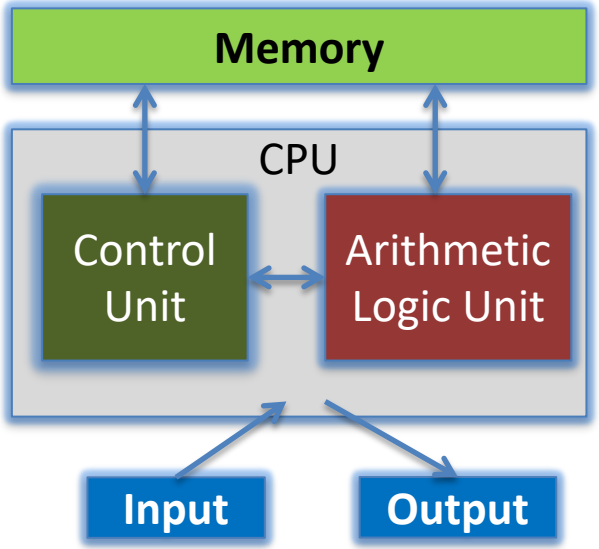
ICHEC
Irish Centre for High-End Computing

The Fundamentals of Computer & HPC Architectures

The von Neumann Architecture

- Hungarian-American mathematician John von Neumann proposed the design in 1945
  - Part of the proposal of computer EDVAC, which followed ENIAC
- Four main components:
  - Memory
  - Control Unit
  - Arithmetic Logic Unit
  - Input/Output

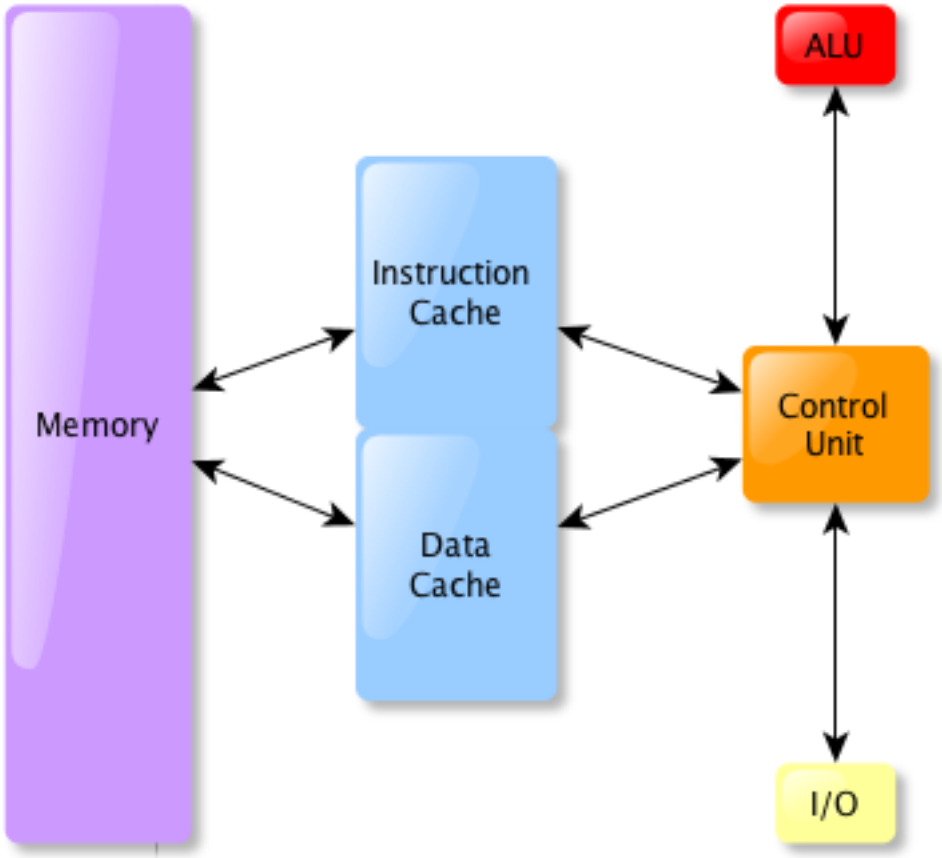


Harvard Architecture

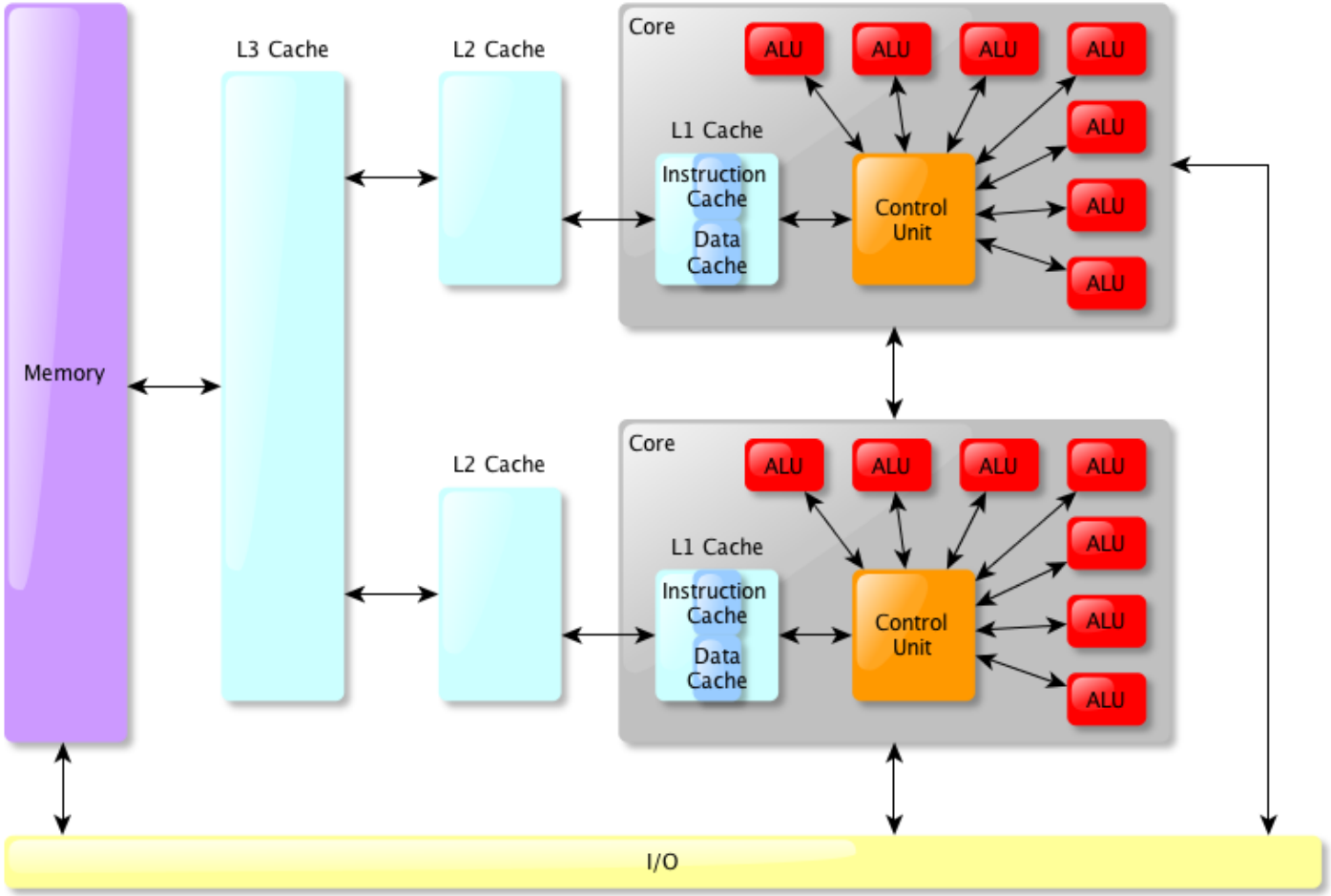
- Instruction read and data memory access can be performed simultaneously
- Different address spaces for program and data memories

