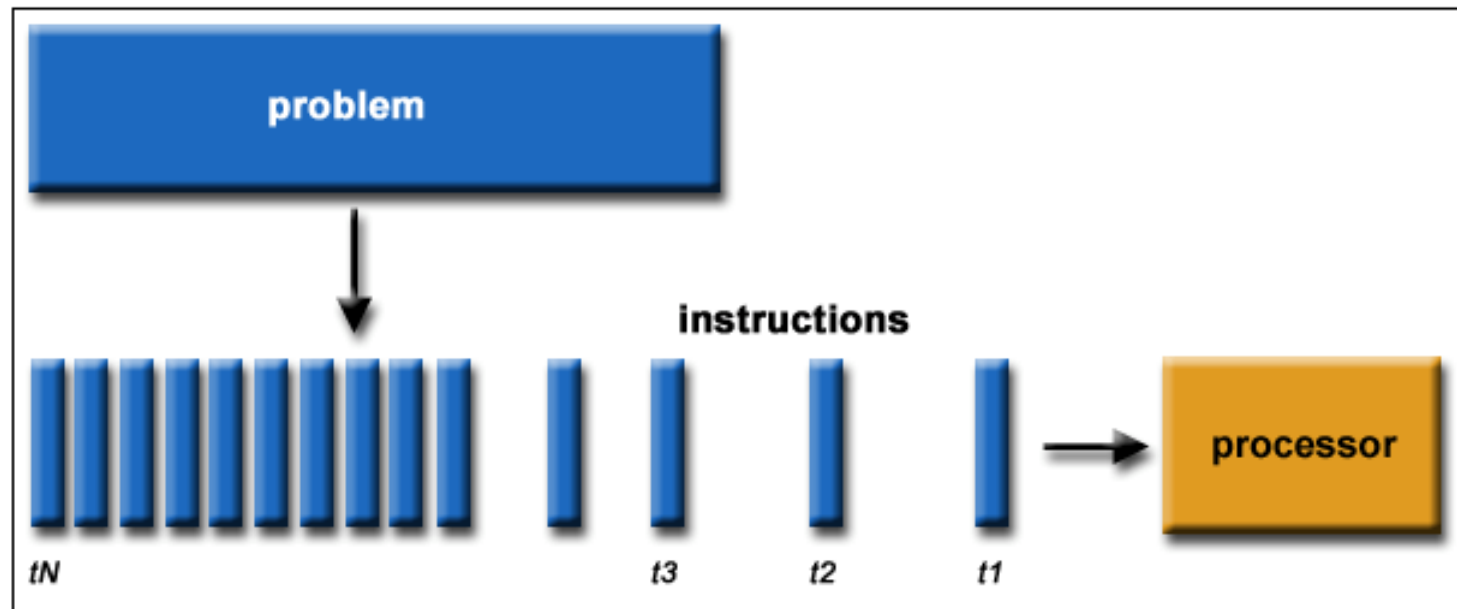


Introduction to parallel computing

Jui-Hung Hung

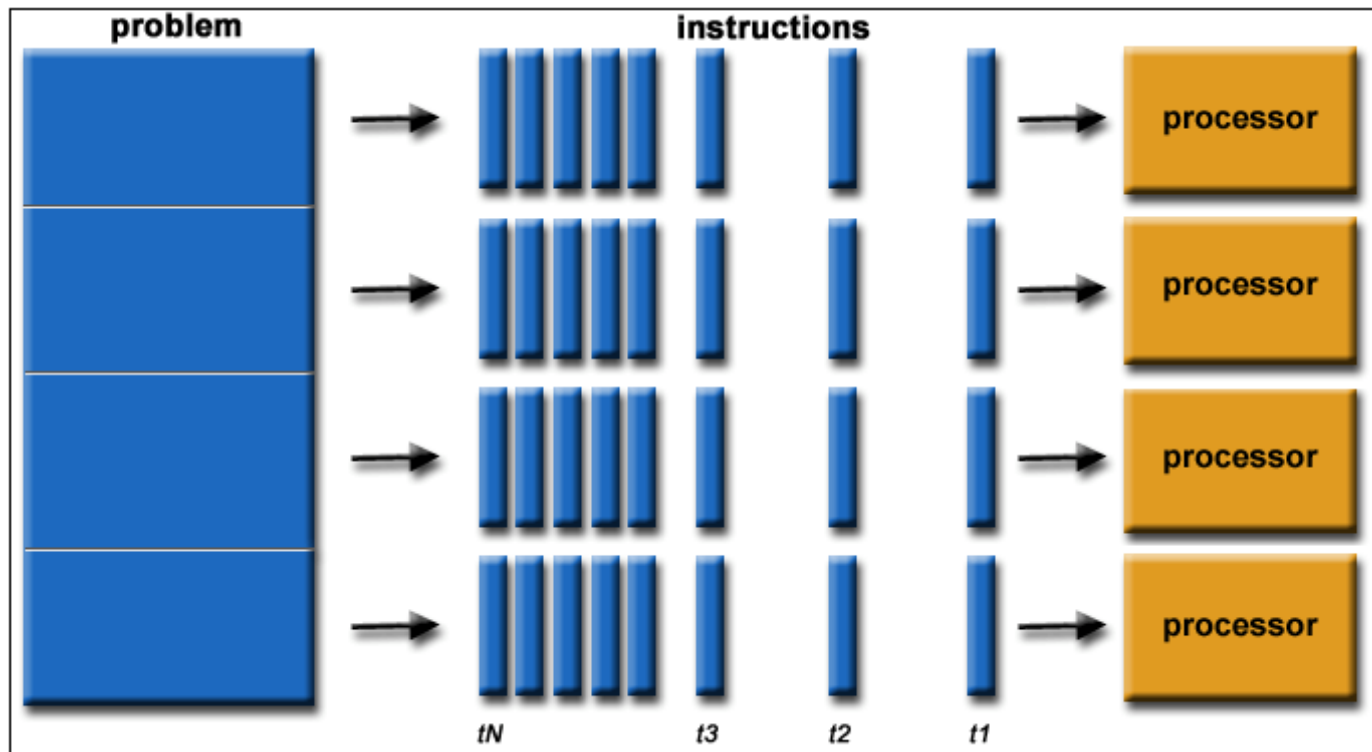
Serial computation

- Only one instruction may execute at any moment in time



Parallel Computing

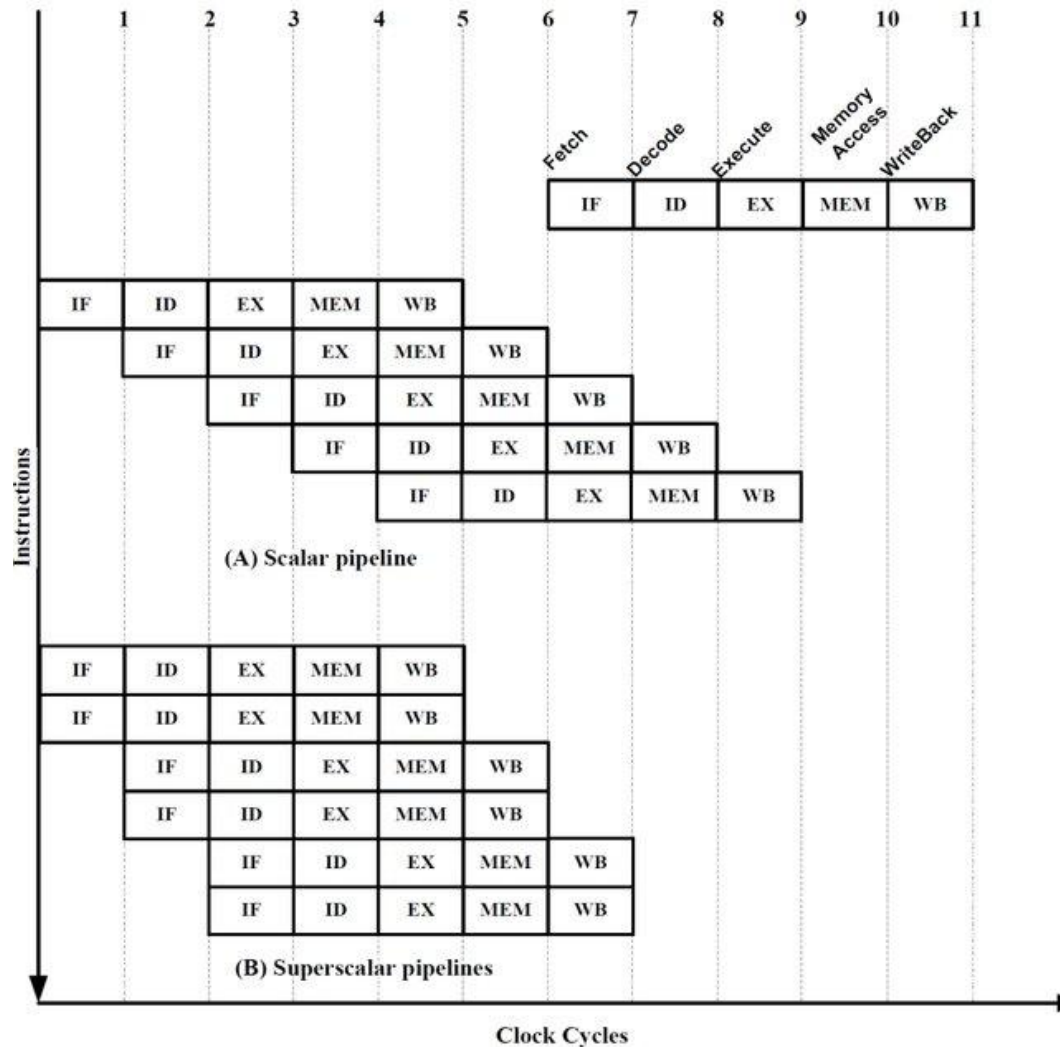
- A problem is broken into discrete parts that can be solved concurrently



