UniTs - University of Trieste

Faculty of Scientific and Data Intensive Computing

Department of mathematics informatics and geosciences

High Performance Computing

Lecturer:
Prof. Stefano Cozzini

Authors:

Andrea Spinelli
Christian Faccio

March 4, 2025

github.com/Spina02,christianfaccio

andreaspinelli2002@gmail.com,christianfaccio@outlook.it

Abstract

Contents

1 Introduction

Definition: ***High Performance Computing***

High Performance Computing (HPC) is the use of servers, clusters and supercomputers - plus associated software, tools, components, storage and services - **for scientific, engineering or AI tasks** that are particularly intense in computation, memory usage or data management.

As models become more complex and new data bring in more information, we require ever increasing computational power to solve these problems in a reasonable amount of time. This is where High-Performance Computing comes in.
Insert image...
Organizations are explaning their definitions of HPC to include worlloads such as AI and high-performance data analytics (HPDA) in addition to traditional HPC simulation and modeling workloads.
Its elements include:

- **Hardware:** servers, clusters, supercomputers, storage, networking, etc.
- **Software:** compilers, libraries, tools, applications, etc.
- **Services:** consulting, training, support, etc.
- **People:** users, administrators, developers, etc.
- **Data:** input, output, storage, management, etc.

Human capital is by far the most important aspect of HPC. The best hardware and software in the world is useless without skilled people to use it.

- HPC providers need to provide training and support to users.
- HPC users need to be trained and supported to use the hardware and software effectively.

What about performance and metrics?

Performance is the most important metric in HPC. It is the measure of how well a system is performing a given task. It is usually measured in terms of time to solution, which is the time it takes to complete a given task. But, it is not always what matters… to reflect a greater focus on the productivity, rather than just the performance, of HPC systems, the term **Productivity** is often used.

$$\text{Productivity} = \frac{\text{Application Performance}}{\text{Application Effort}}$$

- `How fast can I do things on my CPU?`
  We count the number of operations per second that a CPU can perform. This is called **FLOPS** (Floating Point Operations Per Second). The theoretical peak performance is determined by the number of cores, the clock speed, and the number of operations per cycle.

  $$\text{FLOPS} = \text{Cores} \times \text{Clock Speed} \times \text{Operations per Cycle}$$

  It is not easy to be defined for real applications, so benchmarks are used to measure the performance of a system. The TOP500 list is a list of the 500 most powerful supercomputers in the world, ranked by their performance on the LINPACK benchmark. The LINPACK benchmark measures the performance of a system solving a system of linear equations. See `https://www.top500.org/`.

- `How fast can I move data around?`
- `How much data can I store?`

2 Hardware and Software

In the classical Von Neumann architecture there is only one processing unit (CPU) that processes instructions, the ALU is responsible for math and logic operations and the register stores data.

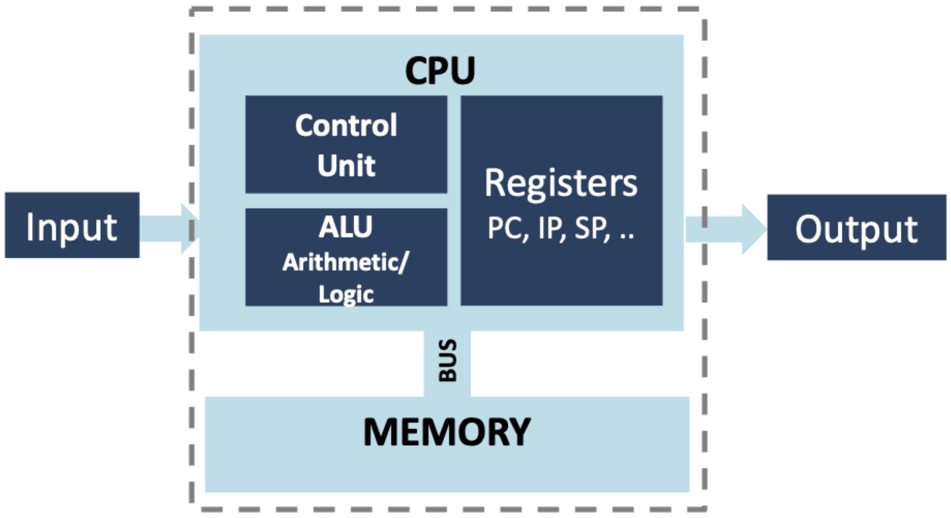


Figure 2.1: Von Neumann architecture

One instruction is executed at a time, the CPU fetches the instruction from the memory, decodes it and executes it. The CPU can access the memory to read or write data. Accessing any location in the mempory has always the same cost, this is called the **uniform memory access** (UMA).

