UNIVERSITÀ DI PARMA

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

Introduzione

Programmazione parallela e HPC - a.a. 2022/2023

Roberto Alfieri – Alessandro Dal Palù

Programmazione Parallela e HPC: sommario

Introduzione all'HPC

Alcuni Rifermenti:

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

Motivazioni dell'HPC: problemi "grand challenge"

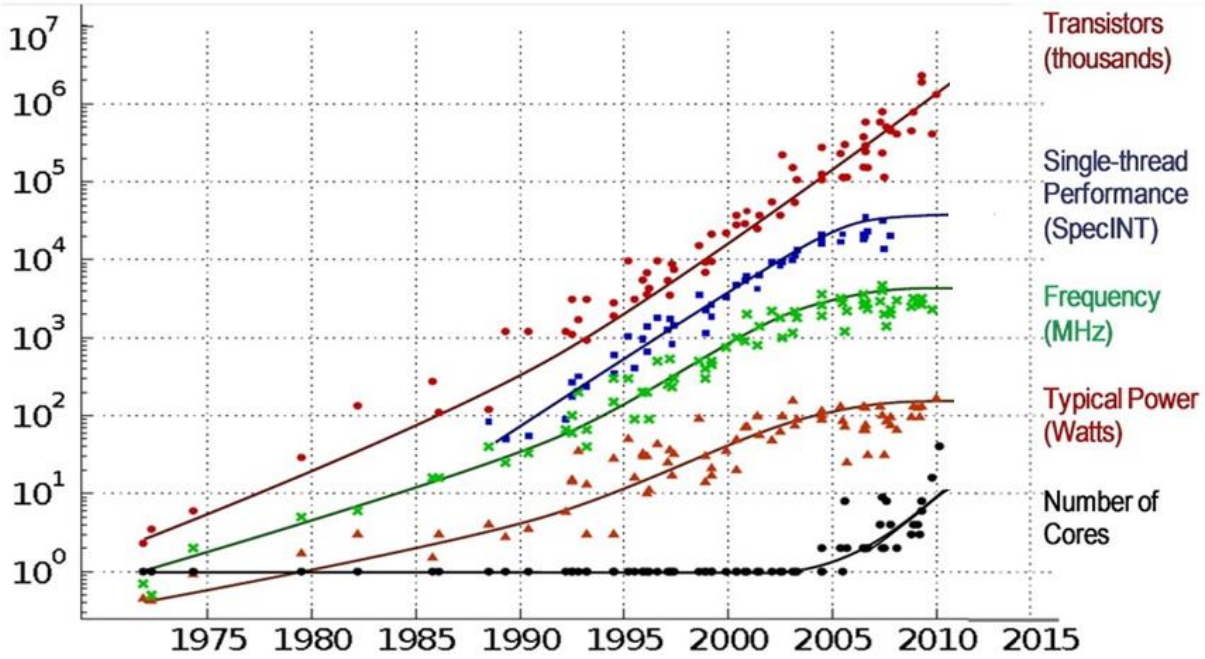
Problemi di notevole impatto in diversi ambiti che possono essere risolti computazionalmente ma richiedono notevoli risorse di calcolo e storage per dare una soluzione in un tempo ragionevole.

Esempi:

- Scienza e Ingegneria in UNIPR
- Difesa (crittografia): Sicurezza RSA
- Climatologia: ECMWF DataCenter ECMWF
- Aerodinamica: Dallara
- Energia: ENI
- Vita extraterrestre: SETI@HOME Boinc
- Data Science: @CINECA

