CPA – Parallel Computing

Informatics Engineering Degree

T1. Introduction to Parallel Computing

J. M. Alonso, F. Alvarruiz, I. Blanquer, D. Guerrero, J. Ibáñez,
E. Ramos, J. E. Román

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Year 2016/17

Contents

Section 1

Introduction

- Parallel Computing Applications
- Supercomputing

Parallel Computing

What is parallel computing?

The computation performed on a parallel computer that

- Executes concurrently several instructions for the resolution of a single problem

On the contrary, a sequential computer

- Follows the conventional (von Neumann) architecture
- Instructions are executed sequentially one after the other

Parallel computing requires

- Redesigning the algorithms
- Changing the data structures

$\longrightarrow$ parallel programming

Motivation

Why parallel computing?

There is a tendency to include more parallelism support in the hardware

- Clock frequency limited by the light speed
- Larger integration scale unfeasible due to heating problems

Examples: Game consoles with more than 400 cores
$\longrightarrow$ There is a lack of programmers capable of exploiting such hardware!

Sometimes, sequential computing does not cover the needs

- Large scale problems
- Real time

Example: many matrix algorithms have a cost $O(n^3)$ requiring problem dimensions of $n = 10^6$ or even larger.

Numerical Simulation

Simulation: emulate a physical system using a computer

In Engineering:

- Virtual prototyping
- Cost and production cycle reduction

In Science:

- Key for scientific progress
- Complex systems: complex geometry, multi-physic models, non-linearity

Often, simulation requires a large computation capacity.

Simulation: Example

Simulation of physiological behaviour of organs, tissues and cells (VPH: *Virtual Physiological Human*)

- Long term vision: To simulate the complete physiological behaviour of the human body

Example: Alya-Red

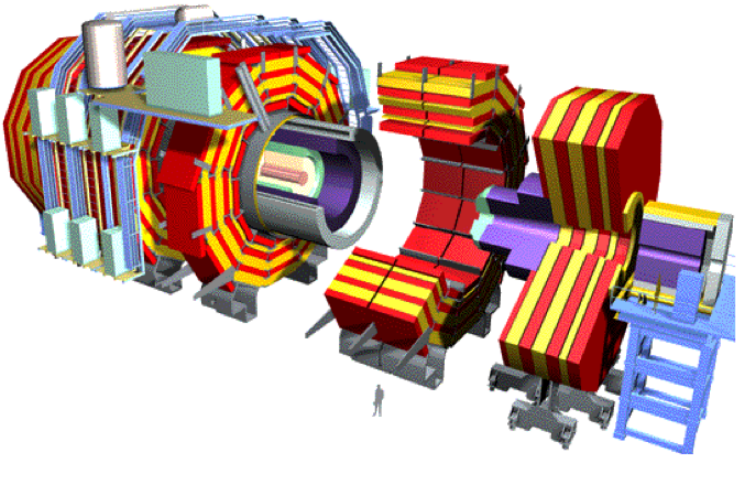
- Is a computational simulator of a heart that models the electric, visco-elastic, fluid-dynamic and physic-chemical behaviour of a whole heart
- 1st award of divulgation scientific videos NSF 2012: www.youtube.com/watch?v=tKD2hfF27rM

Data processing

Large scientific challenges require acquiring, storing and processing large data volumes

- Disciplines such as High Energy Physics, Astrophysics or Biotechnology
- Data coming from radiotelescopes, particle colliders or human genome

Example: The Large Hadron Collider at CERN



Supercomputing

Parallel computing, High Performance Computing or Supercomputing, consist on the concurrent execution of different tasks by different processing units.

