23rd Summer School on PARALLEL COMPUTING

Introduction to High Performance Computing

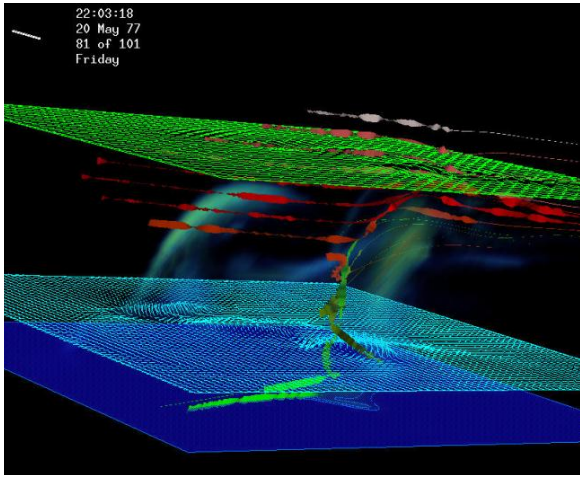
Giovanni Erbacci - g.erbacci@cineca.it
Supercomputing Applications & Innovation Department - CINECA

CINECA

Outline

- Computational Sciences
- High Performance Computing
- von Neumann Model
- Pipelining
- Memory
- Flynn Taxonomy
- Vector Architectures
- Parallel Architectures
- Accelerators

Computational Sciences

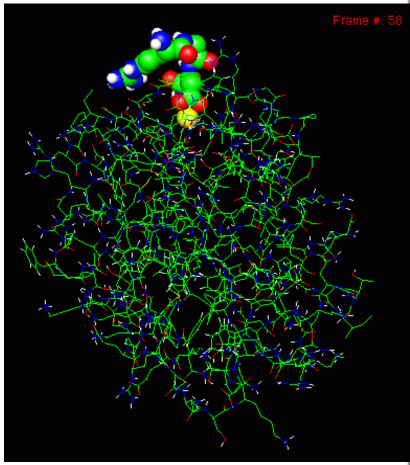


Identify the scientific disciplines that use mathematical models and computing systems to analyze and solve scientific problems.

Computational methods allow us to study complex phenomena, giving a powerful impetus to scientific research.

The use of computers to study physical systems allows to manage phenomena

- **very large**
  (meteo-climatology, cosmology, data mining, oil reservoir)
- **very small**
  (drug design, silicon chip design, structural biology)
- **very complex**
  (*f*fundamental physics, *fluid dynamics, turbolence)*
- **too dangerous or expensive**
  *(fault simulation, **nuclear** tests, crash analysis)*



Computational Sciences / 1

Computational science (with theory and experimentation), is the "third pillar" of scientific inquiry, enabling researchers to build and test models of complex phenomena

The Nobel Prize in Chemistry 1998

"for his development of the density-functional theory"

Walter Kohn

"for his development of computational methods in quantum chemistry"

John A. Pople

Owen Willans Richardson, *Years '20*

Nobel Prize in Physics 1928

for his work on the thermionic phenomenon and especially for the discovery of the law named after him"

John Von Neumann, *Years '40*

Kenneth. Wilson, *Years '80*

Nobel Prize in Physics 1982

"for his theory for critical phenomena in connection with phase transitions"

Size of computational applications

Computational Dimension:

number of operations needed to solve the problem,

in general is a function of the size of the involved data structures (n, n^2 , n^3 , $n \log n$, etc.)

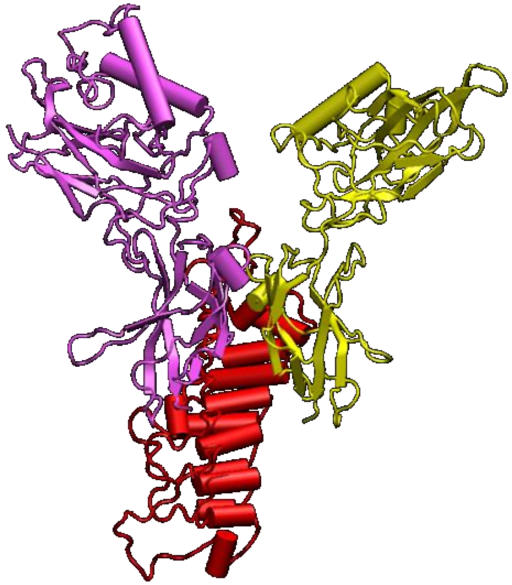
flop - Floating point operations

indicates an arithmetic floating point operation.

flop/s - Floating points operations per second

is a unit to measure the speed of a computer.

computational problems today: $10^{15} – 10^{22}$ flop



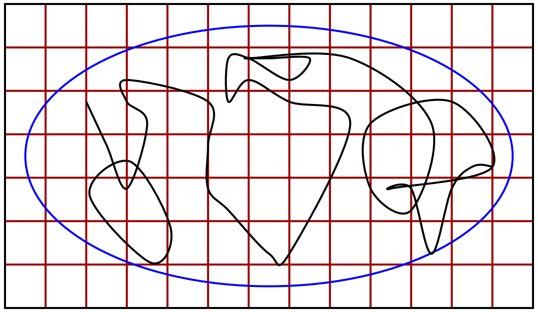
One year has about 3×10^{7} seconds!

Most powerful computers today have reach a sustained performance in the order of Tflop/s (10^{12} flop/s).

Example: Weather Prediction

Forecasts on a global scale (…..too accurate and inefficient!!)

- **3D Grid to represent the Earth**
  - Earth's circumference: ≅ 40000 km
  - radius: ≅ 6370 km
  - Earth's surface: ≅ $4\pi r^2$ ≅ $5 \bullet 10^8$ km^2



- **6 variables:**
  - temperature
  - pressure
  - umidity
  - wind speed in the 3 Cartesian directions

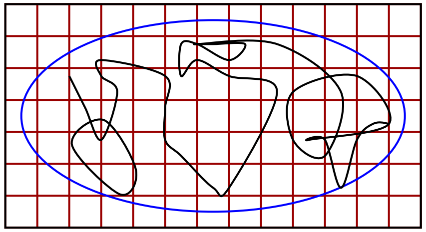
- **cells of 1 km on each side**

- 100 slices to see how the variables evolve on the different levels of the atmosphere
- a 30 seconds time step is required for the simulation with such resolution

- Each cell requires about 1000 operations per time step (Navier-Stokes turbulence and various phenomena)

On a global scale is currently a precision quite impossible. unimaginable!
On a local scale normally the cells are 10-15 km on each side

Example: Weather Prediction / 1



Grid: $5 \bullet 10^8 \bullet 100 = 5 \bullet 10^{10}$ cells

- each cell is represented with 8 Byte
- Memory space:

  (6 var)•(8 Byte)•($5 \bullet 10^{10}$ cells) $\cong 2 \bullet 10^{12}$ Byte = 2TB

A 24 hours forecast needs:

- $24 \bullet 60 \bullet 2 \cong 3 \bullet 10^3$ time-step
- ($5 \bullet 10^{10}$ cells) • (10^3 oper.) • ($3 \bullet 10^3$ time-steps) = $1.5 \bullet 10^{17}$ operations !

A computer with a power of 1Tflop/s will take $1.5 \bullet 10^5$ sec.

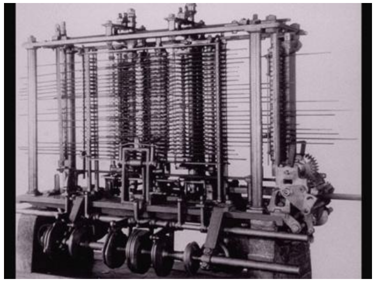
- 24 hours forecast will need 2days to run

  ..... but we shall obtain a very accurate forecast

Supercomputer

supercomputer are defined as the more powerful computers available in a given period of time.

Powerful is meant in terms of execution speed, memory capacity and accuracy of the machine.



Supercomputer:"*new statistical machines with the mental power of 100 skilled mathematicians in solving even highly complex algebraic problems*"..

New York World, march 1920

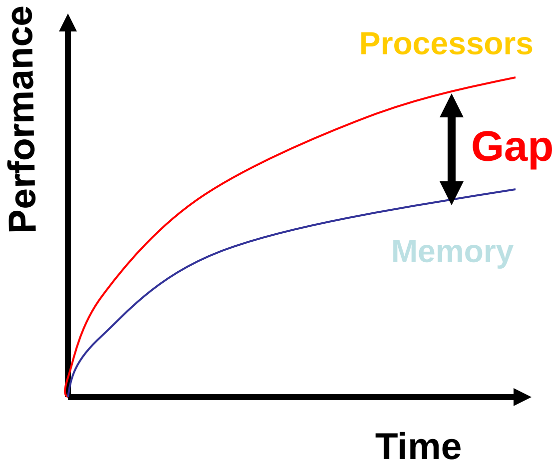
to describe the machines invented by Mendenhall and Warren, used at Columbia University's Statistical Bureau.

More powerful computers

Increase the speed of microprocessors:

Physical limitations to size and speed of a single chip:

- speed of light,
- size of atoms,
- dissipation of heat

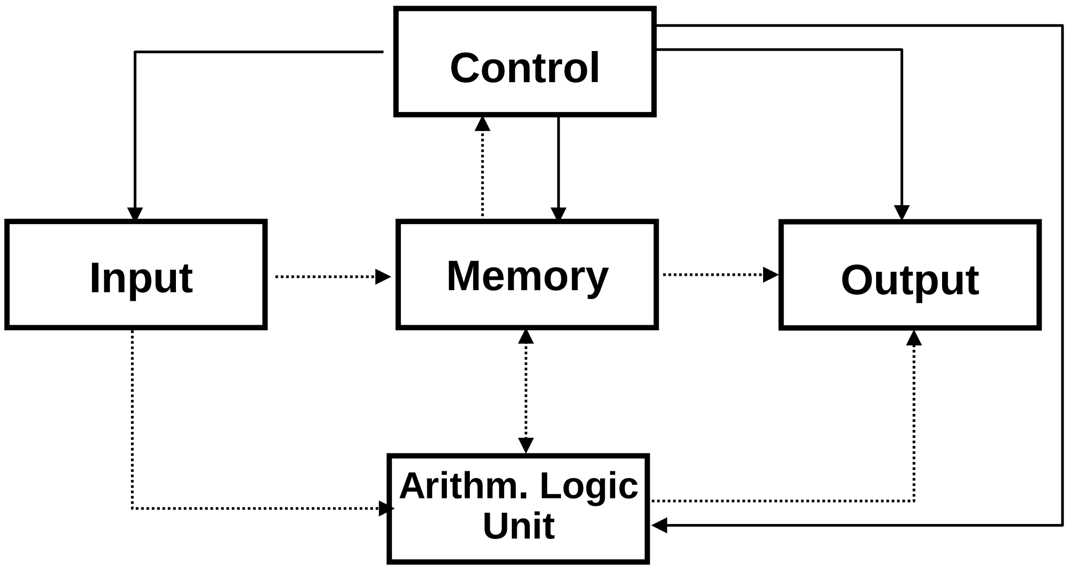


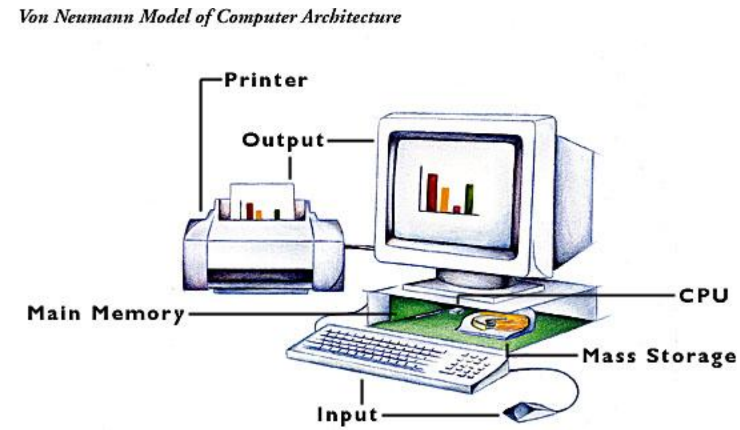
Solution:
- increase the degree of parallelism
– use many CPUs cooperatively on the same problem

"Serial computing is dead, and the parallel computing revolution has begun: Are you part of the solution, or part of the problem?"
Dave Patterson,
UC Berkeley, Usenix conference June 2008

von Neumann Model

Conventional Computer





.... Data

——— Control

Instructionsi are processed sequentially

1 A single instruction is loaded from memory (**fetch**) and decoded
2 Compute the addresses of operands
3 Fetch the operands from memory;
4 Execute the instruction ;
5 Write the result in memory (**store**).

Speed of Processors: Clock Cycle and Frequency

The *clock cycle* τ is defined as the time between two adjacent pulses of oscillator that sets the time of the processor.