2.1 Moore's Law

Definition: ***Moore's Law***

It states that the number of transistors in a dense integrated circuit doubles about every two years.

How can we go from Moore's Law to processor performance? Through Dennard Scaling:

"Power density stays constant as transistors get smaller"

Intuitively,

- **Smaller transistors** → shorter propagation delay → faster frequency
- **Smaller transistors** → smaller capacitance → lower voltage

$$Power \propto Capacitance \times Voltage^2 \times Frequency$$

But… even with smaller transistors, we cannot continue reducting power, there are then two options:

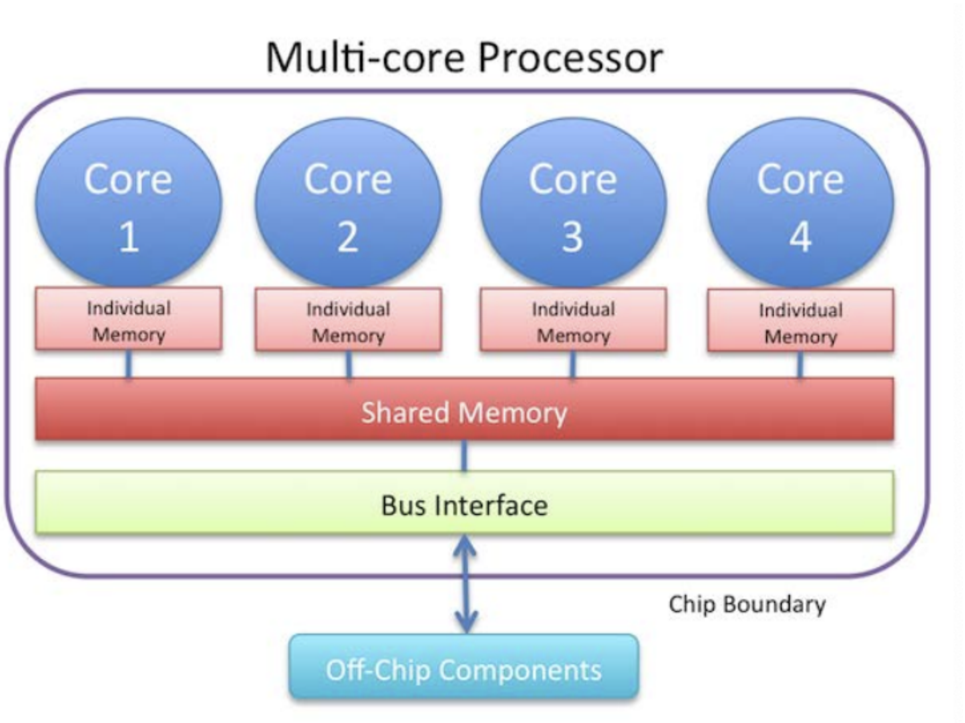


Figure 2.2: Multicore processors

- **Increase power**
- **Stop frequency scaling**

From 2006, single-core performance stopped increasing, so the only way to increase performance is to use more cores. The first solution is to write efficient software to make the efficient use of hardware resources. The second is to use specialized architectural solutions, like GPUs, FPGAs, etc.

Today, **CPUs are multicore processors**, lowering clock frequency because of power and heat dissipation, but packing more computing cores onto a chip. These cores will share some resources (memory, network, disk, . . .) but are still capable of independent calculations.

2.2 Parallel Computers

Flynn Taxonomy (1966) is used to classify parallel computers based on the number of instruction streams and data streams.

