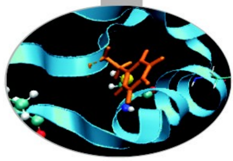


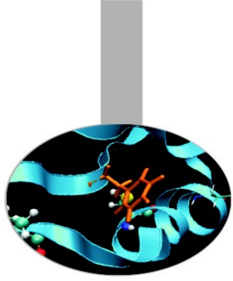
Welcome!

School on Data Analytics and ~~Visualization~~ Deep Learning

Giuseppe Fiameni

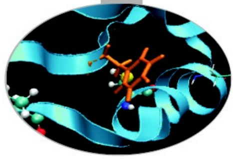
June 12th 2017

School major topics

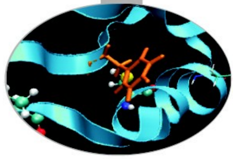


- *Computing Architecture*
- *Data Analytics*
- *Machine Learning*
- *Programming with R and H20*
- *Apache Spark*
- *INTEL Data Analytics Accelerating Library*
- *Deep Learning*
- *Convolutional Networks*
- *Programming with Tensorflow*

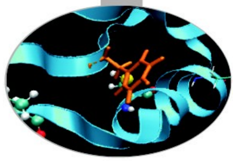
SuperComputing Applications and Innovation



Computing architectures



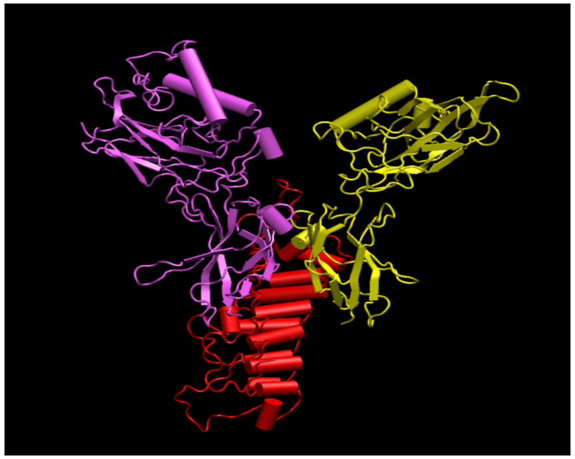
What does it make a problem be relevant from the computational point of view?



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

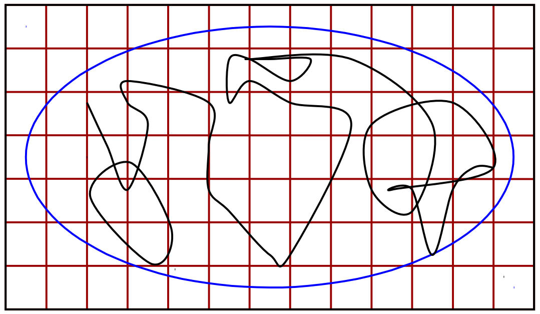
is a unit to measure the speed of a computer.

Computational problems today: 10^{15} – 10^{22} flop

One year has about 3 x 10^7 seconds!

Most powerful computers today have reach a sustained performance is of the order of Tflop/s - Pflop/s (10^{12} - 10^{15} flop/s).

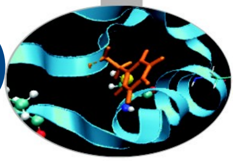
Example: Weather Prediction



Forecasts on a global scale:

- **3D Grid to represent the Earth**
  - Earth's circumference: ≅ 40000 km
  - radius: ≅ 6370 km
  - Earth's surface: ≅ $4\pi r^2$ ≅ $5 \cdot 10^8$ km^2
- **6 variables:**
  - temperature
  - pressure
  - humidity
  - 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)

Example: Weather Prediction (cont.)



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) ≅ $2 \bullet 10^{12}$ Byte = 2TB

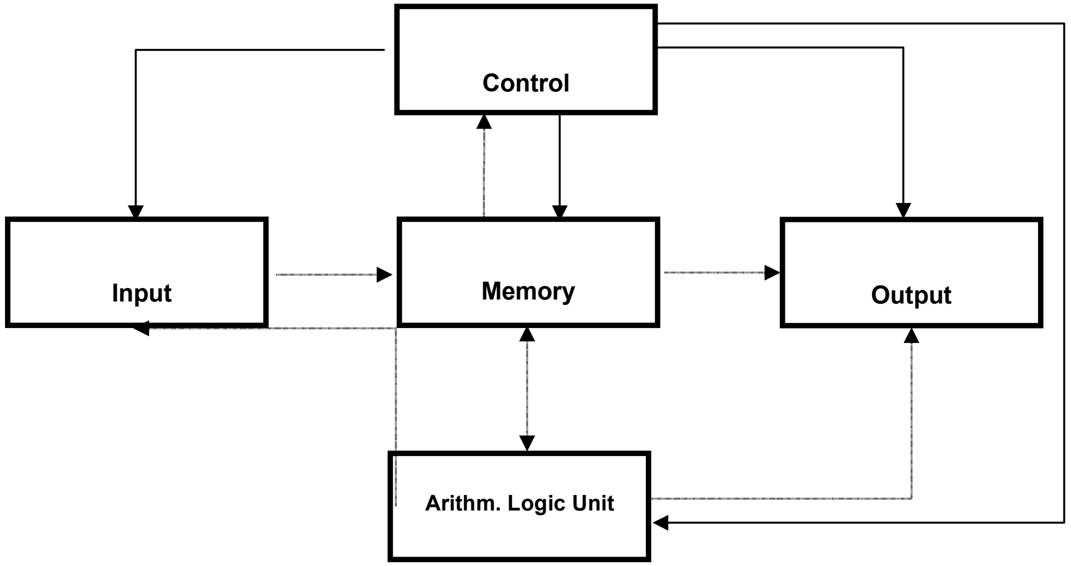
A 24 hours forecast needs:

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

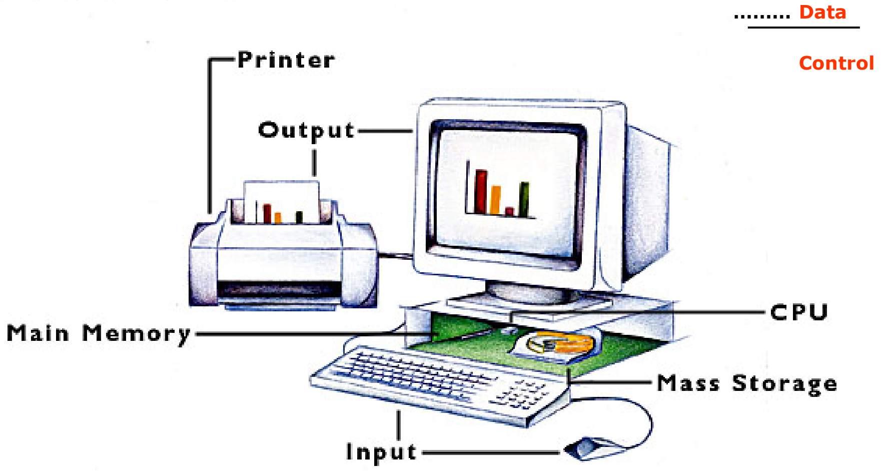
A computer with a power of 1Tflop/s will take $1.5 \bullet 10^{5}$ sec.

➔ **24 hours forecast will need 2days to run ... but we shall obtain a very accurate forecast**

Von Neumann Model



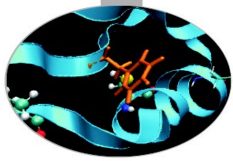
Von Neumann Model of Computer Architecture



Instructions are processed sequentially

- A single instruction is loaded from memory (fetch) and decoded
- Compute the addresses of operands
- Fetch the operands from memory;
- Execute the instruction ;
- 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 .

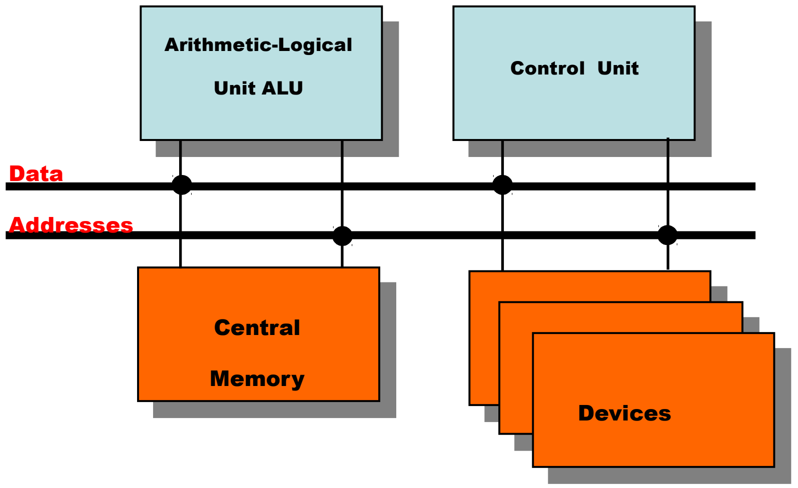
Propagation velocity of a signal in a vacuum: **300.000 Km/s = 30 cm/ns**

Heat dissipation problems inside the processor. Also Quantum tunelling expected to become important.

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 :

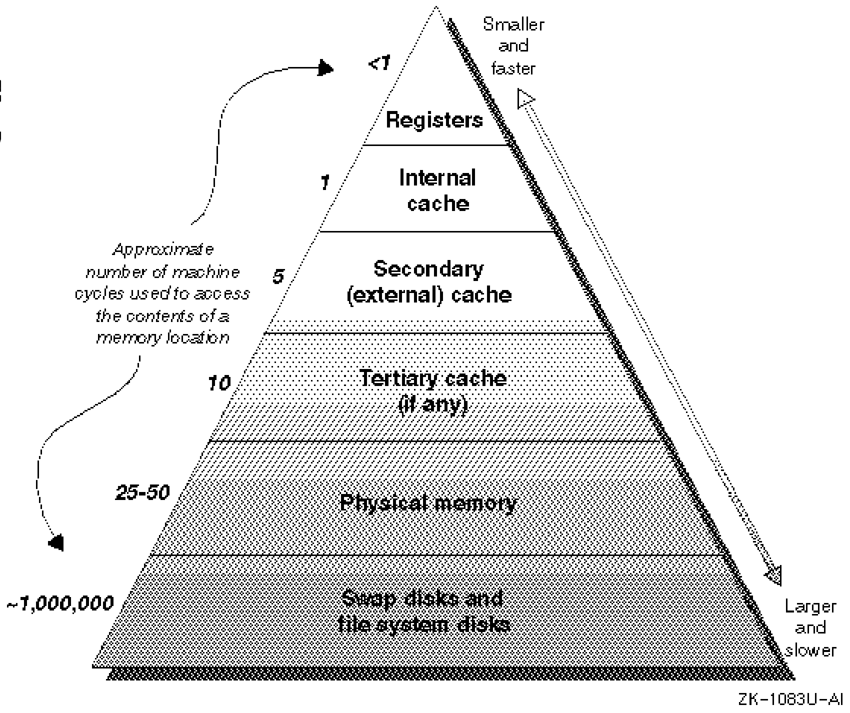
- fast memory is expensive and small
- slow memory is cheap and big