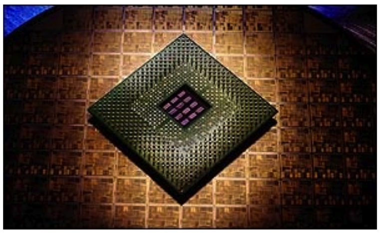
Concurrency may happen at different scales:

- In a single processor
- In a single computer
- In a local network
- In the Internet

Supercomputing

In a single processor

- Several cores in the same processor sharing some key units
- Simultaneous execution of multiple basic instructions



In a single computer

- Integration of different processors in a single computer
- Depending on the coupling of processors:
  - Shared memory
  - Distributed memory

Parallel Computers

Shared Memory

- All processors can access to the whole memory
- Different latencies depending on the memory bank accessed (due to proximity)
- Scalability up to hundreds of processors
- High cost; high performance in fine-grain parallelism

Distributed Memory

- Each processor has only access to its local memory block
- Information is explicitly exchanged through messages
- Higher scalability (thousands of processors)
- Reduced cost; worse performance in the fine grain

TOP500

In the TOP500 list the 500 most powerful computers in the world are cited
http://www.top500.org

- Updated twice a year
- Computers are classified accordingly to their sustained performance, measured in floating-point operations per second (FLOPS)

#	Centre	System	Manuf.	Cores	RMax
1	NSC (China)	Sunway TaihuLight - Sunway SW26010 260C 1.45GHz, Sunway	NRCPC	10.649.600	93014
2	NSC (China)	Tianhe-2 - NUDT Xeon E5-2692 12C + Xeon Phi, TH Express-2	NUDT	3.120.000	33863
3	ORNL (USA)	Titan - Cray XK7, Opteron 6274 16C + Tesla K20x, Cray Gemini	Cray	560.640	17590
4	LLNL (USA)	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom	IBM	1.572.864	17173
5	RIKEN (Japan)	K computer - SPARC64 VIIIfx 2.0GHz, Tofu interconnect	Fujitsu	705.024	10510

RMax in TFLOPS

Spanish Network of Supercomputing (RES)

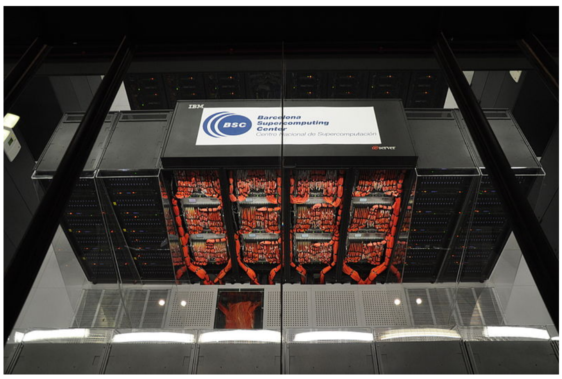
It currently comprises the following centres:

- Barcelona Supercomputing Centre (BSC)
- U. Politécnica Madrid (CeSViMa)
- Universidad de Cantabria (IFCA)
- I. Astrofísica de Canarias
- Universidad de Málaga
- Universitat de València
- Universidad de Zaragoza (BIFI)
- U. Las Palmas de Gran Canaria

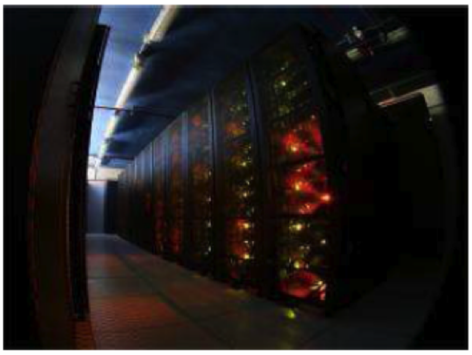
Spanish Network of Supercomputing (RES)

Mare Nostrum III has currently:

- 48448 cores (Intel Xeon E5-2670 8C processors at 2.6 GHz)
- Sustained power: 925 TFLOPS
- 3028 nodes linked using an Infiniband FDR10 network
- 94.6 TB main memory, 1.9 PB disk

Spanish Network of Supercomputing (RES)



