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

Introduzione

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

Roberto Alfieri

Programmazione Parallela e HPC: sommario

PARTE 1 - INTRODUZIONE

PARTE 2 – PERFORMANCE DELL'HARDWARE

PARTE 3 – SISTEMI PER IL CALCOLO AD ALTE PRESTAZIONI

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

Introduzione all'HPC

Alcuni Rifermenti:

- Introduction to Parallel Computing (LLNL)
- «HPC for Science and Engineering» (vedi Materiale Didattico)
- «High Performance Computing» Sterling, Anderson, Brodowicz - Ed. Gordon Bell

Motivazioni dell'HPC: problemi "Grand Challenge"

Problemi di notevole impatto in diversi ambiti che possono essere risolti computazionalmente ma richiedono notevoli risorse di calcolo e storage per dare una soluzione in un tempo ragionevole.

Esempi problemi Grand Challenge:

- HPC & Aerodinamica: Dallara
- HPC & Energia: ENI
- HPC & Climatologia: ECMWF
- HPC & Problemi Big Data: DataScience (CINECA)
- HPC & AI : AI nei sistemi HPC
- HPC & Sicurezza informatica: Fattorizzazione RSA
- HPC & Scienza: aree di ricerca in ICSC

Strumenti per l'HPC

Risorse HW per l'HPC:

- Supercomputers (vedi TOP500)

Altre risorse e iniziative :

- Cloud Computing (AWS Amazon, Google Cloud, Microsoft Azure, ..)
- Volunteer Computing

Risorse SW per l'HPC:

- Application Software
  - Programmi OpenSource e commerciali (vedi Software al CINECA)
- Ambiente per lo sviluppo di programmi
  - Compilatori (C/C++, Fortran), Librerie (openMP, MPI, CUDA), Profiler

TOP500

TOP500 è un sito in cui vengono elencati i 500 calcolatori più potenti al mondo.

La potenza di calcolo viene misurata in Operazioni Floating Point al secondo (Flops)

Primo posto TOP500 : circa 1000 PFlops (10^{18} Flops) - 20 MWatt (2 10^{7} W)

Un personal computer : circa 100 GFlop/s (10^{12} Flops) - 100 Watt (10^{2} W)

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	4,742,808	585.34	1,059.33	24,687
3	**Eagle** - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Microsoft Azure United States	1,123,200	561.20	846.84	
6	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404

Novembre 2023

Consumo energetico, Green500 e PUE

Il consumo energetico dei sistemi HPC cresce di pari passo, esponenzialmente, con la potenza di calcolo.

Dal 2016 TOP500 stila anche una graduatoria, denominata **Green500**, in cui la TOP500 viene riordinata in base all'efficienza nel consumo energetico rispetto alla potenza di calcolo erogata, misurata in GFlops/Watt : https://www.top500.org/lists/green500/2023/11/

In generale l'efficienza energetica di un Data Center viene misurata con il **PUE** (Power Usage Effectiveness) che è il rapporto tra la potenza totale (PT) assorbita dal data center e quella usata dai soli apparati IT (PIT) ovvero: PUE = PT / PIT