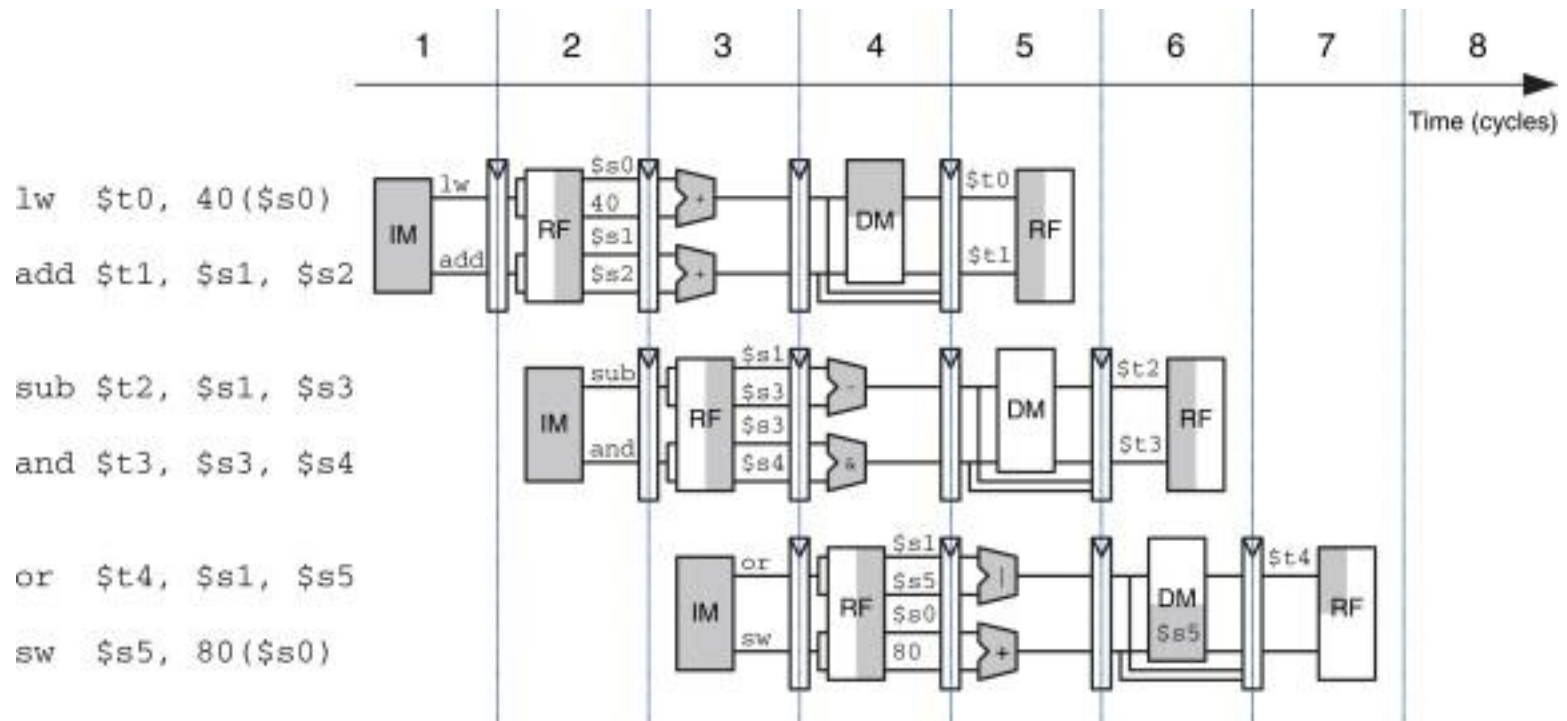
How to do parallel computing

- Hardware support
 - SIMD/GPGPU
 - A single computer with multiple processors/cores
 - An arbitrary number of such computers connected by a network—cluster/grid/supercomputer
- OS support
 - Multithreading/multiprocessing/message passing
- You
 - Know how to write parallel software

Scalar v.s superscalar

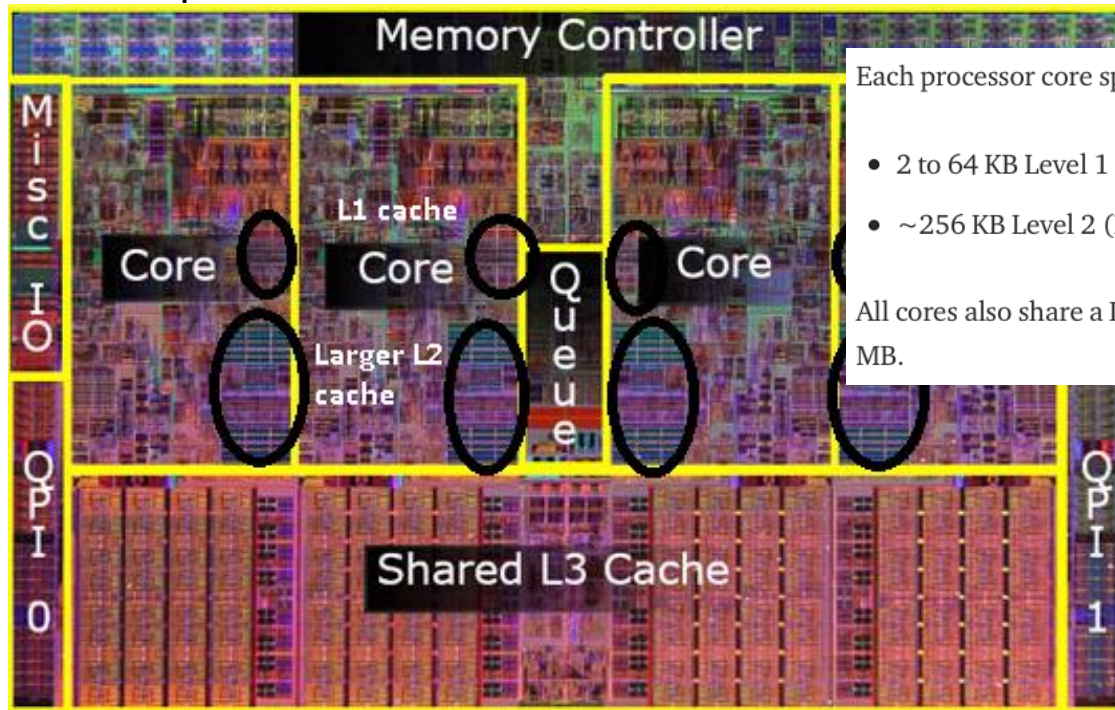


Superscalar = hyperthreading /
SMP(symmetric multiprocessing.)



All modern computers are parallel

- Multiple functional units (L1 cache, L2 cache, branch, prefetch, decode, floating-point, graphics processing (GPU), integer, etc.)
- Multiple execution units/cores
- Multiple hardware threads



Each processor core sports two levels of cache:

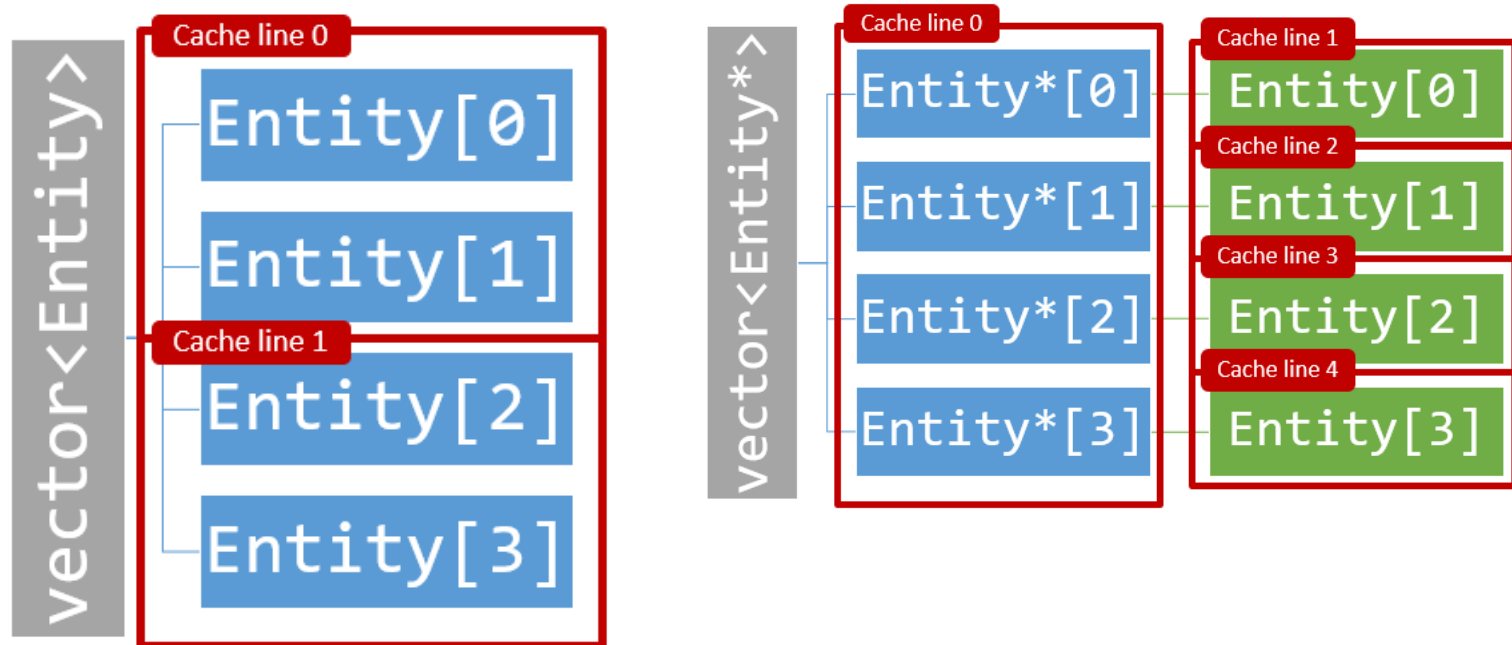
- 2 to 64 KB Level 1 (L1) cache very high speed cache
- ~256 KB Level 2 (L2) cache medium speed cache

All cores also share a Level 3 (L3) cache. The L3 cache tends to be around 8 MB.