The number of these pulses per second is known as clock speed or clock frequency, generally measured in GHz (gigahertz, or billions of pulses per second).

The clock cycle controls the synchronization of operations in a computer: All the operations inside the processor last a multiple of τ.

Processor	τ (ns)	freq (MHz)
CDC 6600	100	10
Cyber 76	27.5	36,3
IBM ES 9000	9	111
Cray Y-MP C90	4.1	244
Intel i860	20	50
PC Pentium	< 0.5	> 2 GHz
Power PC	1.17	850
IBM Power 5	0.52	1.9 GHz
IBM Power 6	0.21	4.7 GHz

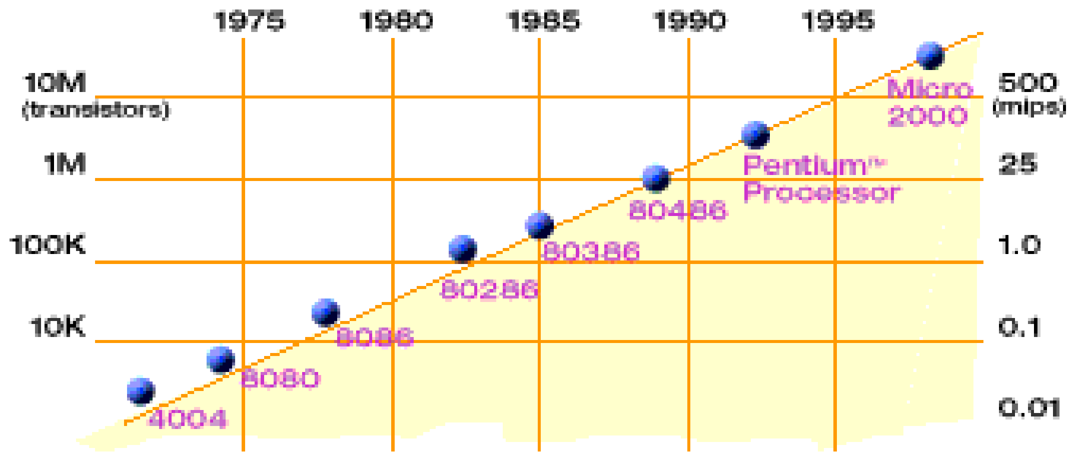
Increasing the clock frequency:

The **speed of light** sets an upper limit to the speed with which electronic components can operate .

Propagation velocity of a signal in a vacuum: **300.000 Km/s = 30 cm/ns**

Heat dissipation problems inside the processor.

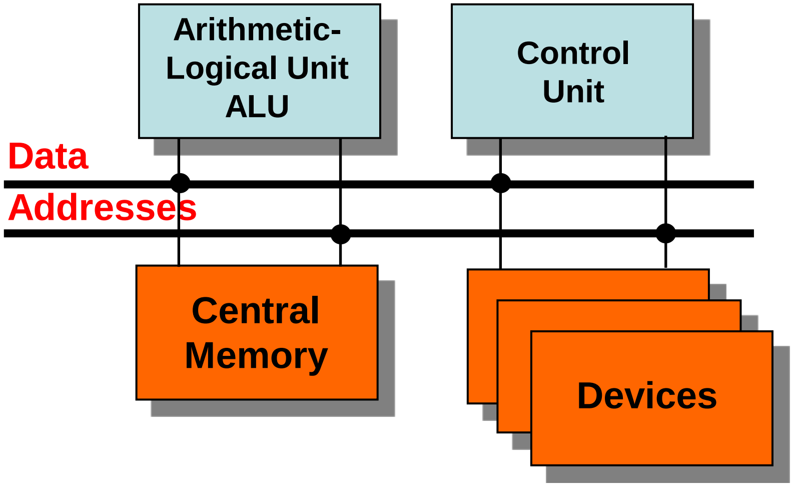
Moore's Law



Empirical law which states that the complexity of devices (number of transistors per
square inch in microprocessors) doubles every 18 months..
Gordon Moore, INTEL co-founder, 1965

It is estimated that Moore's Law still applies in the near future but applied to the number of cores per processor

Other factors that affect Performance



In addition to processor power, other factors affect the performance of computers:

- Size of memory
- Bandwidth between processor and memory
- Bandwidth toward the I/O system
- Size and bandwidth of the cache
- Latency between processor, memory, and I/O system

Memory hierarchies

Time to run code = clock cycles running code + clock cycles waiting for memory

Memory access time: the *time* required by the processor to *access* data or to write data from / to *memory*

The hierarchy exists because

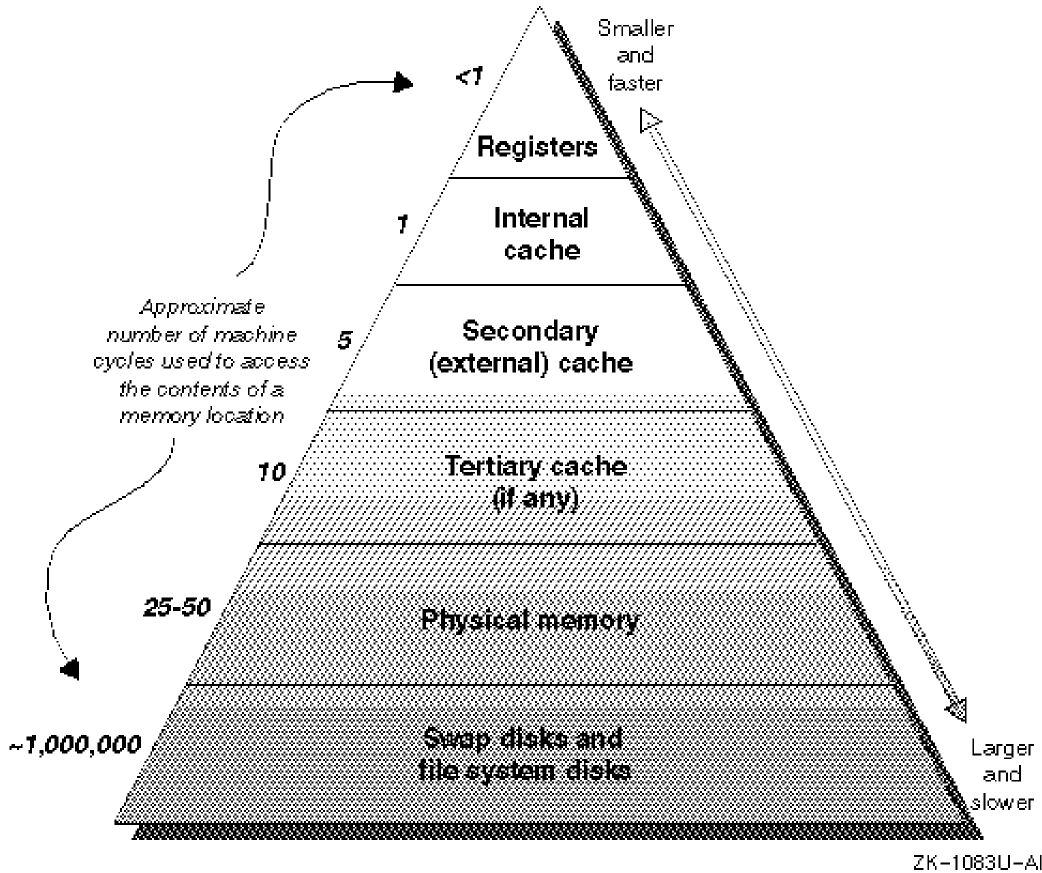
- fast memory is expensive and so is small
- slow memory is cheap and can be big

Latency

- how long do I have to wait for the data?
- (cannot do anything while waiting)

Throughput

- how many Data / second?
- not important when you're waiting



Total time = latency + (amount of data / throughput)

Cache Memory

Memory unit with a small size used as a buffer for data to be exchanged between the main memory and the processor.

- Reduces the time spent by the processor waiting for data from the main memory.
- Cache efficiency depends on the locality of the data references:

Principle of locality

Temporal locality refers to the reuse of specific data within relatively small time durations.

Spatial locality refers to the use of data elements within relatively close storage locations.

A special case of spatial locality, occurs when data elements are arranged and accessed linearly (sequentially), e.g., traversing the elements in a one-dimensional array.

Cache can contain Data, Instructions or both

Cache Memory / 1

The code performance improve when the instructions that compose a heavy computational kernel (eg. a loop) fit into the cache

The same applies to the data, but in this case the work of optimization involves also the programmer and not just the system software.

DEC Alpha 21164 (500 MHz):

Memory access time

Registers	2 ns
L1 On-chip	4 ns
L2 On-Chip	5 ns
L3 Off-Chip	30 ns
Memory	220 ns

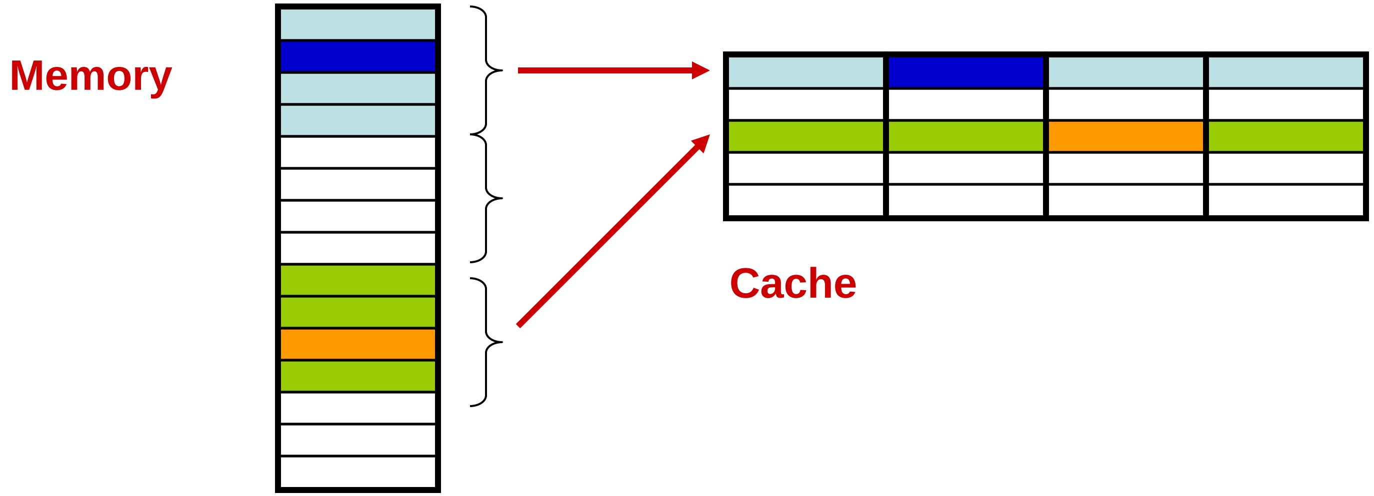
IBM SP Power 6 (4.7 GHz):

Memory access time (in clock cycles)

Registers	
L1: 2 x 64KB	< 5
L2: 2 x 4MB	22 cc
L3: 32 MB	160 cc
Memory 128 GB	400 cc

Cache organisation

The cache is divided into slots of the same size (lines)
Each line contains k consecutive memory locations (ie 4 words).
When a data is required from memory, (if not already in the cache) the system loads from memory, the entire cache line that contains the data, overwriting the previous contents of the line.

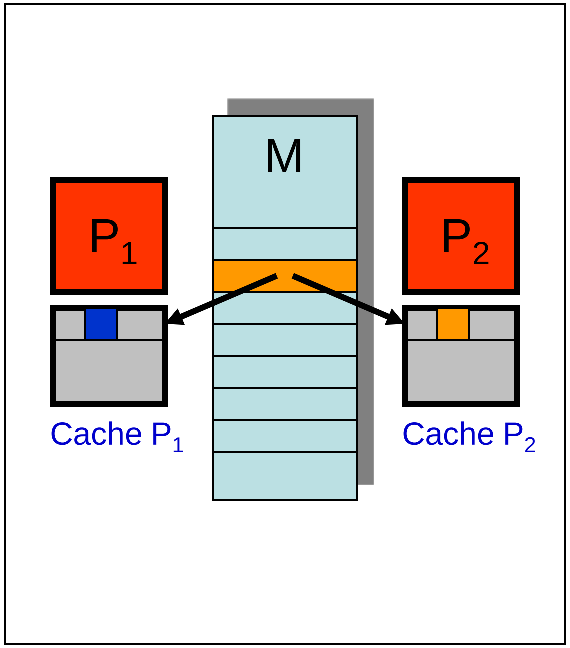


Cache organisation / 1

When the CPU writes out a value the data must be updated in main memory

- Cache *write-back*: data written in the cache will stay there until the cache line is not required to store other data. When the cache line must be replaced, the data is written in memory.
- Cache *write-through*: data are written in cache and immediately to main memory.

