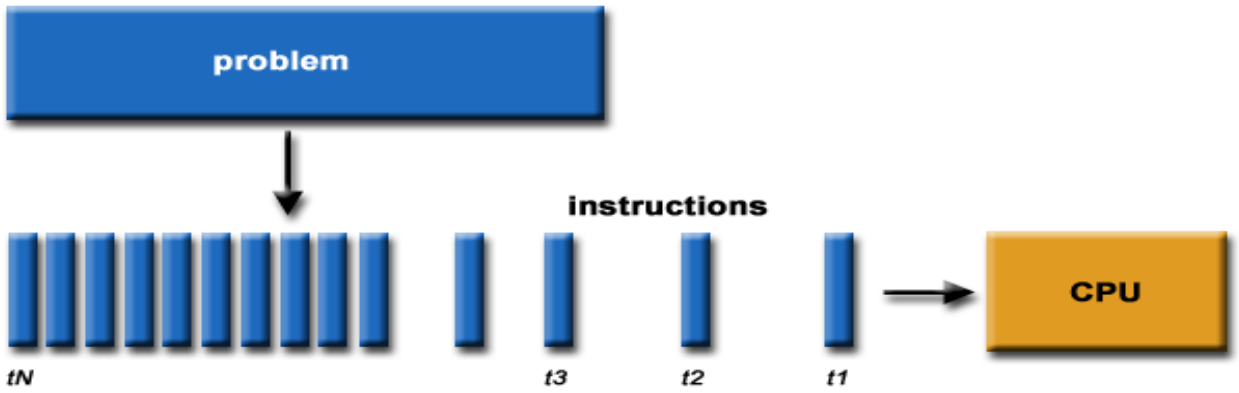
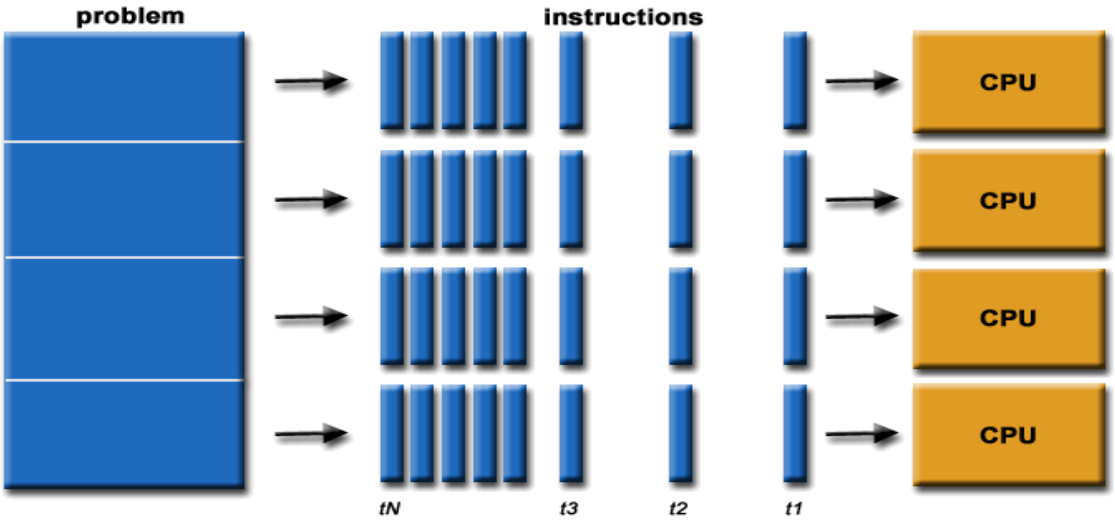
Parallel Computing

- Traditionally, software has been written for serial computation:
  - To be run on a single computer having a single Central Processing Unit (CPU);
  - A problem is broken into a discrete series of instructions.
  - Instructions are executed one after another.
  - Only one instruction may execute at any moment in time.



- In simple terms, **parallel computing** is breaking up a task into smaller pieces and executing those pieces at the same time, each on their own processor or on a set of computers that have been networked together.

  - To be run using multiple CPUs
  - A problem is broken into discrete parts that can be solved concurrently
  - Each part is further broken down to a series of instructions
  - Instructions from each part execute simultaneously on different CPUs



The compute resources might be:

- A single computer with multiple processors;
- An arbitrary number of computers connected by a network;
- A combination of both.

The computational problem should be able to:

- Be broken apart into discrete pieces of work that can be solved simultaneously;
- Execute multiple program instructions at any moment in time;
- Be solved in less time with multiple compute resources than with a single compute resource.

Uses for Parallel Computing:

- Science and Engineering
- Industrial and Commercial

Why Use Parallel Computing?

Main Reasons:

1. **Save time and money:** In theory, throwing more resources at a task will shorten its time to completion, with potential cost savings. Parallel computers can be built from cheap, commodity components.

2. **Solve larger problems:** Many problems are so large and/or complex that it is impractical or impossible to solve them on a single computer, especially given limited computer memory.

3. **Provide concurrency:** A single compute resource can only do one thing at a time. Multiple computing resources can be doing many things simultaneously. For example, the Access Grid provides a global collaboration network where people from around the world can meet and conduct work "virtually".

4. **Use of non-local resources:** Using compute resources on a wide area network, or even the Internet when local compute resources are scarce.

5. **Overcoming memory constraints** - Single computers have very finite memory resources. For large problems, using the memories of multiple computers may overcome this obstacle.

Limitations of Serial Computing

- Both physical and practical reasons pose significant constraints to simply building ever faster serial computers.