Magerit

Altamira

CaesarAugusta

LaPalma

Picasso

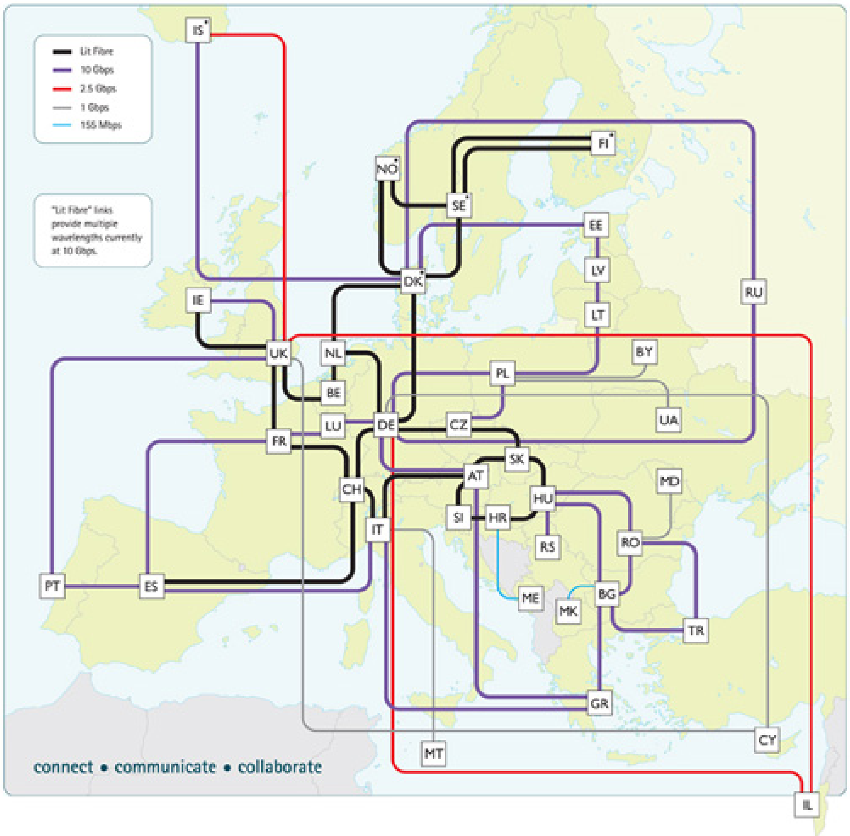
Tirant

Network infrastructures

RedIRIS: Links university and research institutions

Network infrastructures

GÉANT: Links different european research networks



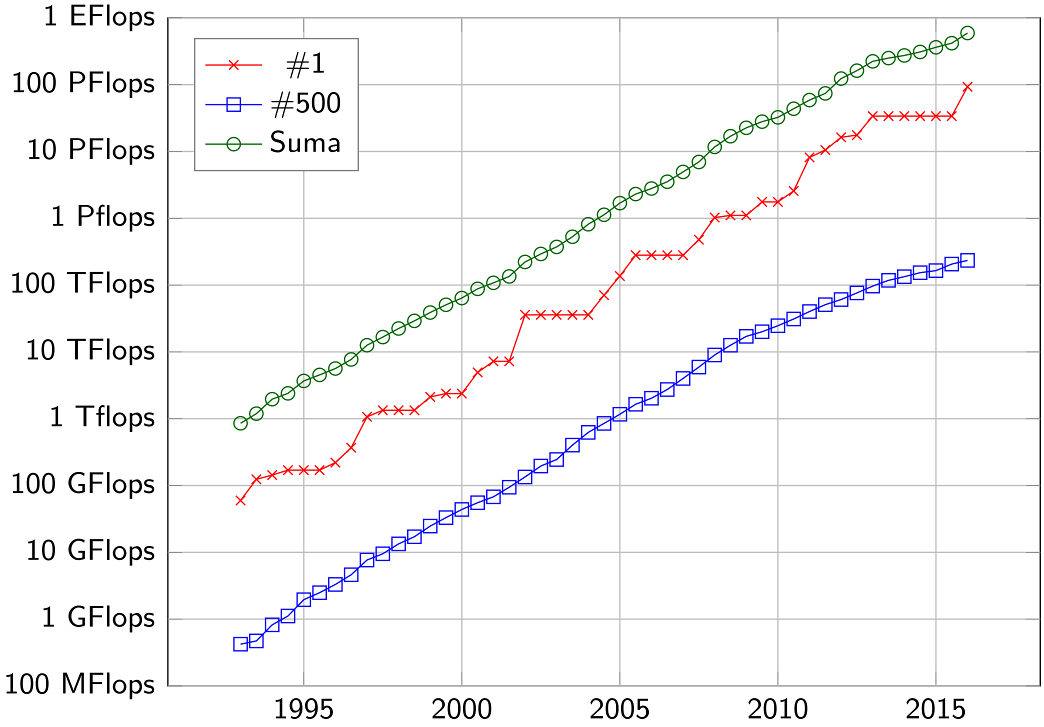
Other Centres

Supercomputing Regional Centres:

- CESGA: Centro de Supercomputación de Galicia
  - FinisTerrae II, 306 nodes, 7344 Haswell cores, 213 TFLOP
- FCSCL: Fundación Centro de Supercomputación de Castilla y León
  - Caléndula, 304 nodes blade bi Xeon quad, 33 TFLOPS
- CénitS: Centro Extremeño de iNvestigación, Innovación Tecnológica y Supercomputación
  - Lusitania, 128 Itanium dual-core processors, 2 TB shared memory
- CSUC: Consorci de Serveis Universitaris de Catalunya

UPV: (Rigel) 72 nodes, with 2 Intel Xeon E5-2450 and 64 GB each; 10 GbE network; performance 20.6 TFLOPS

Top500: Evolution of the Performance



Source: `http://www.top500.org`

Section 2

Parallel Computing Models

- Parallel Computing Architectures
- Parallel Computing Models
- Parallel Programming Methodology

Flynn Taxonomy

SISD	SIMD
Single Instruction, Single Data	Single Instruction, Multiple Data

MISD	MIMD
Multiple Instruction, Single Data	Multiple Instruction, Multiple Data

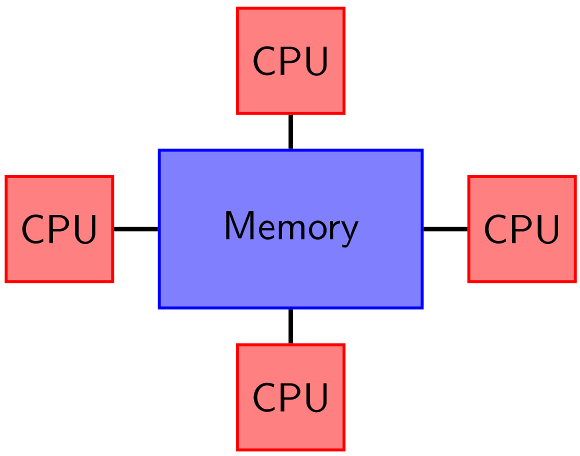
SISD Sequential Computer

SIMD Vector Computers (NEC, Fujitsu, Cray), processors with multimedia extension(SSE3, Altivec)

MIMD Multiprocessors, clusters, multi-core

Shared Memory Architectures

Single address space for all the processors



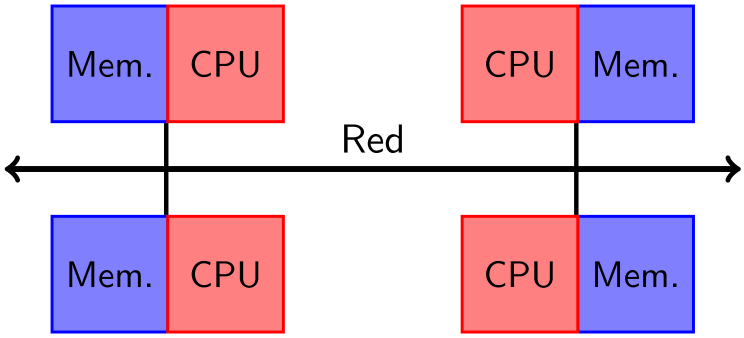
Tipos:

- UMA: Uniform Memory Access
- NUMA: Non-Uniform Memory Access
- cc-NUMA: Cache Coherent NUMA

Advantages: easy programming; Disadvantages: scalability, price

Distributed Memory Architectures

It requires a communication network to let processors access to data outside of the local space