Modified Harvard Architecture

- Introduction of a caching system: smaller but much faster memory closer to the CPU (short access latency)
- Mix of both previous archs: Harvard arch to access caches; Von Neumann to access memory



Recent architecture

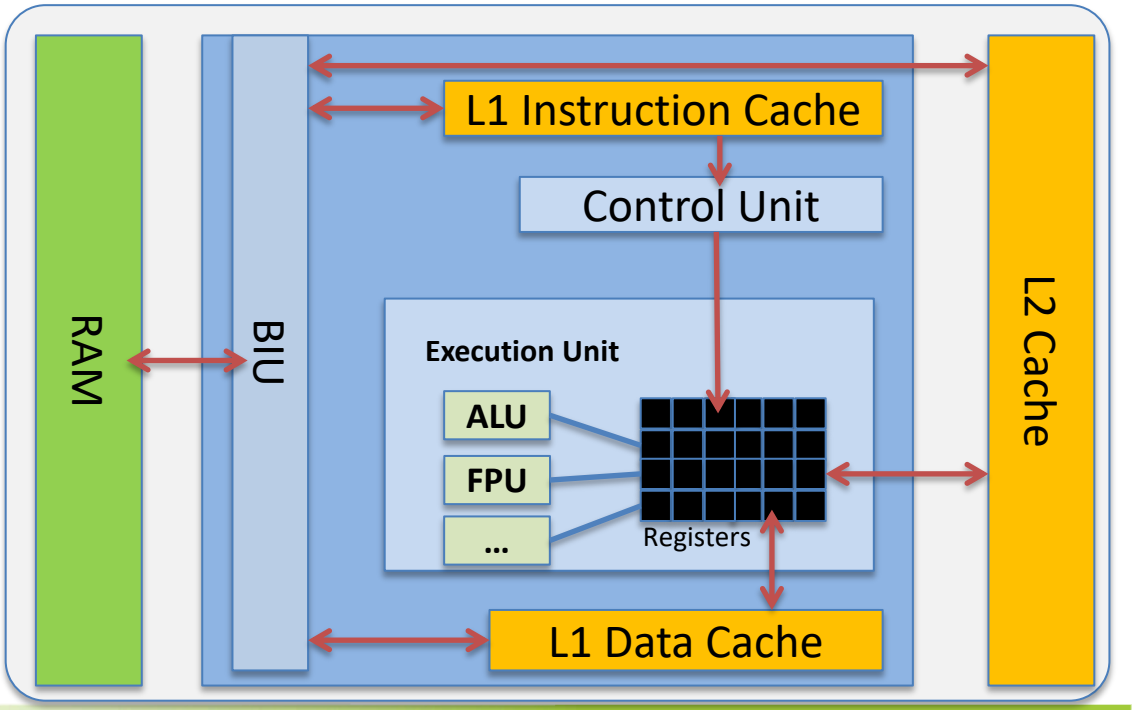


Basic Concepts

- PROCESS: is a task being run by a computer, often simultaneously with many other tasks each taking turns on a single central processing unit;

- CPU (Central Processing Unit): single- or multi-core chips that can handle multiple processes;