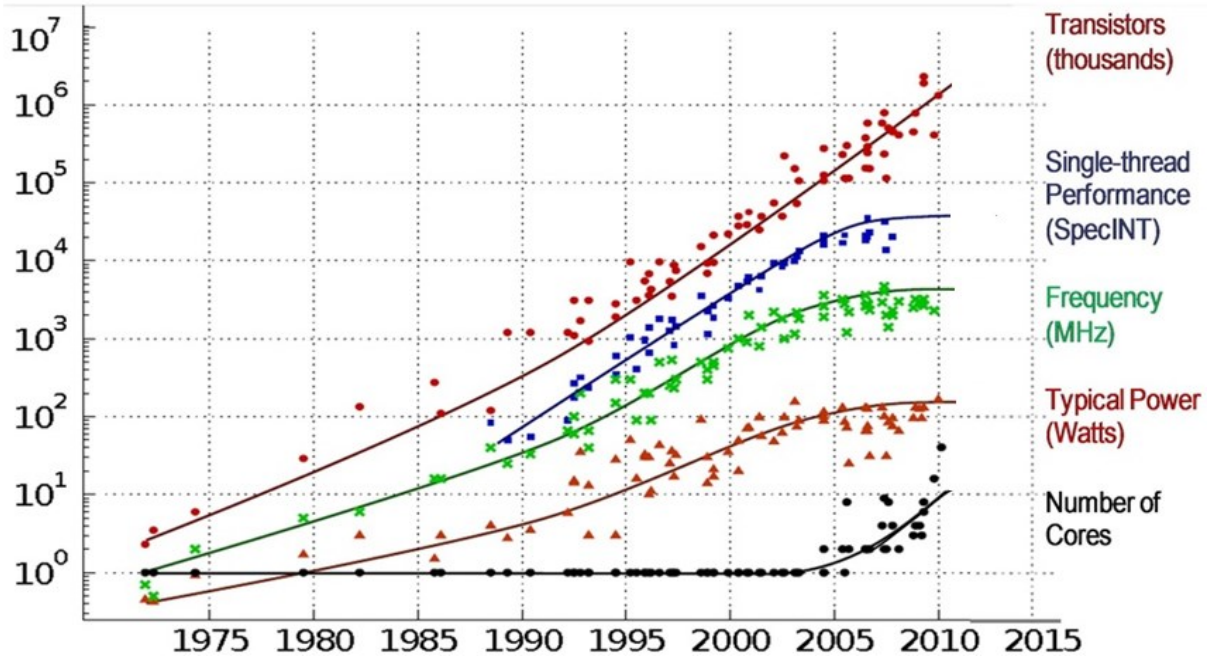
La quota di consumo energetico che eccede l'assorbimento degli apparati IT è principalmente dovuta al raffrescamento.

Il PUE è considerato efficiente se inferiore al 1,8 (vedi articolo del CINECA)

Il data center che presenta il valore di PUE più basso (1,08, consumi non-PIT pari a circa il 7,5% di PT) appartiene a Yahoo! ed è stato costruito nelle vicinanze delle Cascate del Niagara, dunque in condizioni climatiche particolarmente favorevoli (Wikipedia).

Hardware performance: la programmazione parallela

La legge di Moore (1965) ipotizza che le prestazioni dei microprocessori raddoppino ogni 18 mesi.