Latency

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

Throughput

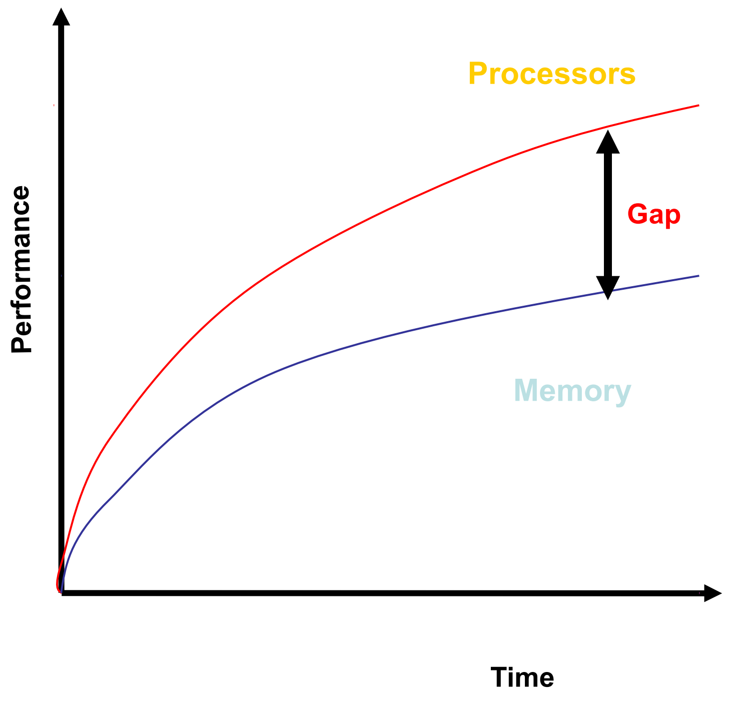
- how many bytes/second. but not important if waiting.


Smaller and faster
<1
Registers
1
Internal cache
Approximate number of machine cycles used to access the contents of a memory location
5
Secondary (external) cache
10
Tertiary cache (if any)
25-50
Physical memory
~1,000,000
Swap disks and file system disks
Larger and slower
ZK-1083U-AI

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

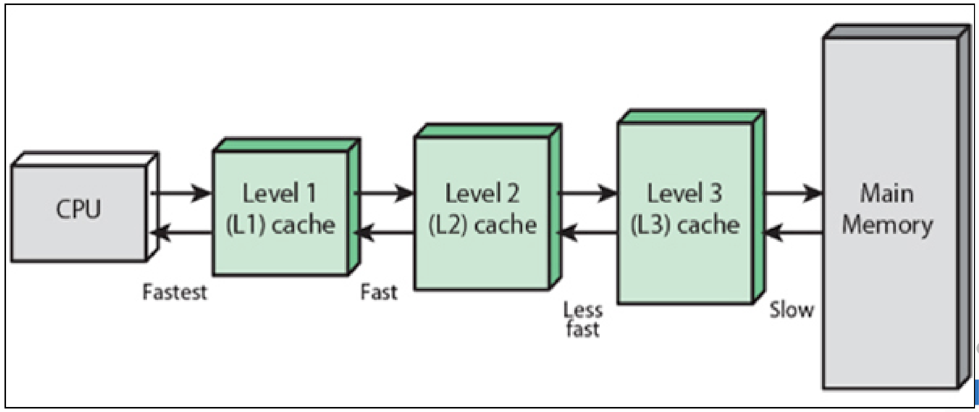
Memory access

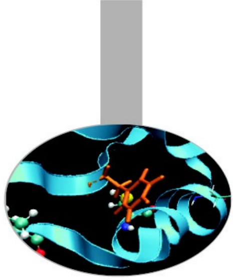
- Important problem for the performance of any computer is access to main memory. Fast processors are useless if memory access is slow!
- Over the years the difference in speed between processors and main memory has been growing.



Cache Memory

- High speed, small size memory used as a buffer between the main memory and the processor. When used correctly, reduces the time spent waiting for data from main memory.
- Present as various “levels” (e.g. L1, L2, L3, etc) according to proximity to the functional units of the processor.
- Cache efficiency depends on the locality of the data references:
  - *Temporal locality* refers to the re-use of data within relatively small time frame.
  - *Spatial locality refers to the use of data within close storage locations (e.g. one dimensional array).*
- *Cache can contain Data, Instructions or both.*





Cache Memory (cont.)

- The code performance improves 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

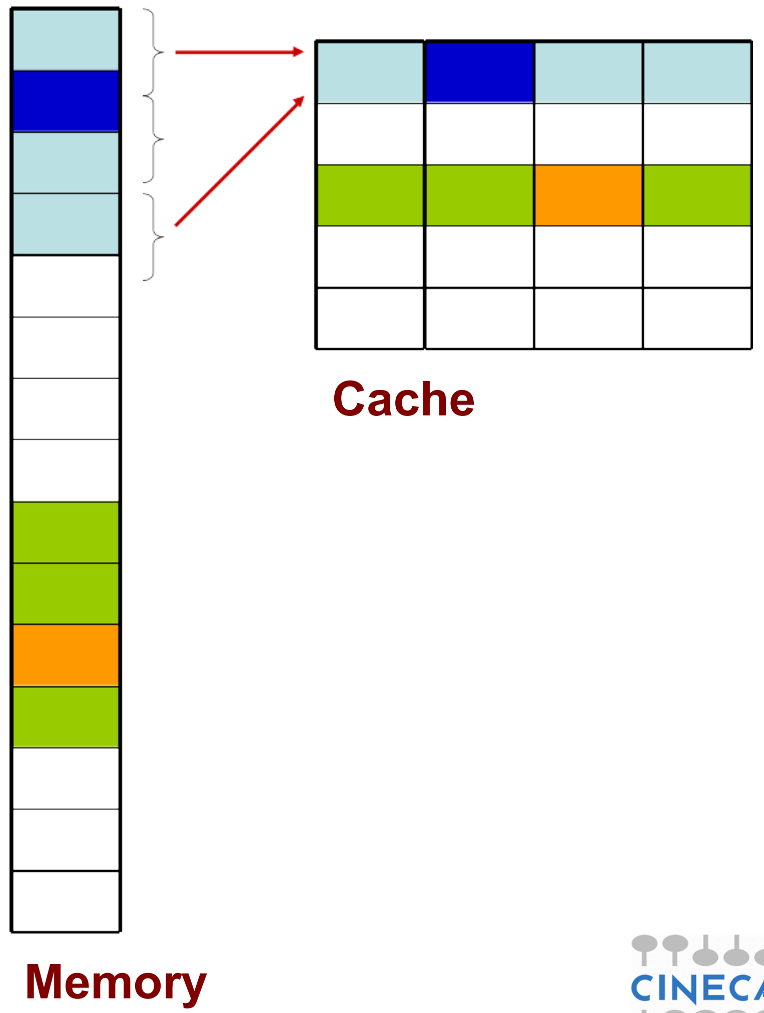
IBM SP Power 6 (4.7 GHz):

Memory access time (in clock cycles)

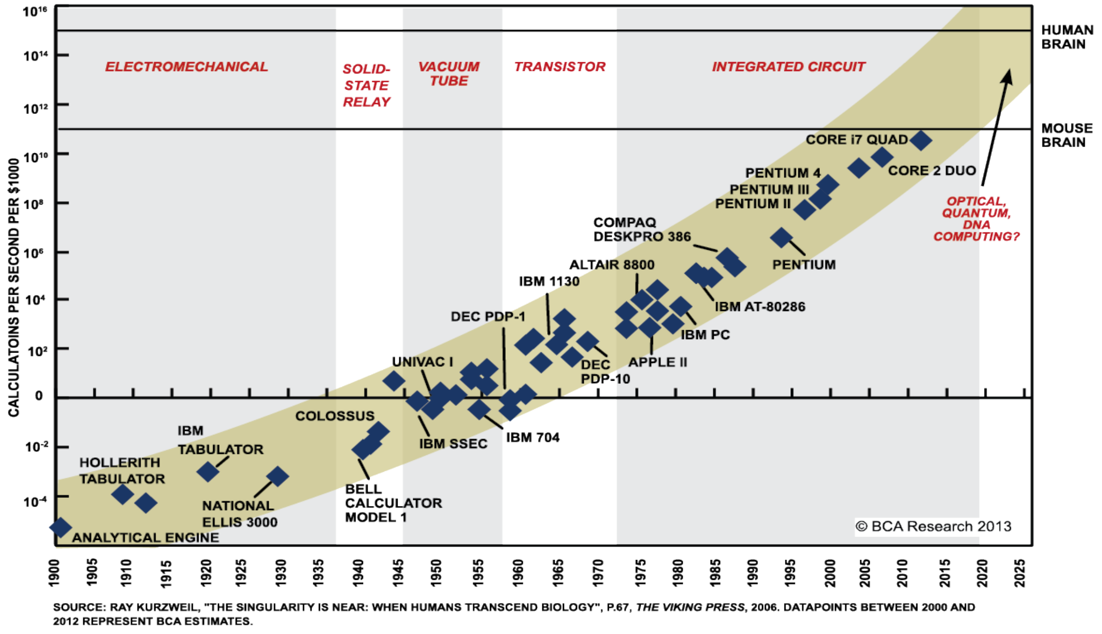
Registers	
L1: 2 x 64KB	< 5
L2: 2 x 4MB	22 cc
L3: 32 MB	160 cc
Memory 128 GB	400 cc

Cache organization

- The cache is divided into slots of the same size (lines)
- Each line contains k consecutive memory locations, i.e. 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.



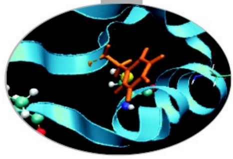
Moore's Law



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

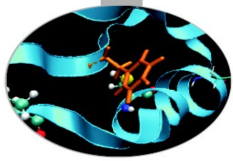
Parallelism



- **Serial Process:**
  - A process in which its sub-processes happen sequentially in time.
  - Speed depends only on the rate at which each sub-process will occur (e.g. processing unit clock speed).

- **Parallel Process:**
  - Process in which multiple sub-processes can be active simultaneously.
  - Speed depends on execution rate of each sub-process AND how many sub-processes can be made to occur simultaneously.

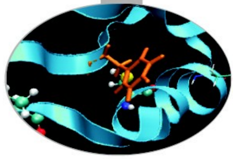
Parallelism



- **Serial Process:**
  - A process in which its sub-processes happen sequentially in time.
  - Speed depends only on the rate at which each sub-process will occur (e.g. processing unit clock speed).

- **Parallel Process:**
  - Process in which multiple sub-processes can be active simultaneously.
  - Speed depends on execution rate of each sub-process AND how many sub-processes can be made to occur simultaneously.

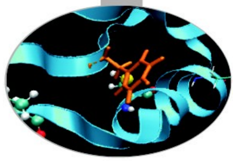
Aspects of parallelism



- It has been recognized for a long time that constant performance improvements cannot be obtained just by increasing factors such as processor clock speed – parallelism is needed.

- Parallelism can be present at many levels:
  - Functional parallelism within the CPU
  - Pipelining and vectorization
  - Multi-processor and multi-core
  - Accelerators
  - Parallel I/O

Multiple Functional Units



Arithmetic logic unit (ALU) executes the operations.