- L1 cache access latency: **4 cycles**
- L2 cache access latency: **11 cycles**
- L3 cache access latency: **39 cycles**
- Main memory access latency: **107 cycles**

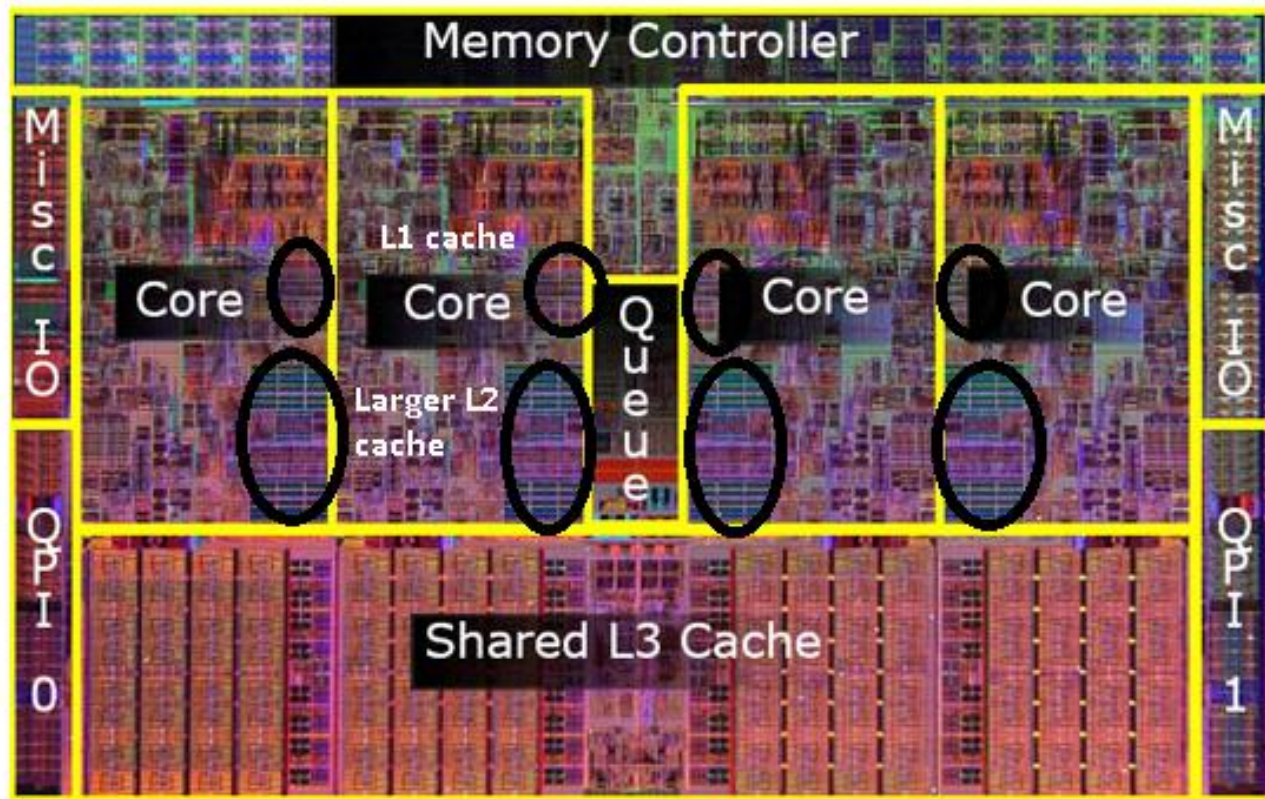
Why you should know about cache

- The memory and object model is essential



Things is not that simple

- Cache coherency induced lag and problem of memory ordering



Compiler and processor memory reordering

Source code

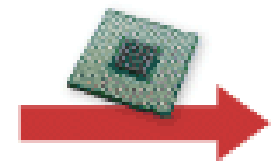
```
if (!PREACTION(gm)) {  
    void* mem;  
    size_t nb;  
    if (bytes <= MAX_SMALL_R  
        binindex_t idx;  
        binmap_t smallbits;  
        nb = (bytes < MIN_REQU  
        idx = small_index(nb);  
        smallbits = gm->smallm  
  
        if ((smallbits & 0x3U)  
            mchunkptr b, p;  
            idx += -smallbits &  
            b = smallbits & b / mem
```



compiler
reordering

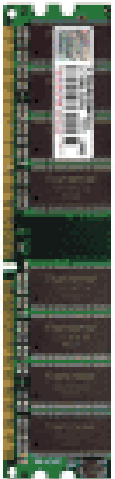
Machine code

```
cmp     esi,0F4h  
ja      dimalloc+1AEh (774  
cmp     esi,0Bh  
jae     dimalloc+20h (774  
mov     esi,10h  
jmp     dimalloc+26h (774  
add     esi,0Bh  
and     esi,0FFFFFFF9h  
mov     eax,dword ptr [_  
mov     edi,esi  
shr     edi,3  
mov     ecx,edi  
shr     eax,cl  
test    al,3  
je      dimalloc+9Ah (774
```