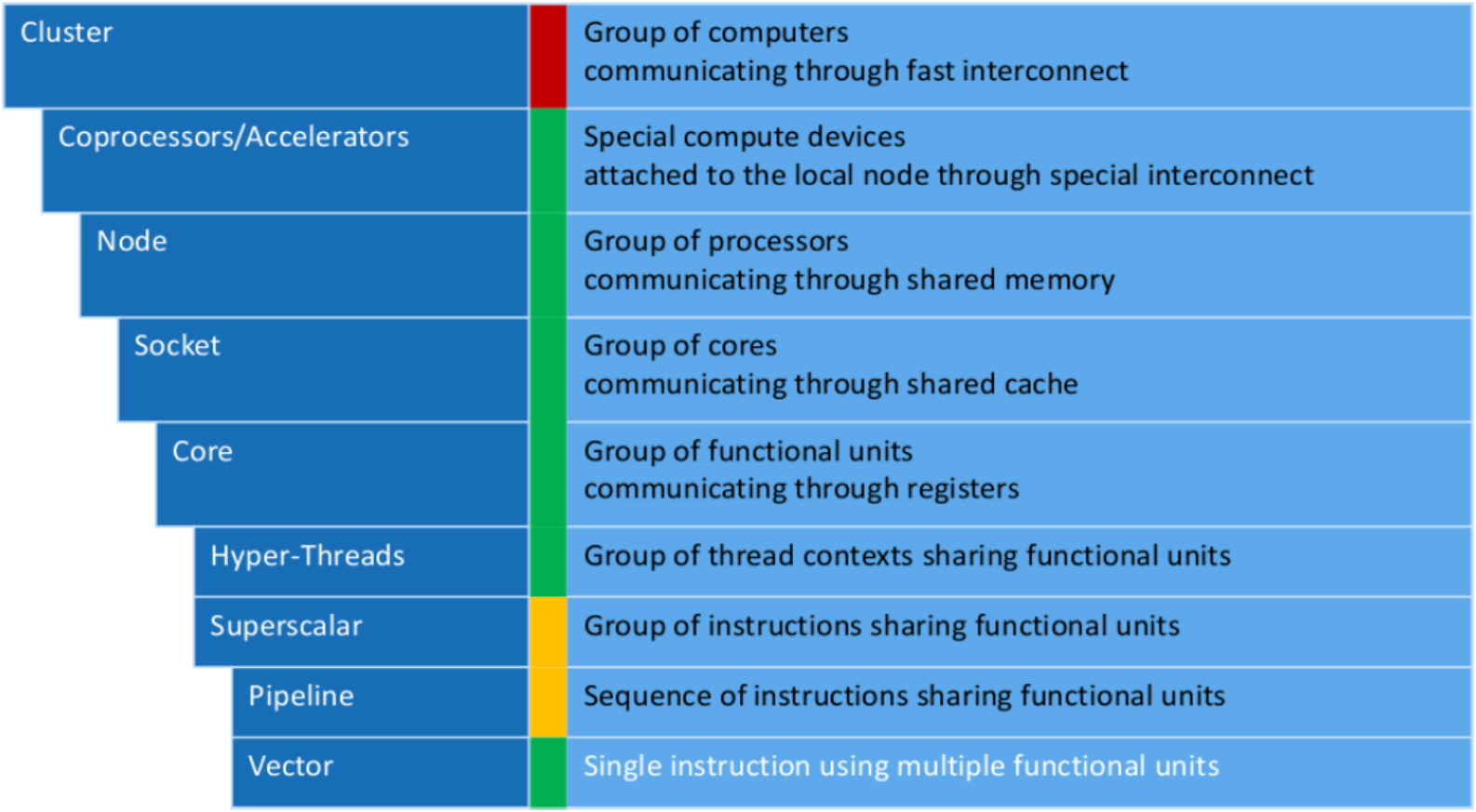
I limiti della legge di Moore si sono raggiunti negli ultimi anni (attorno al 2005) a causa dei limiti fisici imposti per la riduzione delle dimensione dei transistor. La frequenza di lavoro si è stabilizzata attorno a 2-3 Ghz. La conseguenza è stata l'introduzione della tecnologia **multicore** all'interno dello stesso processore.

Le prestazioni complessive continuano a crescere esponenzialmente ma in modo distribuito su più core o attraverso altre tecniche hardware.

Per sfruttare le prestazioni dell'hardware occorre **distribuire in parallelo il carico computazionale dell'applicazione.**