  In multi-processors systems, cache coherency must be managed: we need to access updated data in mail memory

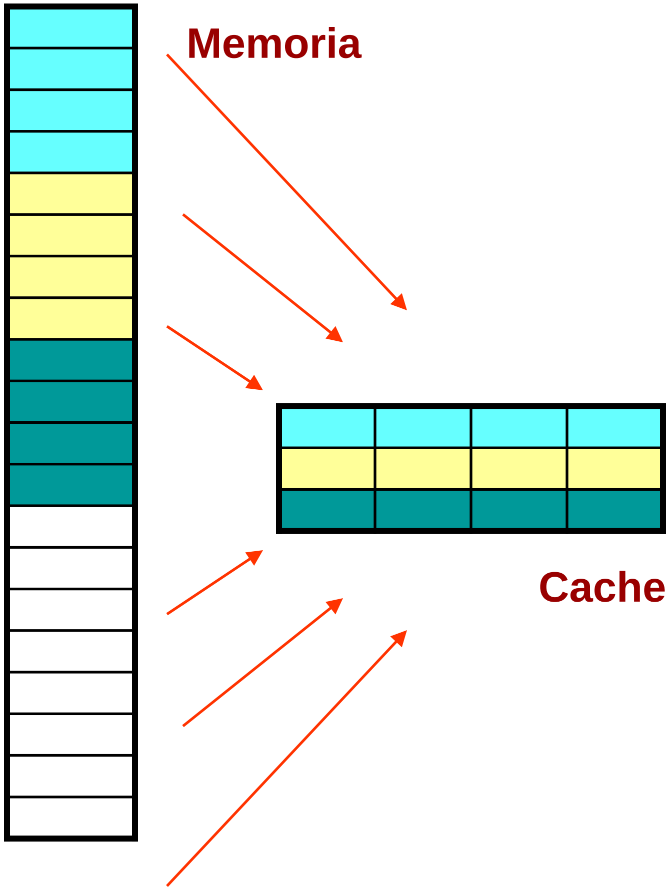


Mapping

Time by time the cache can contain only a sub-set of the data in memory.
A set of memory locations must be associated to a cache line (mapping).

Based on the mapping cache can be organized in one of the following ways:

- ***Direct mapped***
- ***Fully associative***
- ***Set associative***



The way in which memory locations are mapped in the cache lines can affect the performance of the programs: two memory locations heavily used can be mapped or not on the same line.

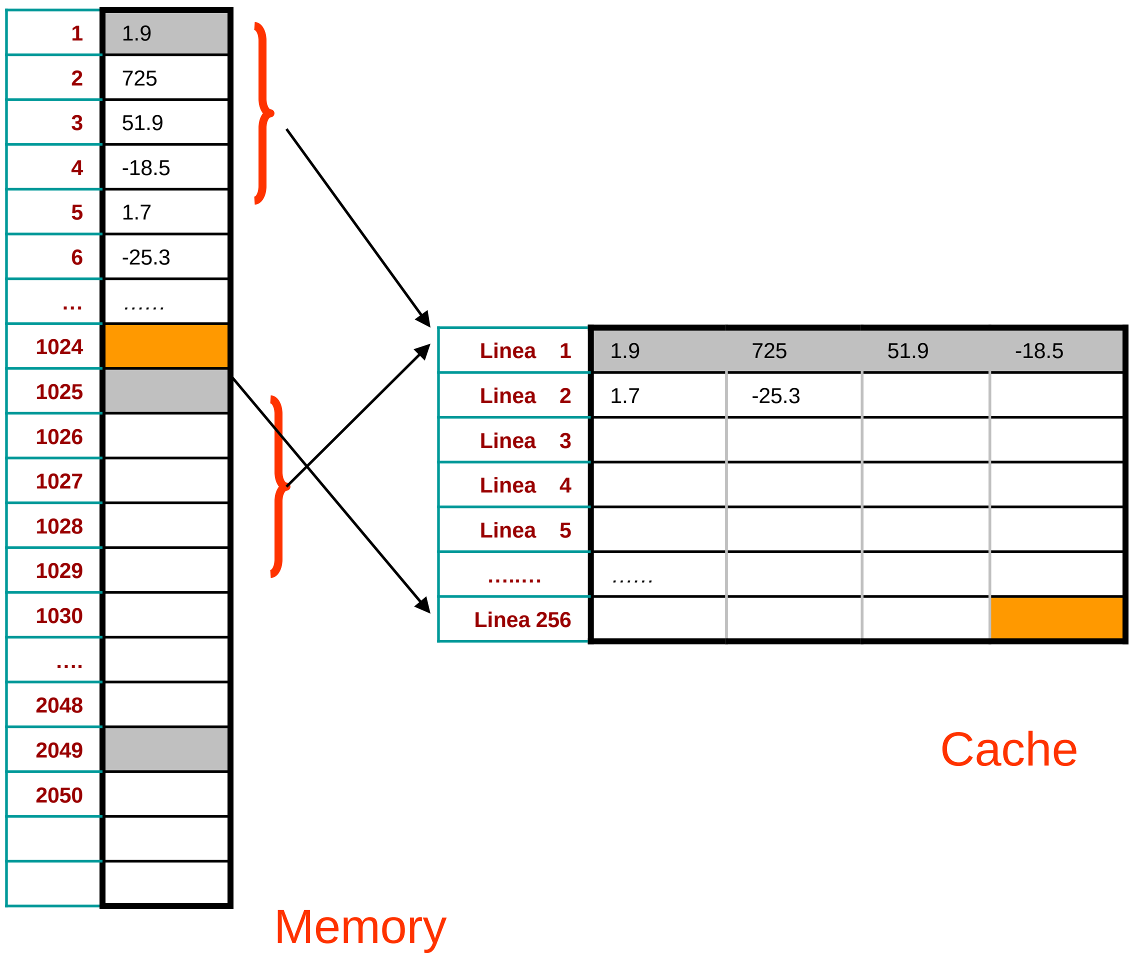
Cache Direct Mapping

With this scheme the first memory location (word 1) is mapped in line 1, as well as the location *d+1*, *2d+1*, *3d+1* etc. where

d = N° $_{\text{lines}}$ * N° $_{\text{words per line}}$

If different memory references in turn point to the same cache line (eg, word 1, word 1025, word 2049), each reference causes a **cache miss** and the line just inserted must be replaced.
A lot of extra work (**overhead**) is generated.

This phenomenon is called **thrashing**.



Cache Direct Mapping / 1

Address

Indicates the word within the line

Indicates the line

Content

Memory

				Content
0	0000	000	00	1.9
1	0000	000	01	725
2	0000	000	10	51.9
3	0000	000	11	-18.5
4	0000	001	00	1.7
5	0000	001	01	-25.3
..	….	…	. .	……
10	0000	010	10	
11	0000	010	11	
12	0000	011	00	
..	….	…	. .	……
29	0000	111	01	
30	0000	111	10	
31	0000	111	11	
32	0001	000	00	
33	0001	000	01	
..	….	…	. .	……
62	0001	111	10	
63	0001	111	11	
64	0010	000	00	
65	0010	000	01	

Linea 0	1.9	725	51.9	-18.5
Linea 1	1.7	-25.3		
Linea 2				
Linea 3				
Linea 4				
Linea 5				
Linea 6				
Linea 7				

Cache
- **8 lines**
- **4 word per line**

Cache Fully Associative

With this scheme, each memory location can be mapped to any cache line, regardless of the memory location.

The name comes from the type of memory used to build this type of cache (**associative memory**).

When the CPU needs a given data, this is required in all the cache lines simultaneously. If a line contains the data, this is sent to CPU, otherwise a cache miss occurs.

In general, the least recently used line will be overwritten with the new data. (**LRU policy**)

Fully associative caches are costly but provide a higher usage when compared to the direct mapped cache

Cache Set Associative

Practically it is a direct mapped cache replicated in multiple copies (banks) i.e. 2 or 4 separate banks of cache (*two-way, four-way set associative*).

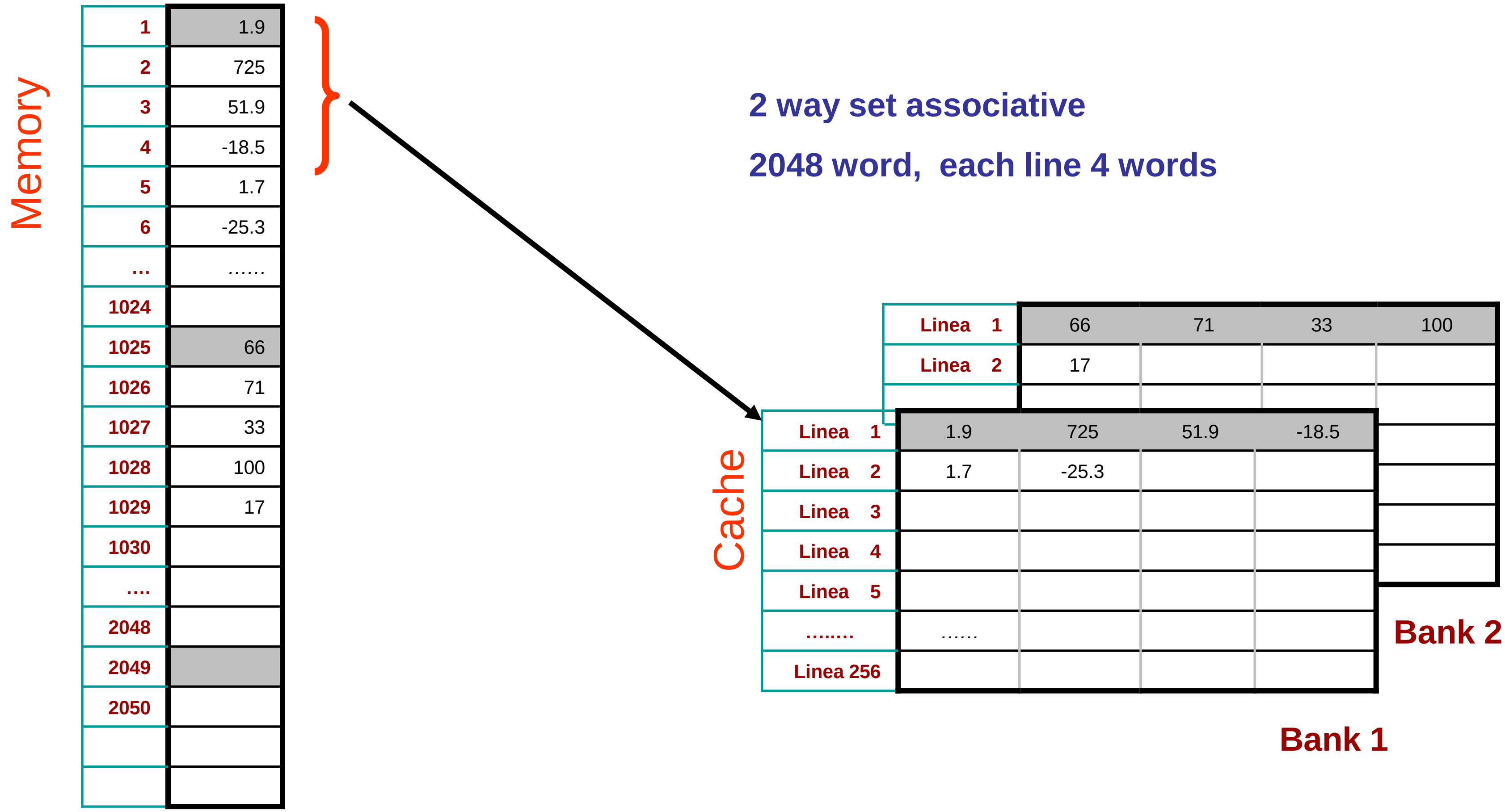
With this scheme, if a memory location is mapped in line k, a further reference to a memory location always mapped to the same line, will be allocated in line K of a different bank.

Set Associative cache is less susceptible to cache thrashing compared to a direct mapped cache with the same size.

Cache Set Associative

2 way set associative

2048 word, each line 4 words



Multiple Functional Units

Arithmetic logic unit (ALU) executes the operations.
ALU is designed as a set of independent functional units, each in charge of executing a different arithmetic or logical operation,

- Add
- Multiply
- Divide
- Integer Add
- Integer Multiply
- Branch ....

The functional units can operate in parallel. This aspect represents the first level of parallelism. It is a parallelism internal to the single CPU.

The compiler analyses the different instructions and determine which operations can be done in parallel, without changing the semantics of the program.