- Transmission speeds - the speed of a serial computer is directly dependent upon how fast data can move through hardware. Absolute limits are the speed of light (30 cm/nanosecond) and the transmission limit of copper wire (9 cm/nanosecond). Increasing speeds necessitate increasing closeness of processing elements.

- Limits to miniaturization - Processor technology is allowing an increasing number of transistors to be placed on a chip. However, even with molecular or atomic-level components, a limit will be reached on how small components can be.

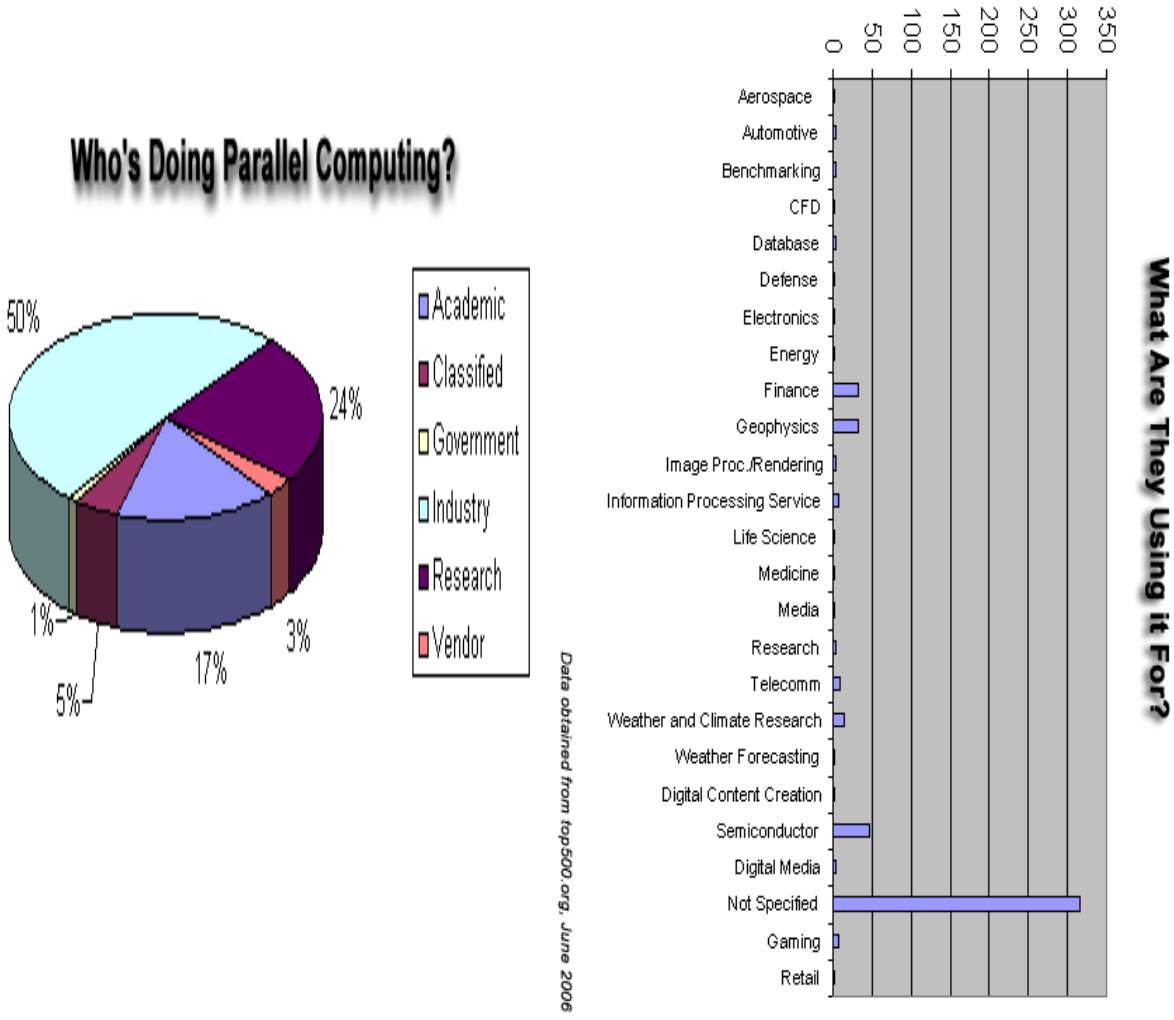
- Economic limitations - It is increasingly expensive to make a single processor faster. Using a larger number of moderately fast commodity processors to achieve the same (or better) performance is less expensive.

Who and What?

- Top500.org provides statistics on parallel computing users - the charts below are just a sample. Some things to note:

  Sectors may overlap - for example, research may be classified research. Respondents have to choose between the two.

- **"Not Specified" is by far the largest application -probably means multiple applications.**



The Future:

- During the past 20+ years, the trends indicated by ever faster networks, distributed systems, and multi-processor computer architectures (even at the desktop level) clearly show that parallelism is the future of computing.
- In this same time period, there has been a greater than 1000x increase in supercomputer performance,
- The race is already on for Exascale Computing!

Classification:

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processing element within a single machine, while clusters and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specific tasks.