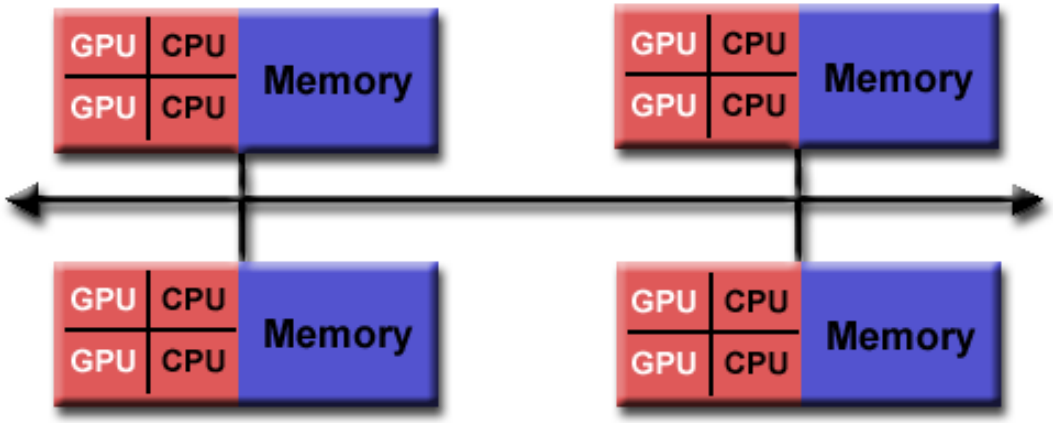
Livelli di parallelismo

E' possibile parallelizzare il flusso di esecuzione di un algoritmo a diversi livelli hardware: all'interno di un core (vettorizzazione, pipeline, superscalare, hyperthreading), all'intero di un nodo (multi-core, multi-socket), utilizzando acceleratori (e.g. GPU), tra più nodi che cooperano all'interno di un cluster.
Negli ultimi anni queste tecnologie stanno avendo una rapida evoluzione.



Architetture parallele

Le elevate prestazioni di un moderno calcolatore HPC si ottengono grazie ad architetture hardware in cui vengono sommate le prestazioni che possono essere fornite dai diversi livelli di parallelismo.



Lo sfruttamento di queste architetture si ottiene suddividendo il carico computazionale il task paralleli progettati attraverso l'utilizzo combinato di modelli di programmazione specifici per i diversi livelli di parallelismo. I principali modelli sono:

- Programmazione multithreading per il parallelismo multicore in architetture a memoria condivisa.
- Programmazione a scambio di messaggi tra processi in architetture multinodo a memoria distribuita.
- Programmazione di kernel di calcolo da eseguire su acceleratori (GPU).

Case study: Leonardo (CINECA)

Info generali: *https://leonardo-supercomputer.cineca.eu/it*
Caratteristiche HW: *https://www.hpc.cineca.it/systems/hardware/leonardo/*

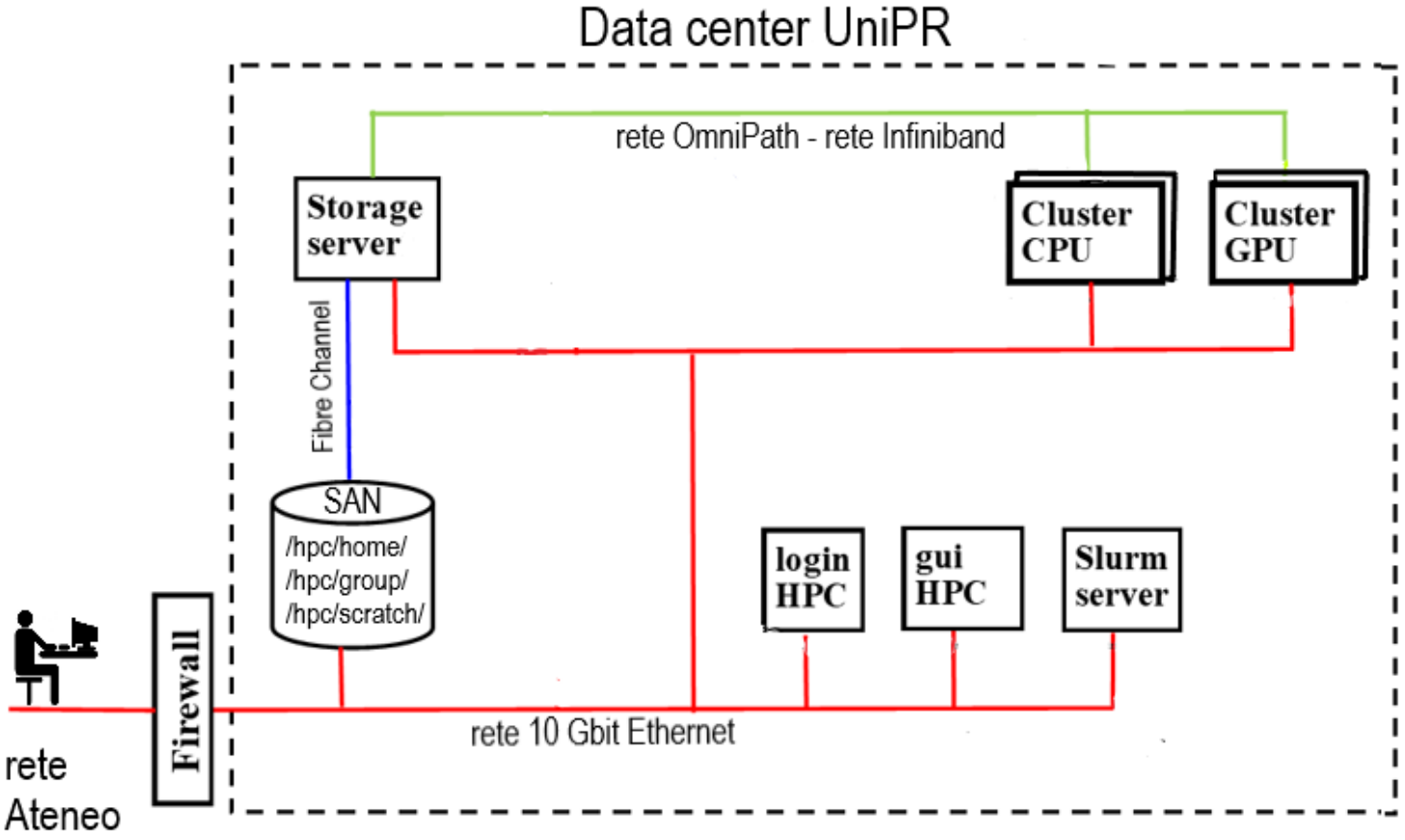
3456 booster nodes (BullSequana X2135)