Pipelining

Is a technique where more instructions, belonging to a stream of sequential execution, overlap their execution

This technique improves the performance of the processor

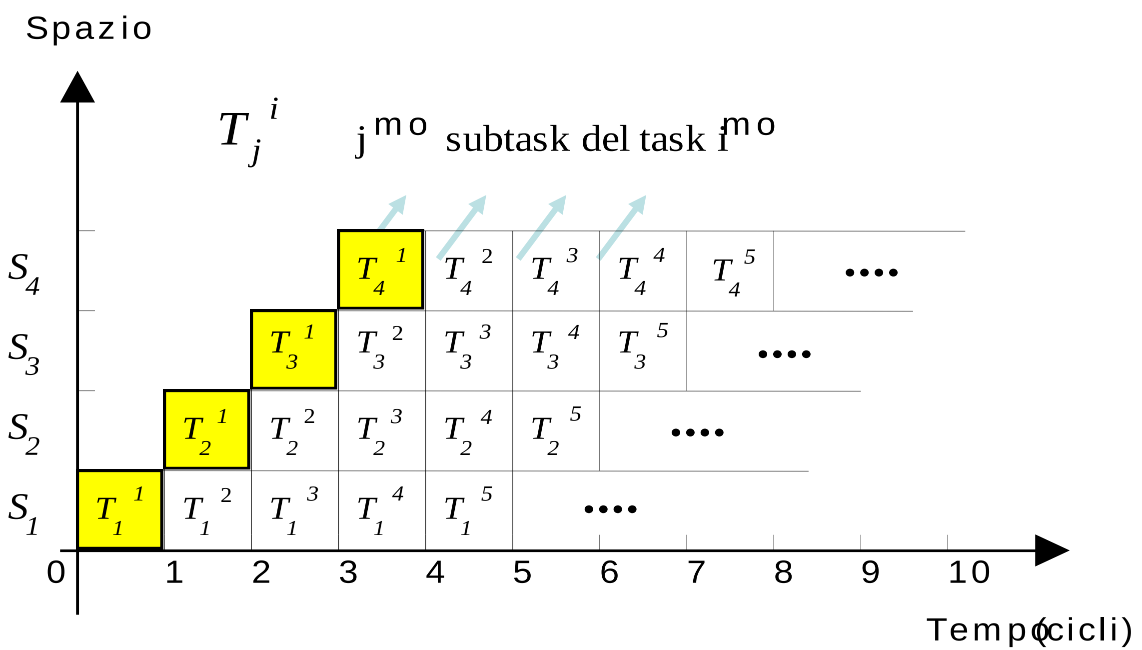
The concept of pipelining is similar to that of assembly line in a factory where in a flow line (pipe) of assembly stations the elements are assembled in a continuous flow.

All the assembly stations must operate at the same processing speed, otherwise the station slower becomes the bottleneck of the entire pipe.

Pipelining / 1

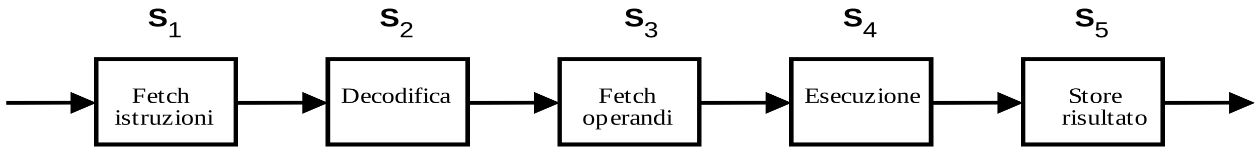
A task T is decomposed into a set of sub-tasks {T1, T2, ... Tk} linked by a dependency relationship: Task Tj can not start until all previous sub-tasks $\{T_i, \forall i < j\}$ are completed

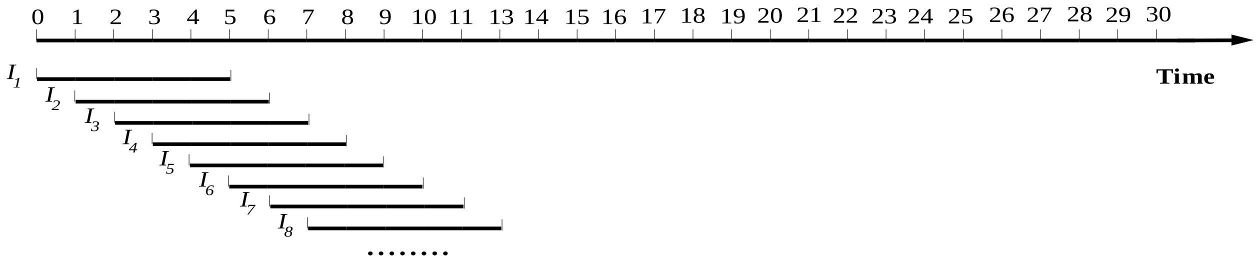
Overlapping operations in a four stages pipe

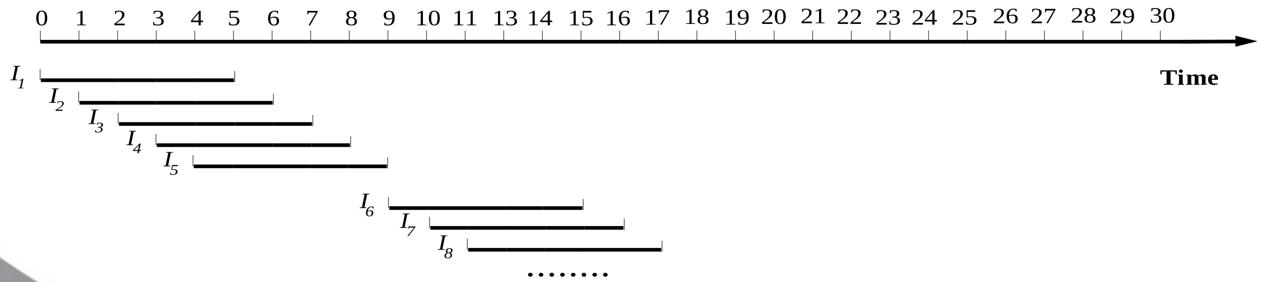


Space-time diagram

Instruction Pipelining







Vector Computers

Vector computer architectures adopt a set of vector instructions, In conjunction with the scalar instruction set.
The vector instructions operates on a set of vector registers each of which is able to contain more than one data element.

- Cray vector systems of the 80s and 90s
- Cray C90: 8 vector registers each with 128 elements at 64-bits
- Also the current microprocessors have a set of vector egisters and a set of vector instructions

The vector instructions implement a particular operation to be performed on a given set of operands called **vector**.

Functional units when executing vector instructions exploit pipelining to perform the same operation on all data operands stored on vector registers.

Data transfer to and from the memory is done through load and store operations operating on vector registers.

Example

```
do i = 1, N
  A(i) = B(i)+C(i)
end do
```

$$V0 \leftarrow V1 + V2$$

Functional Unit Add Floating Point

	Operands	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Result
CP 0	.... B(3) B(2) B(1) .... C(3) C(2) C(1)							
CP 1	.... B(4) B(3) B(2) .... C(4) C(3) C(2)	B(1) C(1)						
CP 2	.... B(5) B(4) B(3) .... C(5) C(4) C(3)	B(2) C(2)	B(1) C(1)					
CP 3	.... B(6) B(5) B(4) .... C(6) C(5) C(4)	B(3) C(3)	B(2) C(2)	B(1) C(1)				
CP 4	.... B(7) B(6) B(5) .... C(7) C(6) C(5)	B(4) C(4)	B(3) C(3)	B(2) C(2)	B(1) C(1)			
CP 5	.... B(8) B(7) B(6) .... C(8) C(7) C(6)	B(5) C(5)	B(4) C(4)	B(3) C(3)	B(2) C(2)	B(1) C(1)		
CP 6	.... B(9) B(8) B(7) .... C(9) C(8) C(7)	B(6) C(6)	B(5) C(5)	B(4) C(4)	B(3) C(3)	B(2) C(2)	B(1) C(1)	
CP 7	.... B(10) B(9) B(8) .... C(10) C(9) C(8)	B(7) C(7)	B(6) C(6)	B(5) C(5)	B(4) C(4)	B(3) C(3)	B(2) C(2)	**B(1) + C(1)**

CINECA

Flynn Taxonomy

M. J. Flynn

Very high speed computing systems, proceedings of the IEEE (1966).

Some computer organizations and their effectiveness, IEEE Transaction on Computers.(1972).

"The multiplicity is taken as the maximum possible number of simultaneous operations (instructions) or operands (data) being in the same phase of execution at the most constrained component of the organization"

A computer architecture is categorized by the multiplicity of hardware used to manipulate streams of instructions (sequence of instructions executed by the computer) and streams of data (sequence of data used to execute a stream of instructions).

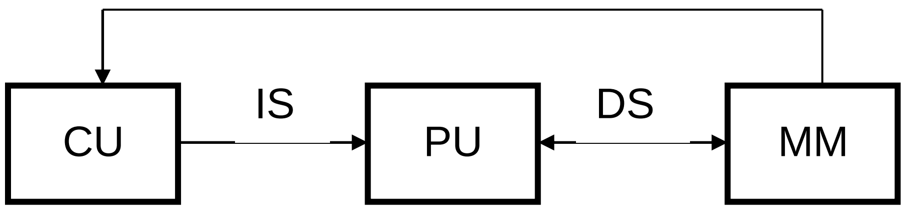
SI	Single Instruction stream	**SD**	Single Data stream
MI	Multilpe Instruction stream	**MD**	Multilpe Data stream

4 possible combinations : **SISD**, **SIMD**, **MISD**, **MIMD**

SISD Systems

Sequential systems

- Classical von Neumann architecture.
- Scalar mono-processor systems (single core).
- The execution of the instructions can be pipelined (Cyber 76).
- Each arithmetic instruction starts an arithmetic operation.



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

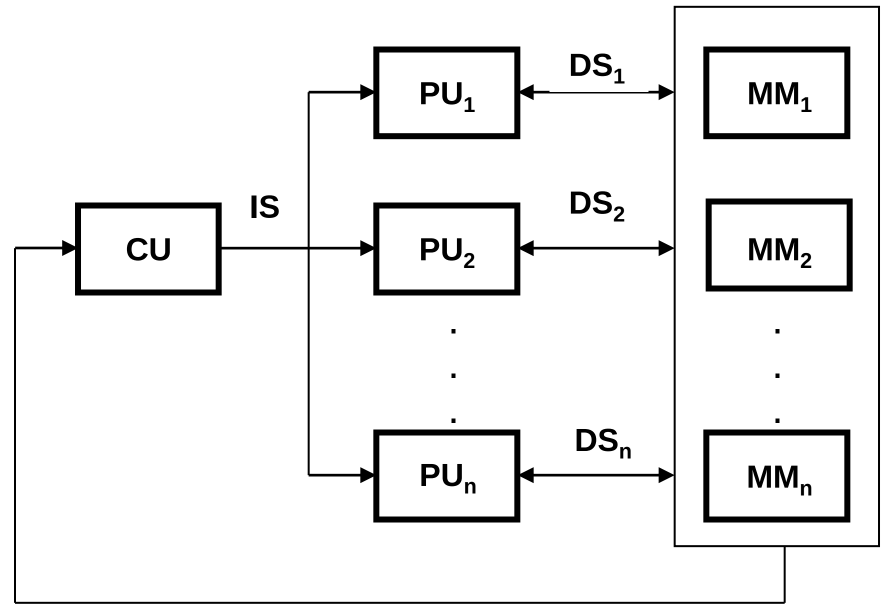
SIMD Systems

Synchronous parallelism

SIMD systems presents a single control unit

A single instruction operates simultaneously on multiple data.

Array processor and vector systems fall in this class



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

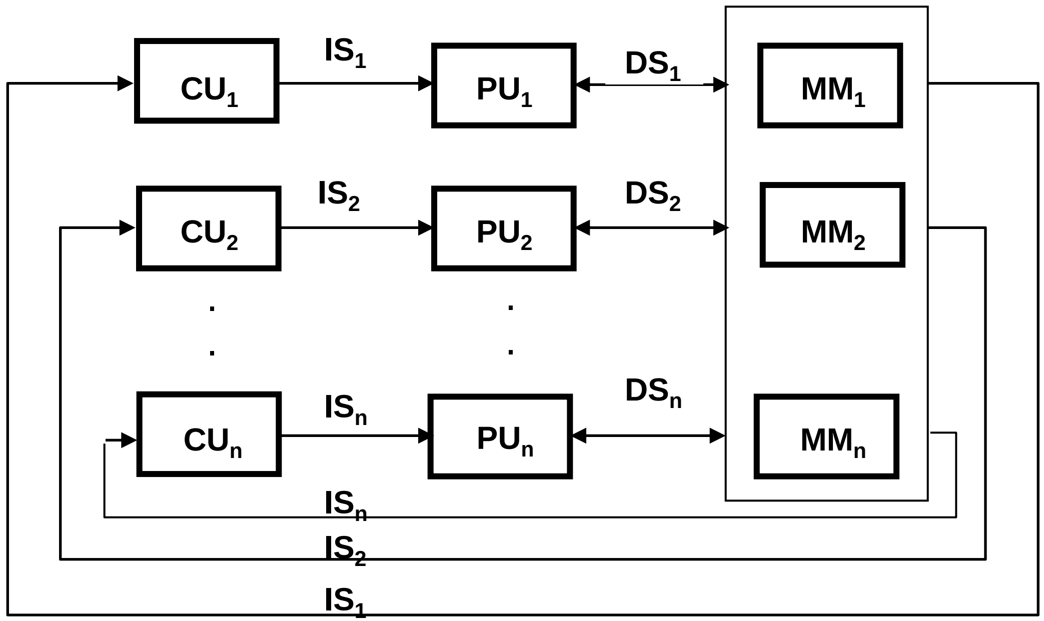
MIMD Systems

Asynchronous parallelism

Multiple processors execute different instructions operating on different data.