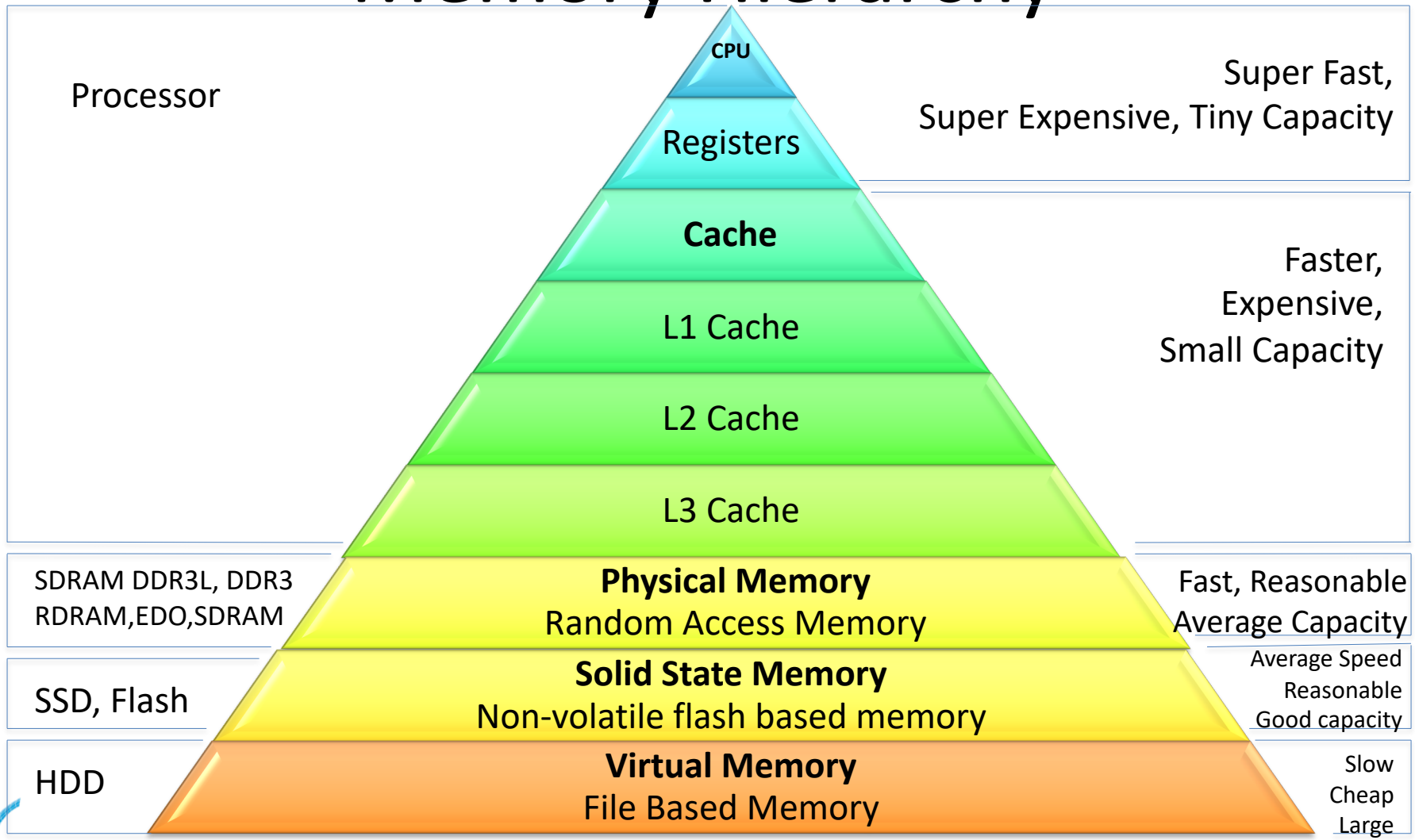
  - CU: Schedules the order of instruction execution
  - ALU: Performs arithmetic and logical calculations
  - FPU: Performs operations on floating point numbers; available in many modern processors



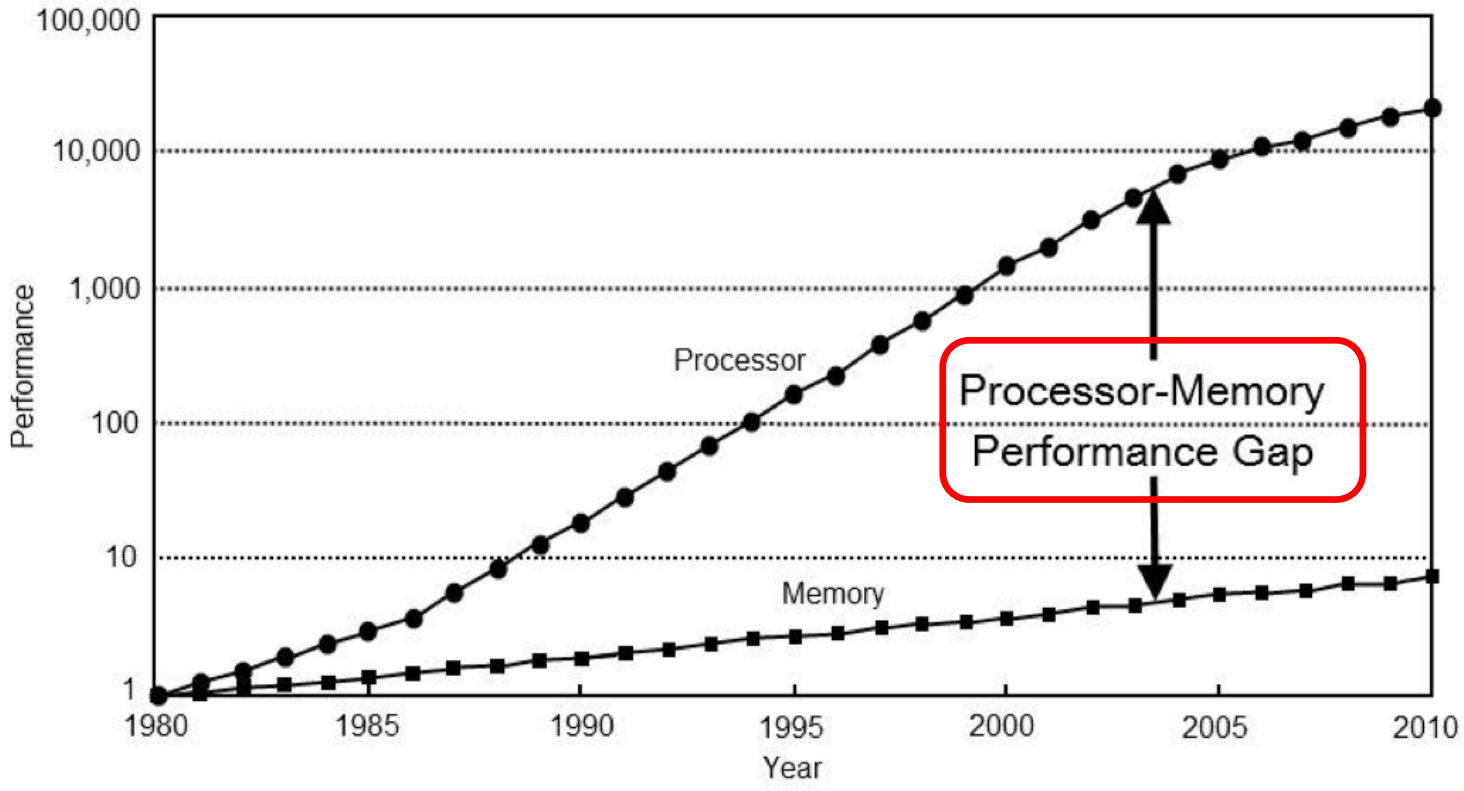
Basic Concepts

- REGISTER: small amount of storage where data resides that is being used right now;
- CACHE: small area of fast memory where data resides when it is about to be used and/or has been used recently;
- MAIN MEMORY (also called RAM “Random Access Memory”): where data resides when it is being used by a program that is currently running. It may be volatile or not.