- Ogni nodo: 1 CPU Intel Xeon 8358 32 cores 2.6 GHz, RAM: 0.5TB, GPU: 4 x Nvidia A100
- 240 PFlops

1536 data-centric nodes (BullSequana X2140)

- Ogni nodo: 2 CPU Intel Saphire 56 cores 4.8 GHz, RAM: 1.5TB
- 9 PFlops

Case study: Il cluster HPC.unipr.it



Riferimenti:

- *www.hpc.unipr.it*
- *User Guide*

Nodi Cluster CPU

22 DualSocketBDW:	11 Tflops
18 DualSocketAMD:	9 Tflops
1 QuadSocketBDW	1 Tflops
8 QuadSocketSKL:	22 Tflops
4 Intel KNL	6 Tflops

51 TFlops d.p.

nodi con GPU

12 NVIDIA P100:	56 Tflops
2 NVIDIA V100:	14 Tflops
14 NVIDIA A100:	135 Tflops

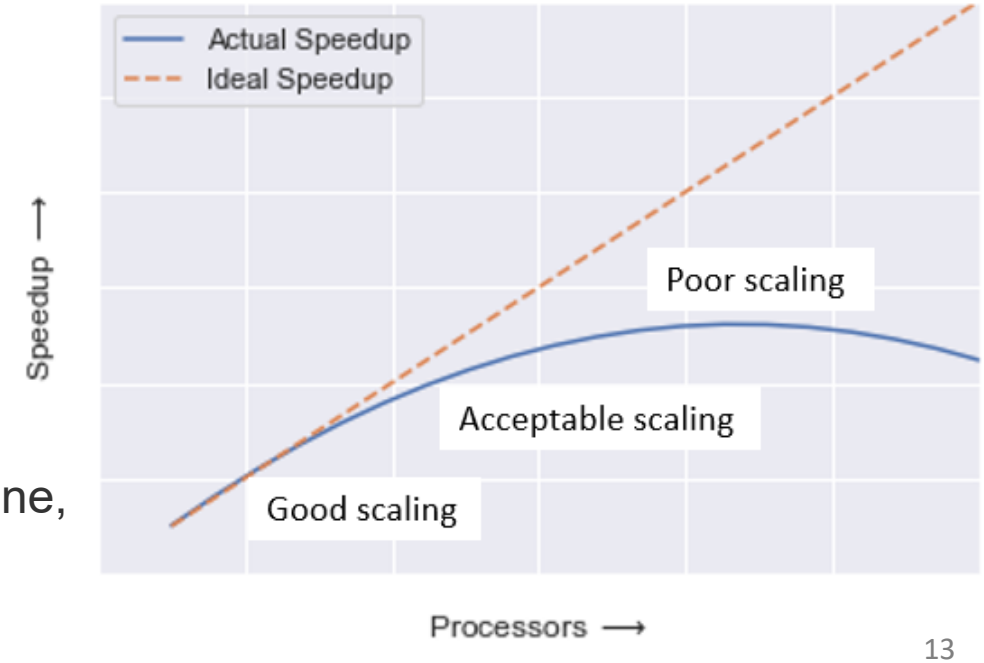
205 Tflops d.p.