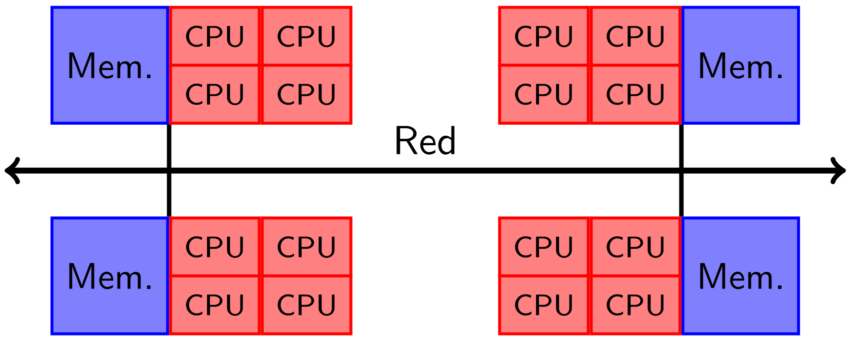
Features

- There is no global memory concept
- Independent processors (no coherence)
- Data exchange explicitly programmed

Advantages: scalability, cost; Drawbacks: programming effort

Hybrid Architectures: *Distributed-Shared Memory*

Combination of the two models



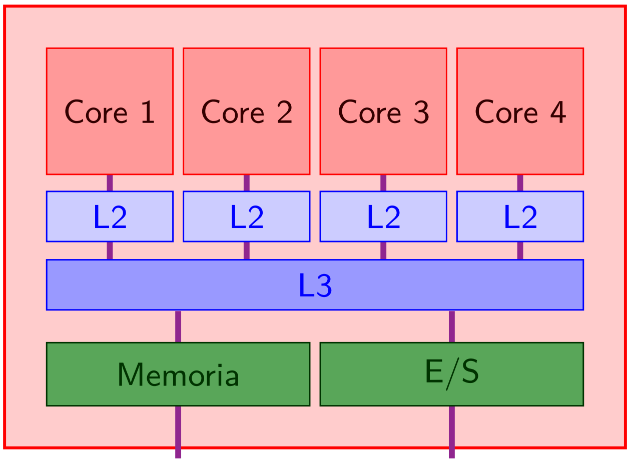
Features

- Each node is a multiprocessor (e.g. cc-NUMA)
- Communication to move data from one node to other

Currently supercomputers follow often this model, (multicores and GPUs are becoming popular).

Multi-Core Processors

Multiple cores: Current tendency in the design of processors



Features

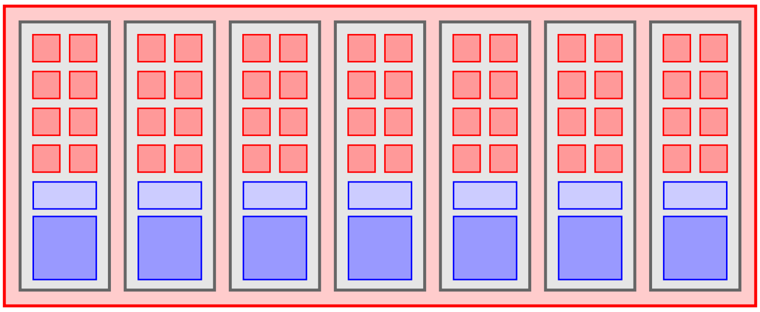
- Simmetric (or not) multiprocessing in a single chip
- Several cache levels in the same chip

Advantages: Cost; Drawbacks: Low Efficiency (bandwidth)

Many-Core Processors

Massively parallel, with a large number of single cores

- Graphic Processor Units (GPU)



Features

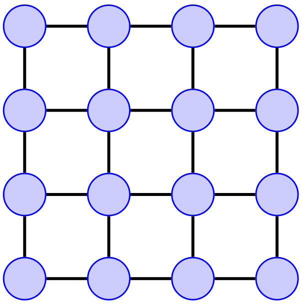
- Many cores (e.g., 448)
- Light threads, very quick context change

Advantages: cost, power; Drawbacks: complex programming

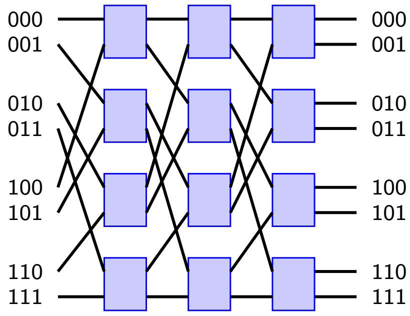
Interconnection Networks

In any case, a network is needed

Static topologies: ring, open 2D grid, 2D grid, 2D or 3D torus, hypercube



Dynamic networks: single-stage, multi-stage, *crossbar*



- Uniform latency networks: low scalability (cost)
- Non-uniform latency networks: cheaper and latency depending on the distance

Clusters

A *cluster* is simply a set of PCs or workstations connected in a network to execute parallel computing algorithms.