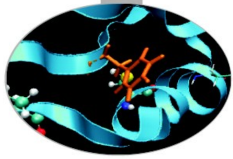
ALU is designed as a set of independent functional units, each in charge of executing a different arithmetic or logical operation:

- *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.

Aspects of parallelism



- It has been recognized for a long time that constant performance improvements cannot be obtained just by increasing factors such as processor clock speed – parallelism is needed.

- Parallelism can be present at many levels:
  - Functional parallelism within the CPU
  - Pipelining and vectorization
  - Multi-processor and multi-core
  - Accelerators
  - Parallel I/O

Pipelining

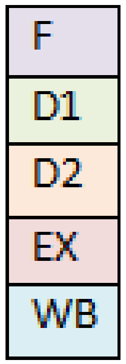
- It is a technique where more instructions, belonging to a stream of sequential execution, overlap their execution

- This technique improves the performance of the single processor

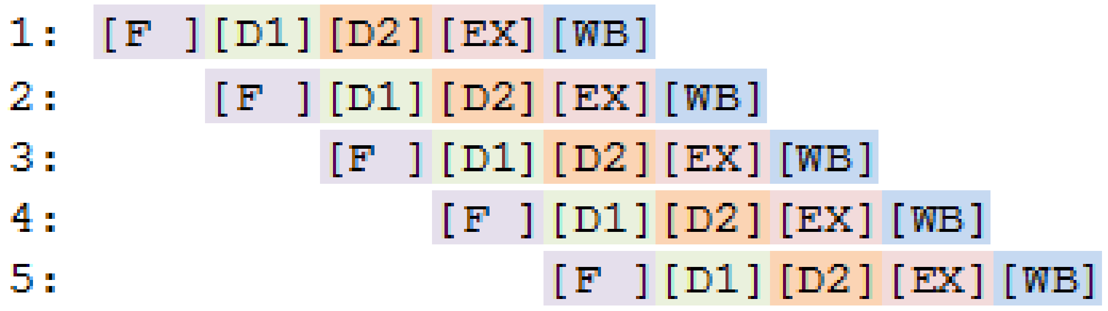
- The concept of pipelining is similar to that of assembly line in a factory where in a flow line (pipe) of assembly stations the elements are assembled in a continuous flow.

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

Instructions Pipelining



The i486 Pipeline

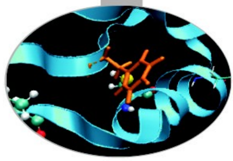


Five instructions going through a pipeline at the same time.

The stages are
1: Fetch
2: D1 (main decode)
3: D2 (secondary decode, also called translate)
4: EX (execute)
5: WB (write back to registers and memory)

http://www.gamedev.net/page/resources/_/technical/general-programming/a-journey-through-the-cpu-pipeline-r3115

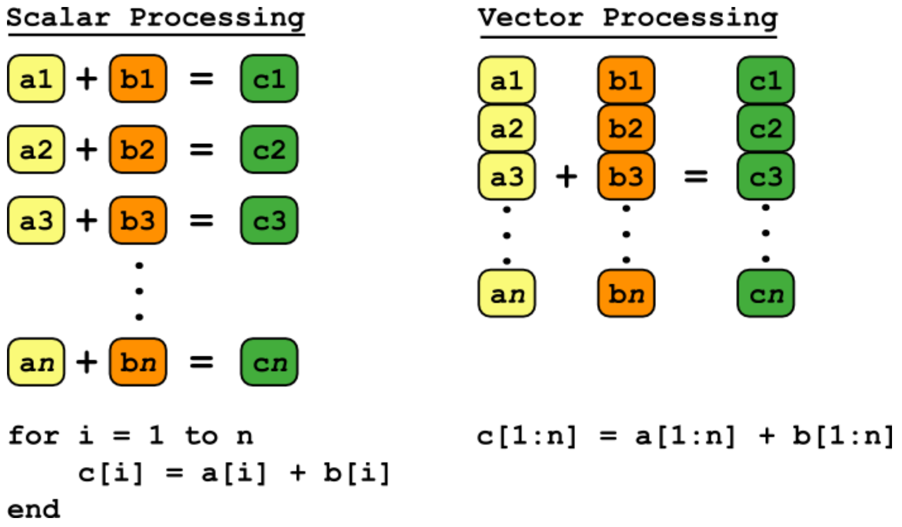
Vector Computers



- 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.
- 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.

CPU Vector units

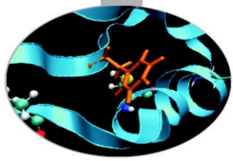
- Vectorisation performed by dedicated hardware on chip
- Compiler generates vector instructions, when it can, from programmer's code
- Important optimization which can lead to 4x, 8x speedups according "size" of vector unit (e.g. 256 bit)



```
for i = 1 to n
    c[i] = a[i] + b[i]
end
```

```
c[1:n] = a[1:n] + b[1:n]
```

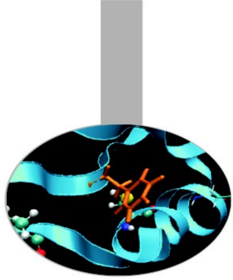
Aspects of parallelism



- It has been recognized for a long time that constant performance improvements cannot be obtained just by increasing factors such as processor clock speed – parallelism is needed.

- Parallelism can be present at many levels:
  - Functional parallelism within the CPU
  - Pipelining and vectorization
  - Multi-processor and multi-core
  - Accelerators
  - Parallel I/O

Flynn Taxonomy



M. J. Flynn

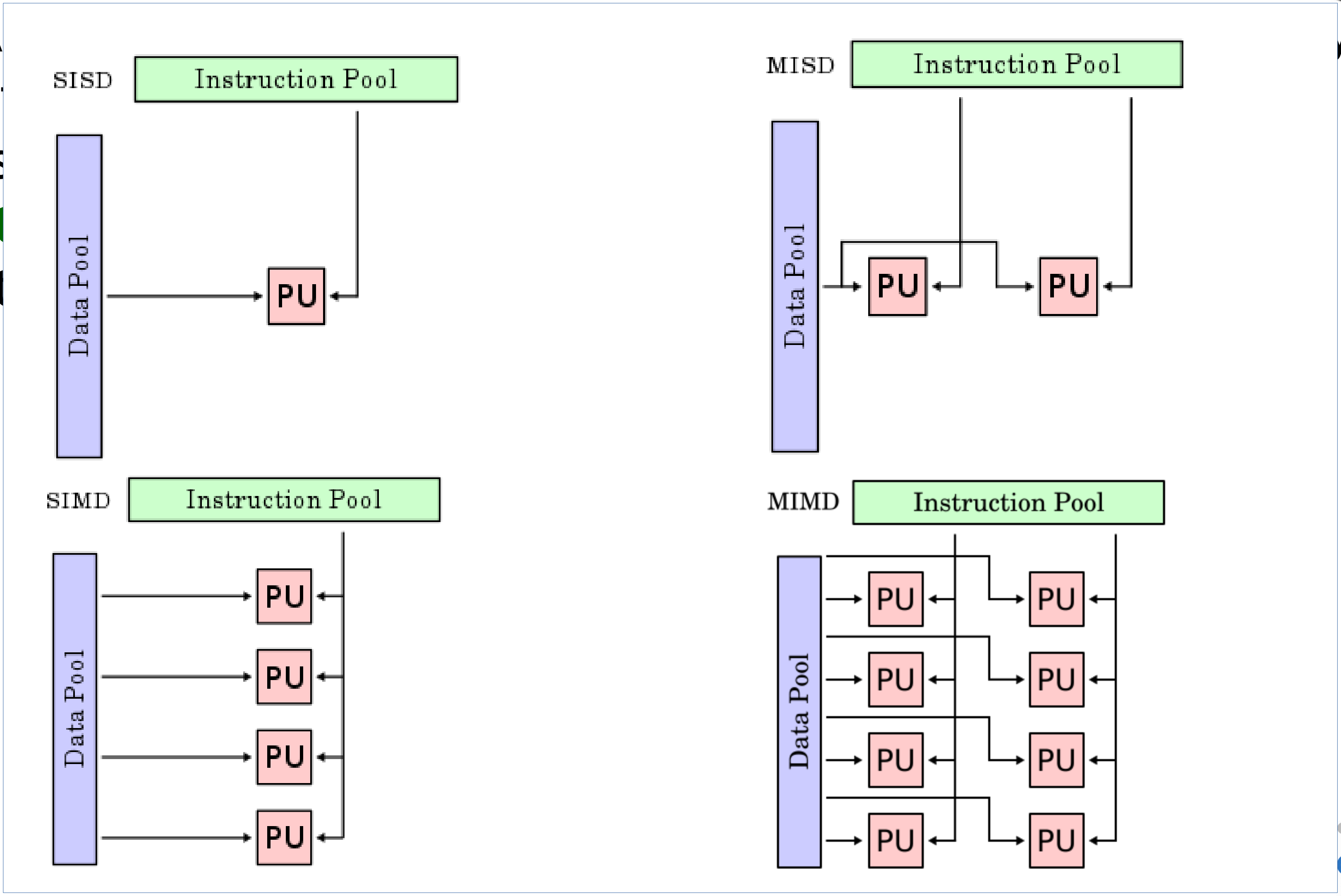
Very high speed computing systems, proceedings of the IEEE (1966).

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"

Flynn Taxonomy (cont.)