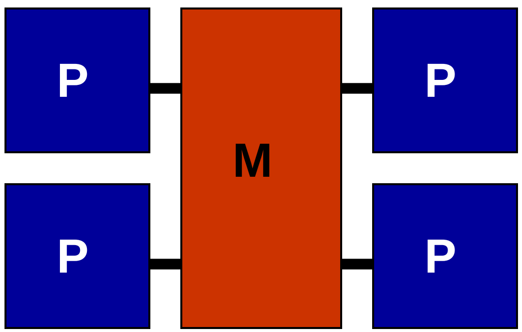
Represents the multiprocessor version of the SIMD class.

Wide class ranging from multi-core systems to large MPP systems.

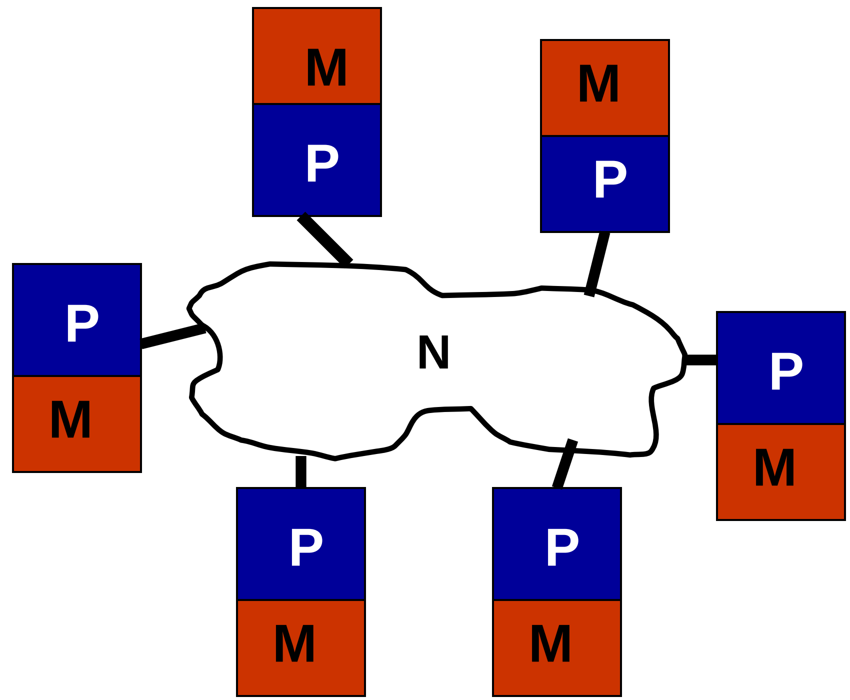


CU	**Control Unit**
PU	Processing Unit
MM	Memory Module
DS	Data stream
IS	Instruction Stream

Classification based on the Memory

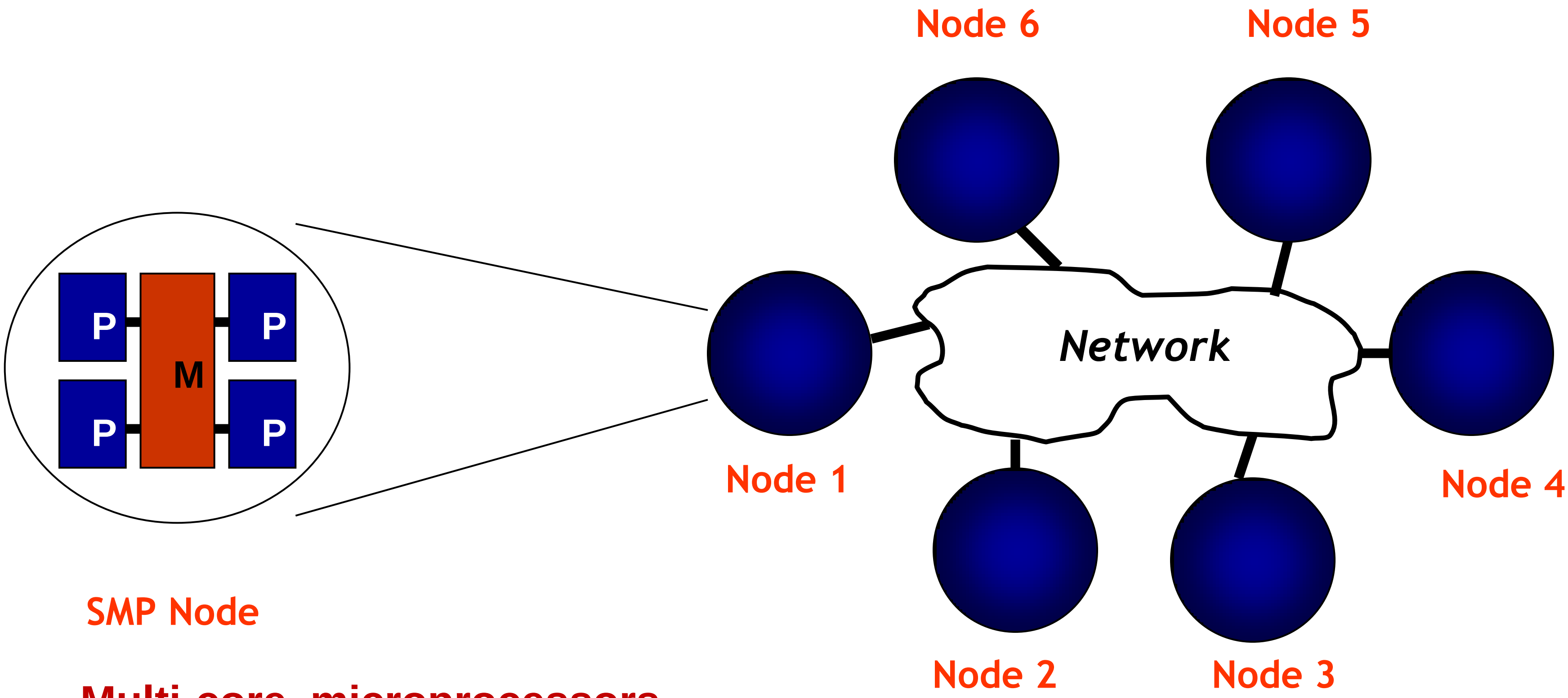


Shared Memory System



Distributed Memory System

The concept of Node



SMP Node

Multi-core microprocessors

Shared memory systems

All the processors (cores) share the main memory.
The memory can be addressed globally by all the processors of the system

Uniform Memory Access (UMA) model <=> SMP: Symmetric Multi Processors

The memory access is uniform: the processors present the same access time to reference any of the memory locations.

Processor-Memory interconnection via common bus, crossbar switch, or multistage networks.

Each processor can provide local caches,

Shared memory systems can not support a high number of processors

-

Distributed memory systems

The memory is physically distributed among the processors (local memory). Each processor can access directly only to his own local memory

- *NO-Remote Memory Access* **(NORMA) model**

Communication among different processors occurs via a specific communication protocol (message passing).
The messages are routed on the interconnection network
In general distributed memory systems can scale-up from a small number of processors $O(10^2)$ to huge numbers of processors $O(10^6)$ but the power of the single cores is not too high, to reduce global costs and power consumption but power is not too high, called processing nodes.
The performance of the system are influenced by:

- **Power of the node**
- **Topology of the interconnection network**

NUMA systems

Non Uniform Memory Access (NUMA) model

Memory is **physically distributed among all the processors** (each processor has its own local memory) but the collection of the different local memories forms a global address space accessible by all the processors

Hw support to ensure that each processor can access directly the memory of all the processors

The time each processor needs to access the memory is not uniform:

- access time is faster if the processor accesses its own local memory;
- when accessing the memory of the remote processors delay occurs, due to the interconnection network crossing.

Interconnection network

It is the set of links (cables) that define how the different processors of a parallel computer are connected between themselves and with the memory unit.

The time required to transfer the data depends on the type of interconnection.

The transfer time is called the communication time.

Features of an interconnection network:

- **Bandwidth**: identifies the amount of data that can be sent per unit time on the network. Bandwidth must be maximized.
  Latency: identifies the time required to route a message between two processors. Latency is defined also as the time needed to transfer a message of length zero. Latency must be minimized.

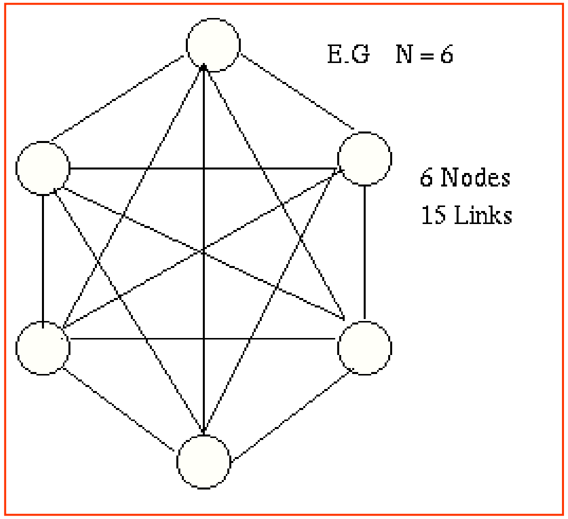
Other points to consider:

- **Cost**
- **Scalability**
- **Reliability**
- **Diameter**
- **Degree**

Connettivity

Complete Interconnection (ideal)

- each node can communicate **directly** with all other nodes (in parallel)
- with n nodes the bandwidth is proportional to n^2
- the cost increases proportionally to n^2



Indirect Interconnection (practical)

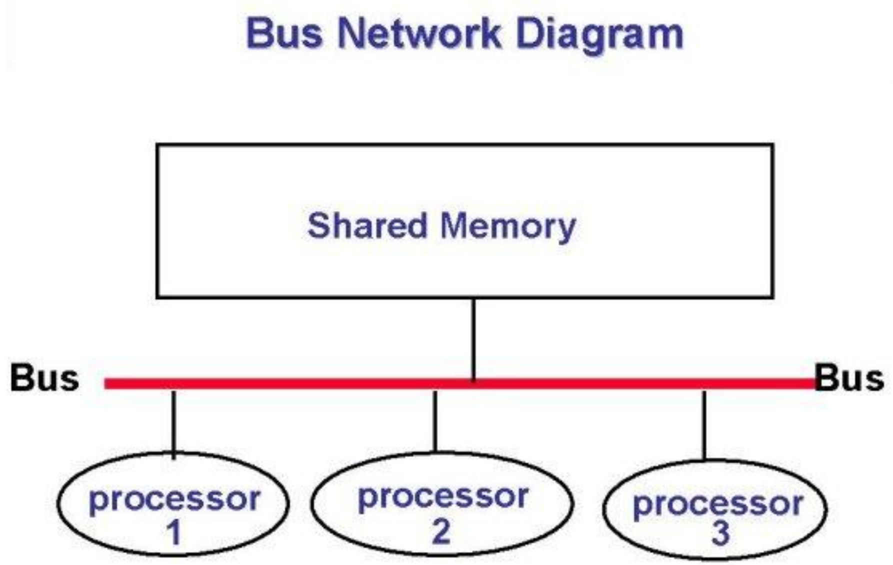
- Only a few nodes are connected directly. A direct or indirect path allows to reach all nodes.
- the worst case is represented by a single communication channel, shared by all the nodes (es. a cluster of workstations connected by a LAN).
- Need for intermediate solutions that **balance cost** and **performance** (mesh, tree, shuffle-exchange, omega, hypercube, ...).

Bus Network

The bus topology is constituted by a coaxial cable (bus) to which are connected all devices

The benefits of a network based on bus are the simplicity to Implement and the very low cost.

The negative aspects are the limited data transmission rate and consequently the non scalability in terms of performance.



Cross-bar Switch Network

A cross-bar switch is a network that operates according to a switching mechanism to connect devices to other devices (processors, memory).