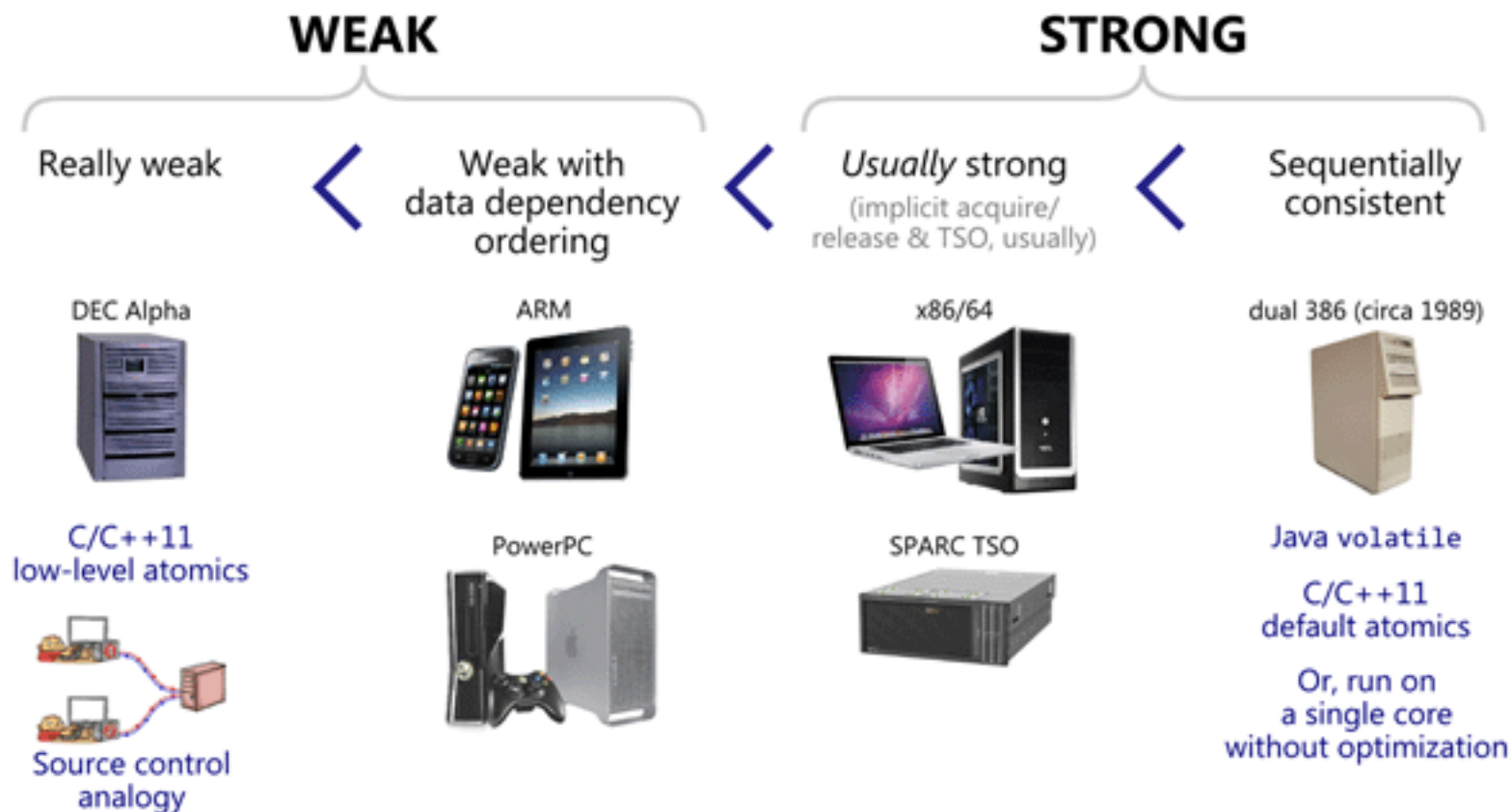
processor
reordering

Memory



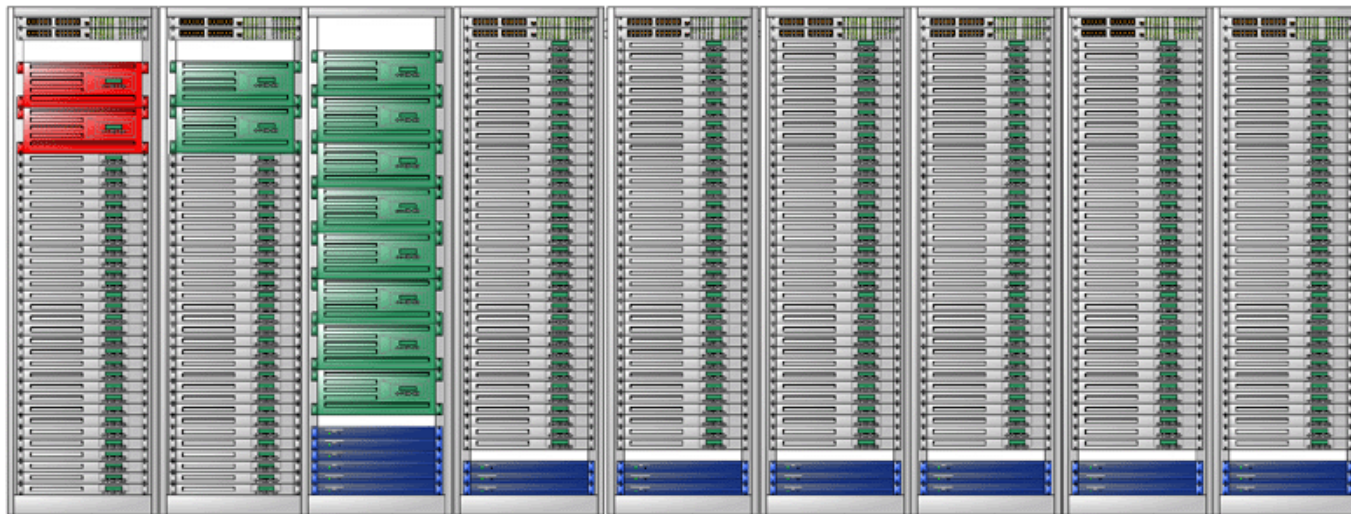
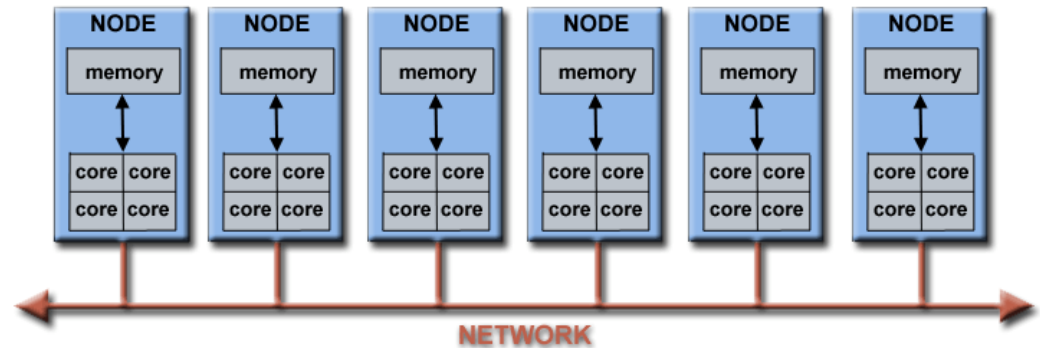
The hardware memory model matters here.




Weak & Strong Memory Order





Distributed computing

- Message passing

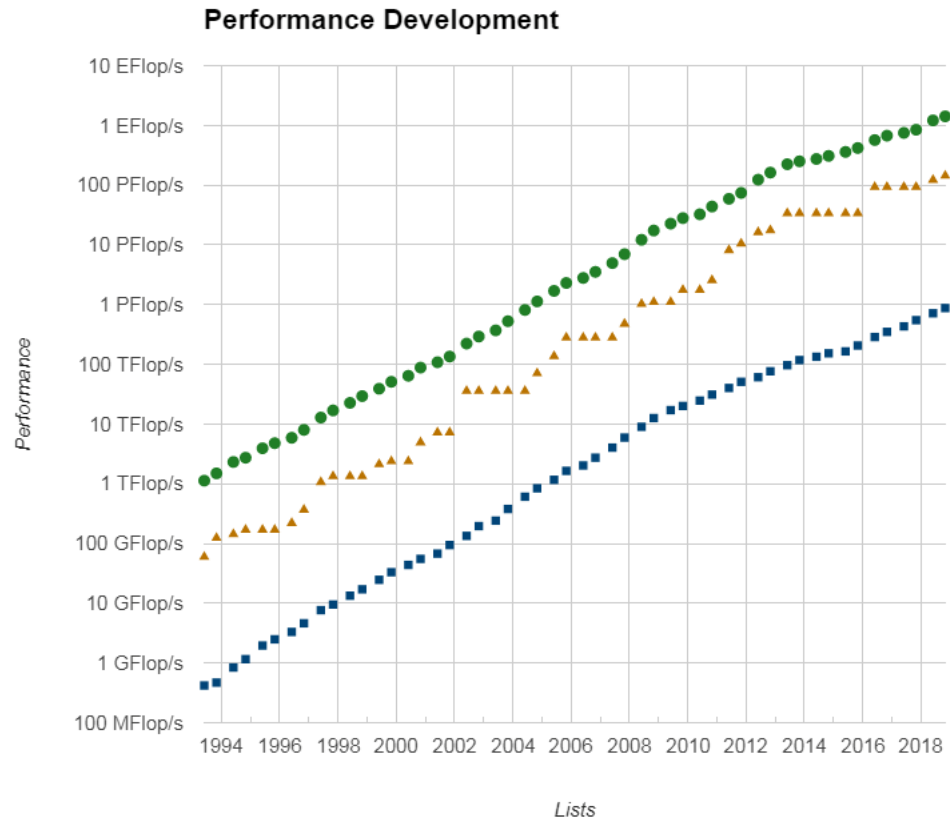


 compute node
 infiniband switch
 management hardware

 login / remote partition server node
 gateway node

The majority of the world's large parallel computers (supercomputers) are clusters of hardware

Power of supercomputing

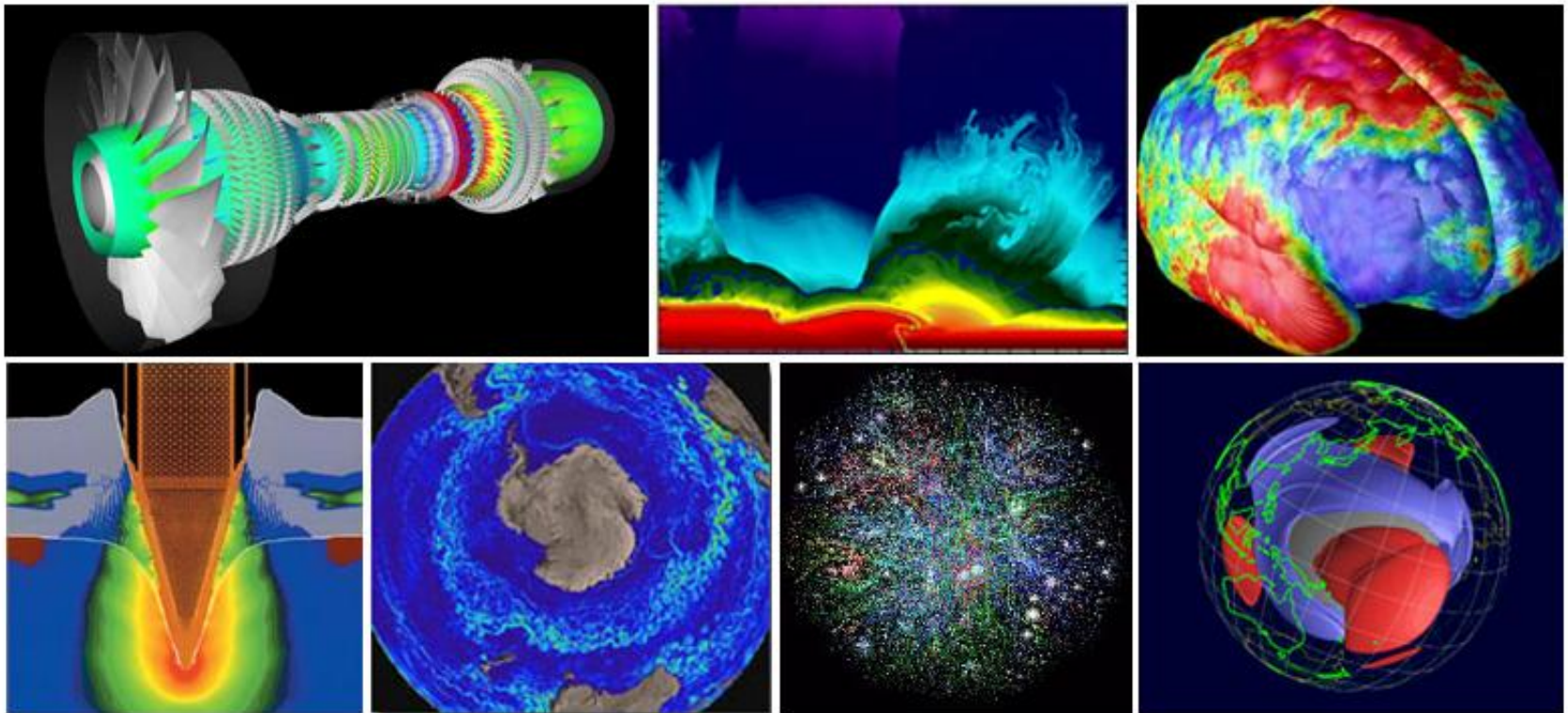


台灣杉二號（英語：Taiwania 2）是超級電腦，位於中華民國國家實驗研究院國家高速網路與計算中心（簡稱「國網中心」），是由科技部國研院國網中心結合廣達、台灣大、華碩等三大國內企業共同建造^[2]，硬體規格由9,072顆CPU（Intel Xeon Gold 6154）及2,016個GPU（NVIDIA Tesla V100 SXM2 32 GB）組成，以每秒執行9千兆次浮點運算的速度，在TOP 500於2018年底發布的世界五百大超級電腦排名中，排名第20名^[3]。

Who is Using Parallel Computing?