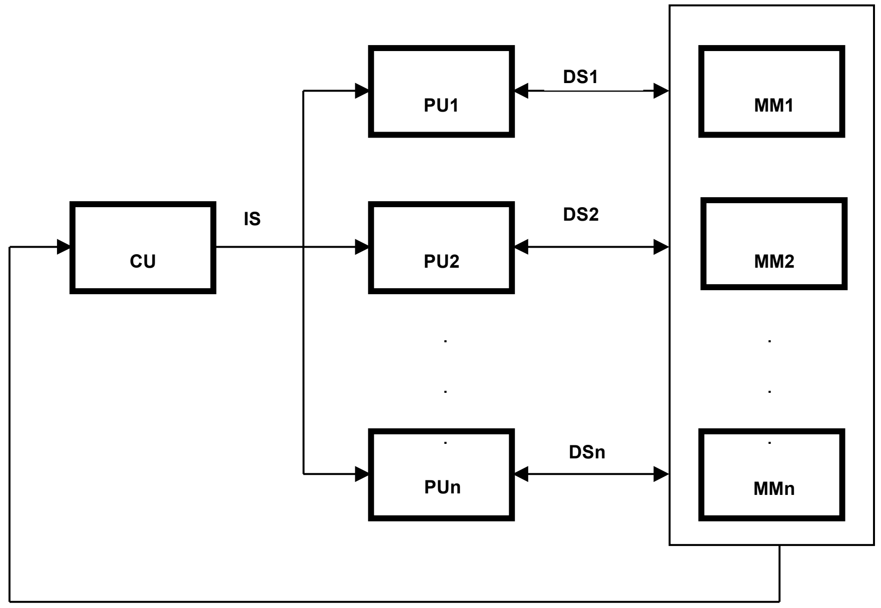
A
o
(s
st
st



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

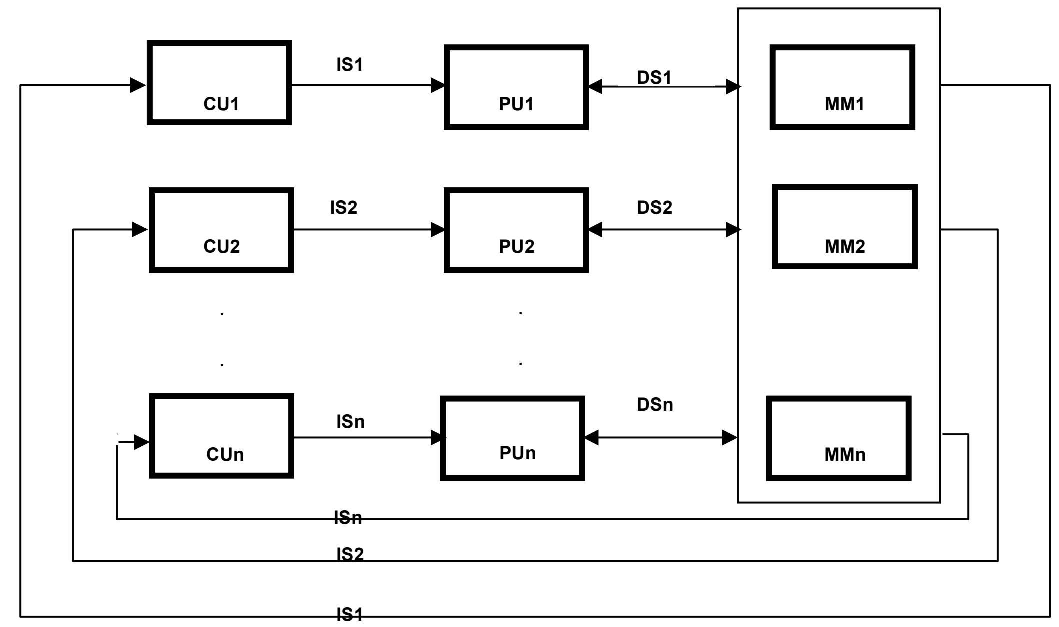


CU	Control Unit
PU	Processing Unit
MM	Memory Module
DS	Data stream
IS	Instruction Stream

MIMD Systems

Asynchronous parallelism

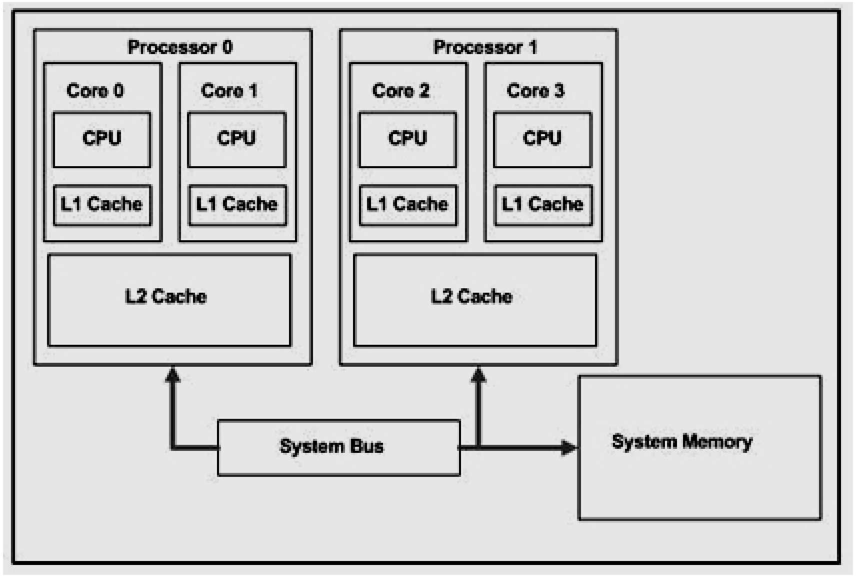
- Multiple processors execute different instructions operating on different data.
- Represents the multiprocessor version of the SIMD class.
- Wide class ranging from multi-core systems to large MPP systems.

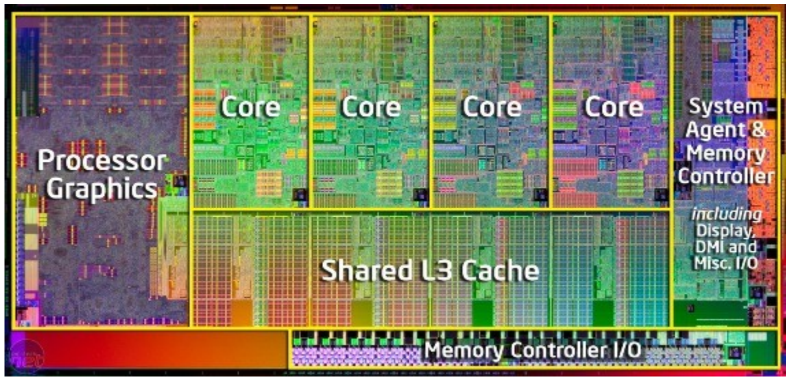


CU	Control Unit
PU	Processing Unit
MM	Memory Module
DS	Data stream
IS	Instruction Stream

Multi-core processors

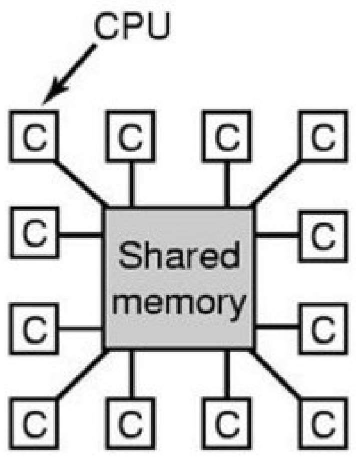
- Because of power, heat dissipation, etc increasing tendency to actually *lower* clock frequency but pack more computing cores onto a chip.
- These cores will share some resources, e.g. memory, network, disk, etc but are still capable of independent calculations

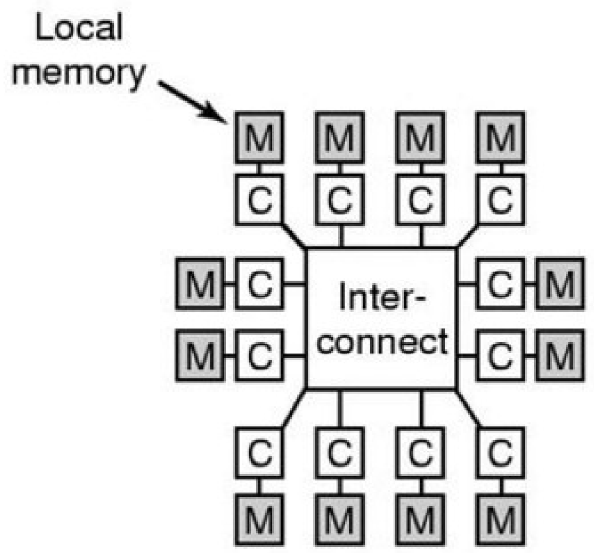


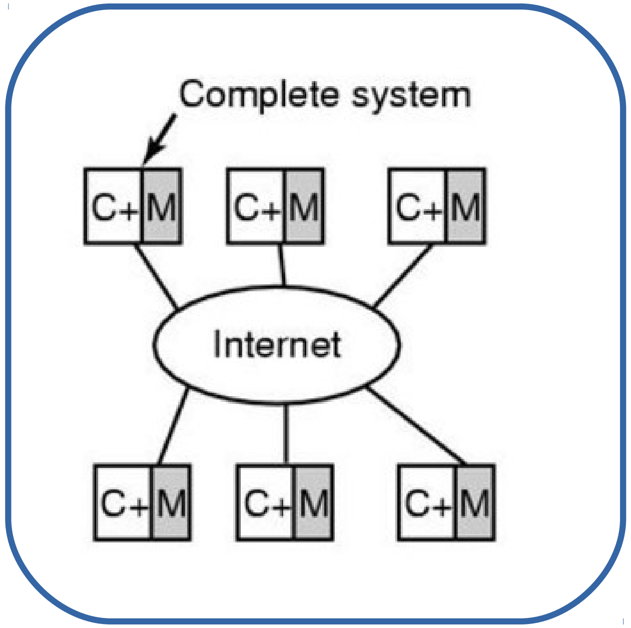


Multi-processor systems

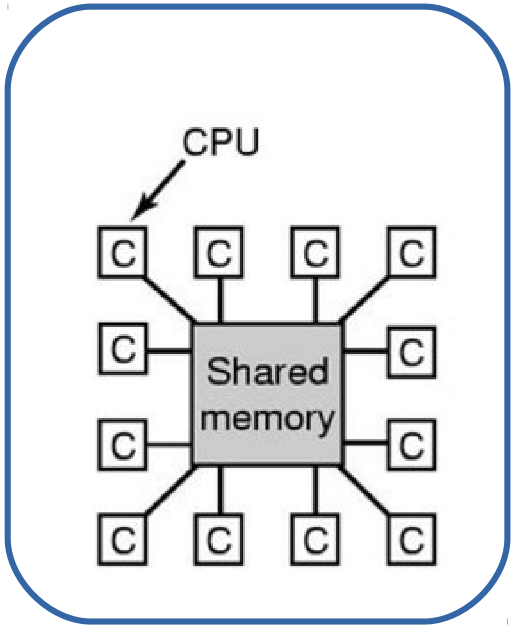
- One way to increase performance is to link (multi-core) processors together in *clusters*, perhaps grouped together first in *nodes*.

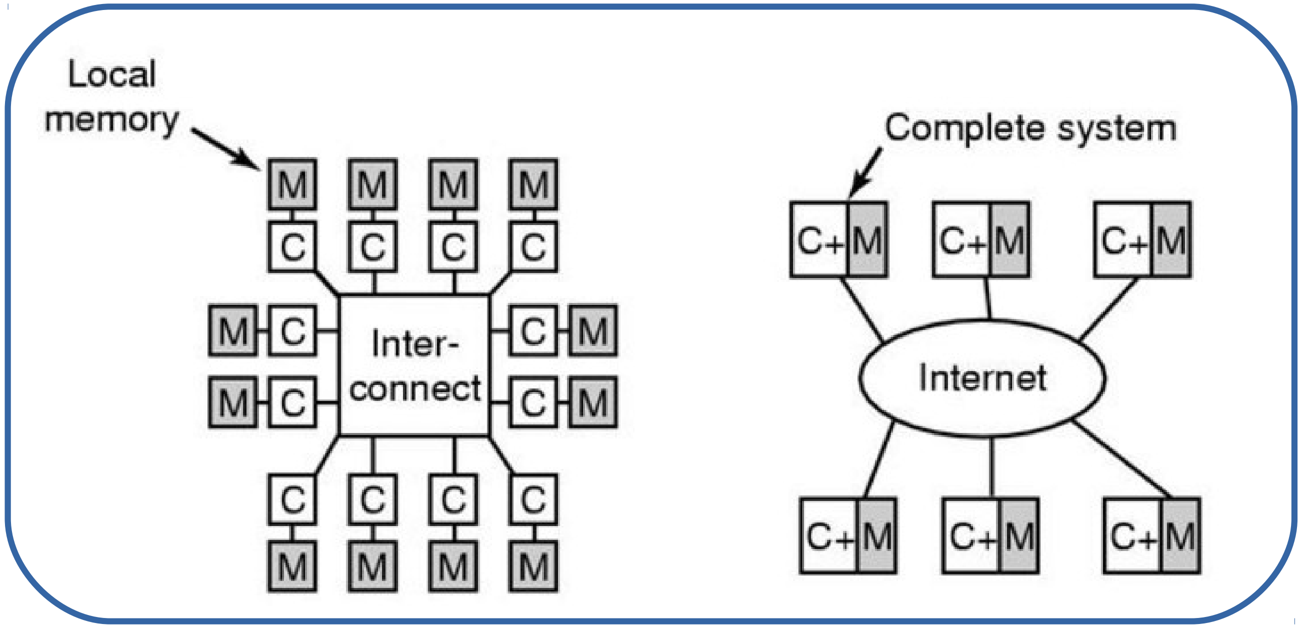






Multi-processor systems

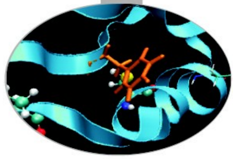




Shared Memory

Distributed Memory

Shared memory systems



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

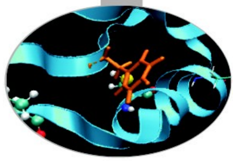
***Uniform Memory Access* (UMA) model <=> SMP: Symmetric Multi Processors**