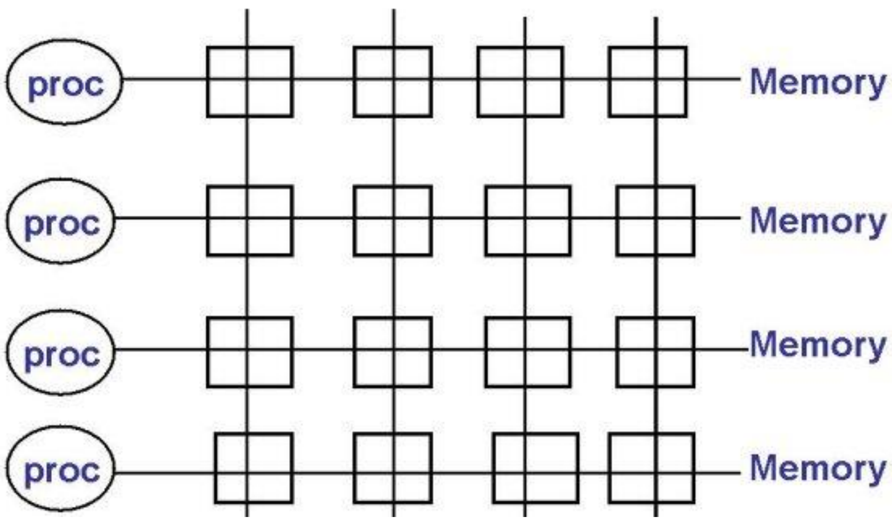
Mechanism similar to that adopted by the telephone systems.
Scales better than the bus but has higher costs

-The processors communicate via the switch boxes to access the memory modules.

- Multiple paths exists for communicate between a processor and a certain memory module.

- The switch determines the optimal path to be taken.

Cross-bar Switch Network Diagram



Mesh topology

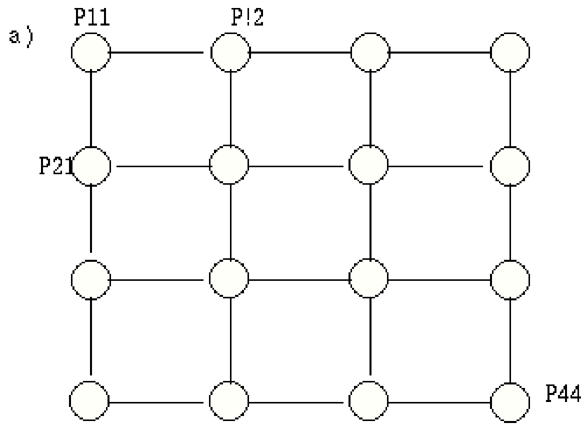
In a mesh network, the nodes are arranged as a **k-**dimensional lattce with width **w**, for a total of w^k nodes

- k=1 (a linear array)
- k=2 (2D array) es. ICL DAP, Intel Paragon,

Direct communication is allowed only between neighboring nodes.

The internal nodes communicate directly with other **2k** nodes.

EXAMPLE



2D mesh of width 4 with no wraparound connections on edge or corner nodes

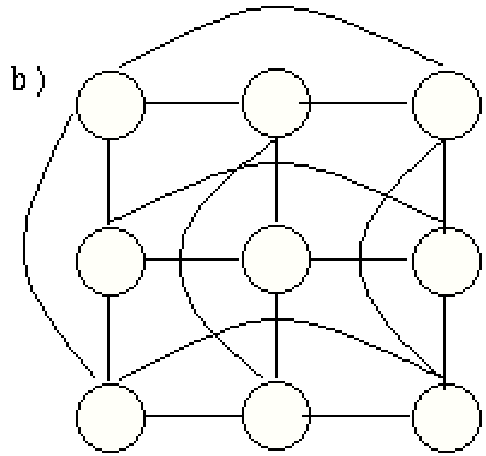
corner nodes have degree 2
edge nodes have degree 3

Toroidal Topology

Some variations of the mesh model have wrap-around type connections between the nodes to the edges of the mesh (torus topology).

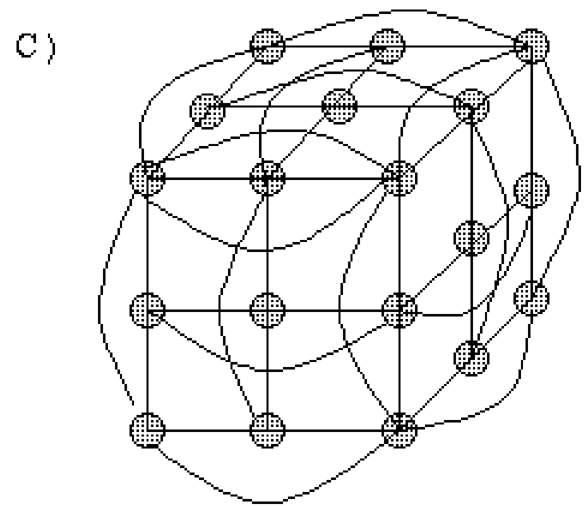
The Cray T3E adopts a 3D torus topology

IBM BG/Q has adopts a 5D torus topology



2D mesh of width 3 with wraparound connections between nodes on same row or column

Edge and corner nodes have degree 4



k = 3 w = 3

i.e. 3 ^ 3 = 27 nodes

with wraparound connections

all nodes have degree 6 (2k)

Hypercube Topology

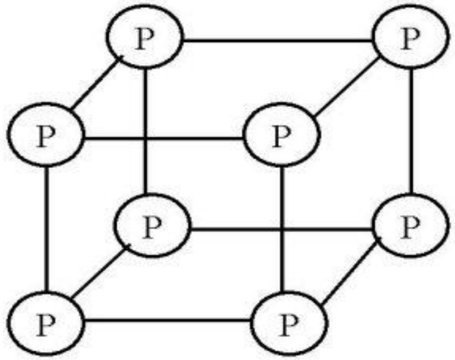
A hypercube topology (cube-connected) is formed by $n = 2^k$ nodes connected according to the vertexes of a cube with **k** dimensions.

Each node is directly connected to **k** other nodes.

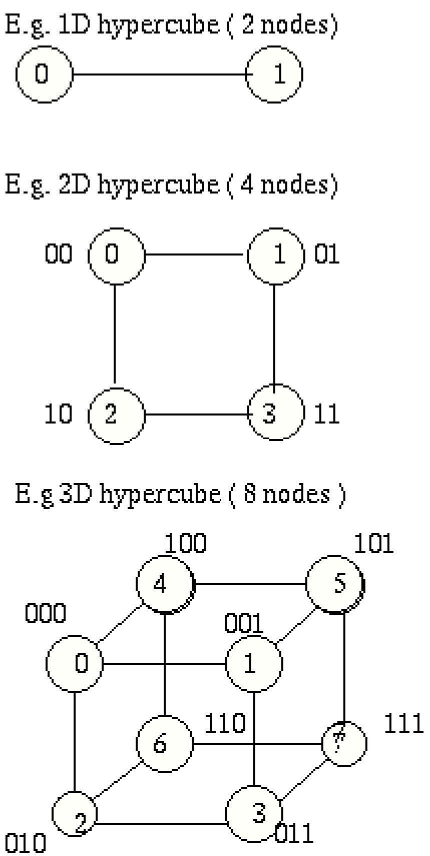
The degree of a hypercube topology is log n and also the diameter is log n.

Examples of computers with this type of network are CM2, Ncube-2, Intel iPSC860, SGI Origin.

Hypercube Network Diagram



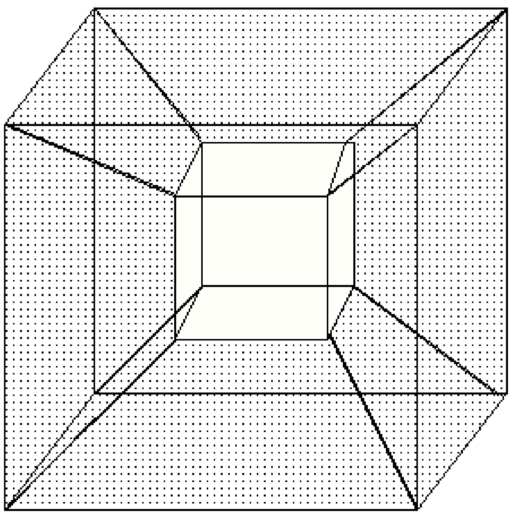
Hypercube Topology / 1



Recursive definition: *H(i) = H(H(i-1),H(i-1))*

Each node is connected to **k** other nodes
k represents the distance between the two most distant points (diameter).
A cube connected network is a network butterfly whose columns are collapsed into single nodes.

4D Hypercube or Binary 4-Cube



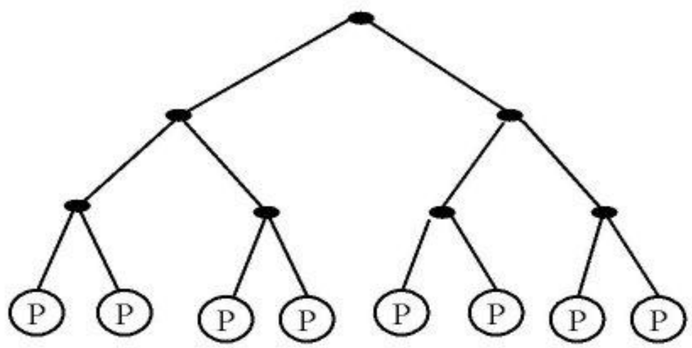
Tree Topology

The processors are located in the terminal nodes (leaves) of a binary tree

The degree of a network with tree topology with N nodes is $\log_2 N$ and the diameter is

$2 \log_2(N) - 2$

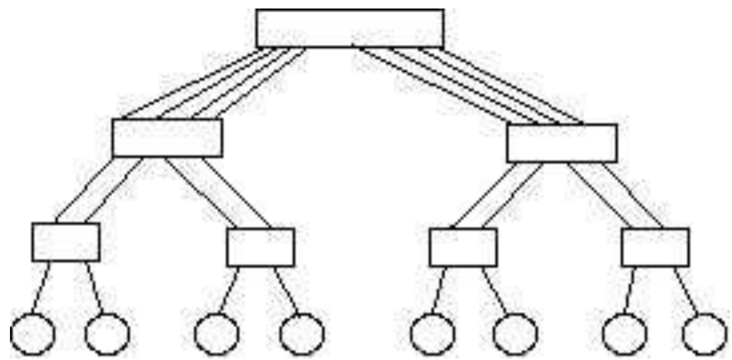
Tree Network Diagram



Fat tree

tree topology where the bandwidh of the network increases with the level of the network.

CM-5, Network Myrinet e QSNet



Pyramid

A pyramidal network of amplitude p is a complete 4-ry tree with levels, where the nodes of each level are connected in a bi-dimensional mesh.

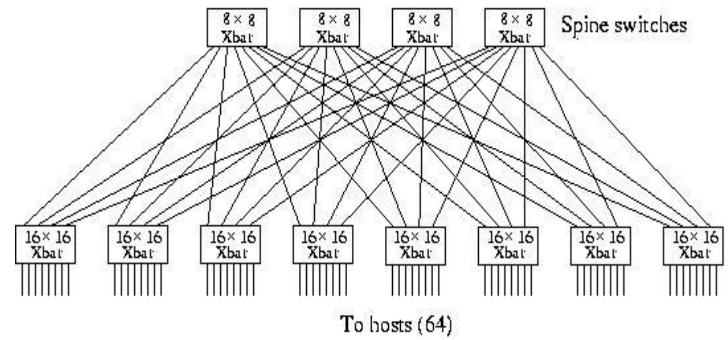
Commodity Interconnects

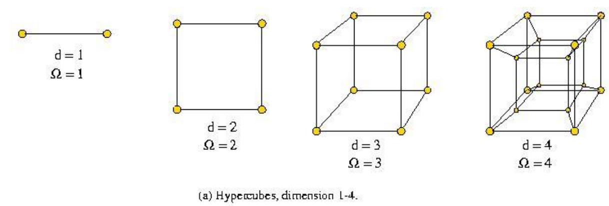
- Gig Ethernet
- Myrinet
- Infiniband
- QsNet
- SCI

Clos

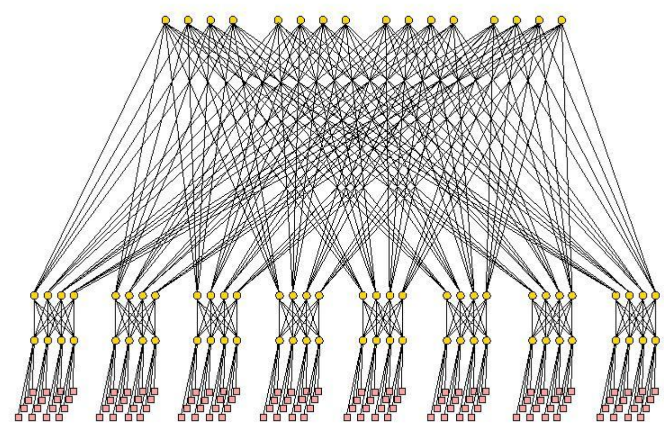
Fat tree

Torus

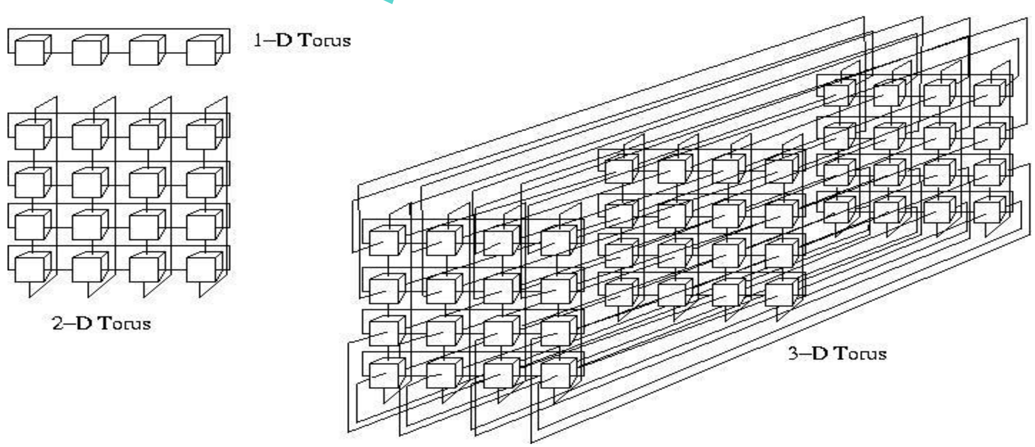




(a) Hypercubes, dimension 1-4.