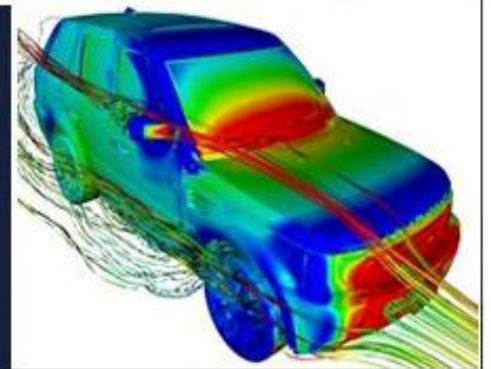
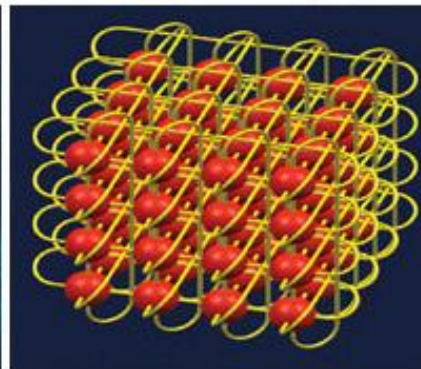
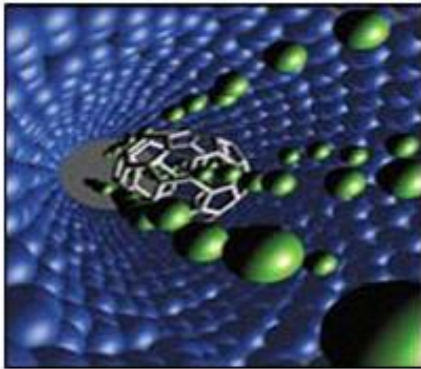
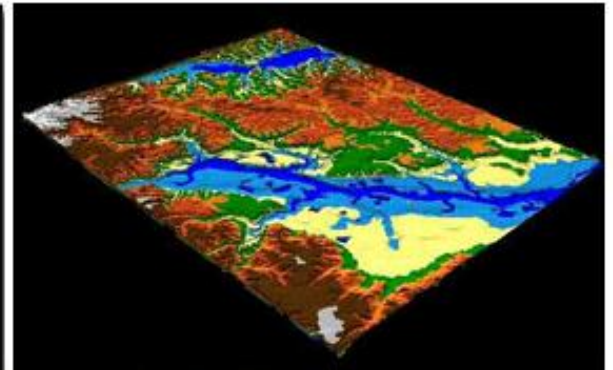
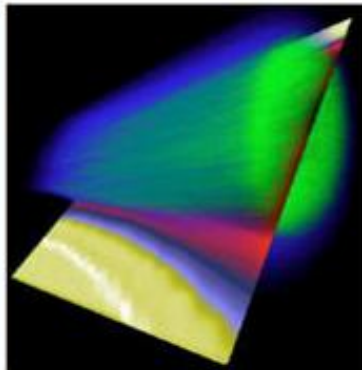
► Science and Engineering:

- Historically, parallel computing has been considered to be "the high end of computing", and has been used to model difficult problems in many areas of science and engineering:
 - Atmosphere, Earth, Environment
 - Physics - applied, nuclear, particle, condensed matter, high pressure, fusion, photonics
 - Bioscience, Biotechnology, Genetics
 - Chemistry, Molecular Sciences
 - Geology, Seismology
 - Mechanical Engineering - from prosthetics to spacecraft
 - Electrical Engineering, Circuit Design, Microelectronics
 - Computer Science, Mathematics
 - Defense, Weapons



► Industrial and Commercial:

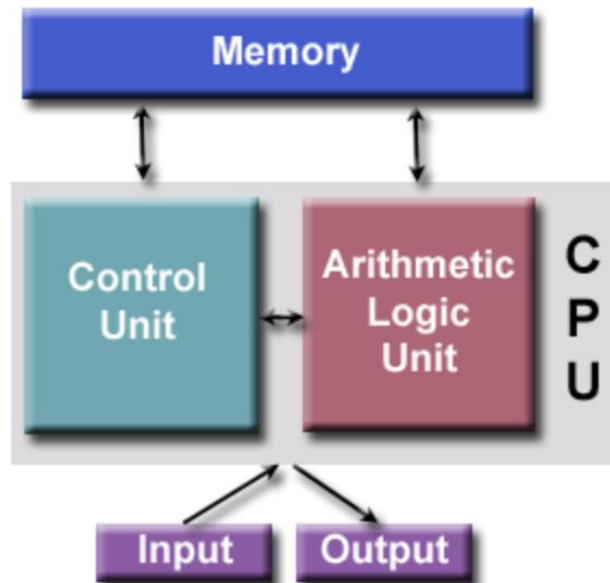
- Today, commercial applications provide an equal or greater driving force in the development of faster computers. These applications require the processing of large amounts of data in sophisticated ways. For example:
 - "Big Data", databases, data mining
 - Artificial Intelligence (AI)
 - Web search engines, web based business services
 - Medical imaging and diagnosis
 - Pharmaceutical design
 - Financial and economic modeling
 - Management of national and multi-national corporations
 - Advanced graphics and virtual reality, particularly in the entertainment industry
 - Networked video and multi-media technologies
 - Oil exploration



Concepts and terminology



- von Neumann Architecture
 - John von Neuman
 - Probably the smartest man who ever lived
 - "stored-program computer"