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

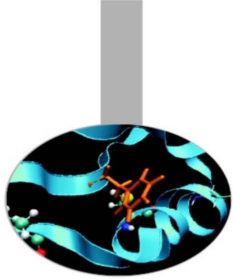
- The memory is physically distributed among the processors (local memory)

- Each processor can access directly only to its 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)

NUMA systems



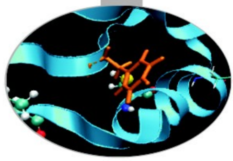
Non Uniform Memory Access (NUMA) model

- 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.

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



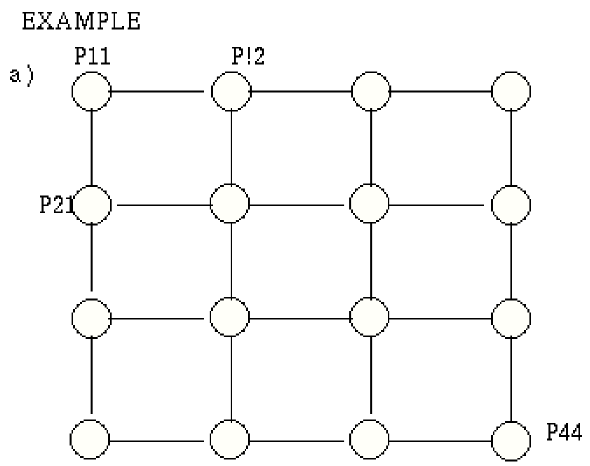
- It is the set of links (cables) that define how the different processors of a parallel computer are connected between themselves and with the memory unit.
- The time required to transfer the data depends on the type of interconnection.
- The transfer time is called the communication time.

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.**

Example network



2D mesh of width 4 with

no wraparound connections

on edge or corner nodes

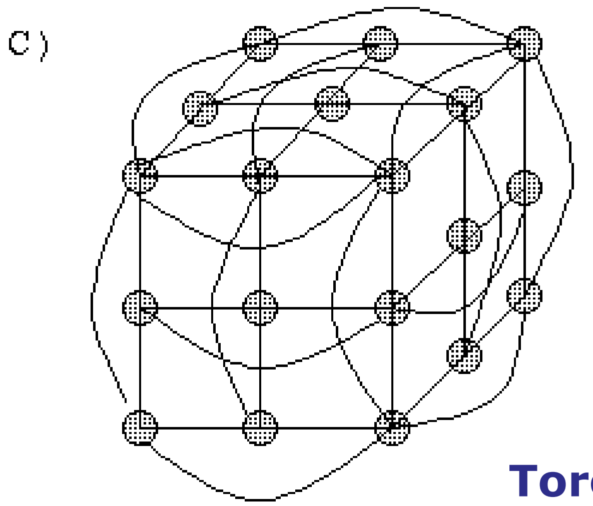
corner nodes have degree 2

edge nodes have degree 3

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** adopts a 5D torus topology

MESH Topology



k = 3 w = 3

i.e. 3 ^ 3 = 27 nodes

with wraparound connections

all nodes have degree 6 (2k)

Toroidal Topology

Commodity Interconnects

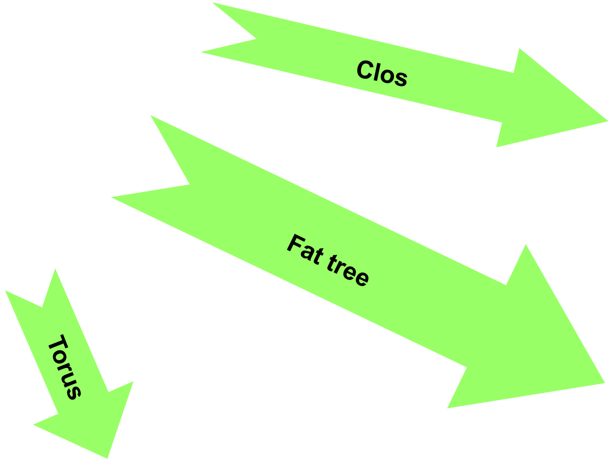
Gig Ethernet

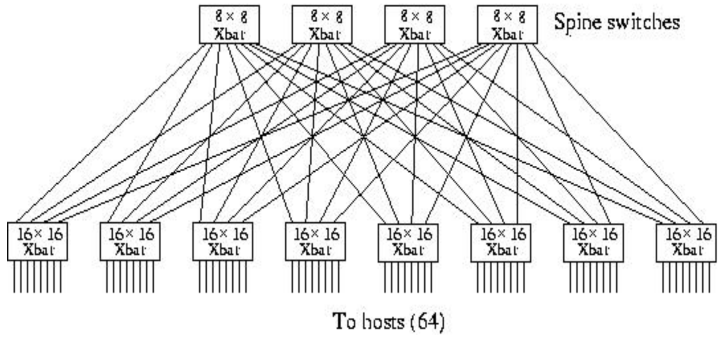
Myrinet

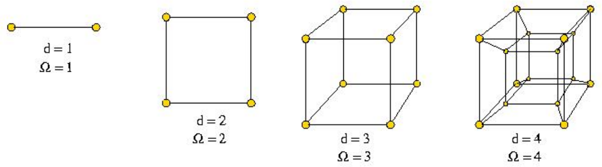
Infiniband

QsNet

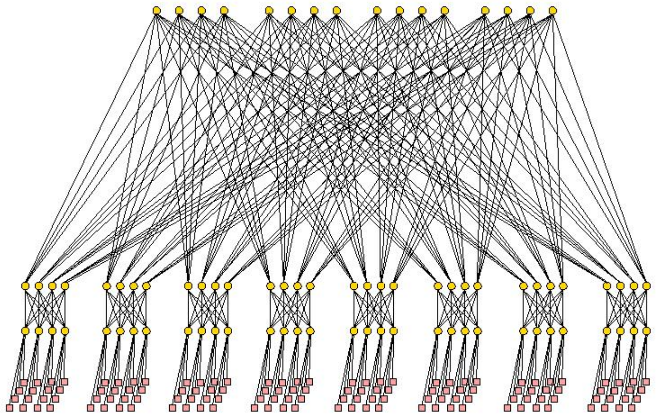
SCI



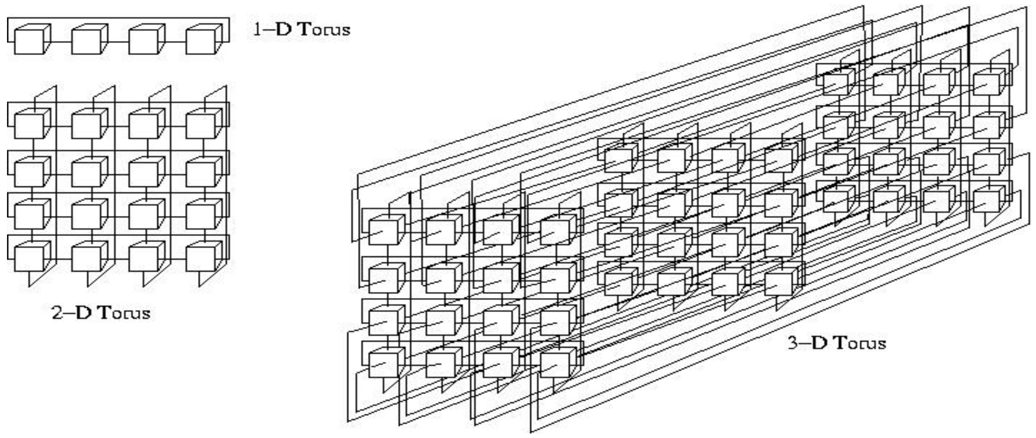




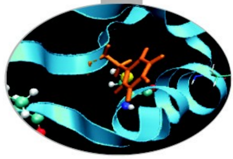
(a) Hypercubes, dimension 1-4.



(b) A 128-way fat tree.



Aspects of parallelism

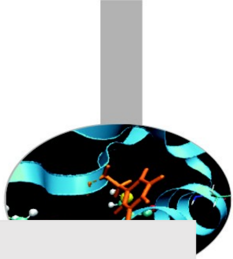


- It has been recognized for a long time that constant performance improvements cannot be obtained just by increasing factors such as processor clock speed – parallelism is needed.

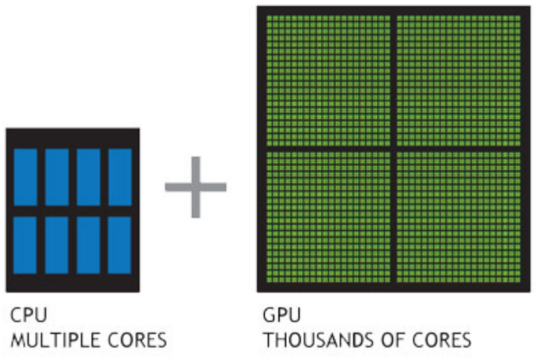
- Parallelism can be present at many levels:
  - Functional parallelism within the CPU
  - Pipelining and vectorization
  - Multi-processor and multi-core
  - Accelerators
  - Parallel I/O

Recent HPC Trends – accelerators/GPUs

- Co-processors or accelerators have been around for a while but it was only when Nvidia released CUDA did GPUs become interesting for HPC (2006).
- GPGPUs or simply GPUs work in a different way to conventional CPUs. Emphasis on stream **processing.**
- **Acceleration can be significant but depends on application.**



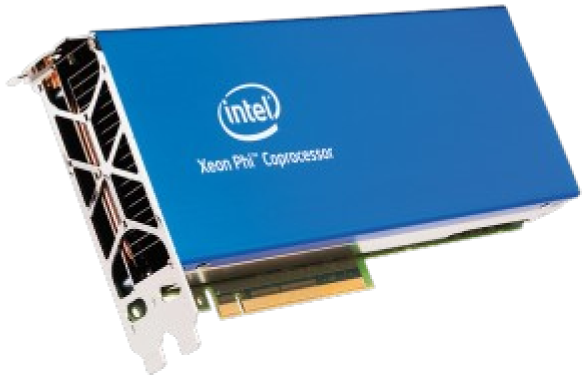
Features	Tesla K80[1]
GPU	2x Kepler GK210
Peak double precision floating point performance	**2.91 Tflops** (GPU Boost Clocks) 1.87 Tflops (Base Clocks)
Peak single precision floating point performance	8.74 Tflops (GPU Boost Clocks) 5.6 Tflops (Base Clocks)
Memory bandwidth (ECC off)[2]	480 GB/sec (240 GB/sec per GPU)
Memory size (GDDR5)	24 GB (12GB per GPU)
CUDA cores	4992 (2496 per GPU)



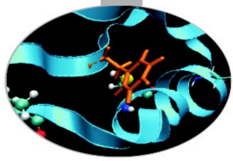
See more at: http://www.nvidia.com/object/tesla-servers.html#sthash.ENyyzyxw.dpuf

Recent HPC Trends- Accelerators/Intel Xeon PHI (MIC)

- Also an accelerator but more similar to a conventional multicore CPU.
- Current version, Knight's Corner (KNC) has 57-61 1.0-1.2 GHz cores, 8-16GB RAM. 512 bit vector unit.
- Cores connected in a ring topology and MPI possible.