Flynn's Classification of Computers

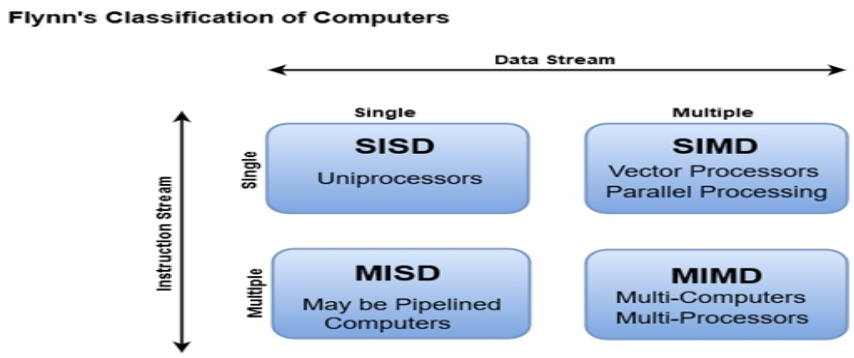
- There are different ways to classify parallel computers. For example: One of the more widely used classifications, in use since 1966, is called Flynn's Classification.
- Flynn proposed a classification for the organization of a computer system by the number of instructions and data items that are manipulated simultaneously.
- The sequence of instructions read from memory forms an **instruction stream**.
- The operations performed on the data in the processor forms a **data stream**.
- Parallel processing may occur in the instruction stream, in the data stream, or both.
- Flynn's classification divides computers into four major groups that are:

➢ Single Instruction stream, Single Data stream (SISD)

➢ Single Instruction stream, Multiple Data stream (SIMD)

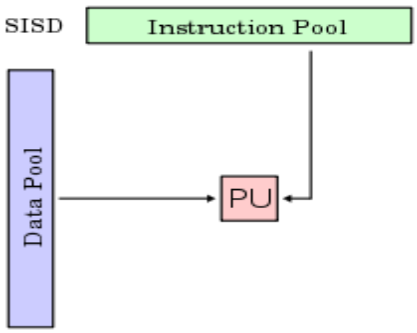
➢ Multiple Instruction stream, Single Data stream (MISD)

➢ Multiple Instruction stream, Multiple Data stream (MIMD)



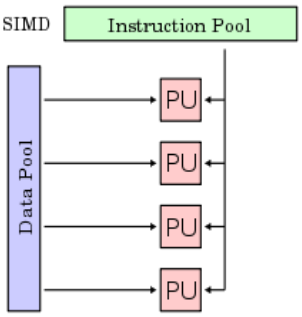
➢ SISD (Single Instruction, Single Data stream)

- SISD represents the organization of a single computer containing a control unit, a processor unit, and a memory unit.
- SISD refers to an Instruction Set Architecture in which a single processor (one CPU) executes exactly one instruction stream at a time and also fetches or stores one item of data at a time to operate on data stored in a single memory unit.
- Most conventional computers have SISD architecture like the traditional Von-Neumann computers.
- The SISD model is a typical non-pipelined architecture with the general-purpose registers, as well as dedicated special registers such as the Program Counter (PC), the Instruction Register (IR), Memory Address Registers (MAR) and Memory Data Registers (MDR).
- Data Stream flows between the processors and memory bi-directionally.
- Example: Older generation computers, minicomputers, and workstations.



➢ SIMD (Single Instruction, Multiple Data streams)

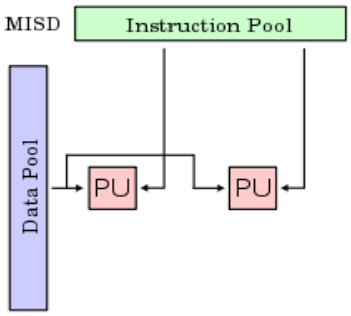
- SIMD represents an organization that includes many processing units under the supervision of a common control unit.
- SIMD is an Instruction Set Architecture that have a single control unit (CU) and more than one processing unit (PU) that operates like a Von Neumann machine by executing a single instruction stream over PUs, handled through the CU.
- The CU generates the control signals for all of the PUs and by which executes the same operation on different data streams. The SIMD architecture, in effect, is capable of achieving data level parallelism just like with vector processor.



- Some of the examples of the SIMD based systems include IBM's AltiVec and SPE for PowerPC, HP's PA-RISC Multimedia Acceleration eXtensions (MAX), Intel's MMX and iwMMXt Etc.

➢ MISD (Multiple Instruction, Single Data stream)

- MISD is an Instruction Set Architecture for parallel computing where many functional units perform different operations by executing different instructions on the same data set.
- This type of architecture is common mainly in the fault-tolerant computers executing the same instructions redundantly in order to detect and mask errors.

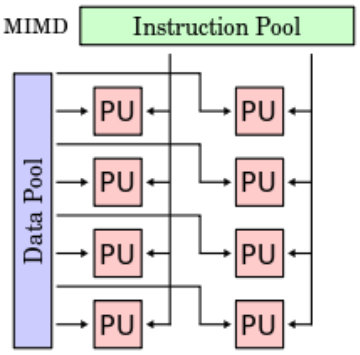


➢ MIMD (Multiple Instruction, Multiple Data streams)

- Multiple Instruction stream, Multiple Data stream (MIMD) is an Instruction Set Architecture for parallel computing that is typical of the computers with multiprocessors. Using the MIMD, each processor in a multiprocessor system can execute different set of the instructions independently on the different set of data units.

- In MIMD, each processor has a separate program and an instruction stream is generated from each program.

- MIMD based computer systems can use the shared memory in a memory pool or work using distributed memory across heterogeneous network computers in a distributed environment.