Hardware performance: la legge di Moore e i sui limiti

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



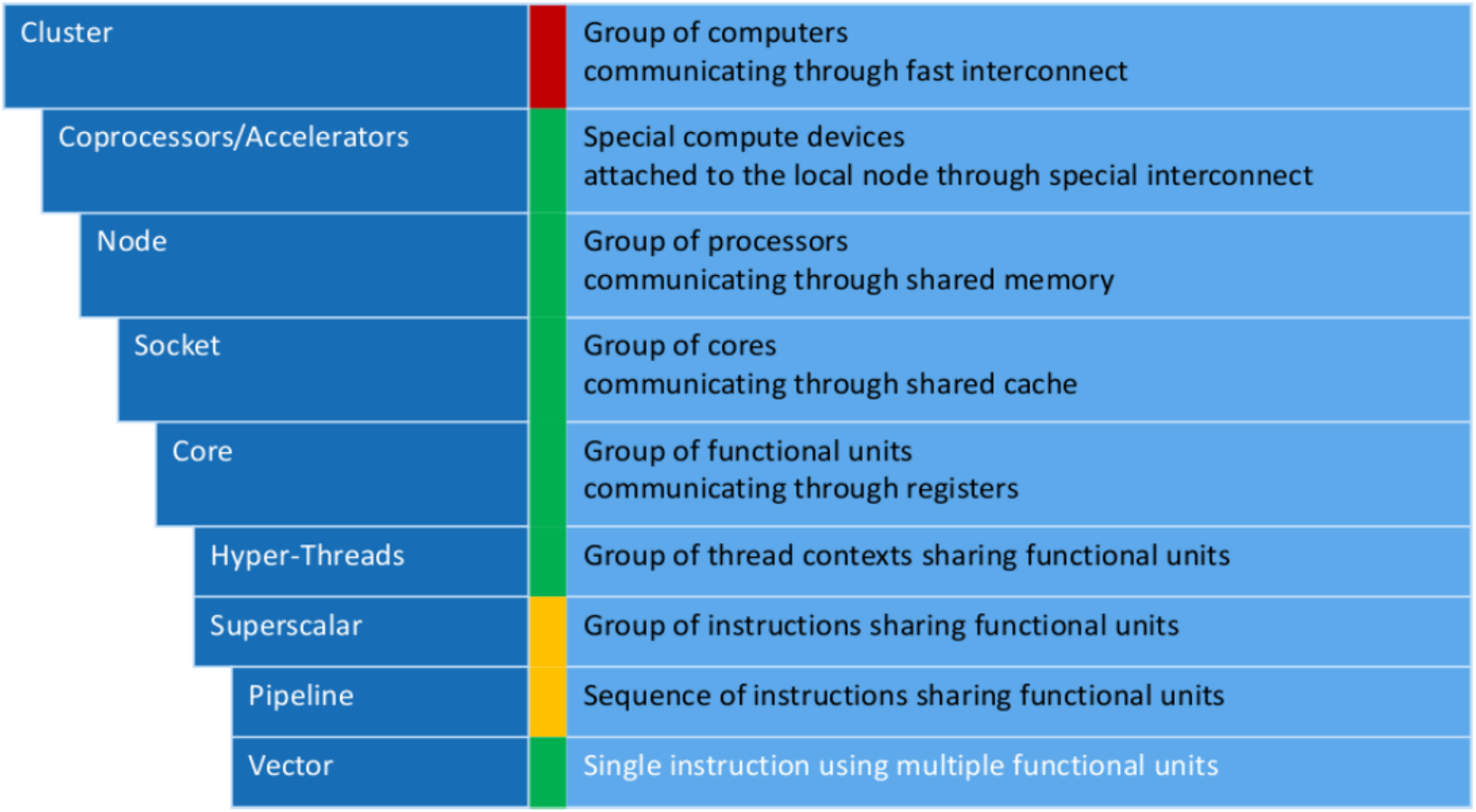
I limiti della legge di Moore si sono raggiunti negli ultimi anni (attorno al 2005) a causa dei limiti fisici imposti per la riduzione delle dimensione dei transistor. La frequenza di lavoro si è stabilizzata attorno a 2-3 Ghz. La conseguenza è stata l'introduzione della tecnologia **multicore** all'interno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

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 di un cluster. Negli ultimi anni queste tecnologie stanno avendo una rapida evoluzione.



Efficiency trand and Efficiency Gap

Il parallelismo dell'hardware cresce ed evolve continuamente consentendo di avere nel tempo una **crescita esponenziale** delle prestazioni ma al contempo la complessità nella programmazione de diversi livelli di parallelismo porta ad una crescente **difficoltà nello sfruttamento delle risorse disponibili**.