3 Mathematics

3.1 Set theory

3.1.1 Von Neumann Paradox

3.2 Ergodic theory

3.3 Operator theory

3.4 Measure theory

3.5 Geometry

3.6 Lattice theory

3.7 Mathematical formulation of quantum mechanics

3.7.1 Von Neumann Entropy

3.7.2 Quantum mutual information

3.7.3 Density matrix

3.7.4 Von Neumann measurement scheme

3.8 Quantum logic

3.9 Game theory

3.10 Mathematical economics

3.11 Linear programming

3.12 Mathematical statistics

3.13 Fluid dynamics

3.14 Mastery of mathematics

4 Nuclear weapons

4.1 Manhattan Project

4.2 Atomic Energy Commission

4.3 Mutual assured destruction

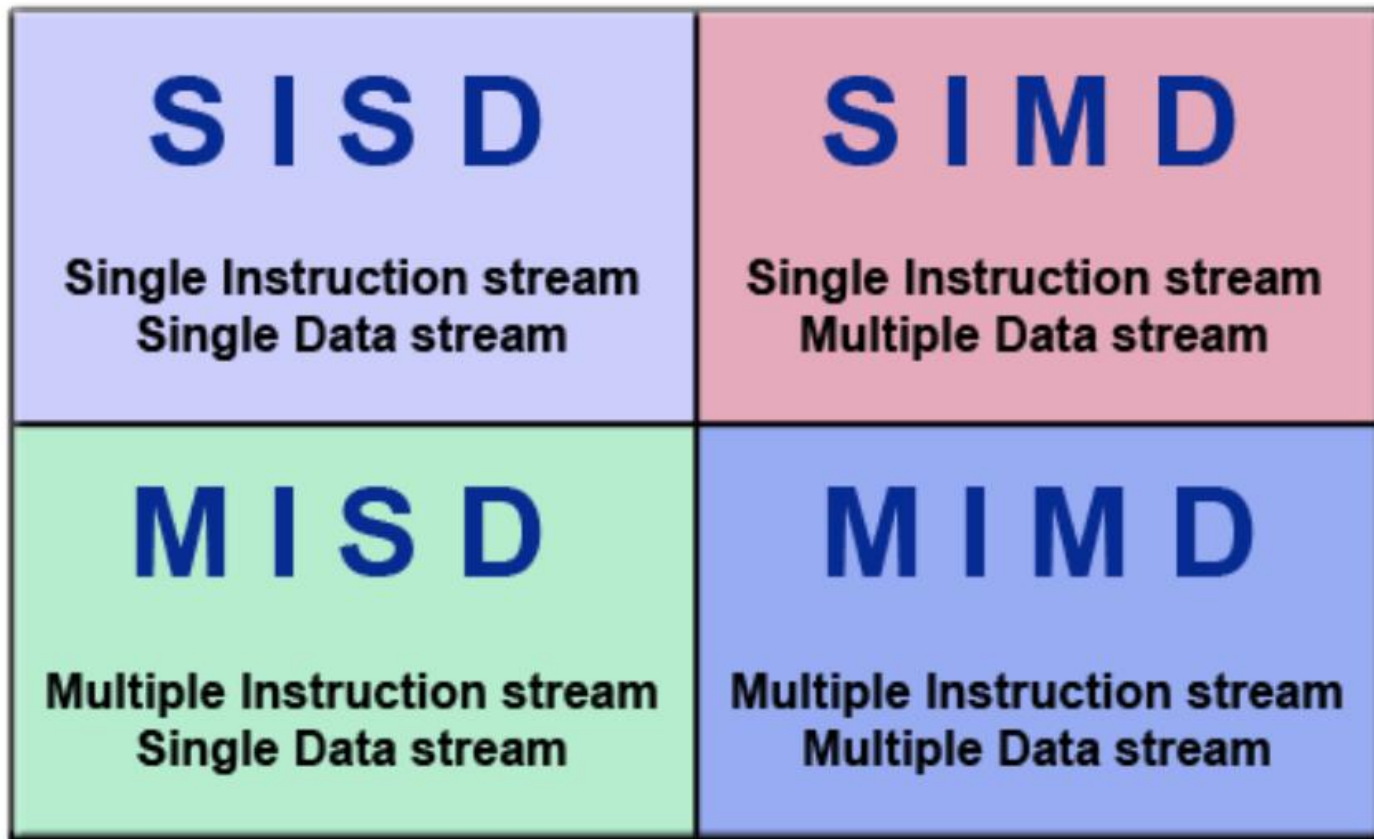
5 Computing

5.1 Cellular automata, DNA and the universal constructor

5.2 Weather systems and global warming

5.3 Technological singularity hypothesis

Flynn's Classical Taxonomy



Single Instruction, Single Data (SISD)

- A serial (non-parallel) computer
- **Single Instruction:** Only one instruction stream is being acted on by the CPU during any one clock cycle
- **Single Data:** Only one data stream is being used as input during any one clock cycle

Single Instruction, Multiple Data (SIMD)

- A type of parallel computer
- **Single Instruction:** All processing units execute the same instruction at any given clock cycle
- **Multiple Data:** Each processing unit can operate on a different data element
- Most modern computers, particularly those with graphics processor units (GPUs) employ SIMD instructions and execution units.

Multiple Instruction, Single Data (MISD)

- A type of parallel computer (not really exist)
- **Multiple Instruction:** Each processing unit operates on the data independently via separate instruction streams.
- **Single Data:** A single data stream is fed into multiple processing units.

Multiple Instruction, Multiple Data (MIMD)

- A type of parallel computer
- **Multiple Instruction:** Every processor may be executing a different instruction stream
- **Multiple Data:** Every processor may be working with a different data stream
- Most current supercomputers, clusters and grids, multi-processor SMP computers, multi-core PCs
- MIMD architectures also include SIMD execution sub-components

Speedup

- Observed speedup of a code which has been parallelized, defined as

$$\frac{\text{wall-clock time of serial execution}}{\text{wall-clock time of parallel execution}}$$

- One of the simplest and most widely used indicators for a parallel program's performance

Overhead

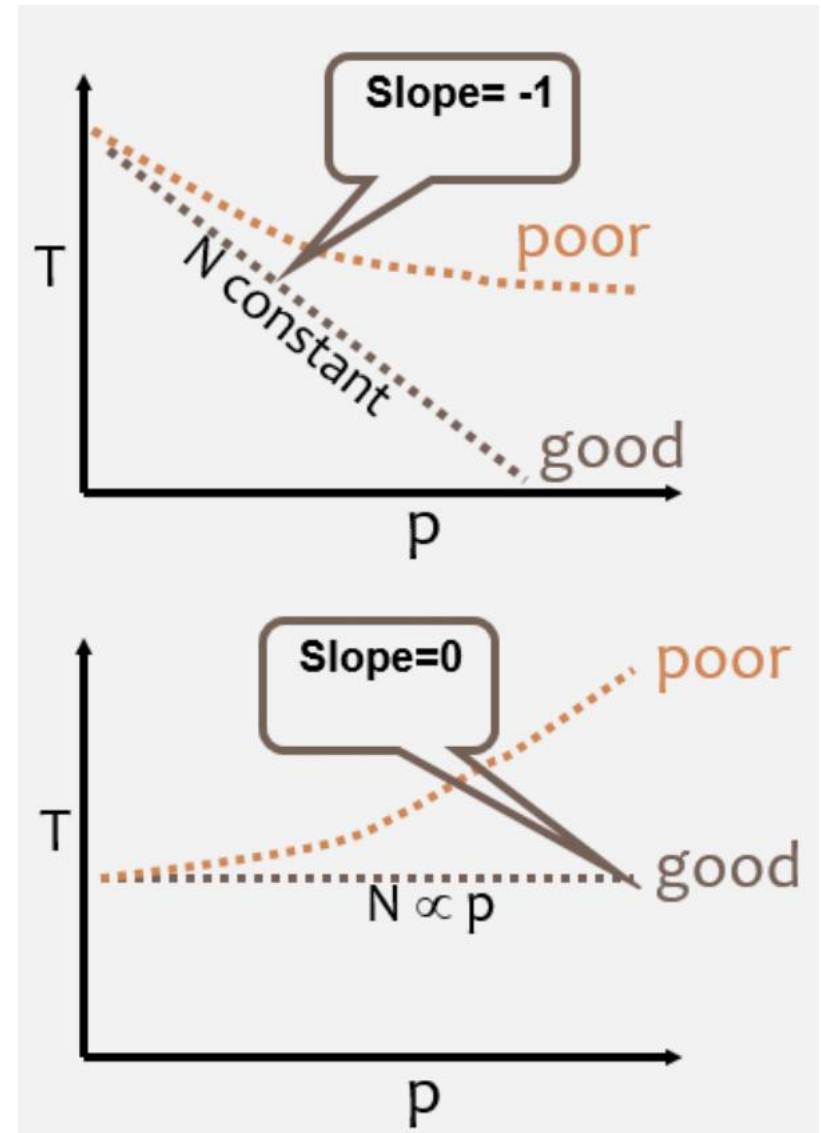
- The amount of time required to coordinate parallel tasks, as opposed to doing useful work.
- Parallel overhead can include factors such as:
 - Task start-up time
 - Synchronizations
 - Data communications
 - Software overhead imposed by parallel languages, libraries, operating system, etc.
 - Task termination time

Embarrassingly Parallel

- Solving many similar, but independent tasks simultaneously; little to no need for coordination between the tasks.
 - Linear speedup (the best you can get)
- The perfect solution (if exists) is embarrassingly easy

Scalability

- **Strong scaling:**
 - The total problem size stays fixed as more processors are added.
 - run the same problem size faster
- **Weak scaling:**
 - The problem size *per processor* stays fixed as more processors are added.
 - run larger problem in same amount of time
 - How much I can increase the size of my problem such that the execution time is the same without parallelism



Amdahl's Law

- For strong scaling
- potential program speedup is defined by the fraction of code (P) that can be parallelized

$$\text{speedup} = \frac{1}{\frac{P}{N} + S}$$

N	speedup			
	P = .50	P = .90	P = .95	P = .99
10	1.82	5.26	6.89	9.17
100	1.98	9.17	16.80	50.25
1,000	1.99	9.91	19.62	90.99
10,000	1.99	9.91	19.96	99.02
100,000	1.99	9.99	19.99	99.90