- The MIMD architectures is primarily used in a number of application areas such as computer- aided design/ computer-aided manufacturing, simulation, modeling, communication switches etc.
  - Examples: Cray T90, Cray T3E, IBM-SP2

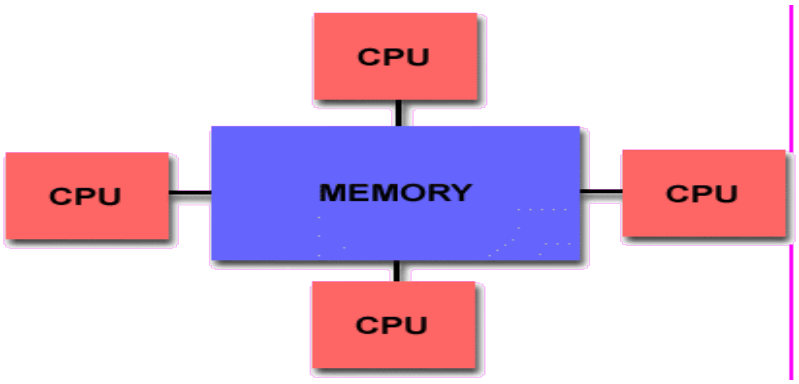
Parallel Computer Memory Architectures

- Shared Memory
- Distributed Memory
- Hybrid Distributed-Shared Memory

Shared Memory

General Characteristics:

- Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.
- Multiple processors can operate independently but share the same memory resources.
- Changes in a memory location effected by one processor are visible to all other processors.
- Shared memory machines can be divided into two main classes based upon memory access times: UMA and NUMA.



Uniform Memory Access (UMA):

- Most commonly represented today by Symmetric Multiprocessor (SMP) machines,
- Identical processors,
- Equal access and access times to memory,
- Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.

Non-Uniform Memory Access (NUMA):

- Often made by physically linking two or more SMPs,
- One SMP can directly access memory of another SMP,
- Not all processors have equal access time to all memories,
- Memory access across link is slower,
- If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA.

Advantages:

- Global address space provides a user-friendly programming perspective to memory,
- Data sharing between tasks is both fast and uniform due to the proximity of memory to CPUs

Disadvantages:

- Primary disadvantage is the lack of scalability between memory and CPUs. Adding more CPUs can geometrically increases 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.
- Expense: it becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing numbers of processors.

◆ **Distributed Memory**

General Characteristics:

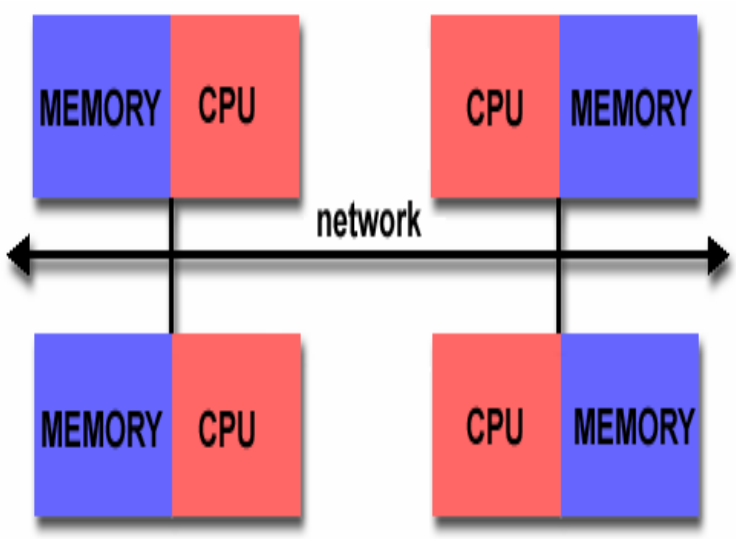
- Like shared memory systems, distributed memory systems vary widely but share a common characteristic. Distributed memory systems require a communication network to connect inter-processor memory.

- Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.

- Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.

- When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.

- The network "fabric" used for data transfer varies widely, though it can be as simple as Ethernet.



Advantages

- Memory is scalable with number of processors. Increase 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

- The programmer is responsible for many of the details associated with data communication between processors.
- It may be difficult to map existing data structures, based on global memory, to this memory organization.
- Non-uniform memory access (NUMA) times.

◆ **Hybrid Distributed-Shared Memory**

General Characteristics:

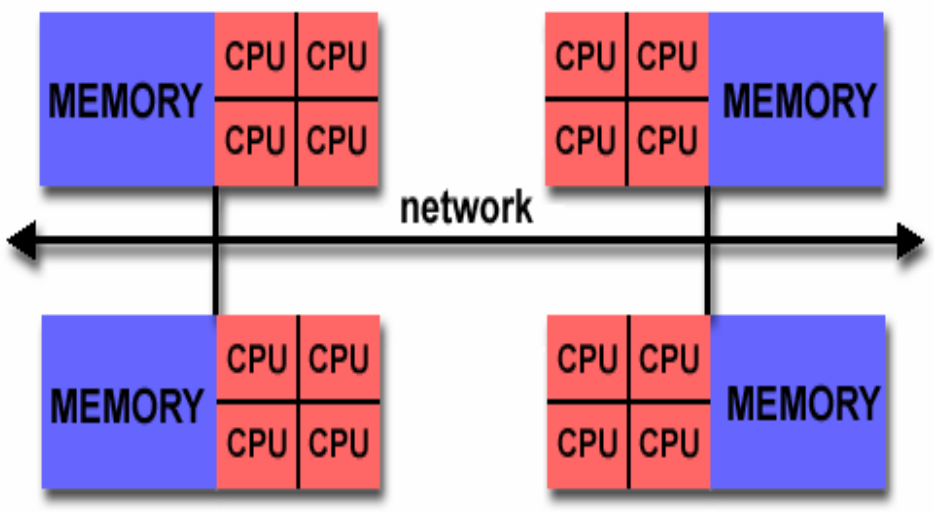
- The largest and fastest computers in the world today employ both shared and distributed memory architectures.

- The shared memory component is usually a cache coherent SMP machine. Processors on a given SMP can address that machine's memory as global.