(b) A 128-way fat tree.



HPC Architectures

Performance comes from:

- Device Technology
- Computer Architecture

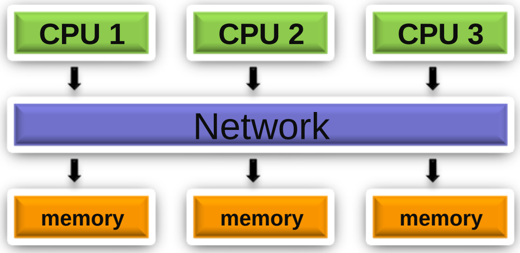
Characteristics of HPC system:

- Multi-processor/multi-core nodes

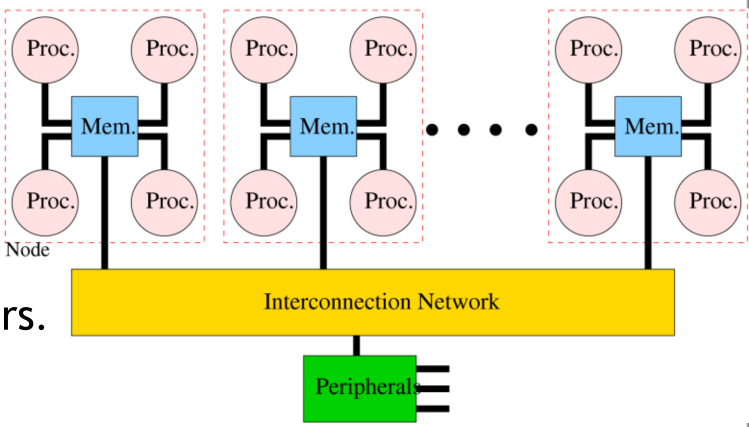
  (not necessarily SMP nodes anymore!)

- Node architecture may differ, even with similar components
- Network interconnecting nodes
- Parallelism
  - Number of operations per cycle per processor
    - Instruction level parallelism (ILP)
    - Vector processing
  - Number of processors per node
  - Number of nodes in a system
- Hybrids, combining standard processors with accelerators.

Shared Memory Systems



Distributed Memory



The macro-architecture of HPC systems is presently remarkably uniform: but this aspect will change soon

HPC architectures /1

The are several factors that have an impact on the system architectures in the presently:

1. Power consumption has become a primary headache.
2. Processor speed is never enough.
3. Network complexity/latency is a main hindrance.
4. There is still the memory wall.

Interestingly, solutions for point 1 and 2 can often be combined.

Since a few years computational accelerators of various kinds are offered that may address speed and power consumption.

Fitting accelerators into a general purpose sytem:

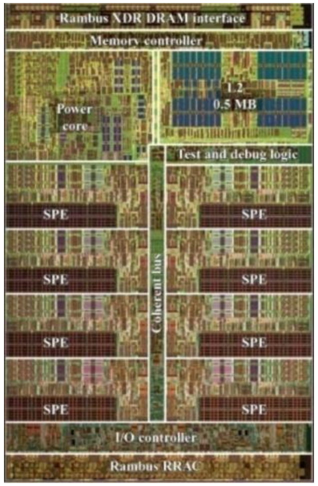
This is a general problem that is met by at least AMD and Intel.

Goal: Let accelerator communicate directly with the system's memory and General Purpose Computational Cores.

HPC Architectures: Parameters affecting Performance

- Peak floating point performance
- Main memory capacity
- Bi-section bandwidth
- I/O bandwidth
- Secondary storage capacity
- Organization
  - Class of system
  - # nodes
  - # processors per node
  - Accelerators
  - Network topology
- Control strategy
  - MIMD
  - Vector, PVP
  - SIMD
  - SPMD

Heterogeneous Multicore Architecture



Combines different types of processors

- Each optimized for a different operational modality
  - Performance > *N*x better than other *N* processor types
- Synthesis favors superior performance
  - For complex computation exhibiting distinct modalities

Conventional co-processors

- Graphical processing units (GPU)
- MIC
- Network controllers (NIC)
- Efforts underway to apply existing special purpose components to general applications

Purpose-designed accelerators

- Integrated to significantly speedup some critical aspect of one or more important classes of computation
- IBM Cell architecture

ClearSpeed SIMD attached array processor

Drawback of all accelerators: **No standard (yet).**

– Software Development Kits far from uniform (but improving rapidly, OpenFPGA initiative, OpenCL ...).

- – Programming is hard work.

HPC systems evolution

Vector Processors
- Cray-1

SIMD, Array Processors
- Goodyear MPP, MasPar 1 & 2, TMC CM-2

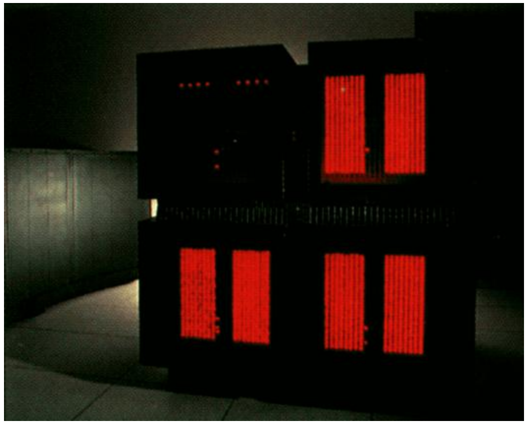
Parallel Vector Processors (PVP)
- Cray XMP, YMP, C90 NEC Earth Simulator, SX-6

Massively Parallel Processors (MPP)
- Cray T3D, T3E, TMC CM-5, Blue Gene/L

Commodity Clusters
- Beowulf-class PC/Linux clusters
- Constellations

Distributed Shared Memory (DSM)
- SGI Origin
- HP Superdome

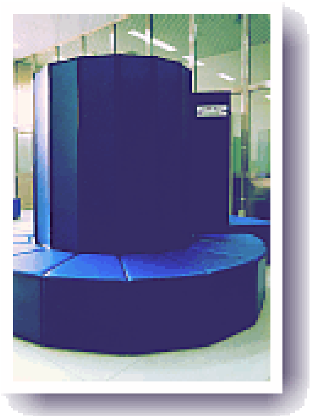
Hybrid HPC Systems
- Roadrunner
- Chinese Tianhe-1A system
- GPGPU systems



HPC systems evolution in CINECA

1969:	CDC 6600	1st system for scientific computing
1975:	CDC 7600	1st supercomputer
1985:	Cray X-MP / 4 8	1st vector supercomputer
1989:	Cray Y-MP / 4 64	
1993:	Cray C-90 / 2 128	
1994:	Cray T3D 64	1st parallel supercomputer
1995:	Cray T3D 128	
1998:	Cray T3E 256	1st MPP supercomputer
2002:	IBM SP4 512	1 Teraflops
2005:	IBM SP5 512	
2006:	IBM BCX	10 Teraflops
2009:	IBM SP6	100 Teraflops
2012:	IBM BG/Q	2 Petaflops

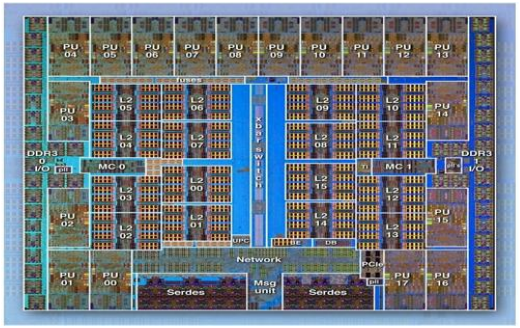







BG/Q in CINECA

10 BGQ Frame
10240 nodes
1 PowerA2 processor per node
16 core per processor
163840 cores
1GByte / core
2PByte of scratch space
2PFlop/s peak performance
1MWatt liquid cooled



-16 core chip @ 1.6 GHz

- a crossbar switch links the cores and L2 cache memory together.
- 5D torus interconnect

The Power A2 core has a 64-bit instruction set (unlike the prior 32-bit PowerPC chips used in BG/L and BG/P

The A2 core have four threads and has in-order dispatch, execution, and completion instead of out-of-order execution common in many RISC processor designs.

The A2 core has 16KB of L1 data cache and another 16KB of L1 instruction cache.

Each core also includes a quad-pumped double-precision floating point unit: Each FPU on each core has four pipelines, which can be used to execute scalar floating point instructions, four-wide SIMD instructions, or two-wide complex arithmetic SIMD instructions.

FERMI@CINECA

- Architecture: 10 BGQ Frames
- Model: IBM-BG/Q
- Processor type: IBM PowerA2 @1.6 GHz
- Computing Cores: 163840
- Computing Nodes: 10240
- RAM: 1GByte / core (163 PByte total)
- Internal Network: 5D Torus
- Disk Space: 2PByte of scratch space
- Peak Performance: 2PFlop/s
- N. 12 in Top 500 rank (June 2013)
- National and PRACE Tier-0 calls

PLX@CINECA

IBM Cluster linux

Processor type: 2 six-cores Intel Xeon (Esa-Core Westmere)

X 5645 @ 2.4 GHz, 12MB Cache

N. of nodes / cores: 274 / 3288

RAM: 48 GB/Compute node (14 TB in total)

Internal Network: Infiniband with 4x QDR switches (40 Gbps)

Accelerators: 2 GPUs nVIDIA M2070 per node

548 GPUs in total

Peak performance: 32 Tflops

565 TFlops SP GPUs

293 TFlops DP GPUs

N. 266 in Top 500 rank (June 2013)

National and PRACE Tier-1 calls

Eurora: EURopean many integrated cORe Architecture

- Hybrid cluster based on the evolution of the AURORA architecture by Eurotech
- 64 nodes Intel Sandy Bridge dual socket, (1024 cores in total)
  - 32 nodes at 2.1 GHz and
  - 32 nodes at 3.1 GHz).
- 16 GByte DDR3 RAM, 160 GByte SSD, 1 FPGA Altera Stratix V. per Node
- Interconnection networks: Infiniband QDR and 3D Torus
- Hot water cooling
- The system is equipped with:
  - 64 MIC processors (2 per node on 32 nodes)
  - 64 NVIDIA K20 accelerators (2 per node on 32 nodes)
- Peak performance (K20 accelerated) 175,7 Tflop/s
- N. 467 in Top 500 rank (June 2013)
- N. 1 in Green500 rank (June 2013)

LINPACK Benchmark

http://www.top500.org/

The TOP500 project was started in 1993 to provide a reliable basis for tracking and detecting trends in high-performance computing. Twice a year, a list of the sites operating the 500 most powerful computer systems is assembled and released. The best performance on the **Linpack** benchmark is used as performance measure for ranking the computer systems. The list contains a variety of information including the system specifications and its major application areas.

The LINPACK Benchmark was introduced by Jack Dongarra. The LINPACK Benchmark is to solve a dense system of linear equations. For the TOP500, is used that version of the benchmark that allows the user to scale the size of the problem and to optimize the software in order to achieve the best performance for a given machine.
This performance does not reflect the *overall performance* of a given system, as no single number ever can. It does, however, reflect the *performance of a dedicated system for solving a dense system of linear equations.* Since the problem is very regular, the performance achieved is quite high, and the performance numbers give a good correction of peak performance.
By measuring the actual performance for different problem sizes n, a user can get not only the maximal achieved performance Rmax for the problem size Nmax but also the problem size N1/2 where half of the performance Rmax is achieved. These numbers together with the theoretical peak performance Rpeak are the numbers given in the TOP500.
In an attempt to obtain uniformity across all computers in performance reporting, the algorithm used in solving the system of equations in the benchmark procedure must conform to the standard operation count for LU factorization with partial pivoting. In particular, the operation count for the algorithm must be 2/3 n^3 + O(n^2) floating point operations. This excludes the use of a fast matrix multiply algorithm like "Strassen's Method". This is done to provide a comparable set of performance numbers across all computers.