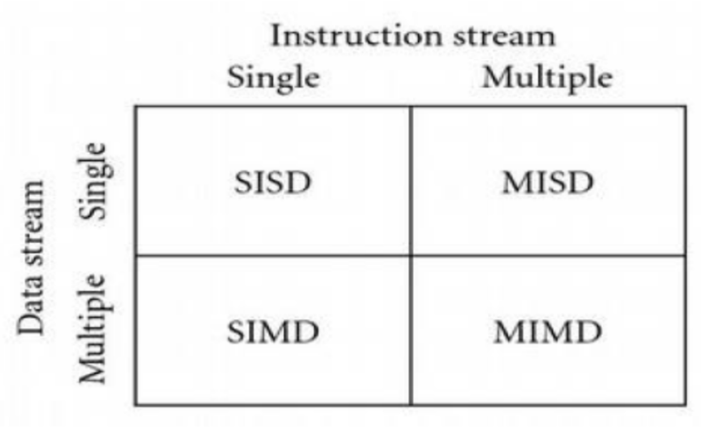
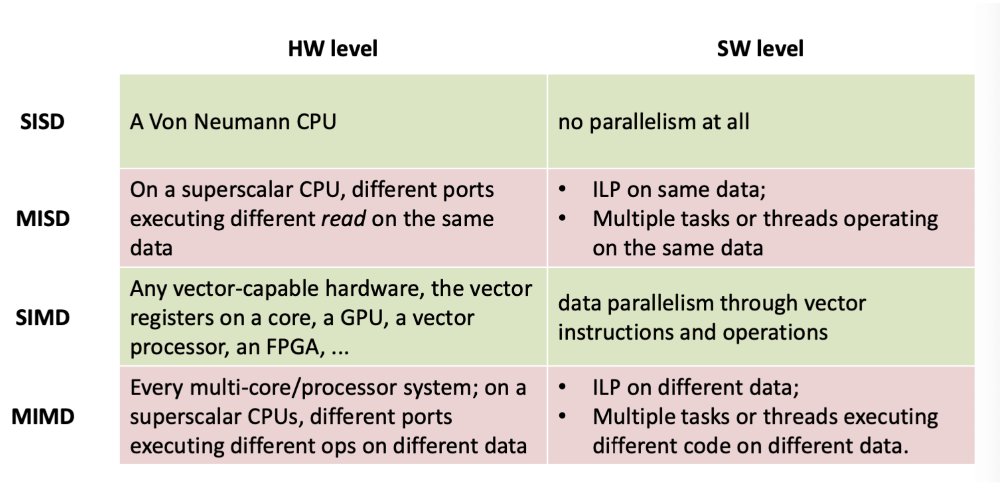


Figure 2.3: Flynn Taxonomy



	HW level	SW level
SISD	A Von Neumann CPU	no parallelism at all
MISD	On a superscalar CPU, different ports executing different *read* on the same data	• ILP on same data; • Multiple tasks or threads operating on the same data
SIMD	Any vector-capable hardware, the vector registers on a core, a GPU, a vector processor, an FPGA, …	data parallelism through vector instructions and operations
MIMD	Every multi-core/processor system; on a superscalar CPUs, different ports executing different ops on different data	• ILP on different data; • Multiple tasks or threads executing different code on different data.

Figure 2.4: Parallel computers

The essential components of a HPC cluster are:

- **Compute nodes**: the actual computers that perform the calculations
- **Interconnect**: the network that connects the compute nodes
- **Storage**: the disk space where data is stored
- **Software**: the operating system and the software stack that runs on the cluster

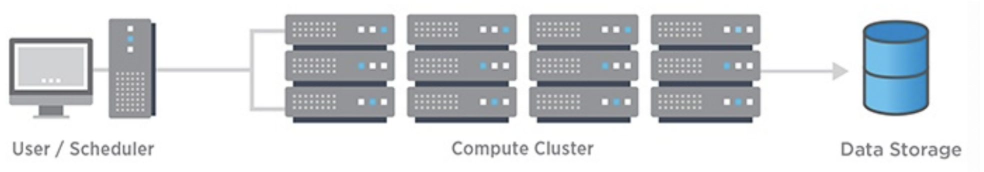


Figure 2.5: HPC cluster

A core is the smallest unit of computing, having one or more threads and is responsible for executing instructions.

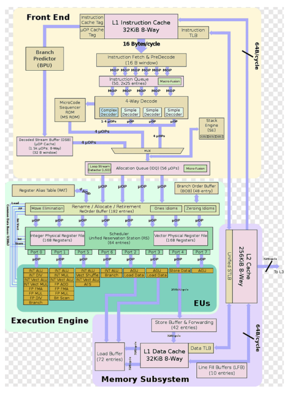


Figure 2.6: Core

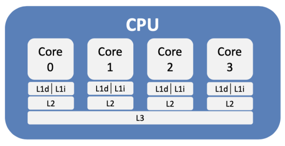


Figure 2.7: Cache hierarchy

Cache hierarchy can have different topologies.

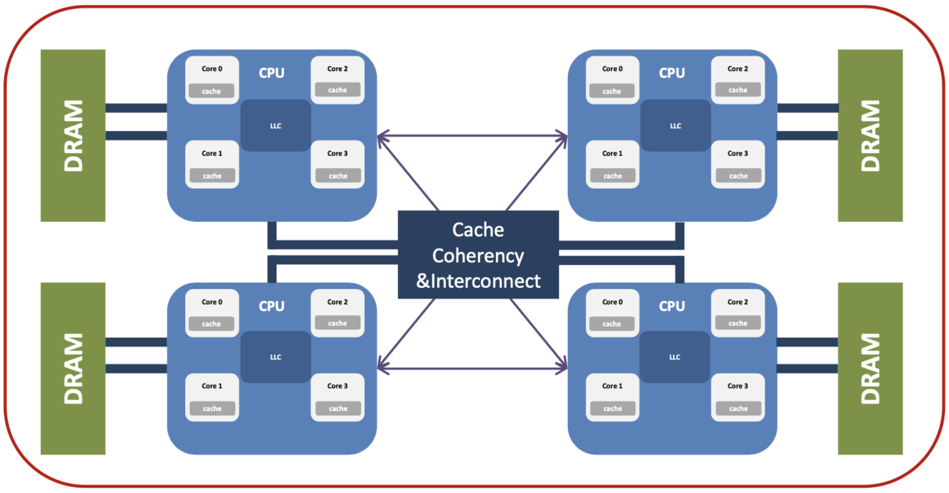


Figure 2.8: Node topology

Tip:

Cache hierarchy has been invented cause accessing memory (moving data) takes almost 100x times computing (doing operations). The cache is a small memory that stores the most frequently accessed data. The cache is faster than the main memory but smaller. The cache is divided into levels, the first level is the fastest but the smallest, the second level is slower but bigger, and so on.

In a cluster there are different types of networks:

- **High-speed network**: used for communication between nodes (parallel computation, low latency/high bandwidth, infiniband or ethernet as examples)
- **I/O NETWORK**: used for communication with storage (I/O requests, latency not fundamental/-good bandwidth, NFS, Lustre, GPFS as examples)
- **In band Management Network**: used for cluster management, monitoring, and control, LRMS (Load Resource Management System) as examples
- **Out of band Management Network**: used for cluster management, remote control of nodes and any other device, IPMI (Intelligent Platform Management Interface) as examples

What about **memory**?

It is fundamental and on a supercomputer there is a hybrid approach as for the memory placement. The memory on a single node can be accessed directly by all the cores on that node (**shared-memory**). When many nodes are used at a time, a process cannot directly access the memory on a different node, it needs to issue a request for that. That is named **distributed-memory**.

Shared Memory

- **Uniform Memory Access (UMA)**:
  Each processor has uniform access to memory. Also known as symmetric multiprocessors (SMP).

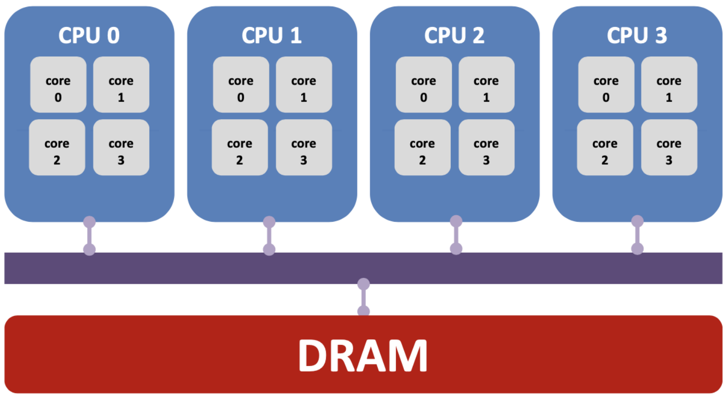


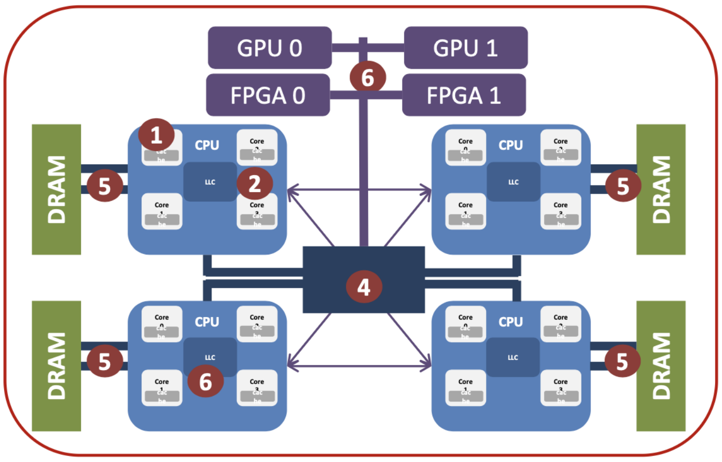
Figure 2.9: Shared memory - UMA

- **Non-Uniform Memory Access (NUMA)**: Time for memory access depends on location of data. Local access is faster on non-local access.

Warning: ***Challenges for multicore***

It aggravates the **Memory Wall problem**, where the memory access time is much slower than the CPU speed.

- **Memory bandwidth**: the memory bandwidth is limited and shared among all the cores
- **Memory latency**: the memory latency is high and can be a bottleneck
- **Memory contention**: the memory contention can be a problem when multiple cores access the same memory location



1. ILP/SIMD units
2. Cores
3.
4. Socket/ccNuma domains
5. Inner cache levels
6. Multiple accelerators

Figure 2.10: Parallelism within a HPC node