- In 1994 a distribution using PCs and Linux get popular (Beowulf)

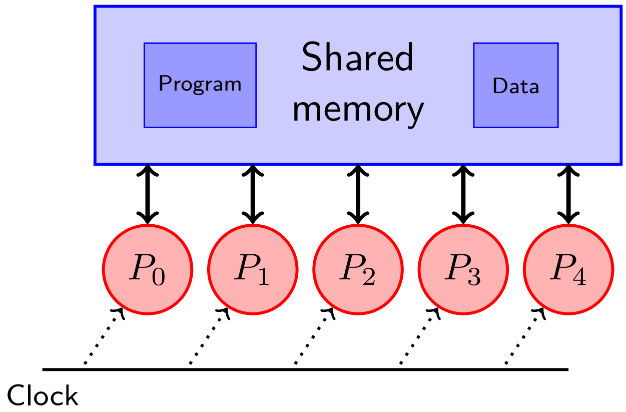
Currently, a commercial cluster typically comprises:

- Rack structure
- A set of nodes
  - Typically, two multi-core processor and 1 disk
  - Optionally, 1 GPU
  - Compact format: 1U, 2U, *blades*
- Network infrastructure
  - Typically, two networks are used: ethernet and low-latency network
  - Low latency networks: Infiniband, Myrinet, Quadrics, ...
  - Components: adapter, *switches*, cables
- Node *front-end*

Shared-memory model

PRAM: Parallel Random Access Machine

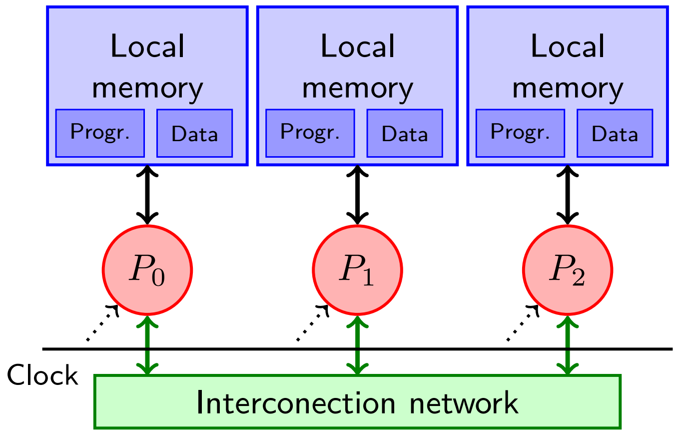
p processors: $P_0, P_1, \ldots, P_{p-1}$



- All processes can access to any memory position (constant time)
- All processes execute the same code (eventually some parts may depend on the process index)
- ~~Synchonized execution, one instruction per cycle~~
- Information exchanged through variables

Distributed memory model

p Processors, each one with its own address space (local memory)



- All processes share the same code (with parts depending on the process index)
- ~~Synchonized execution, one instruction per cycle~~
- Information is exchanged explicitly using messages
- Local processing and communication instructions (one sends and another receives through an "interconnection network")

Parallel Programming Methodologies

Different methodologies can be applied:

- Automatic Parallelisation (parallel compilers)
- Semi-automatic Parallelisation (compiler directives)
- New programming languages
- Extensions of conventional languages
- Software libraries (API)

Examples

- OpenMP is based in directives and/or APIs
- CUDA C is an extension of C
- MPI is based on software libraries

Single/Multiple Program Multiple Data

High-level programming models, suitable for any of the previous models

SPMD: The same program is executed in all the tasks

a.out	a.out	a.out	$\cdots$	a.out
tarea 1	tarea 2	tarea 3	$\cdots$	tarea n

MPMD: each task may have a different program

a.out	b.out	c.out	$\cdots$	b.out
tarea 1	tarea 2	tarea 3	$\cdots$	tarea n

SPMD programs are easy to maintain, although they may require conditional instructions