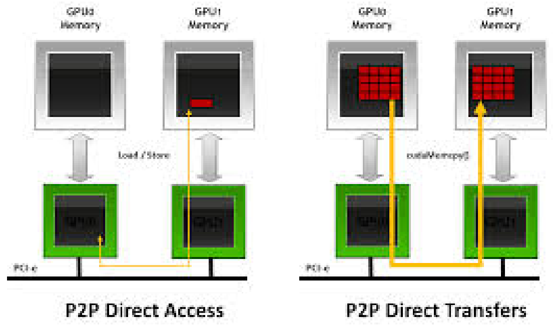


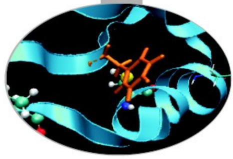
- No need to write CUDA or OpenCL as Intel compilers will compile Fortran or C code for the MIC.
- ~ 1-2 Tflops



Recent HPC Trends - Accelerators

- GPUs and MICs are attracting interest in HPC because of high performance and efficiency (i.e. Flops/watt).
- Currently, they need to be attached to host CPUs via the PCIe bus (a standard PC-like connection).
- Both device families have limitations:
  - low device memory
  - slow transfer rate via PCIe link
  - difficulty in programming (particularly CUDA).
  - speedup is highly application and data dependent.
- But future models are likely to be standalone models (e.g Knight's Landing) and with faster connections (**Nvlink**).



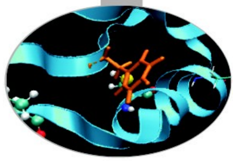


Aspects of parallelism

- It has been recognized for a long time that constant performance improvements cannot be obtained just by increasing factors such as processor clock speed – parallelism is needed.

- Parallelism can be present at many levels:
  - Functional parallelism within the CPU
  - Pipelining and vectorization
  - Multi-processor and multi-core
  - Accelerators
  - Parallel I/O

Parallel I/O



- Parallel I/O means using many I/O resources in a coordinated way to solve a single problem more quickly

Parallel I/O is becoming mandatory for applications

- "It's not working like it used to?"
- A single BG/L compute node has no more than 60 Mbyte/sec of I/O bandwidth
- But the whole machine might have 30 Gbyte/sec of I/O bandwidth!

I/O software determines how well we can make use of the available I/O hardware

Organization of I/O Software

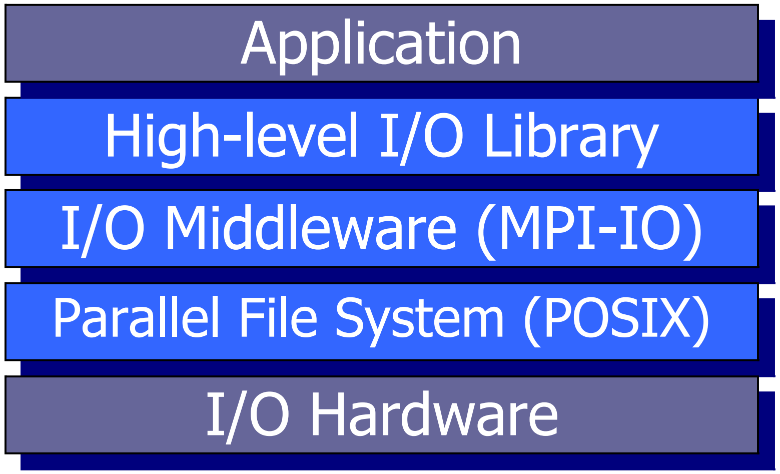
I/O components layered to provide needed functionality (I/O stacks)

- Common APIs allow combination of components

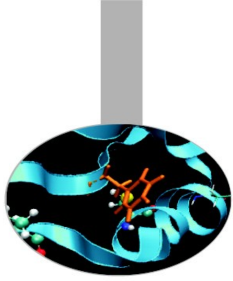
Parallel file system organizes hardware into single, fast storage space

- I/O middleware matches to programming model, provides optimizations
- Example: collective I/O operations in MPI-IO

High-level I/O libraries (HLLs) provide usability



High-level I/O Libraries



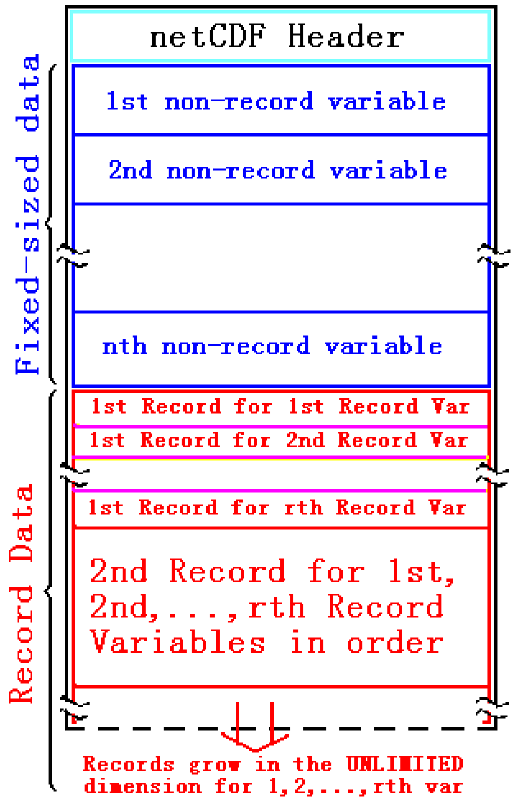
Provide structured data storage

- Multidimensional, typed datasets
- Attributes of data, provenance

Metadata is placed in the file itself, simplifying data movement, archiving

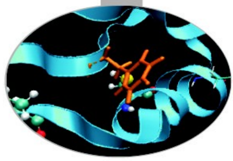
Two good examples

- **HDF5** – first to use MPI-IO, widely used
- **PnetCDF** – parallel API for netCDF data


netCDF Header
1st non-record variable
2nd non-record variable
nth non-record variable
Fixed-sized data
1st Record for 1st Record Var
1st Record for 2nd Record Var
1st Record for rth Record Var
2nd Record for 1st, 2nd,...,rth Record Variables in order
Record Data
Records grow in the UNLIMITED dimension for 1,2,...,rth var

SuperComputing Applications and Innovation

Performance and Scalability



Goal: Minimize the time applications spend performing I/O-related operations

- Maximize time applications spend computing

End-to-end I/O performance includes

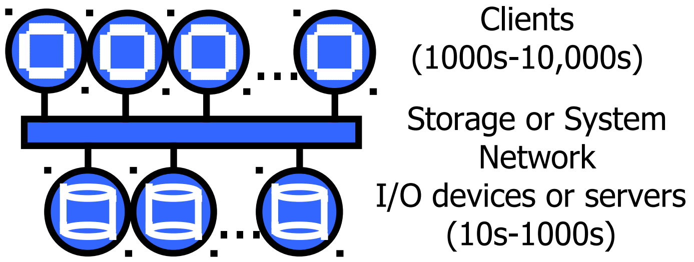
- Concurrent access to files
  - For real application access patterns
- Metadata operations
  - Creating files, traversing directories, etc.
- Overhead of all I/O software layers
  - Features aren’t free

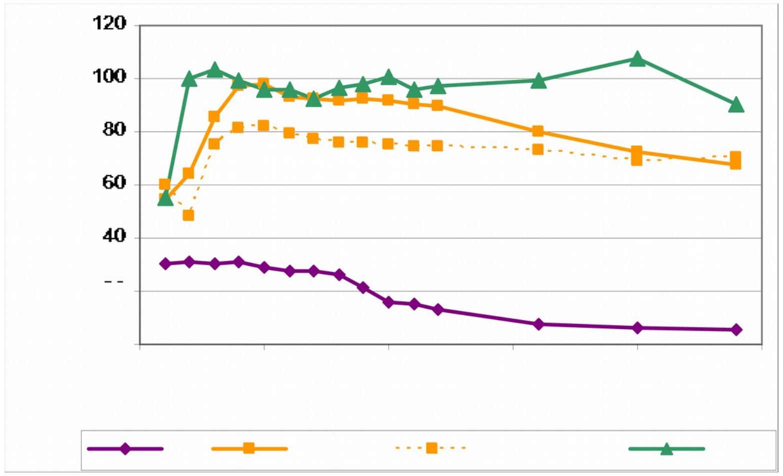
Parallel File Systems

Three popular parallel file system solutions

- GPFS
- Lustre
- PVFS/PVFS2

All capable of 10GByte/sec+ I/O rates, given adequate storage hardware and easy access patterns





Complication: I/O Access Patterns

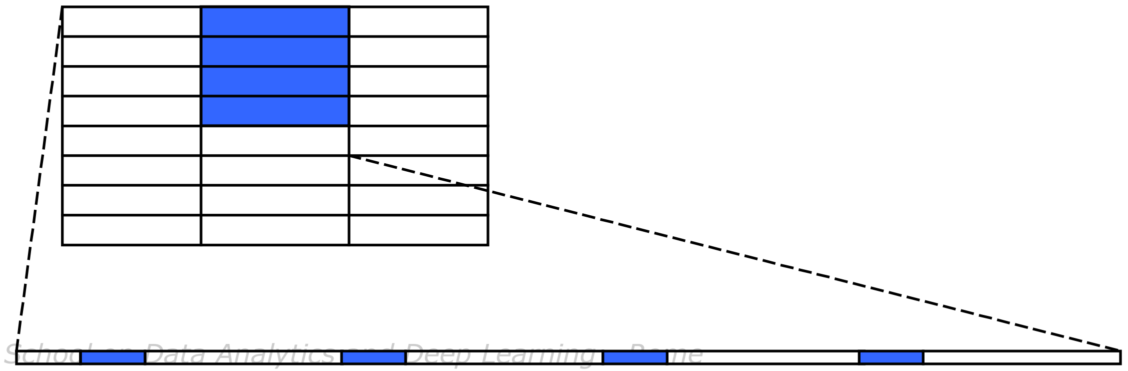
Application I/O is often complex, not just big blocks