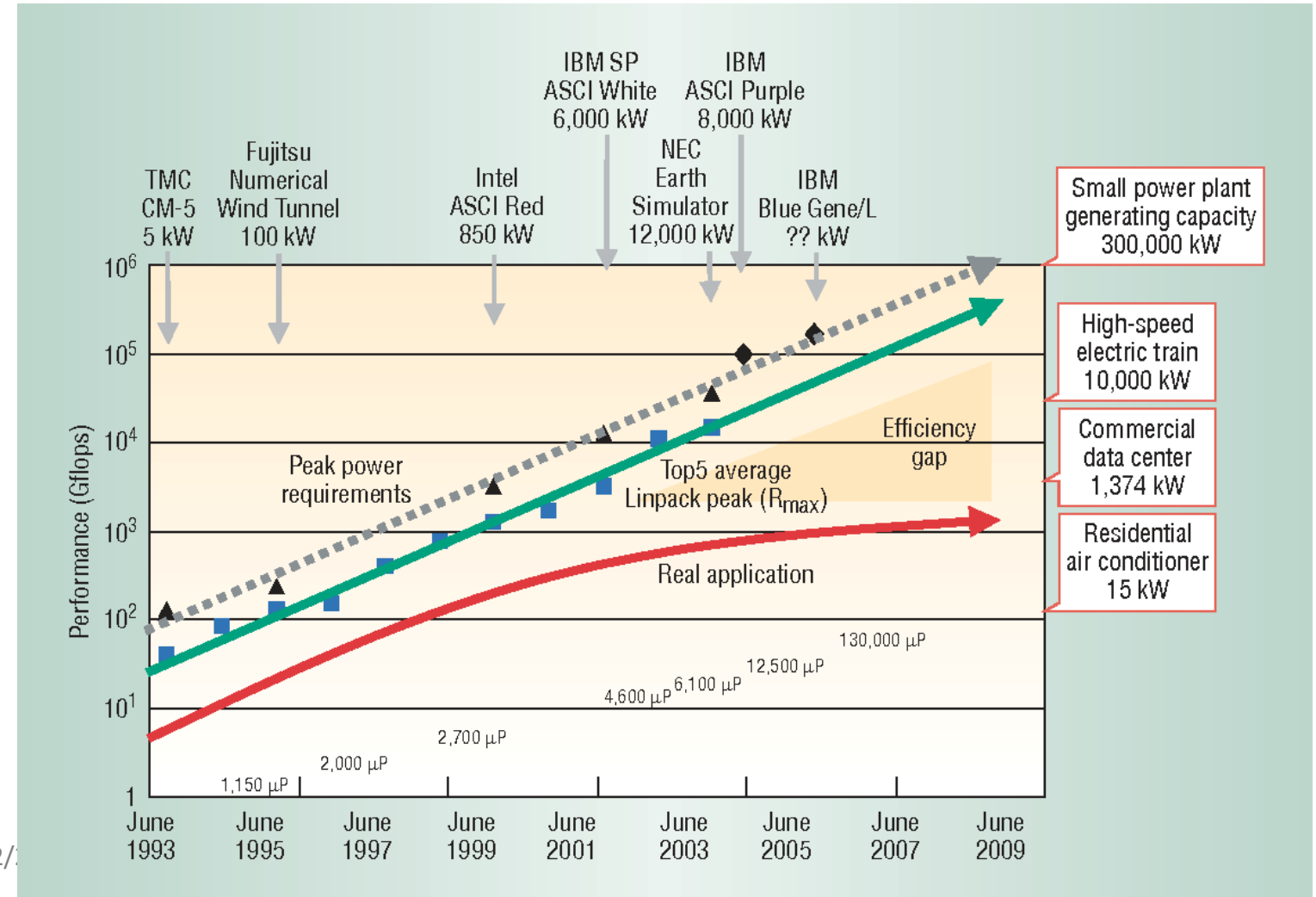


Figure 1. Super-computer performance and power trends. Scientific applications' growing computational demands have led to exponential increases in system peak performance, power consumption, and complexity requirements.

TOP500

TOP500 è un sito (www.top500.org) in cui vengono elencati i 500 calcolatori più potenti al mondo.

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	8,730,112	1,102.00	1,685.65	21,100
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,463,616	174.70	255.75	5,610

Novembre 2022

Consumo energetico

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

Alcune definizioni:

TDP (Thermal Design Power) si riferisce al consumo di energia sotto il massimo carico teorico.

La sua unità di misura è il watt e il suo valore viene dichiarato dai produttori.

PUE (Power Usage Effectiveness) 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.