Gustafson's Law

- For weak scaling
- Speedup

$$S_p = f + p(1 - f)$$

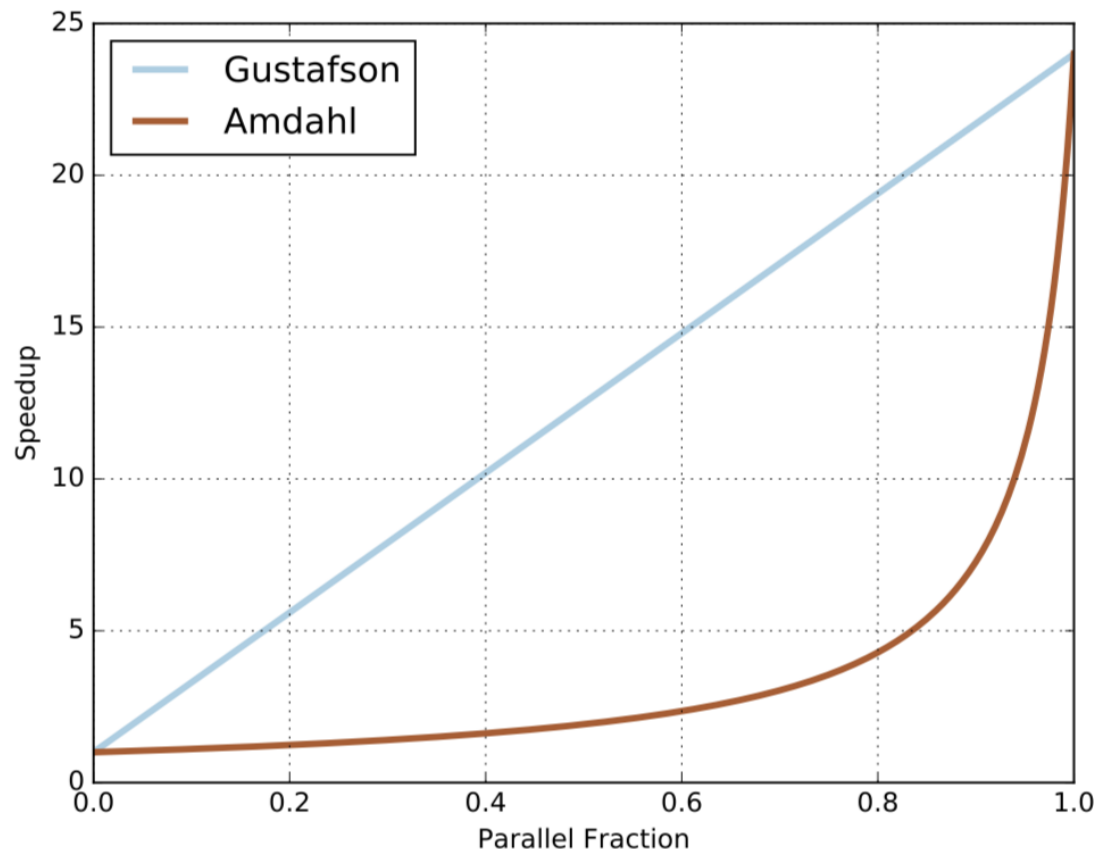
- Efficiency

$$E_w = \frac{S_p}{p} = \frac{t_1}{t_p}$$

scales relative to the
parallel fraction of a
code

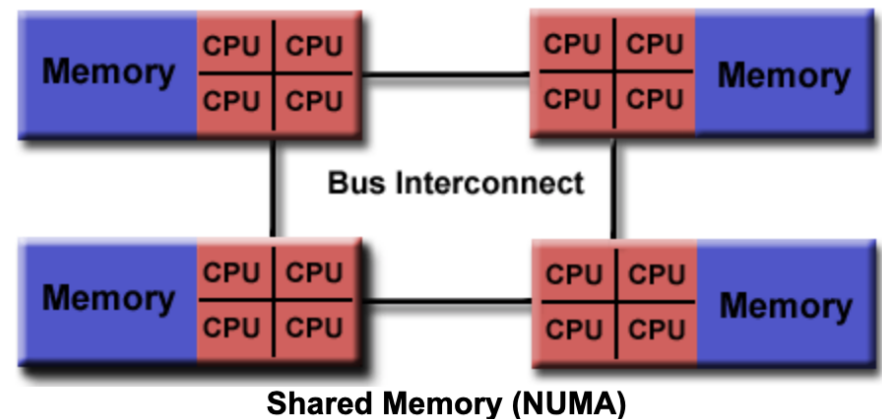
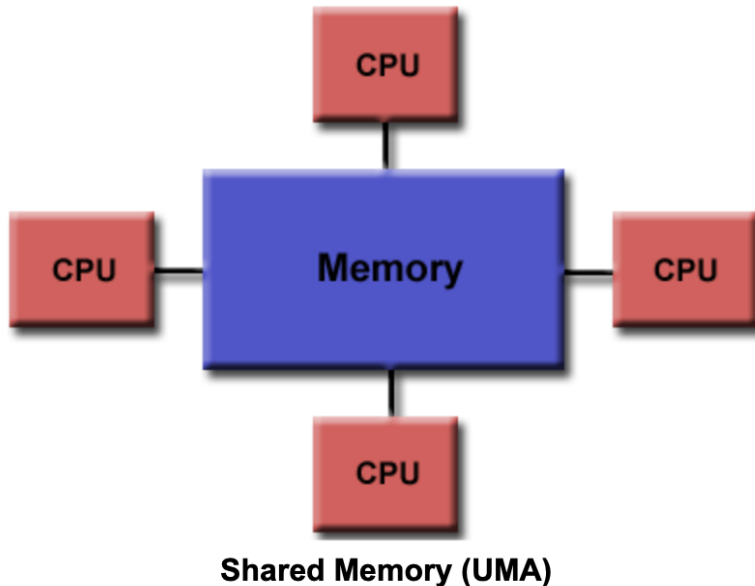
Gustafson's Point of View:

(Example shown for $p = 24$ processors)

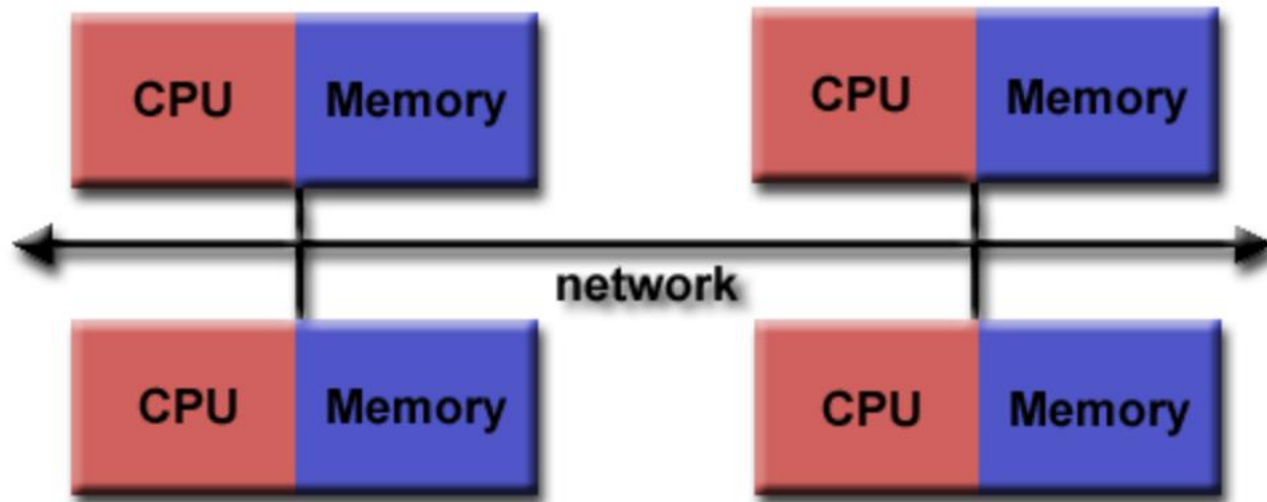


Shared Memory

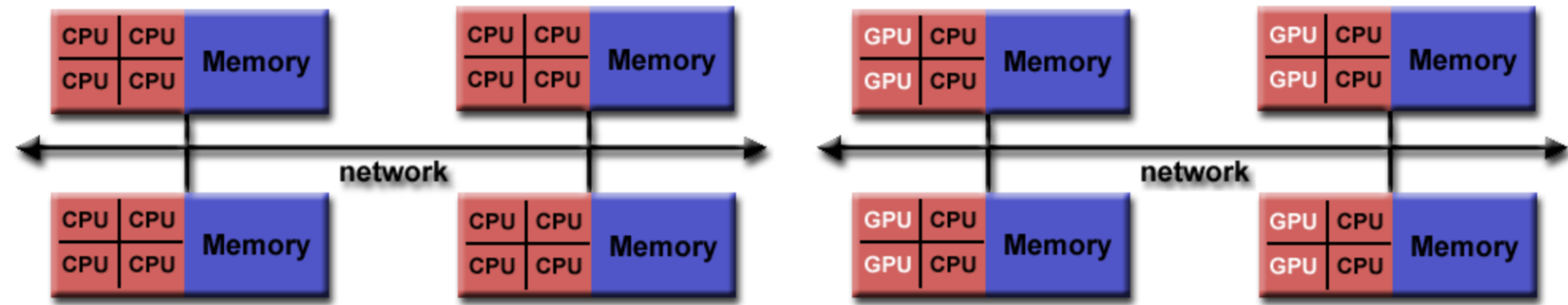
- **Uniform Memory Access (UMA)**
 - Most commonly represented today by *Symmetric Multiprocessor (SMP)* machines
- **Non-Uniform Memory Access (NUMA)**



Distributed Memory



Hybrid Distributed-Shared Memory

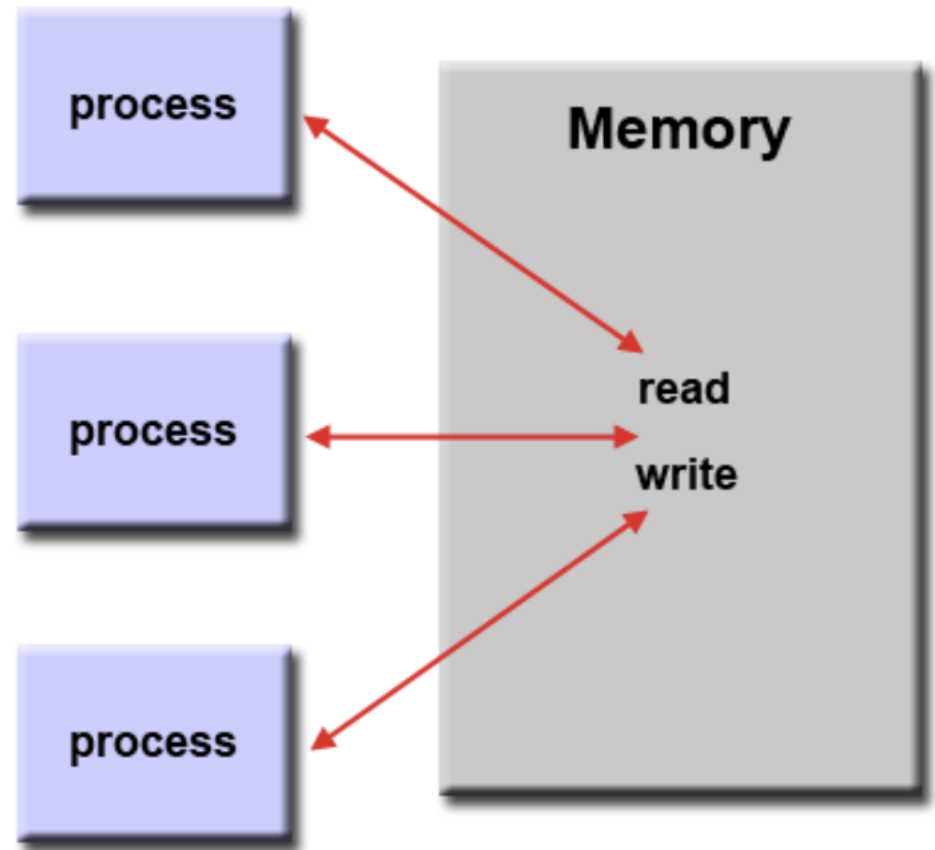


Parallel Programming Models

- There are several parallel programming models in common use
 - Shared Memory (without threads)
 - Threads
 - Distributed Memory / Message Passing
 - Data Parallel
 - Hybrid