- The distributed memory component is the networking of multiple SMPs. SMPs know only about their own memory - not the memory on another SMP. Therefore, network communications are required to move data from one SMP to another.

- Current trends seem to indicate that this type of memory architecture will continue to prevail and increase at the high end of computing for the foreseeable future.

- Advantages and Disadvantages: whatever is common to both shared and distributed memory architectures.



Distributed System

- A distributed system is a collection of independent computers that appears to its users as a single comprehensive system.
- A distributed system is one in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing.
- WWW is the biggest example of distributed system.
- Others are: - The internet,
  - An intranet which is a portion of the internet managed by an organization

- Nearly all systems today are distributed in some way, e.g. :

  - they use email - they access files over a network - they access printers over a network - they are backed up over a network - they share other physical or logical resources - they cooperate with other people on other machines - they receive video, audio, etc.

Why Distributed System? - Objectives of DS

1. **Cost**. Better price/performance as long as commodity hardware is used for the component computers.

2. **Reliability**. By having redundant components the impact of hardware and software faults on users can be reduced.

3. **Transparency.** The end users can be hidden from the actual separation of the distributed system so that the user feels that everything is transparent to everyone.

4. **Scalability**. Resources such as processing and storage capacity can be increased significantly.

5. **Dependability**. The dependence of a system on another system can be achieved to solve a particular task jointly.

6. **Performance**. By using the combined processing and storage capacity of many nodes, performance levels can be reached that are beyond the range of centralized machines.

7. **Flexibility**. Easily can be added or removed a node.

Loosely-coupled System

- Most distributed systems are “loosely-coupled”
- Each CPU runs an independent autonomous OS
- Hosts communicate through *message passing*
- Computers/systems don’t really trust each other
- Some resources are shared, but most are not
- The system may look differently from different hosts
- Typically, communication times are long relative to processing times

Closely-Coupled Systems

- **Distributed system becomes more “closely coupled” as it:**
  - appears more uniform in nature
  - runs a “single” operating system (cooperating across all machines)
  - shares all logical resources (e.g., files)
  - shares all physical resources (CPUs, memory, disks, printers, etc.)
- **In the limit, a closely coupled distributed system: –**
  - Multi computer

- Multiple computers – CPU and memory and network interface (NIC)
- High performance interconnect

- Looks a lot like a single system

E.g., Beowulf clusters

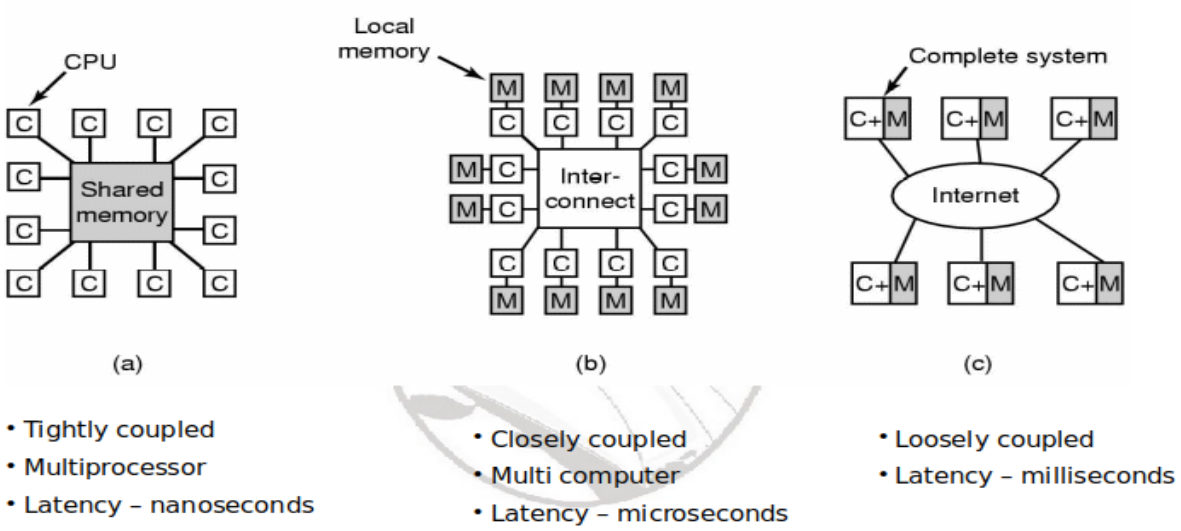
Tightly Coupled Systems

- **Tightly coupled systems usually are *multiprocessor systems***

- Have a single address space
- Usually has a single bus or backplane to which *all* processors and memories are connected
- Low communication latency
- Shared memory for processor communication
- Shared I/O device access

Example:

- Multiprocessor Windows PC



Multiprocessor

Multiprocessing is the use of two or more central processing unit (CPUs) within a single computer system- MIMD. The term also refers to the ability of a system to support more than one processor or the ability to allocate tasks between them. The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching.

Benefits of using a Multiprocessor –

- Enhanced performance.
- Multiple applications.
- Multi-tasking inside an application.
- High throughput and responsiveness.
- Hardware sharing among CPUs.

Multiprocessor operating systems

Multiprocessor operating systems are the server operating systems with special features for the communication and the connectivity.

Multiprocessor operating systems are used where multiple CPUs connected into a single system.

Multiprocessor operating system (OS) is almost a regular OS as they also handle system calls, do memory management, provide file system, and also manage input/output devices.

Here are the list of some various organizations of multiprocessor operating systems:

- Each CPU has its own OS
- Master-Slave multiprocessors
- Symmetric multiprocessors

Describing briefly all the above three different-different organizations of multiprocessor operating systems.