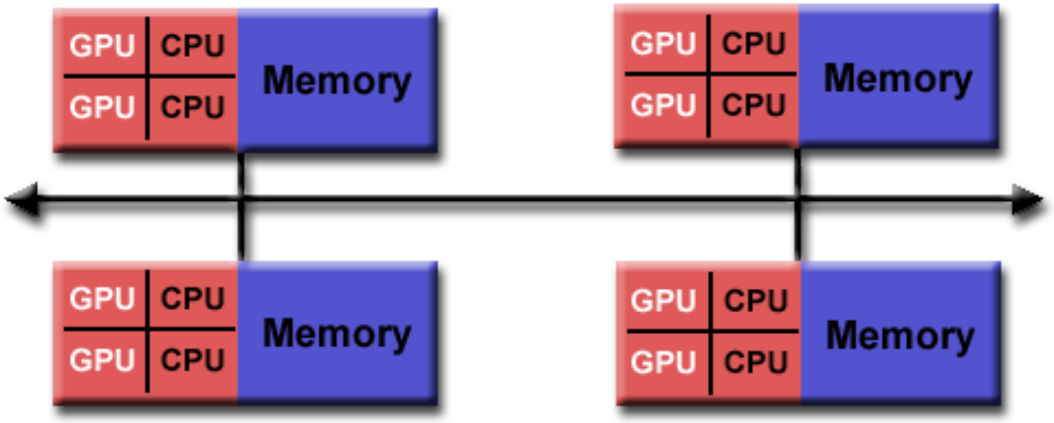
PUE rappresenta una misura di quanto efficiente sia un centro di calcolo, o data center, nell'usare l'energia elettrica che lo alimenta Il PUE è considerato efficiente se inferiore al 1,8. (CINECA)

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).