PROGETTAZIONE DI PROGRAMMI PARALLELI

Dal punto di vista della programmazione esistono diverse metodologie di progettazione per scomporre il carico computazionale in **Task** che verranno poi distribuiti sulle diverse unità di processamento. La scomposizione dovrà tenere conto **dell'organizzazione dei dati**, delle **componenti funzionali dell'algoritmo** e dei **livelli di parallelismo messi a disposizione dal sistema di calcolo**.

Con il termine **Speedup** si intende la misura dell'accelerazione del tempo di calcolo rispetto al programma non parallelizzato: **SpeedUp = Tseriale / Tparallelo**, che nel caso ideale **coincide con il numero di processori**

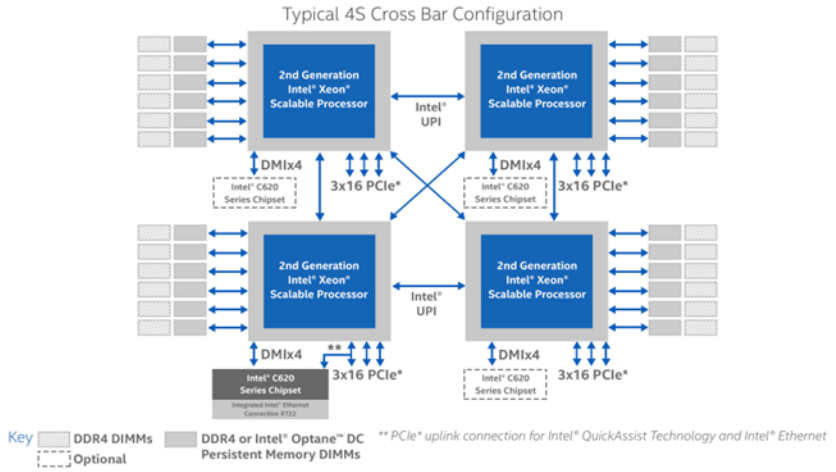
Scalabilità è la capacità del programma parallelo di mantenere una crescita proporzionata dello Speedup al crescere delle unità di processamento.

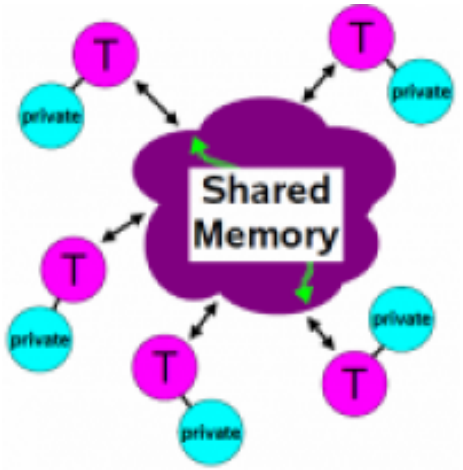


Le limitazioni della scalabilità sono dovute a **Overhead** introdotti dalla parallelizzazione, come al esempio la comunicazione e il sincronismo tra i Task.

Programmazione a memoria condivisa

Una tipica architettura a memoria condivisa è realizzata con **sistemi SMP** (Symmetric Multi Processor) in cui un pool di processori omogenei operano in modo indipendente ma condividono lo spazio di indirizzamento della memoria RAM.