➢ Each CPU has its own OS

To statically divide the memory into as many partitions as there are central processing units and given central processing unit its own private memory and its own private copy of the OS is basically the simplest way to organize a multiprocessor OS or multiprocessor operating system.

➢ Master-Slave Multiprocessors

The master-slave models basically solves almost all the problems of the first model.
In this model, there is a single data structure that keeps track of ready processes.
Now, when a central processing unit goes idle in this model, then it asks the OS for a process to run and it is assigned one.

➢ Symmetric Multiprocessors

Symmetric multiprocessors (SMP) is the third model. In this model, there is one copy of the OS in memory, but any central processing unit can run it. Now, when a system call is made, then the central processing unit on which the system call was made traps to the kernel and then processes that system call. This model balances processes and memory dynamically.

Types of Multiprocessors

There are mainly two types of multiprocessors i.e. symmetric and asymmetric multiprocessors. Details about them are as follows:

Symmetric Multiprocessors

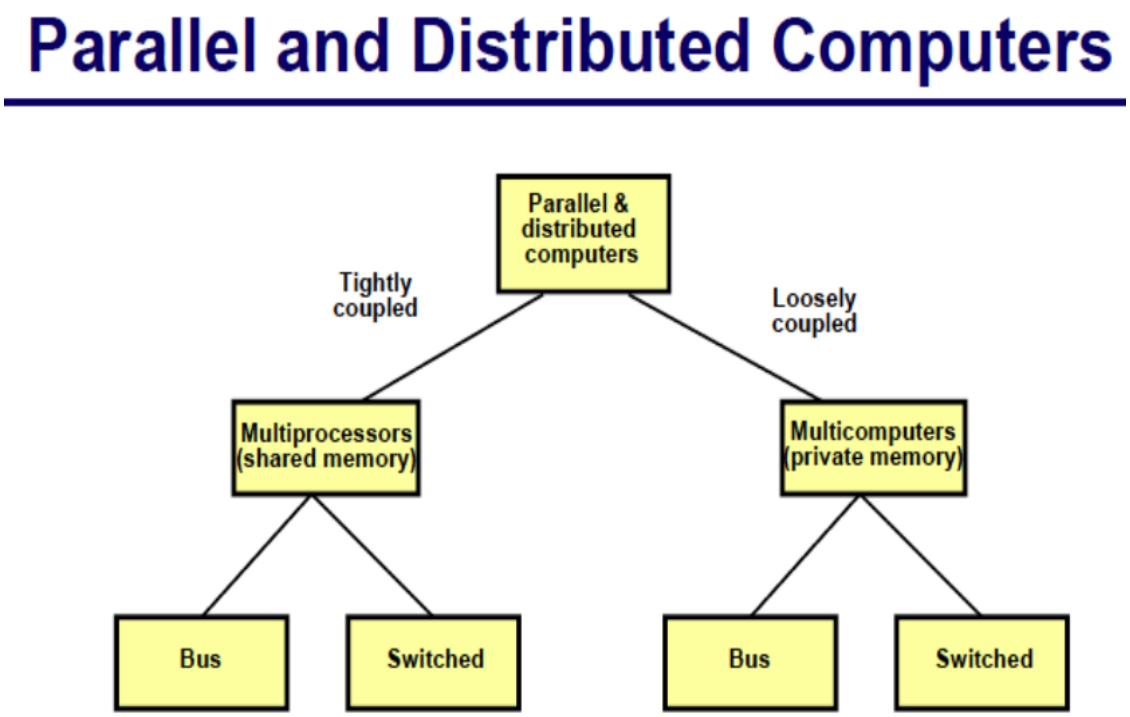
In these types of systems, each processor contains a similar copy of the operating system and they all communicate with each other. All the processors are in a peer to peer relationship i.e. no master - slave relationship exists between them.

An example of the symmetric multiprocessing system is the Encore version of Unix for the Multimax Computer.

Asymmetric Multiprocessors

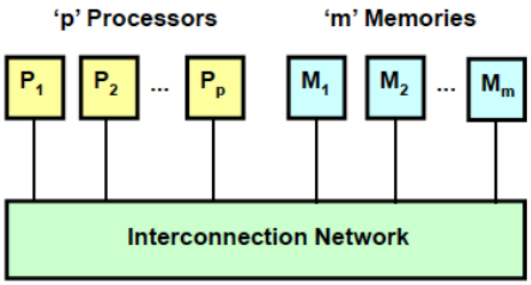
In asymmetric systems, each processor is given a predefined task. There is a master processor that gives instruction to all the other processors. Asymmetric multiprocessor system contains a master slave relationship.

Asymmetric multiprocessor was the only type of multiprocessor available before symmetric multiprocessors were created.