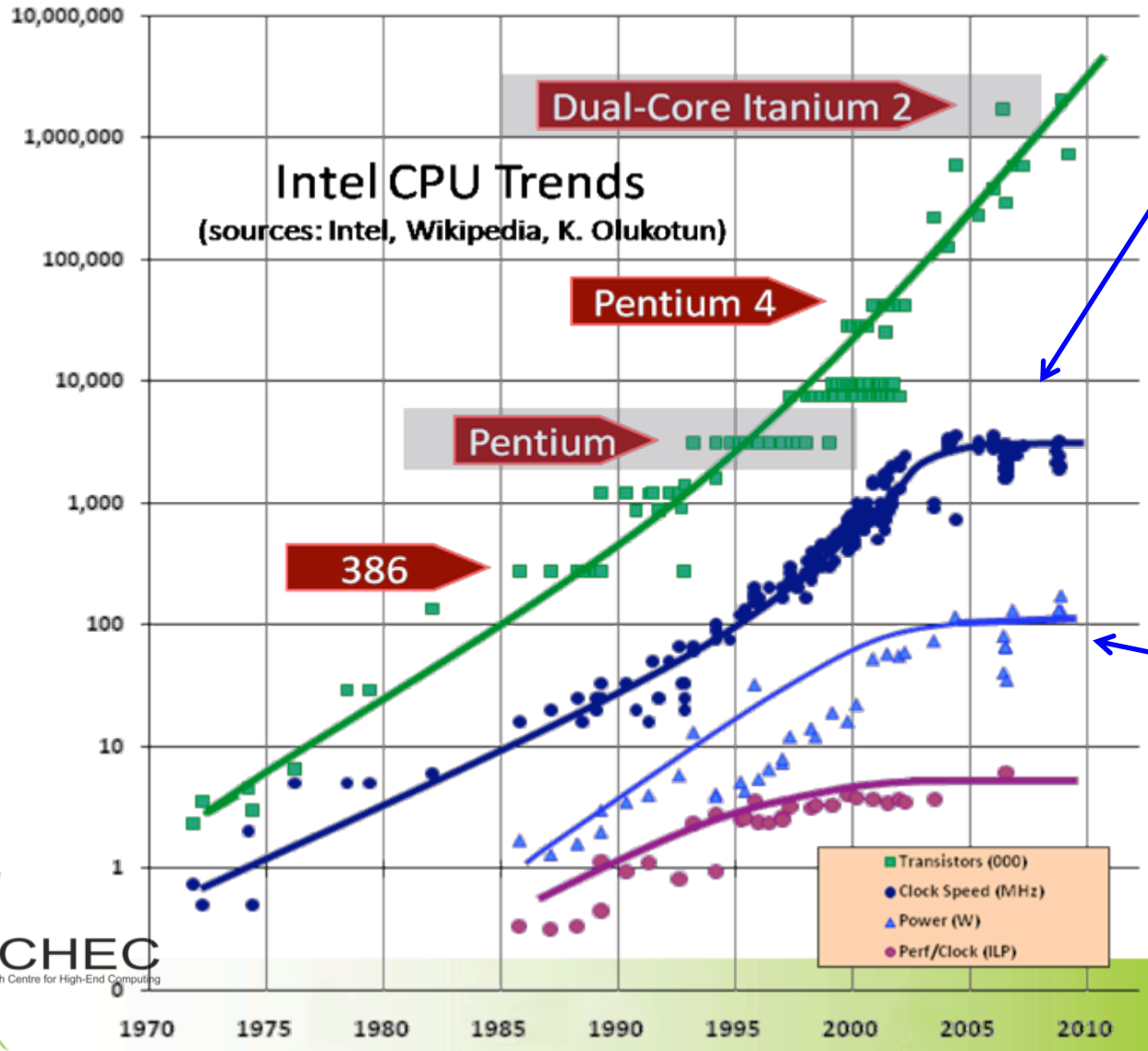
Memory Hierarchy



CPU vs Memory performance evolution



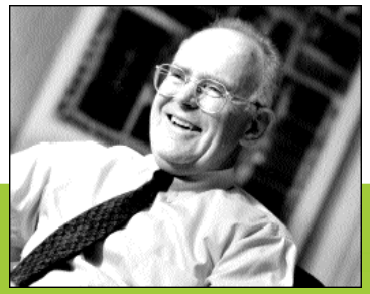
The Free Lunch is over - (Moore's Law)



"The Free Lunch is over":

- Processor clock-speeds are not getting any faster. Will remain ≤ 3 GHz
- Improved future performance must come from finding and exploiting *parallelism*.

Holding down power consumption is the new imperative.