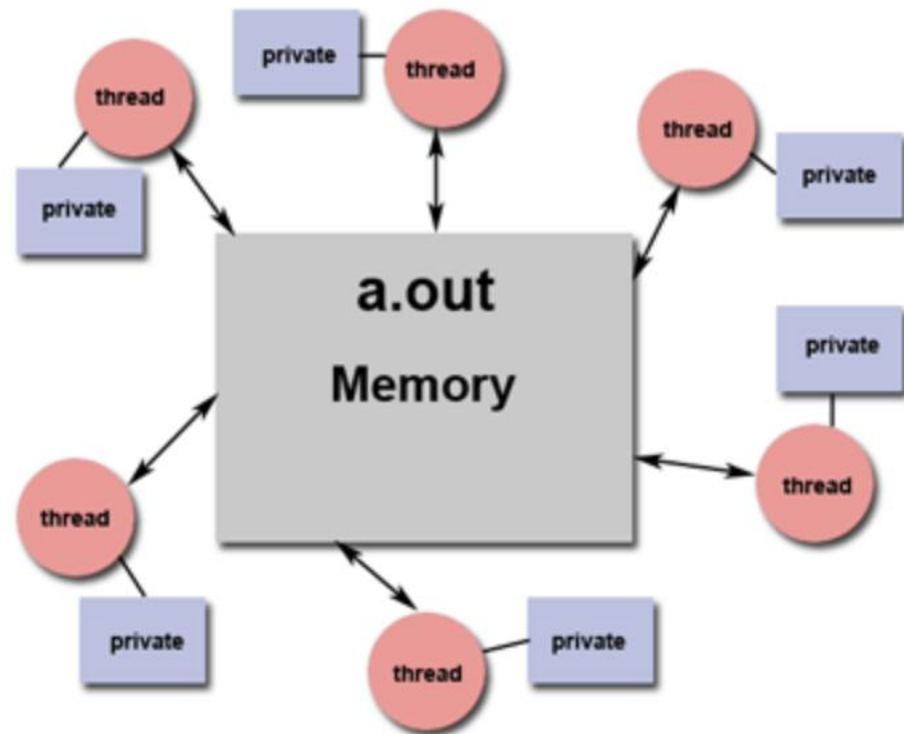
Shared Memory Model (without threads)

- Various mechanisms such as locks / semaphores are used to control access to the shared memory, resolve contentions and to prevent race conditions and deadlocks.
- The simplest
- data locality is hard to maintain



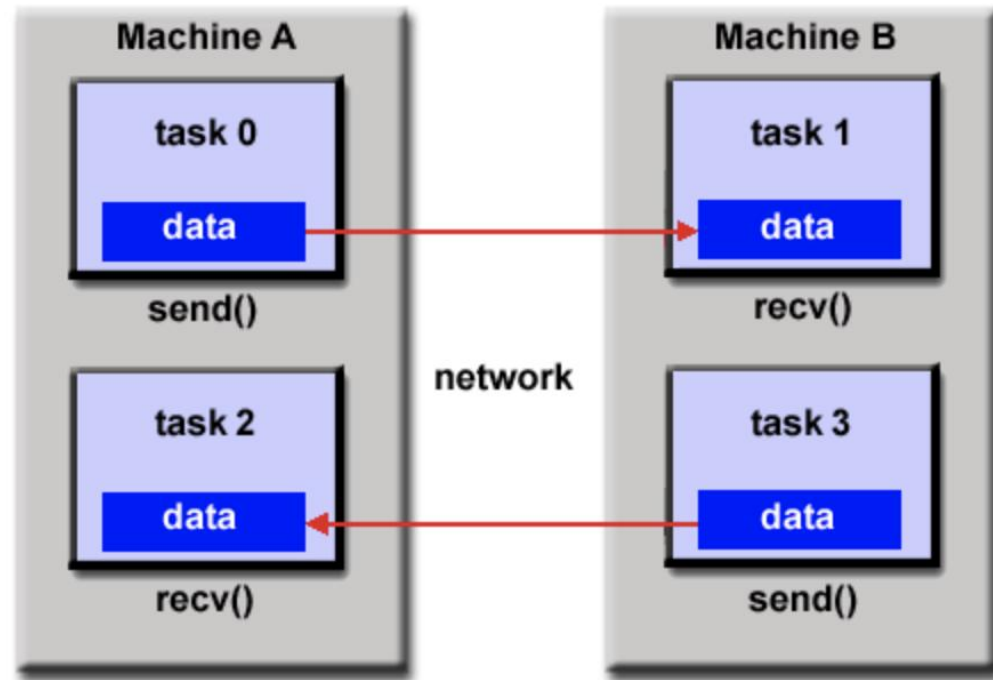
Threads Model

- Lightweight process
- Each thread has local data, but also, shares the entire resources of the process.
- This saves the overhead associated with replicating a program's resources for each thread.
- Each thread also benefits from a global memory view because it shares the memory space.

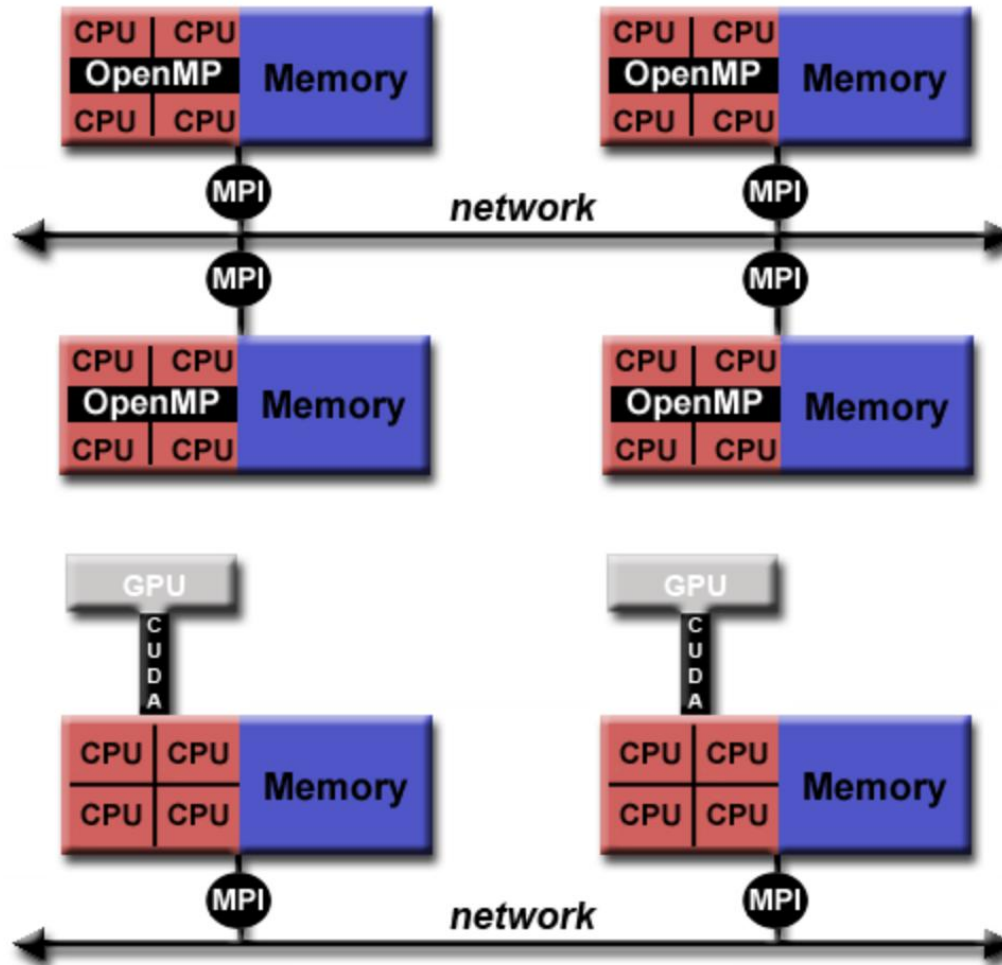


Distributed Memory / Message Passing Model

- A set of tasks that use their own local memory during computation.
- Tasks exchange data through communications by sending and receiving messages



Hybrid Model



More details in this course

- **Designing Parallel Programs**
 - Identify the program's *hotspots* and *bottlenecks*
- **Partitioning**
- **Communications**
- **Synchronization**
- **Data Dependencies**
- **Load Balancing**
- **Granularity**
- **I/O**
- **Performance Analysis**