- Ignoring ghost cells, extracting subarrays
- Additional data stored by high-level I/O libraries
- These result in noncontiguous I/O

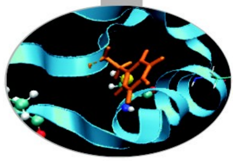
I/O interfaces determine ability to extract performance

- Define the knowledge that the I/O system has to work with

Standard (POSIX) file system interface does not allow for efficient noncontiguous access



	SmartPhone	PlayStation	IBM BG/q
Freq.	2.7 GHz	1.6GHz	1.GHz
# Core	4	8	163840
Peak Perf.	2TF/s	1.84TF/s	2PF/s
RAM (GB)	3	8	2048
Disk (GB)	128	500	2048
Power (watt)	3-5	140	1000000
Cost	650€	300€	20M€



Real HPC Crisis is with Software

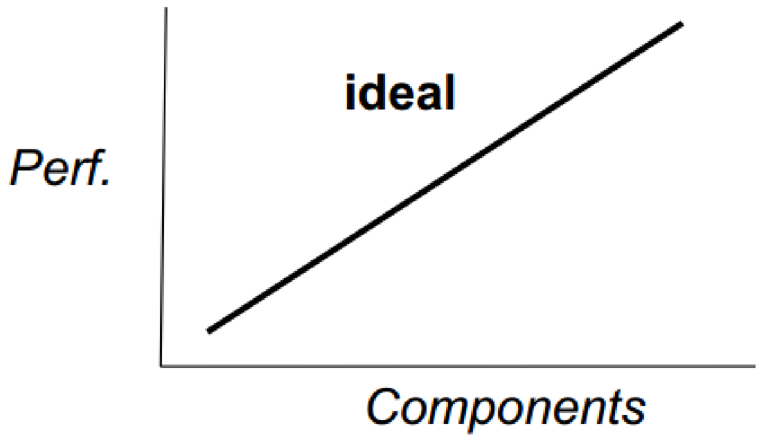
- A supercomputer application is usually much more long-lived than hardware
  - Hardware typically 4-5 years
  - FORTRAN and C still main programming models (hasn’t changed much since the 1970s)
- Porting applications to Petaflop systems is a major challenge.
  - New parallelization strategies are needed.
  - Not just program code – some datasets cannot scale to thousands of cores.
  - Also using supercomputer systems hasnt changed. Users are still expected to know UNIX and batch systems

SuperComputing Applications and Innovation

Speed-up

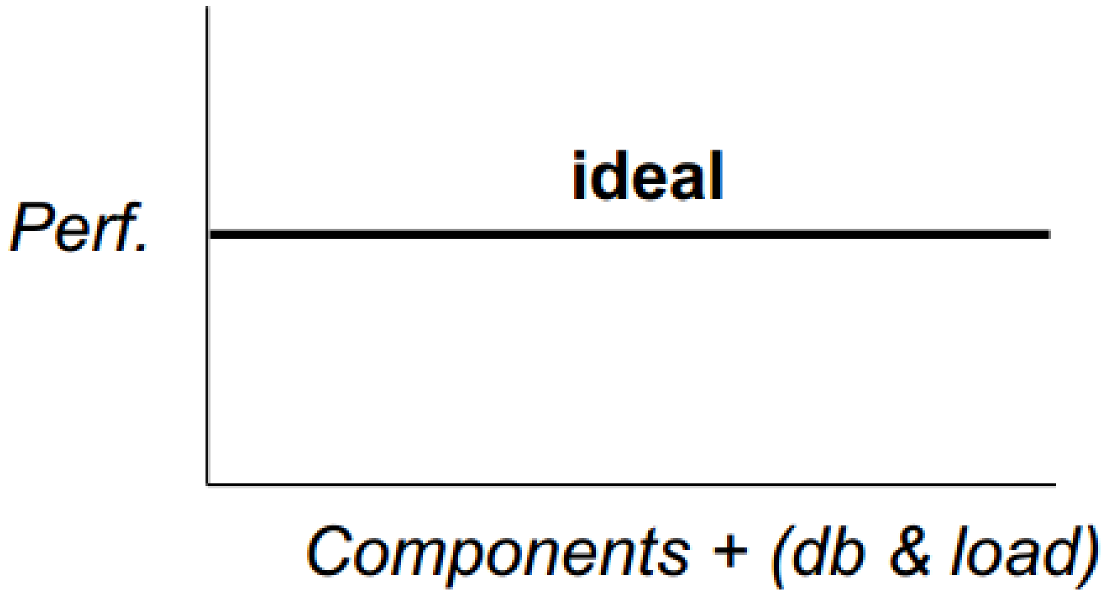
- Linear increase in performance for a constant database size and load, and proportional increase of the system components (CPU, memory, disk)

$$S_p = \frac{T_1}{T_p} = \frac{\text{Execution time of the sequential algorithm}}{\text{Execution time of the parallel algorithm with } p \text{ processors}}$$



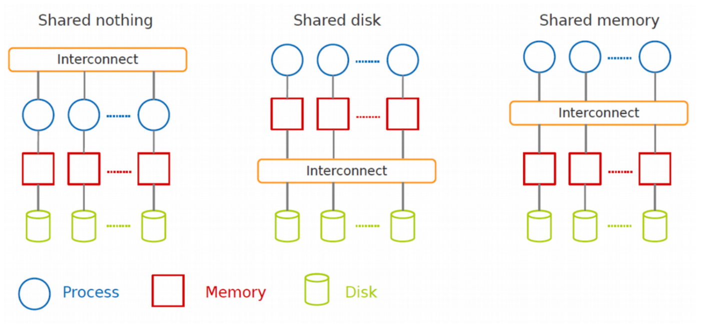
Scale-up

- Sustained performance for a linear increase of database size and load, and proportional increase of components



Parallel Architectures for Data Processing

- Three main alternatives, depending on how processors,memory and disk are interconnected
  - ***Shared-memory computer***
  - ***Shared-disk cluster***
  - ***Shared-nothing cluster***



DeWitt, D. and Gray, J. "Parallel database systems: the future of high performance database systems". ACM Communications, 35(6), 85-98, 1992.

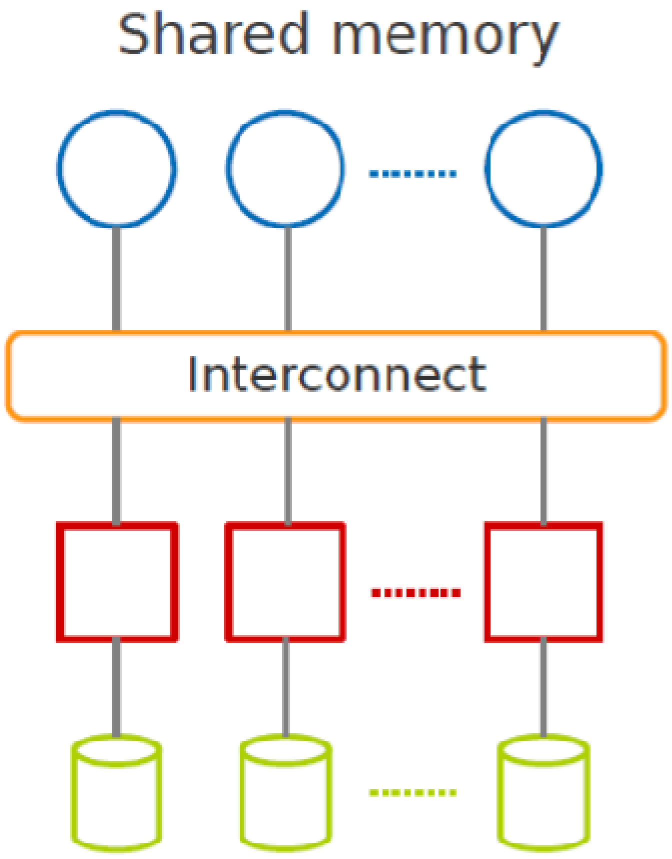
Shared Memory

- All memory and disk are shared
    - Symmetric Multiprocessor (SMP)
    - Recent: Non Uniform Memory

+ Simple for apps, fast com., load balancing

- Complex interconnect limits extensibility, cost

- For write-intensive workloads, not for big data



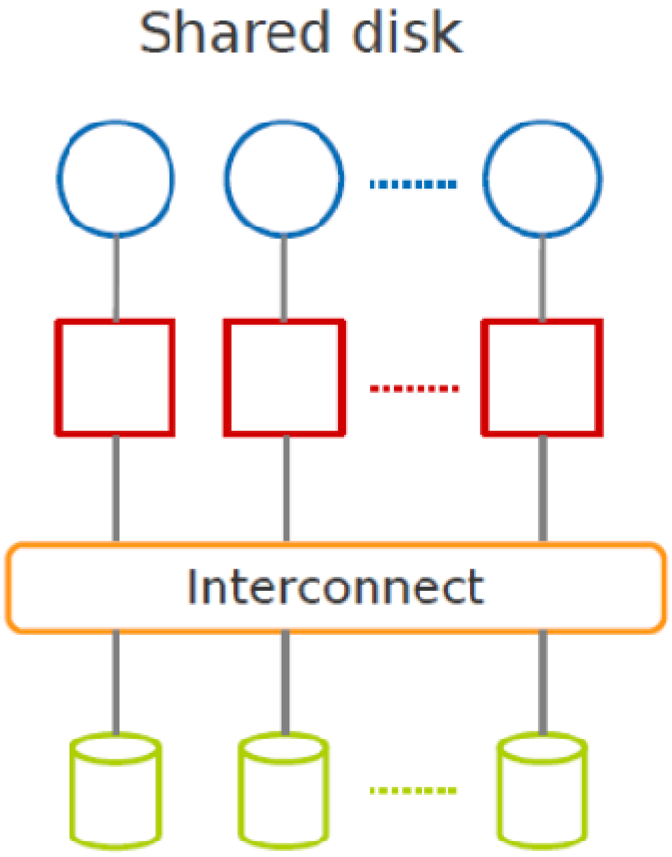
Shared Disk

- Disk is shared, memory is private
  - Storage Area Network (SAN) to interconnect memory and disk (block level)
  - Needs distributed lock manager (DLM) for cache coherence