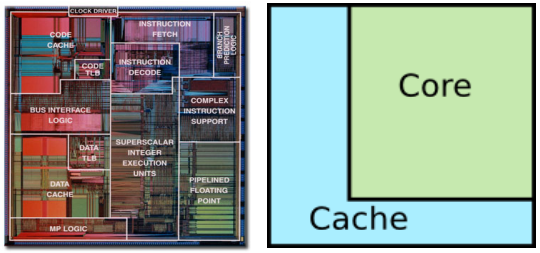


ICHEC
Irish Centre for High-End Computing

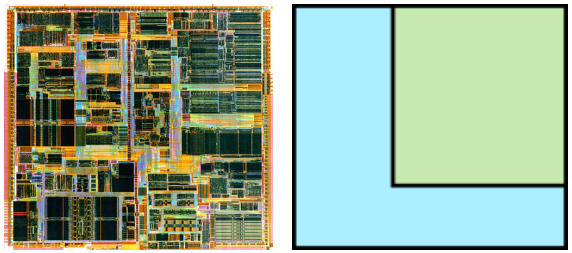
From single- to multi-core

how and why CPU evolved

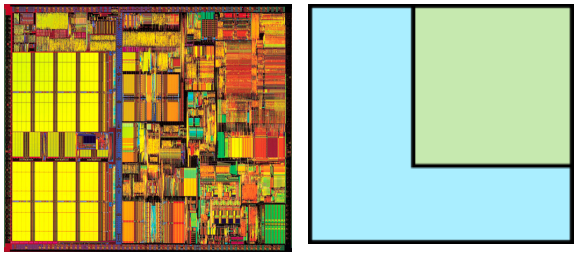
Pentium I – 1993, 60MHz



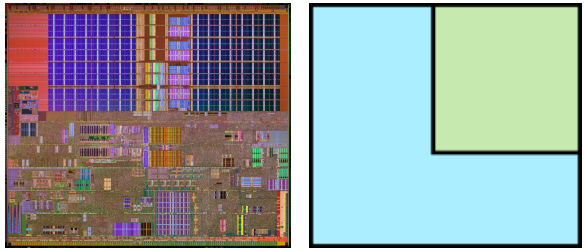
Pentium II – 1997, 233 MHz



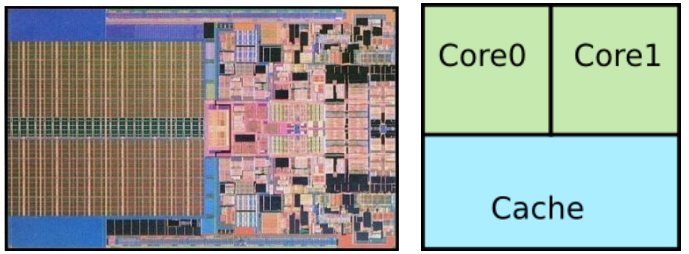
Pentium III – 1999, 500 MHz



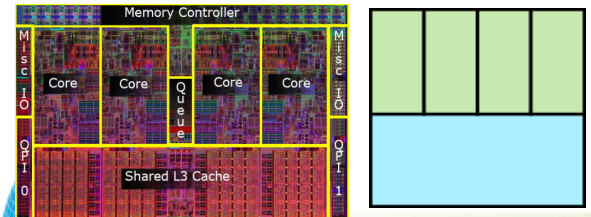
Pentium IV - 2000, 1.4 GHz



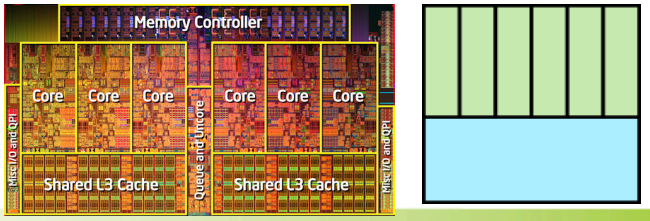
Pentium D – 2005



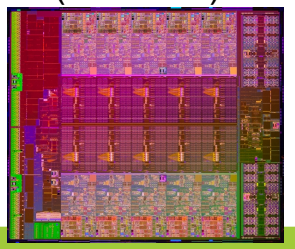
Core i7- 9xx (4 cores)



Westmere (6 cores)



Intel Ivy Bridge-EP (12 cores)

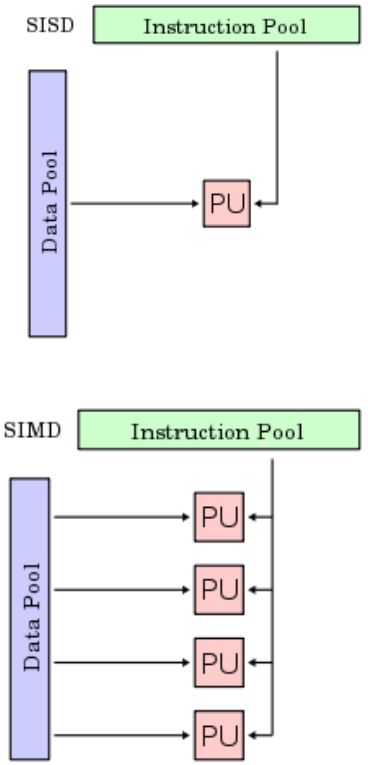


HPC Architectures

FLYNN'S TAXONOMY:

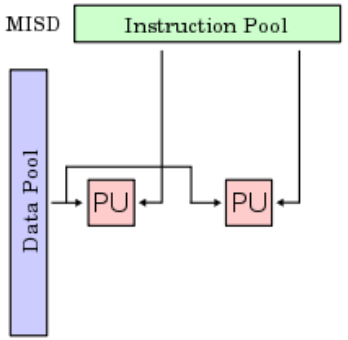
- Single Instruction Single Data (SISD): at any one time only a single instruction is executed, operating on a single data item.

- Single Instruction Multiple Data (SIMD): multiple processors, each operating on its own data item, but they are all executing the same instruction on that data item