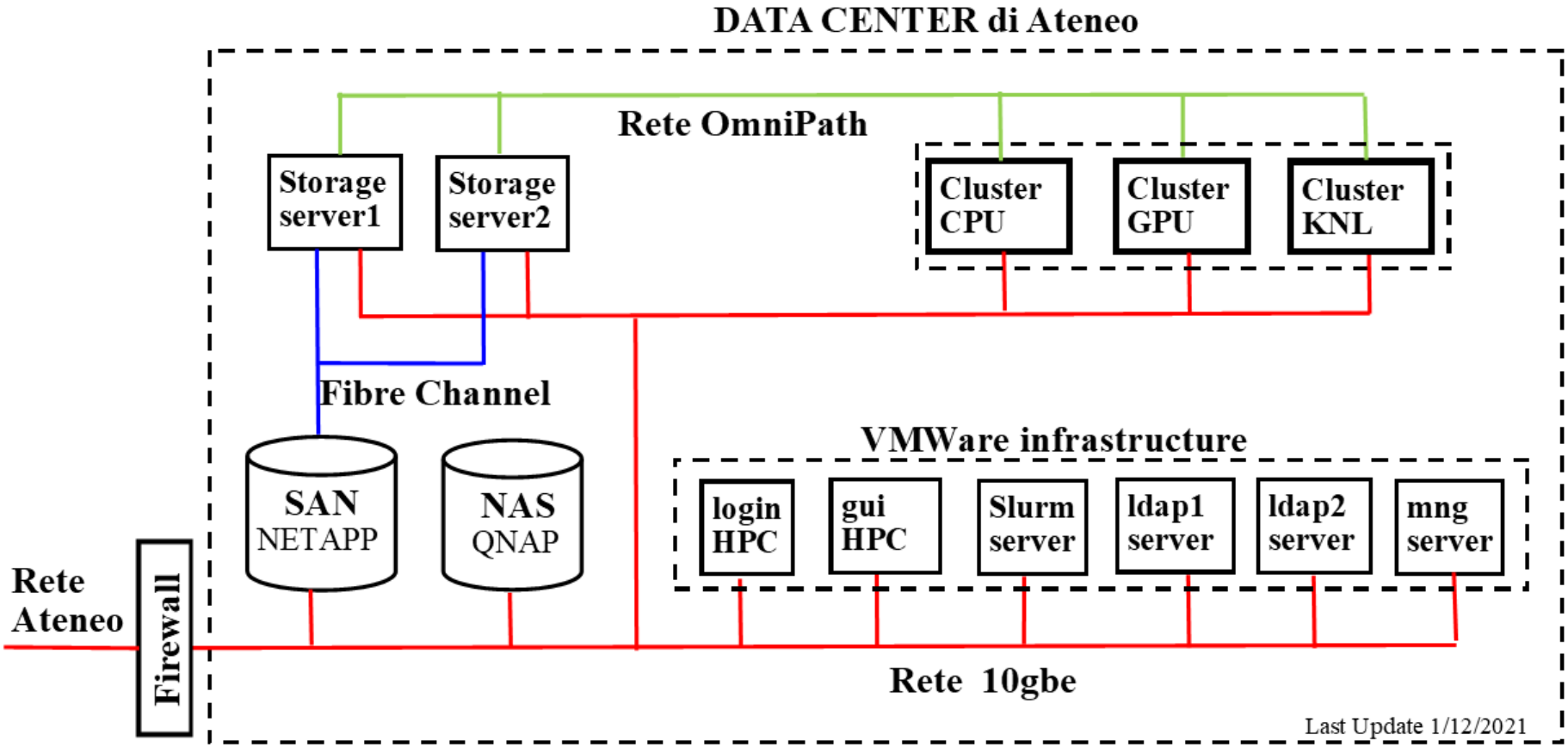
Architetture parallele

Nei sistemi di calcolo moderni l'accelerazione (speedup) si ottiene distribuendo il carico computazionale su di una architettura hardware in grado di sfruttare le performance dei diversi livelli di parallelismo su

- Diversi thread o processi su un singolo nodo (memoria condivisa)
- Diversi processi su più nodi interconnessi (memoria distribuita)
- Kernel di calcolo eseguiti su acceleratori (GPU)