+ Simple for apps, extensibility

- Complex DLM, cost

- For write-intensive workloads or big data



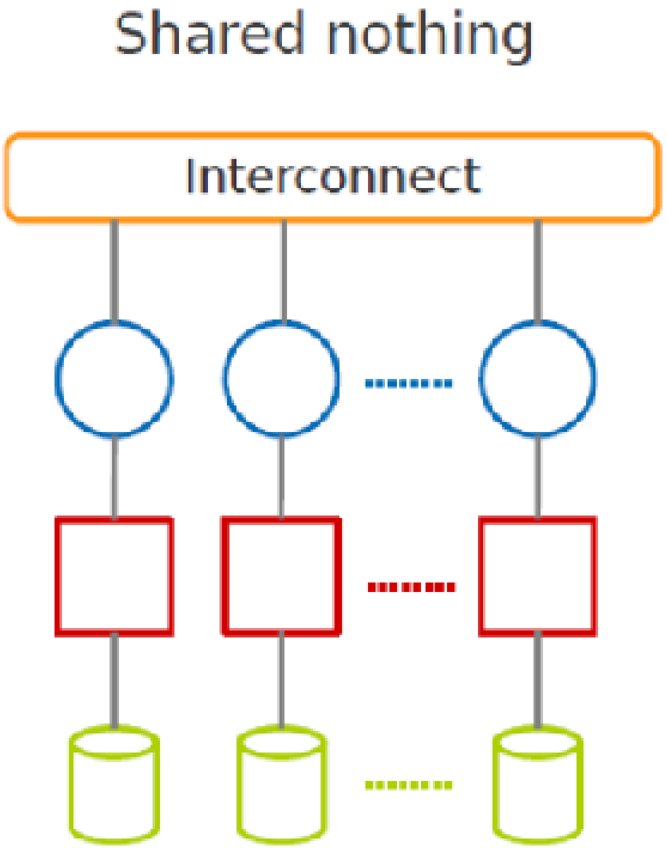
Shared Nothing

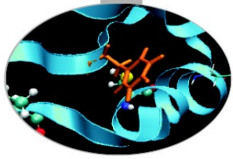
- No sharing of memory or disk across nodes
    - No need for DLM
    - But needs data partitioning

+ highest extensibility, cost

- updates, distributed trans

- For **big data** (read intensive)





Simple Model for Parallel Data

Shared-nothing architecture

- The most general and scalable

Set-oriented

- Each dataset **D** is represented by a table of rows

Key-value

- Each row is represented by a <key, value> pair where
  - *Key uniquely identifies the value in D*
  - *Value is a list of (attribute name : attribute value)*

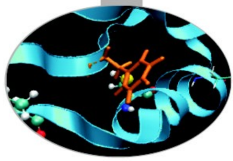
Can represent structured (relational) data or NoSQL data

- But graph is another story (see Pregel, DEX or Spark)

Examples

- *<row-id#5, (part-id:5, part-name:iphone5, supplier:Apple)>*
- *<doc-id#10, (content:<html> html text … </html>)>*
- *<akeyword, (doc-id:id1, doc-id:id2, doc-id:id10)>*

Data Partitioning

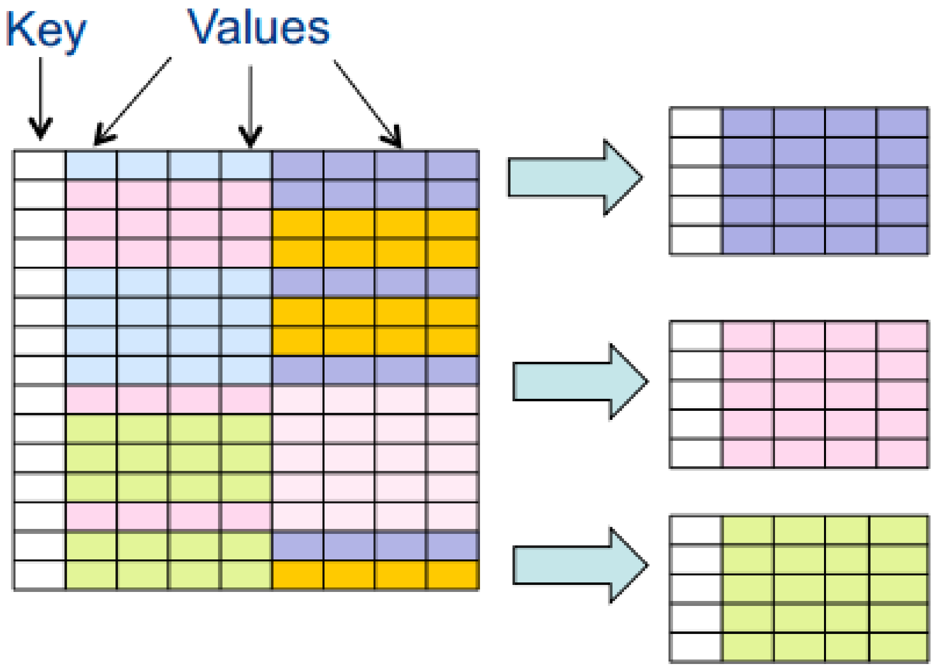


Vertical partitioning

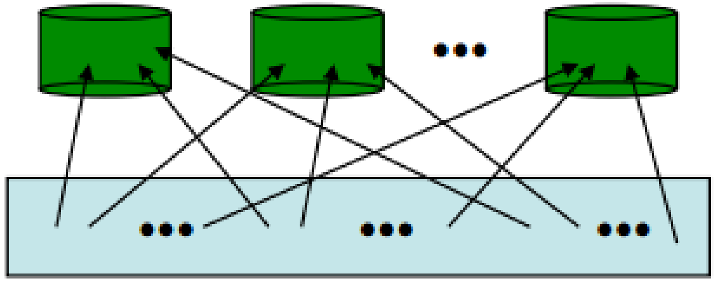
- Basis for column stores (e.g. MonetDB): efficient for OLAP queries
- Easy to compress, e.g. using Bloom filters

Horizontal partitioning (sharding)

- Shards can be stored (and replicated) at different nodes

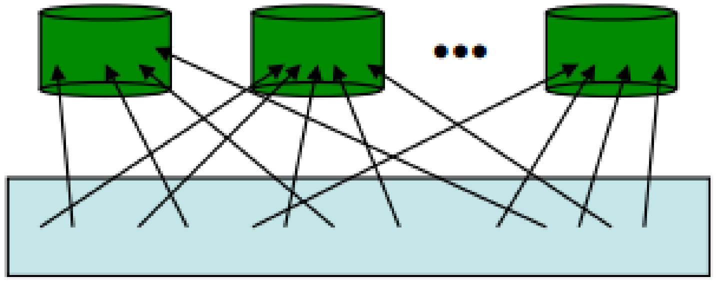


Sharding Schemes



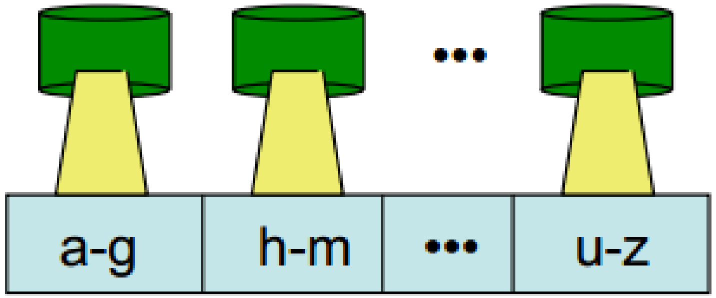
Round-Robin

- *i*th row to node (*i mod n*)
- perfect balancing
- but full scan only



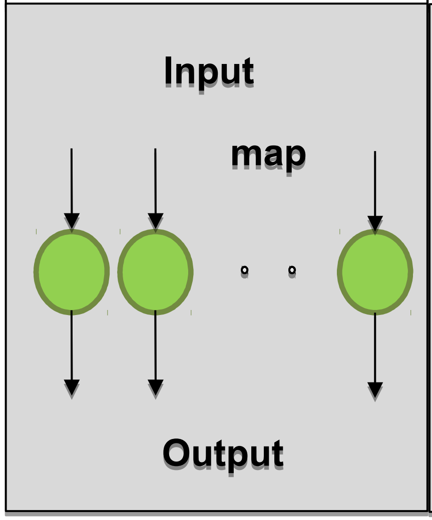
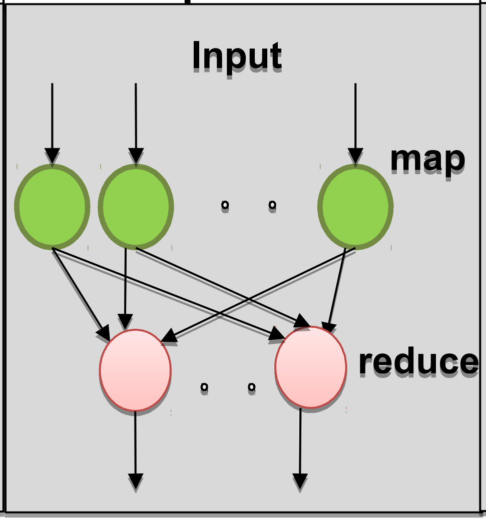
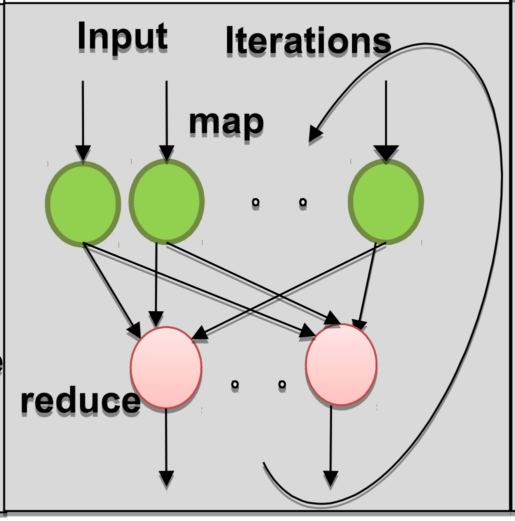
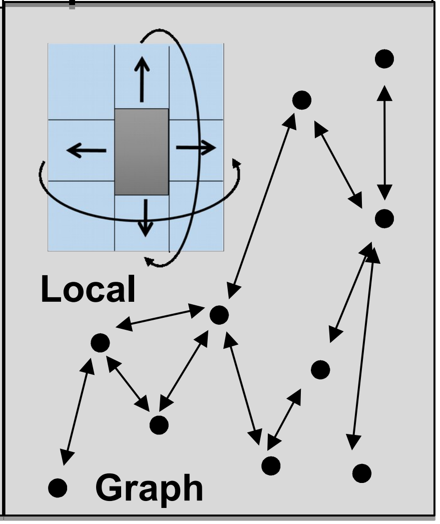
Hashing

- (k,v) to node h(k)
- exact-match queries
- but problem with skew

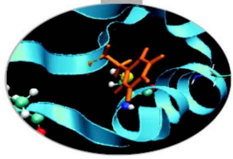


Range

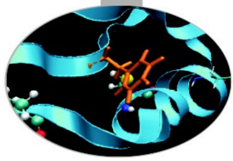
- (k,v) to node that holds k's interval
- exact-match and range queries
- deals with skew

(1) Map Only	(2) Classic MapReduce	(3) Iterative Map Reduce or Map-Collective	(4) Point to Point or Map-Communication
			
BLAST Analysis Local Machine Learning Pleasingly Parallel	High Energy Physics (HEP) Histograms Distributed search Recommender Engines	Expectation maximization Clustering e.g. K-means Linear Algebra, PageRank	Classic MPI PDE Solvers and Particle Dynamics Graph Problems
MapReduce and Iterative Extensions (Spark, Twister)			MPI, Giraph
Integrated Systems such as Hadoop + Harp with Compute and Communication model separated			

Courtesy of Prof. Geoffrey Charles Fox – Indiana University



The End



HPC vs HTC

High Performance (Capability)

High Throughput (Capacity)

Fine-grained Applications
- Many-node
- Few concurrent runs
- High interconnect use

Course-grained Applications
- Single-node
- Many concurrent runs
- No interconnect use