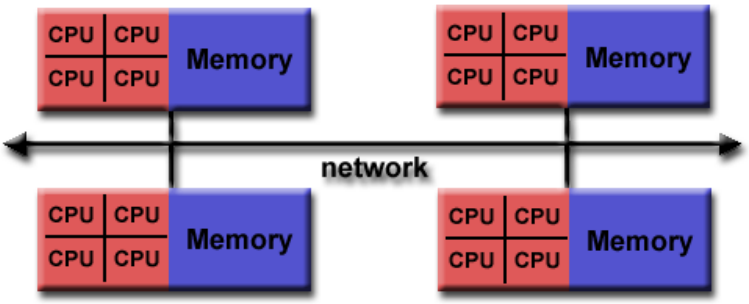


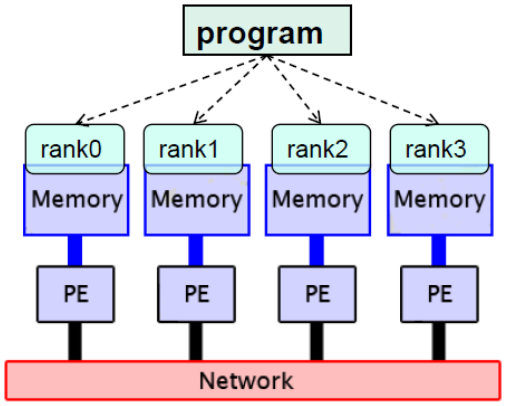
I task del programma possono essere assegnati a thread in esecuzione su differenti unità di processamento. La comunicazione tra i task avviene mediante l'accesso sincronizzato a variabili condivise.

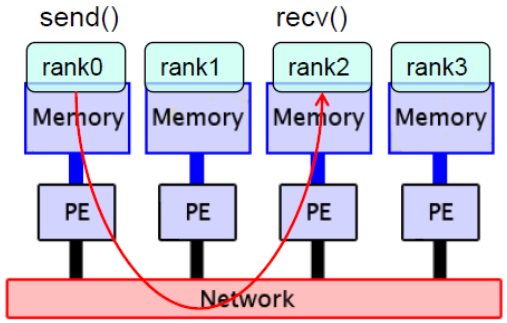
Le librerie che possiamo utilizzare per la programmazione sono Posix Threads e **openMP** (Open Multiprocessing www.openmp.org) che è una libreria ideata specificamente per applicazioni parallele a memoria condivisa ed è supportata da vari linguaggi di programmazione come il C/C++ e il Fortran.

Programmazione a memoria distribuita

Una architettura a memoria distribuita è costituita da un cluster di computer interconnessi da una rete di comunicazione.







I task (processi) sono eseguiti sulle CPU del cluster e hanno il proprio spazio di indirizzamento.

La comunicazione richiede una specifica libreria in grado di implementare diversi modelli di comunicazione come punto–punto (tra coppie di task) o globali (tra tutti i task) utilizzando percorsi fisici diversi in base alla localizzazione dei task.

MPI (Message Passing Interface) è la specifica emanata dall'MPI Forum per lo sviluppo di una libreria standard per lo scambio di messaggi. Le implementazioni possono essere opensource (esempio Open MPI) o commerciali (come Intel MPI).

Programmazione GPU

La GPU è nata come coprocessore per il rendering di immagini su un dispositivo di visualizzazione, ma vista la sua programmabilità dal 2005 si è iniziato ad utilizzarla come coprocessore della CPU.

Sono quindi nate le GP-GPU (General Purpose GPU) sprovviste di uscita video.

Ad oggi il costruttore principale di GP-GPU è NVIDIA che fornisce un modello di programmazione **CUDA**, la libreria e il relativo compilatore **nvcc**.

La programmazione consiste nel costruire un kernel di calcolo che verrà tipicamente inviato assieme ai dati dall'host alla GPU, la quale elabora i dati e al termine invia i risultati all'host.