Il cluster HPC.unipr.it



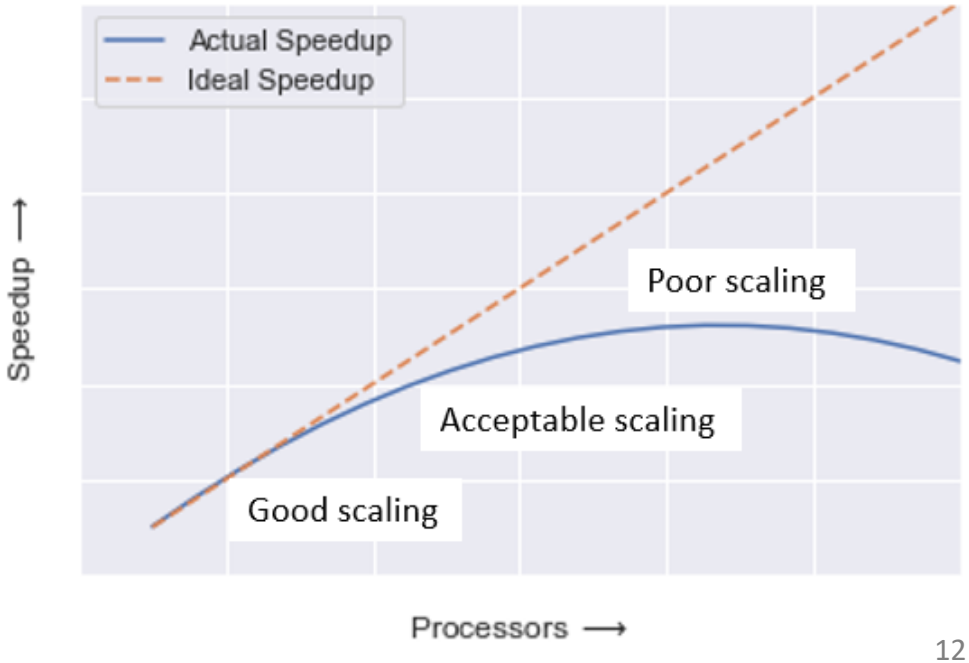
PROGETTAZIONE DI PROGRAMMI PARALLELI

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

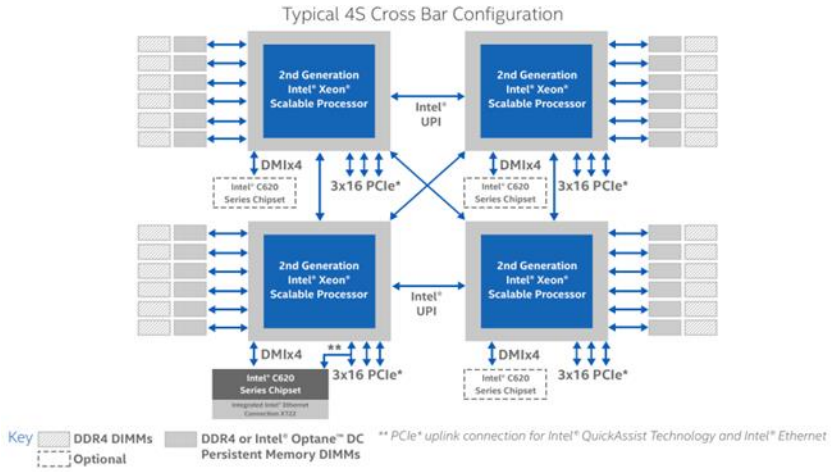
Scalabilità è la capacità del programma parallelo di mantenere una crescita proporzionata dello Speedup al crescere delle unità di processamento.

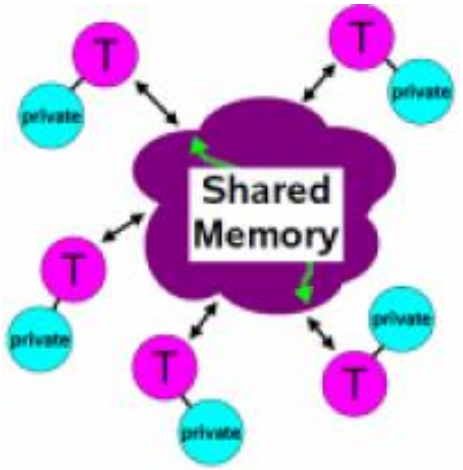
Le limitazioni della scalabilità sono dovute a **Overhead** introdotti dalla parallelizzazione, come al esempio la comunicazione e il sincronismo tra i Task.



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.



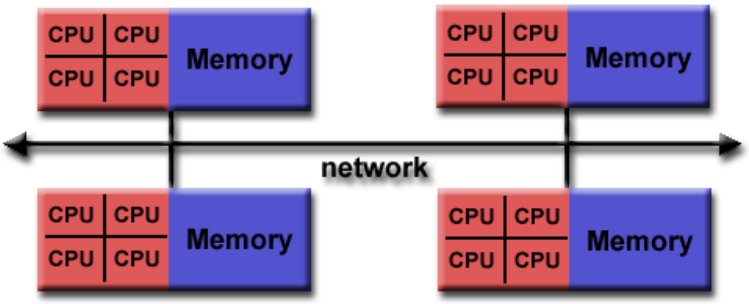


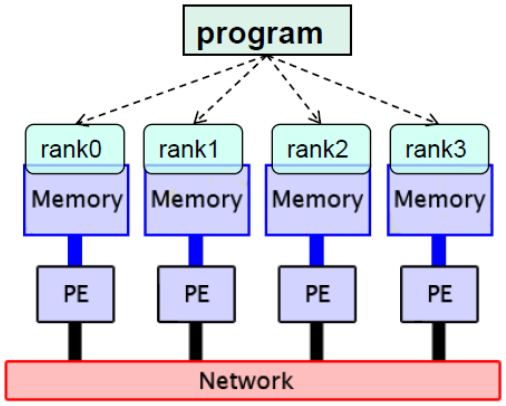
I task del programma possono essere assegnati a thread in esecuzione su differenti unità di processamento. La comunicazione tra i task avviene mediante l'accesso sincronizzato a variabili condivise.

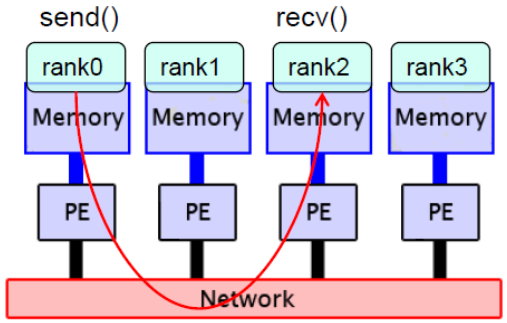
Le librerie che possiamo utilizzare per la programmazione sono Posix Threads e openMP (Open Multiprocessing www.openmp.org) che è una libreria ideata specificamente per applicazioni parallele a memoria condivisa ed è supportata da vari linguaggi di programmazione come il C/C++ e il Fortran.

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.