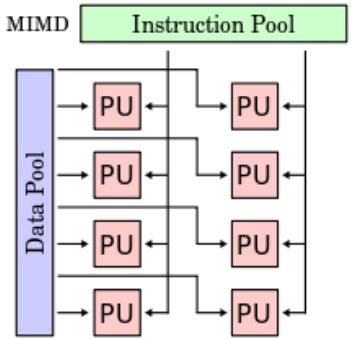


HPC Architectures

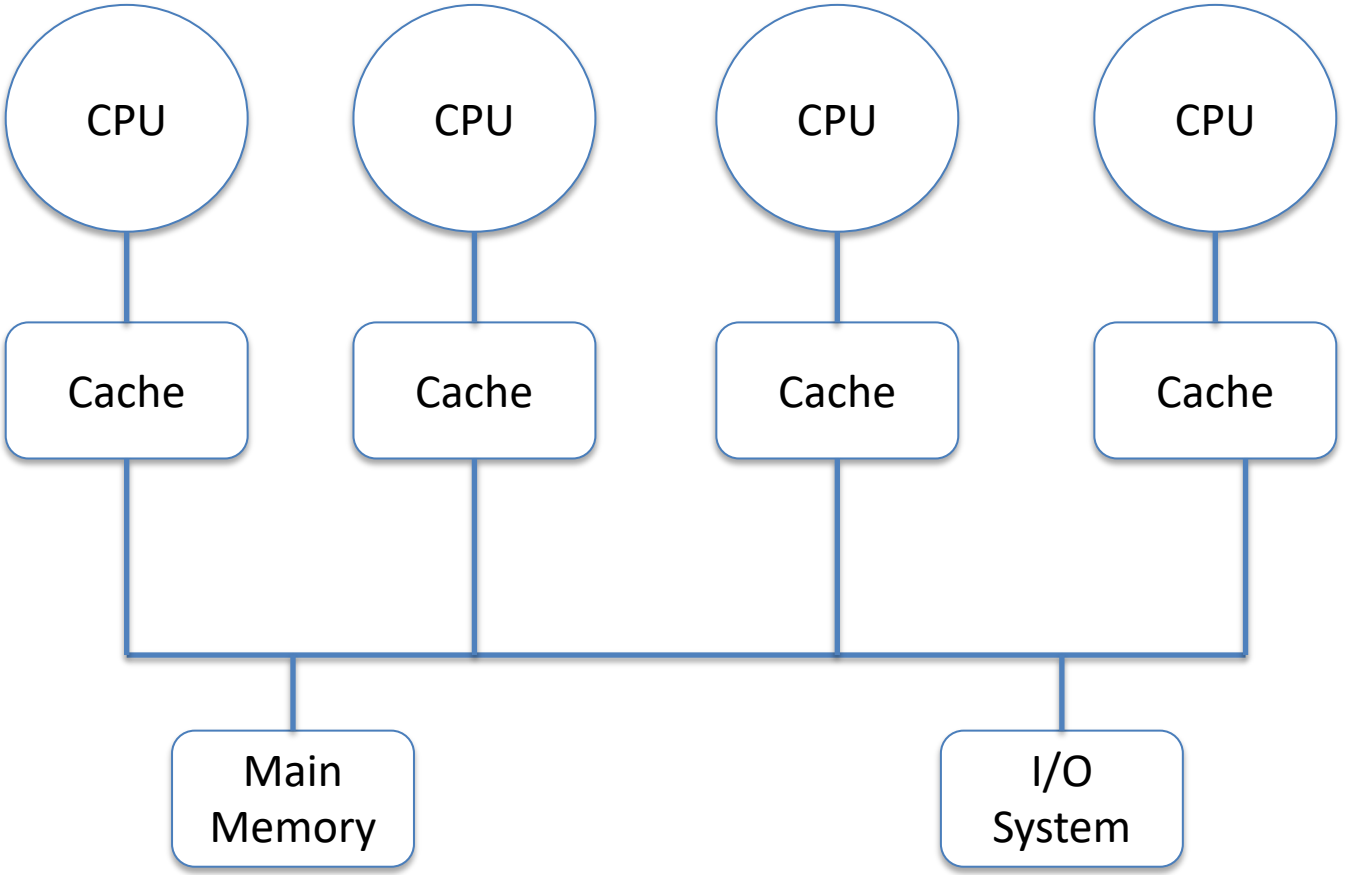
- Multiple Instructions Single Data (MISD): All processors do different operations on the same data



- Multiple Instructions Multiple Data (MIMD): multiple CPUs operate on multiple data items, each executing independent instructions
  - Shared Memory
  - Distributed Memory



Shared Memory



Shared Memory

- UMA (Uniform Memory Access), SMP (Symmetric Multi-Processing)
  - The latency to access any address in the logical memory space is the same for each CPU.
- NUMA (Non-Uniform Memory Access)
  - The latency to access any address in the logical memory space is determined by the physical distance from the CPU.
- Cache-Coherency (cc)
  - To ensure cache consistency (i.e. local cache has the most up-to-date copy of a shared memory resource), cache-coherency protocols are implemented on modern systems.

Shared Memory

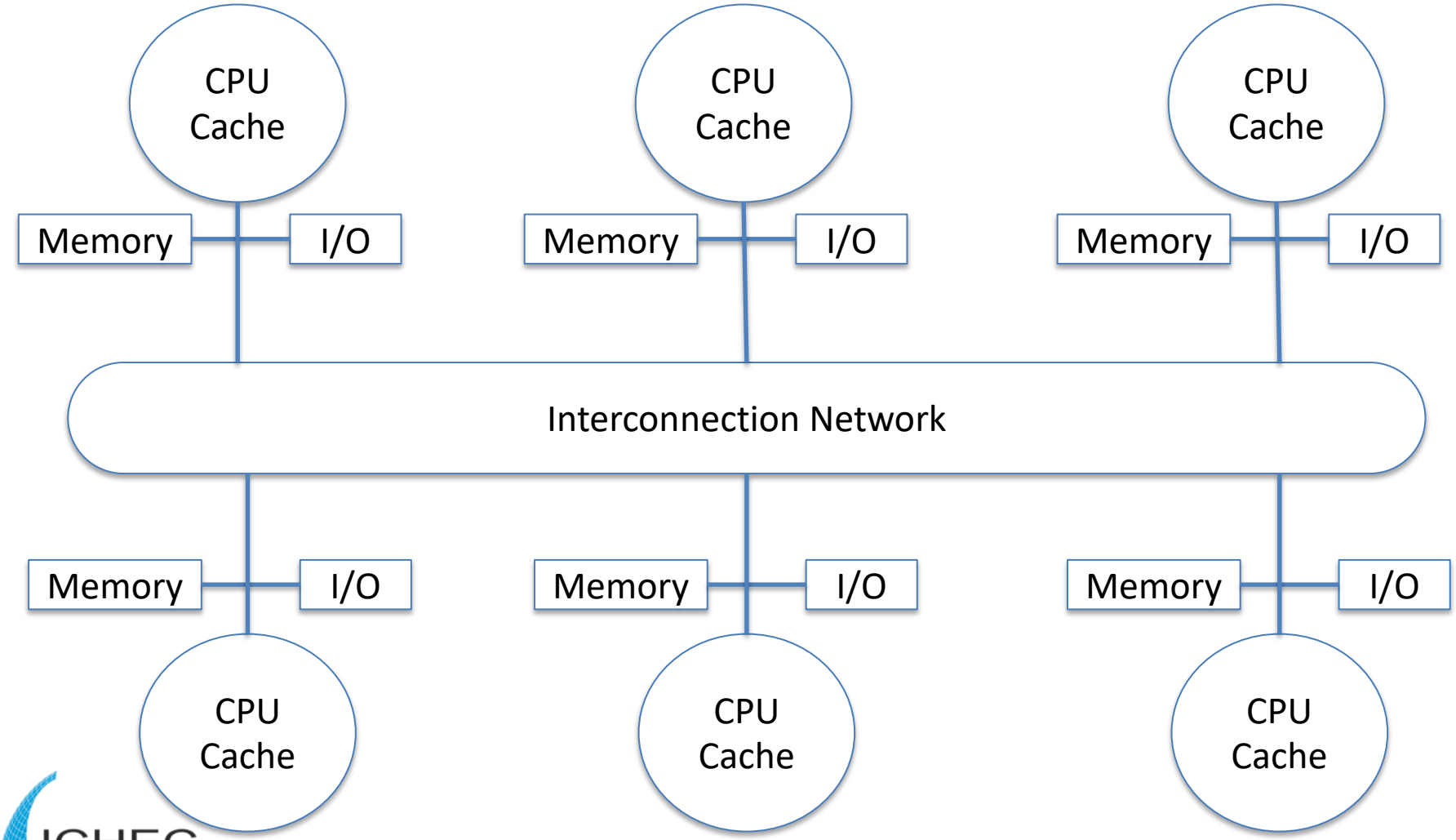
ADVANTAGES:

- Global address space provides a user-friendly programming perspective to memory (such as with the OpenMP API)
- Data sharing between tasks is both fast and uniform due to the proximity of memory to the CPUs

DISADVANTAGES:

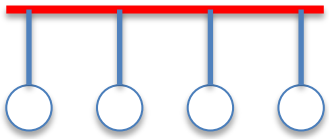
- Need for cache-coherency
- Lack of scalability between memory and CPUs. Adding more CPUs can geometrically increase traffic on the shared memory-CPU path, and for cache coherent systems, geometrically increase traffic associated with cache/memory management.
- Programmer responsibility for synchronization constructs that ensure "correct" access of global memory (i.e. prevent race conditions)

Distributed Memory



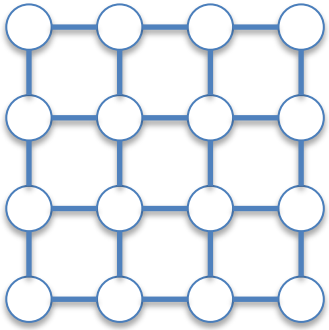
Distributed Memory

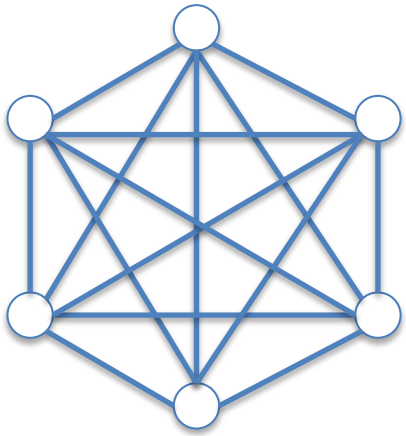
bus



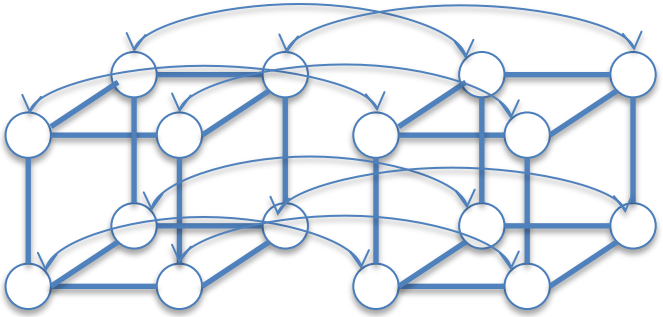
line

2D mesh





fully connected



hypercube

The choice of the interconnection has a **strong** impacts of the communication performance

Topology	Degree	Diameter	Avg. dist.	Bisection	Tot. BW (links)
bus	1	1	1	1	1
line	2	N-1	N/2	1	N-1
2D mesh	4	$2*(N^{1/2}-1)$	$2N^{1/2}/3$	N	2N-2
hypercube 2-cube	$\log_2 N$	$\log_2 N$	$\log_2 N/2$	N/2	$N/(2 \log_2 N)$
fully Connected	N-1	1	1	$N^2/4$	N(N-1)/2

Distributed Memory

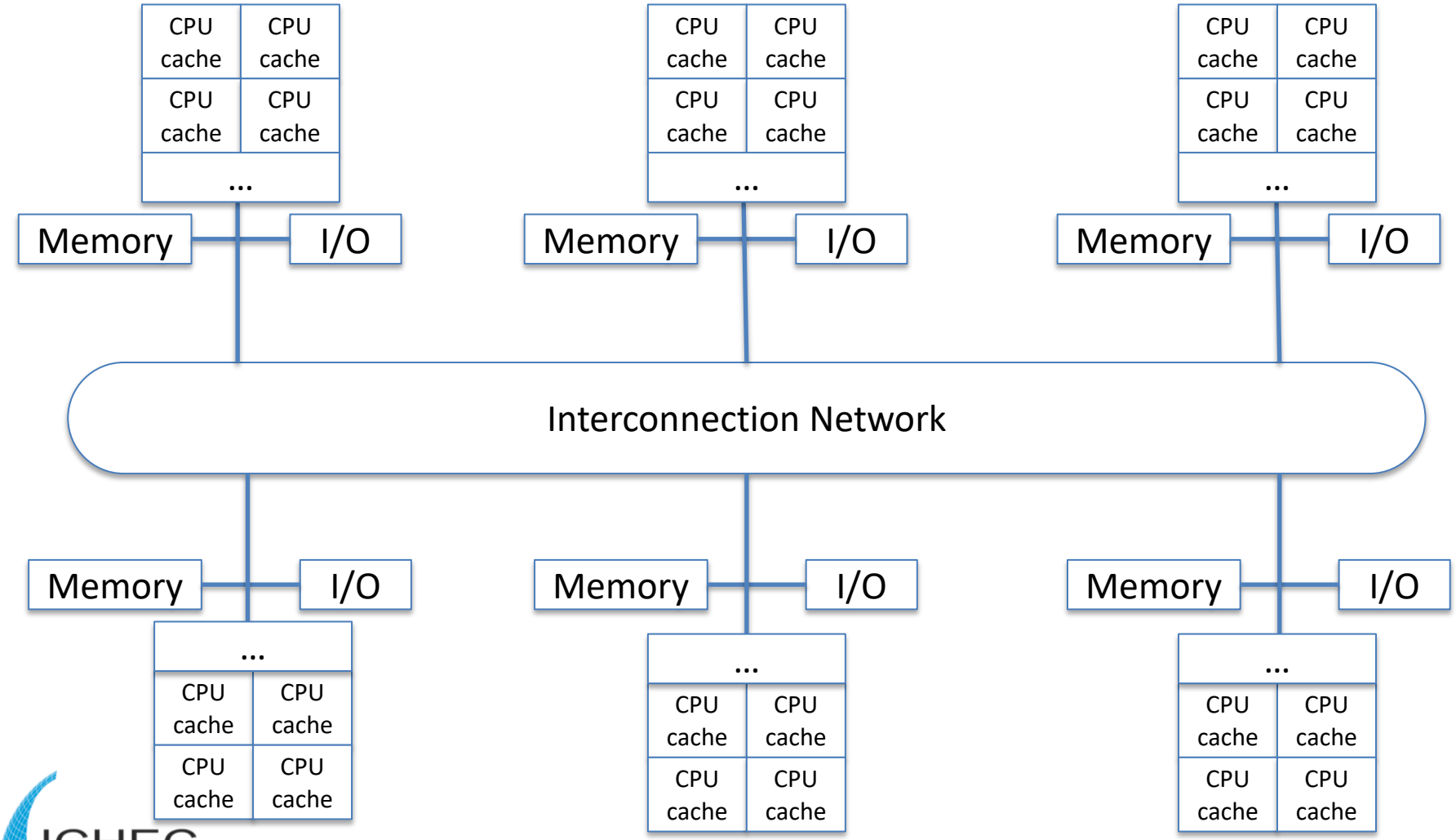
ADVANTAGES:

- Memory is scalable with number of processors. The number of processors and the size of memory increases proportionately.
- Each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherency.
- Cost effectiveness: can use commodity, off-the-shelf processors and networking.

DISADVANTAGES:

- Programmer is responsible for mapping data structures across processors
- Programmer is responsible for coordinating communication between nodes when remote data is required in a local computation (called message-passing)
- Load balancing problem.
- Currently, only low-level programming API's (such as MPI) are available to perform message-passing

Distributed-Shared Memory



Distributed-Shared Memory

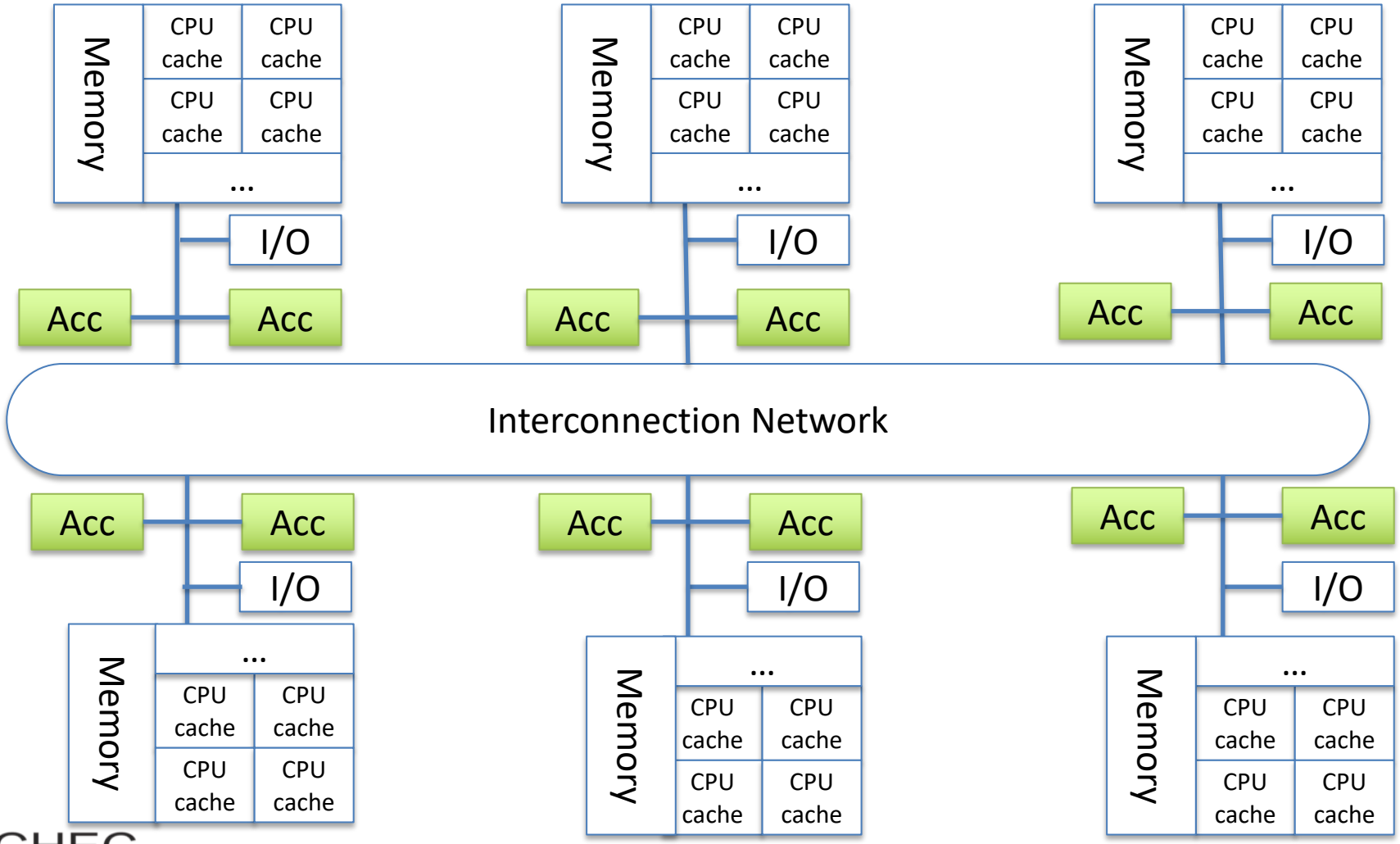
ADVANTAGES:

- Exploits the advantages of both shared and distributed-memory architectures at different levels.
- Can exploit both shared and distributed-memory programming paradigms (OpenMP and MPI) to solve difficult tasks

DISADVANTAGES:

- The same disadvantages of a pure Shared Memory system
- The same disadvantages of a simple Distributed system

Heterogeneous Systems



Heterogeneous Systems

ADVANTAGES:

- Accelerators can speed-up the calculation

DISADVANTAGES:

- Accelerators are separated from the system (like a PCI card), they do not share the memory with the host system;
- Accelerators require their own programming environment.