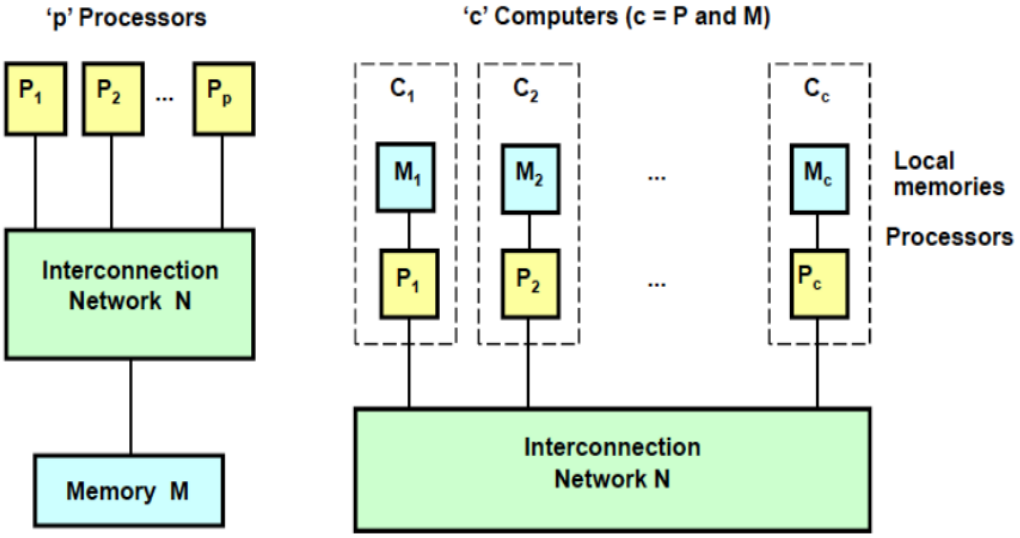
Structural Classification

- Computer system is essentially
  - 'p' processing elements = (CPU + registers + cache)
  - 'm' memory units
  - joined by an inter-connection network
- Memory may be local to a processor, shared or both

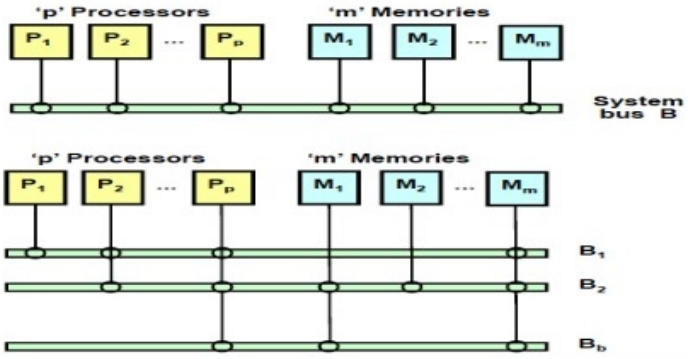


Shared Memory (Multiprocessor)

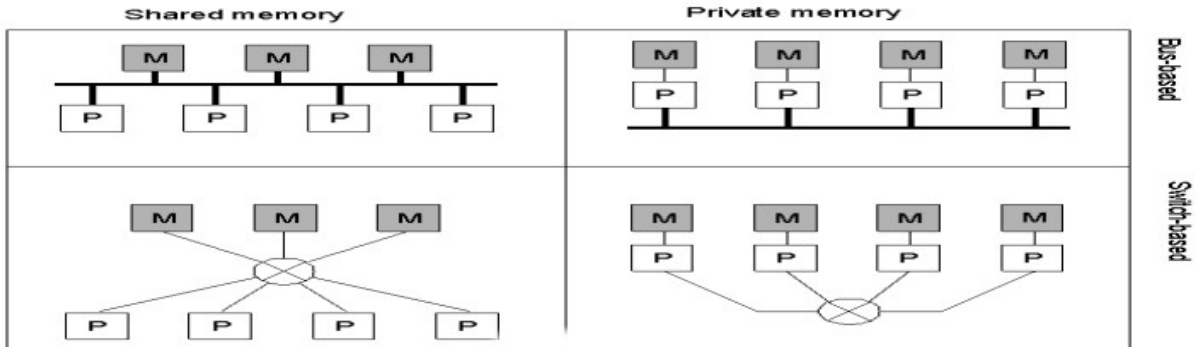
Distributed Memory (Multicomputer or distributed computer system)



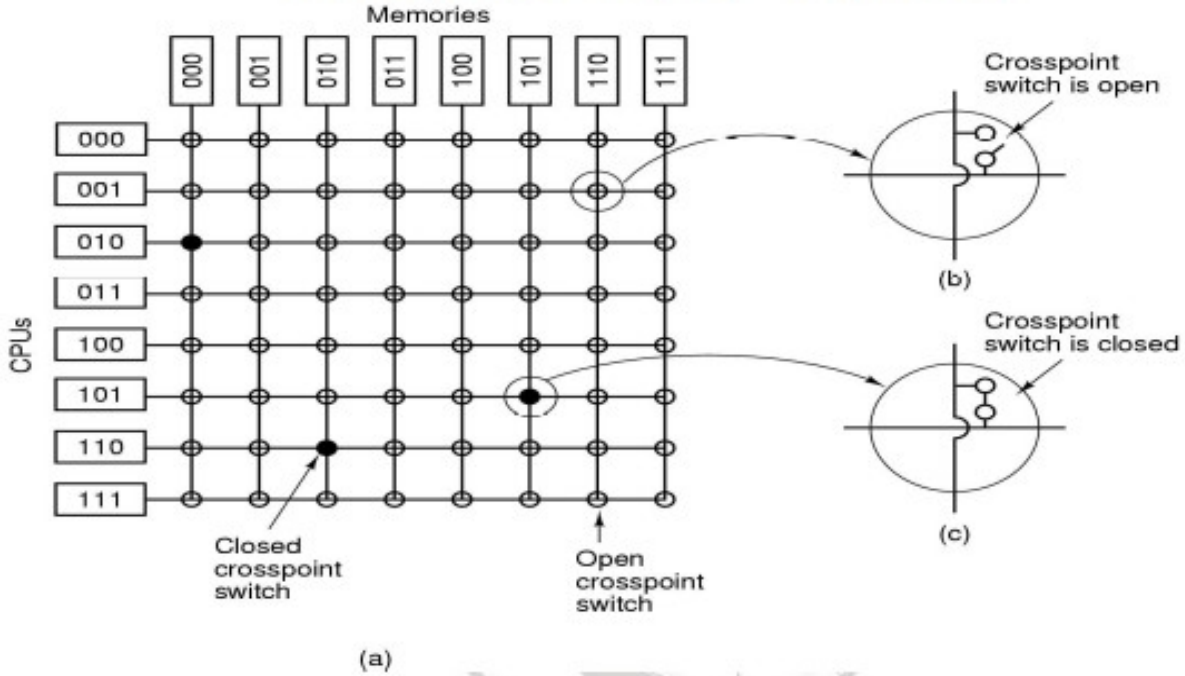
Bussed Systems



- Single shared bus
  - widely used
- Multiple busses
  - relieve bus contention
  - provides some redundancy