Top 500: some facts

1976 Cray 1 installed at Los Alamos: peak performance 160 MegaFlop/s (10^6 flop/s)

1993 (1° Edition Top 500) N. 1 59.7 GFlop/s (10^{12} flop/s)

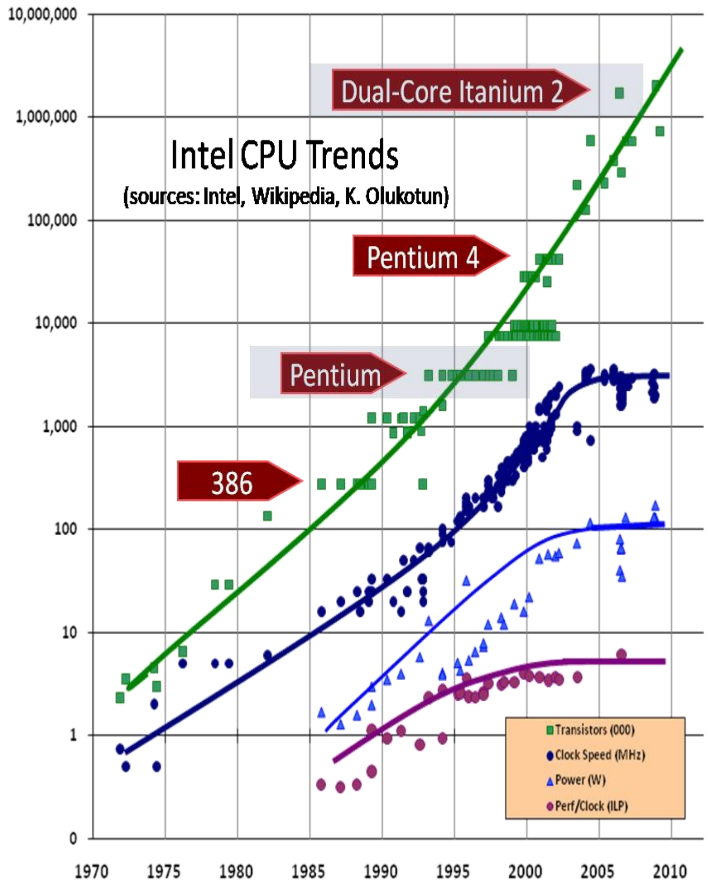
1997 Teraflop/s barrier (10^{12} flop/s)

2008 Petaflop/s (10^{15} flop/s): Roadrunner (LANL) Rmax 1026 Gflop/s, Rpeak 1375 Gflop/s hybrid system: 6562 processors dual-core AMD Opteron accelerated with 12240 IBM Cell processors (98 TByte di RAM)

2011 11.2 Petaflop/s : K computer (SPARC64 VIIIfx 2.0GHz, Tofu interconnect) RIKEN Japan

- 62% of the systems on the top500 use processors with six or more cores
- 39 systems use GPUs as accelerators (35 NVIDIA , 2 Cell , 2 ATI Radeon)

HPC Evolution



From Herb Sutter<hsutter@microsoft.com>

Moore's law is holding, in the number of transistors

– Transistors on an ASIC still doubling every 18 months at constant cost

– 15 years of *exponential* clock rate growth has ended

Moore's Law reinterpreted

– Performance improvements are now coming from the increase in the number of cores on a processor (ASIC)

– #cores per chip doubles every 18 months *instead* of clock

– 64-512 threads per node will become visible soon

- Million-way parallelism

Heterogeneity: Accelerators

-GPGPU

- MIC

Real HPC Crisis is with Software

A supercomputer application and software are usually much more long-lived than a hardware

- Hardware life typically four-five years at most.
- Fortran and C are still the main programming models

Programming is stuck

- *Arguably hasn't changed so much since the 70's*

Software is a major cost component of modern technologies.

- The tradition in HPC system procurement is to assume that the software is free.

It's time for a change

- Complexity is rising dramatically
- Challenges for the applications on Petaflop systems
- Improvement of existing codes will become complex and partly impossible
- The use of O(100K) cores implies dramatic optimization effort
- New paradigm as the support of a hundred threads in one node implies new parallelization strategies
- Implementation of new parallel programming methods in existing large applications has not always a promising perspective

There is the need for new community codes

multidisciplinary applications

PRACE

Partnership for Advanced Computing in Europe

http://www.prace-project.eu/

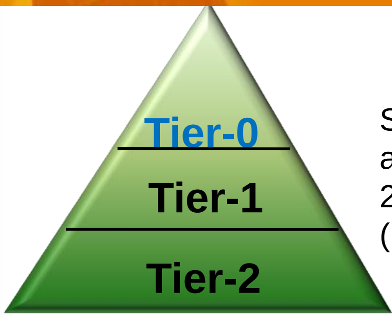
PRACE is part of the ESFRI roadmap and has the aim of creating a European Research Infrastructure providing world class systems and services and coordinating their use throughout Europe.

It covers both hardware at the multi petaflop/s level and also very demanding software (parallel applications) to exploit these systems.

The HPC European e-infrastructure (ESFRI)

PRACE 1.0

- **25** members, AISBL since 2010
- **530 M€** for 2010-2015 (inc 70M€ from UE)
- **6** supercomputers in **4** hosting countries, different architectures
- research and industrial access (open R&D) for all disciplines **based on excellence in science, free of charge**
- Nearly **15 Pflop/s**
- **5 billion** hours granted since 2010

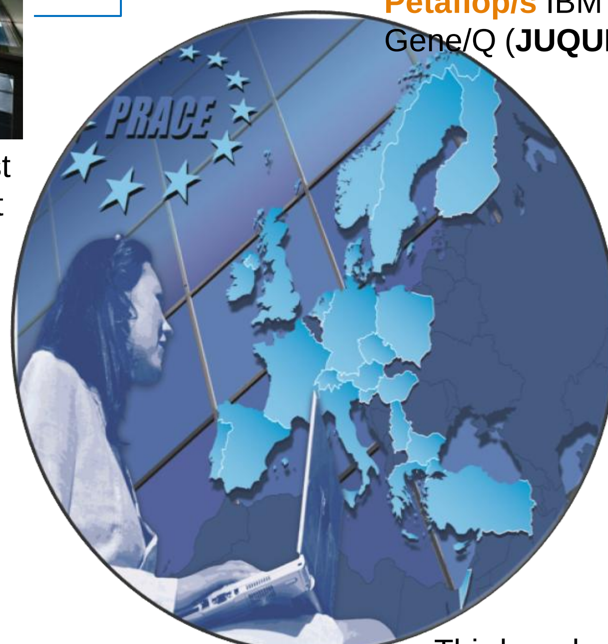


Sixth production system available by January 2013: **1 Petaflop/s** IBM (**MareNostrum**) at BSC.

Upgrade: **5.87 Petaflop/s** IBM Blue Gene/Q (**JUQUEEN**)

First production system available: **1 Petaflop/s** IBM BlueGene/P (**JUGENE**) at GCS (Gauss Centre for Supercomputing) partner FZJ (Forschungszentrum Jülich)

Fifth production system available by August 2012: **2 Petaflop/s** IIBM BG/Q (**FERMI**) at CINECA.



Second production system available: Bull Bullx **CURIE** at GENCI partner CEA. Full capacity of **1.8 Petaflop/s** reached by late 2011.

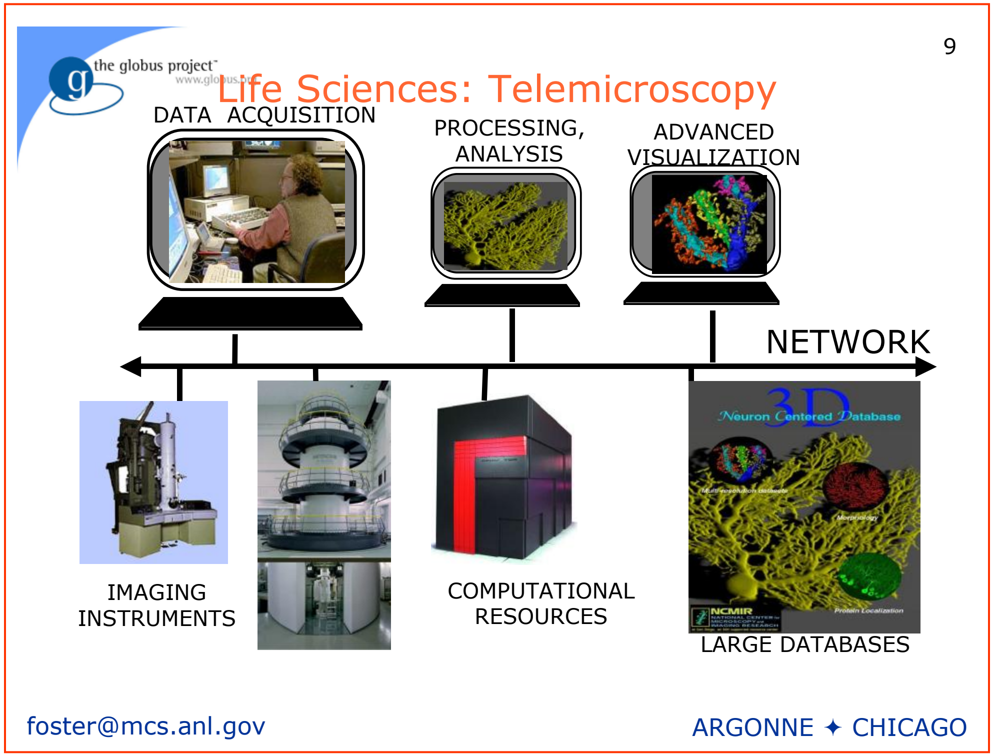
Fourth production system available by mid 2012: **3 Petaflop/s** IBM (**SuperMUC**) at GCS partner LRZ (Leibniz-Rechenzentrum).

Third production system available by the end of 2011: **1 Petaflop/s** Cray (**HERMIT**) at GCS partner HLRS Stuttgart.

HPC and Grid Computing

Grid computing

Networked infrastructure for advanced research that can seamlessly connect distributed teams to the high-end computing systems, instruments, advanced simulation and visualization software, and sensor networks they need to work collaboratively on data- and computation-intensive problems.



HPC and Grid are non-orthogonal but complementar

Bit and Byte on Information Society

Gigabyte [1,000,000,000 bytes OR 10^9 bytes]

500 megabytes: A CD-ROM

100 megabytes: 1 meter of shelved books

10 megabytes: A minute of high-fidelity sound

5 megabytes:The complete works of Shakespeare

2 megabytes: A high-resolution photograph
1 megabyte: A small novel OR a 3.5-inch floppy disk

Megabyte [1,000,000 bytes OR 10^6 bytes]

200 kilobytes: A box of punched cards

100 kilobytes: A low-resolution photograph

50 kilobytes: A compressed document image page
10 kilobytes: An encyclopaedia page

2 kilobytes: A typewritten page

1 kilobyte: A very short story

Kilobyte [1,000 bytes OR 10^3 bytes]

100 bytes: A telegram or a punched card

10 bytes: A single word

1 byte: A single character

Byte [8 bits]

Bit [A binary digit - either 0 or 1]

Bits and Bytes on Information Society/1

Zettabyte [1,000,000,000,000,000,000,000 bytes OR 10^{21} bytes]

5 exabytes: All words ever spoken by human beings
2 exabytes: Total volume of information generated worldwide annually

Exabyte [1,000,000,000,000,000,000 bytes OR 10^{18} bytes]

200 petabytes: All printed material
8 petabytes: All information available on the Web
2 petabytes: All U.S. academic research libraries
1 petabyte: 3 years of Earth Observing System (EOS) data (2001)

Petabyte [1,000,000,000,000,000 bytes OR 10^{15} bytes]

400 terabytes: National Climatic Data Center (NOAA) database
50 terabytes:The contents of a large mass storage system
10 terabytes:The printed collection of the U.S. Library of Congress
2 terabytes: An academic research library
1 terabyte: 50,000 trees made into paper and printed OR daily rate of EOS data (1998)

Terabyte [1,000,000,000,000 bytes OR 10^{12} bytes]

500 gigabytes: The biggest FTP site
100 gigabytes: A floor of academic journals
50 gigabytes: A floor of books
2 gigabytes: 1 movie on a Digital Video Disk (DVD)
1 gigabyte: A pickup truck fi lled with paper