn21 numactl -H
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
node 0 size: 96376 MB
node 0 free: 94007 MB
node 1 cpus: 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
node 1 size: 96762 MB
node 1 free: 94043 MB
node 2 cpus: 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
node 2 size: 96746 MB
node 2 free: 95169 MB
node 3 cpus: 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
node 3 size: 96761 MB
node 3 free: 94860 MB
node distances:
node  0  1  2  3
0: 10 21 21 21
1: 21 10 21 21
2: 21 21 10 21
3: 21 21 21 10
```



numactl consente di controllare al momento del run la locazione fisica del processo e della memoria. Ad esempio per eseguire myprog sul socket 1 e memoria del socket1:

```
numactl --cpubind=1 --membind=1 myprog
```

Sistemi a memoria distribuita

Ogni CPU possiede una propria memoria locale, che non fa parte dello spazio di indirizzamento degli altri processori. Ogni sistema CPU/Memoria è detto **nodo**.

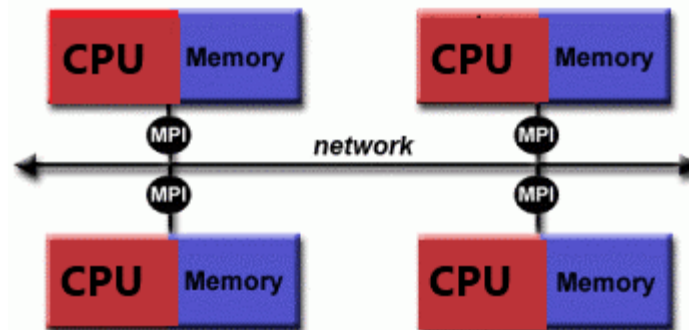
L'infrastruttura di rete (network) per lo scambio di messaggi può essere Ethernet, anche se viene generalmente utilizzata una tecnologia specifica a bassa latenza come Intel OmniPath Architecture (OPA) e Infiniband.

Vantaggi:

- ▶ Il **numero di processori** e la **memoria complessiva** scalano con il numero di nodi.
- ▶ Costi contenuti (l'hardware può essere commodity).

Svantaggi:

- ▶ il programmatore deve gestire i dettagli della comunicazione tra i nodi.
- ▶ Il tempo per l'accesso alla memoria remota dipende dal tipo di infrastruttura di rete, ma generalmente può essere elevato, soprattutto verso i nodi più lontani

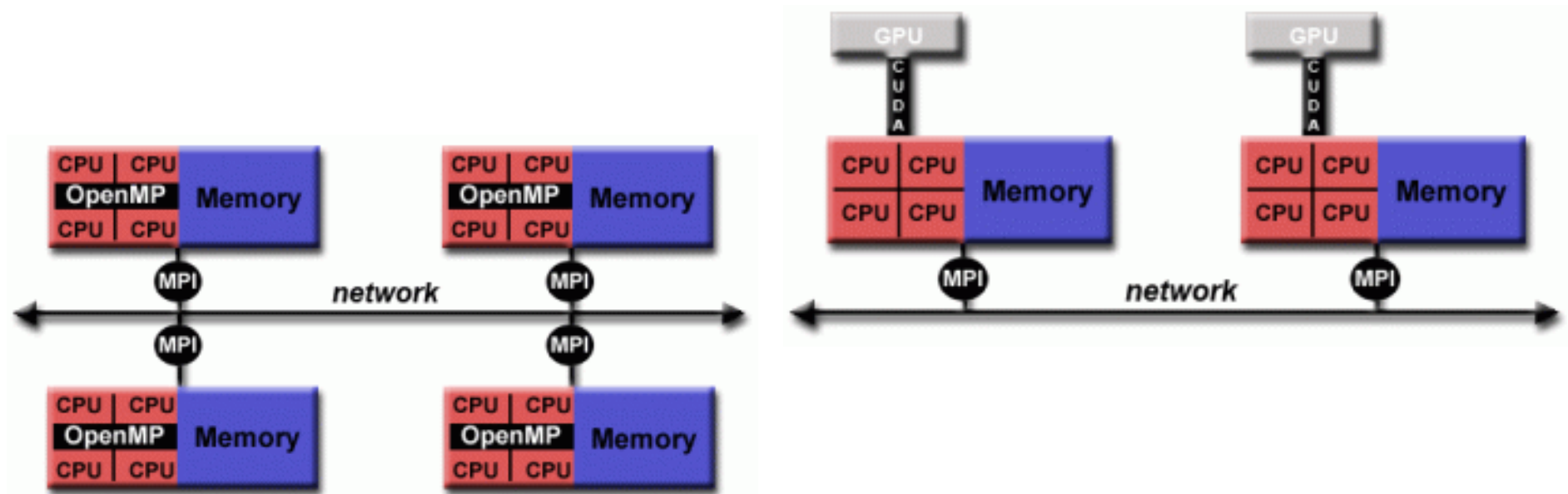


Sistemi Ibridi

I principali sistemi paralleli oggi sono ibridi, ovvero sono composti da più nodi a memoria condivisa (UMA o NUMA) interconnessi tra loro da una rete ad alta velocità.

Nelle ultime generazioni di sistemi paralleli i nodi a memoria condivisa possono disporre di acceleratori basati su GPU.

Questi acceleratori dispongono di un proprio spazio di memoria e comunicano con il nodo attraverso i bus di I/O (e.g. PCI).



Case study: il cluster HPC.unipr.it

Riferimenti: [HPC.unipr.it User Guide](https://hpc.unipr.it/UserGuide)

Partizione	nodi	core	Ram GB
CPU	8	32	128
CPU	16	32	256
CPU	4	28	128
CPU	1	88	1024
CPU	2	96	512
CPU_GUEST	2	72	384
CPU_GUEST	4	64	384

Partizione	nodi	core	Ram GB	GPU
GPU	2	32	128	6 P100
GPU	1	48	512	4 A100 40G
GPU	1	48	512	8 A100 80G
GPU_GUEST	1	32	512	3 A100 80G
GPU_GUEST	1	48	512	2 V100 2 A100 80G