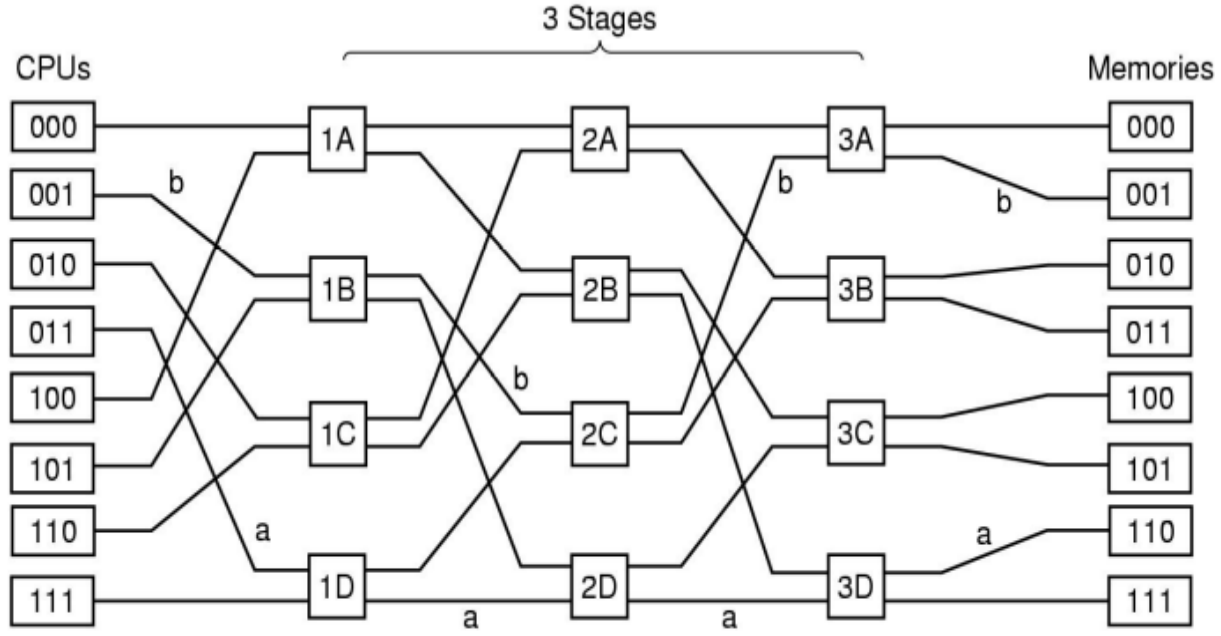


Multiprocessors - Crossbar



- Can support a large number of CPUs -
- Non-blocking network
- Cost/performance effective up to about 100 CPU

Multiprocessors – Multistage Switching Networks



- **Omega Network – blocking**

- With n CPUs and n memories, total of $(n \log_2 n)/2$ switches.

- Better than crossbar switch, but still big.

Multiprocessor Scheduling

- **When processes are independent (e.g., timesharing)**
  - Allocate CPU to highest priority process
  - For a process with a spinlock, let it run until it releases the lock
  - To reduce translation lookaside buffer and memory cache flushes, try to run a process on the same CPU each time it runs

For groups of related processes

- Attempt to simultaneously allocate CPUs to all related processes (*space sharing*)
- Run all threads to termination or block
- **Gang schedule –** apply a scheduling policy to related processes together

Multicomputers

- **Multiprocessor size is limited**
- **Multi computers – closely coupled processors that do not physically share memory**
  - Cluster computers
  - Networks or clusters of computers
  - Can grow to a very large number of processors
- **Consist of**
  - Processing nodes – CPU, memory and network interface (NIC)
  - I/O nodes – device controller and NIC
  - Interconnection network
  - Many topologies – e.g. grid, hypercube,

- Can be packet switched or circuit switched

Interprocess communication

Interprocess communication is the mechanism provided by the operating system that allows processes to communicate with each other. This communication could involve a process letting another process know that some event has occurred or the transferring of data from one process to another.

A diagram that illustrates interprocess communication is as follows:

Approaches to Interprocess Communication

The different approaches to implement interprocess communication are given as follows:

- **Pipe**

  A pipe is a data channel that is unidirectional. Two pipes can be used to create a two-way data channel between two processes. This uses standard input and output methods.

• **Socket**

The socket is the endpoint for sending or receiving data in a network. This is true for data sent between processes on the same computer or data sent between different computers on the same network. Most of the operating systems use sockets for interprocess communication.

• **File**

A file is a data record that may be stored on a disk or acquired on demand by a file server. Multiple processes can access a file as required. All operating systems use files for data storage.

• **Signal**

Signals are useful in interprocess communication in a limited way. They are system messages that are sent from one process to another. Normally, signals are not used to transfer data but are used for remote commands between processes.

• **Shared Memory**

Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other.

• **Message Queue**

Multiple processes can read and write data to the message queue without being connected to each other. Messages are stored in the queue until their recipient retrieves them. Message queues are quite useful for interprocess communication and are used by most operating systems.