1. **What do you mean by parallel processing?**

A system is said to be parallel system in which multiple processor have direct access to shared memory which forms a common address space. Usually tightly coupled system are referred to as parallel system. In these systems, there is a single system wide primary memory (address space) that is shared by all the processors. In short parallel processing in the use of two or more processors in combination to solve a single problem.

1. **What are the** **application areas of parallel processing?**

Application areas of parallel processing:

Parallel computing us an evolution of serial computing that attempts to emulate what has been always been that state of affairs in the natural world: many complex, interrelated events happening at the same time, yet within a sequence. For Examples:

* Planetary and galactic orbits
* Weather and ocean patterns
* Tectonic plate drift
* Rush hour traffic in Paris
* Automobile assembly line
* Daily operations within a business
* Building a shopping mall
* Ordering a hamburger at the drive through

Traditionally, parallel computing has been considered to be “the high end of computing” and has been motivated by numeric simulations of complex systems and “grand challenge problems”, such as:

* Weather and climate
* Chemical and nuclear reactions
* Biological, human resource
* Geological, seismic activity
* Mechanical devices – from prosthetics to spacecraft
* Electronic circuits
* Manufacturing processes

Today, commercial applications are providing an equal or greater driving force in the development of faster computers. These applications require that processing of large amounts of data in sophisticated ways. Example includes:

* Parallel database, data mining
* Oil exploration
* Web search engines web based business services
* Computer-aided diagnosis in medicine
* Management of national and multi-national corporations
* Networked video and multimedia technologies

1. **Distinguish SIMD and MIMD architectures.**

|  |  |
| --- | --- |
| SIMD | MIMD |
| Stands for Single Instruction Multiple Data | Stands for Multiple Instruction Multiple Data |
| It is also called as array processor | It is also called multiprocessor |
| Single program, processing element operate synchronously | Multiple communication programs, processing element operate asynchronously. |
| One copy of the program stored. | Each Processing element stores it own program |
| One decoder in control unit. | One decoder in each Processing element |
| Architecture is simple | Architecture is complex. |
| Cost is low. | Cost is high. |
| Scalability in size and performance | Complex in size and good performance. |
| Single stream of instruction is fetched | Multiple streams of instruction fetched |
| Instruction stream is fetched by shared memory | Instruction streams are fetched by control unit |
| Instruction is broadcasted to multiple processing elements | Instruction streams are decoded to get multiple decoded instruction stream |

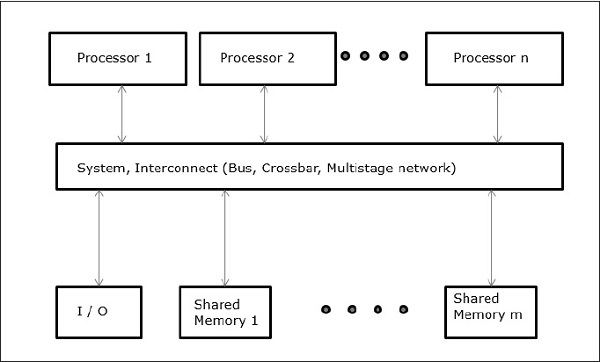
1. **Describe the physical models of shared memory multiprocessors briefly.**

A system with multiple CPUs sharing the same main memory is called shared memory multiprocessor. In a shared memory multiprocessor system all processes on the various CPUs share a unique logical address space, which is mapped on a physical memory that can be distributed among the processors. Each process can read and write data item simply using load and store operations and process communication through shared memory. There are three classes of shared memory multiprocessors according to the way each CPU sees main memory:

1. UMA
2. NUMA
3. COMA

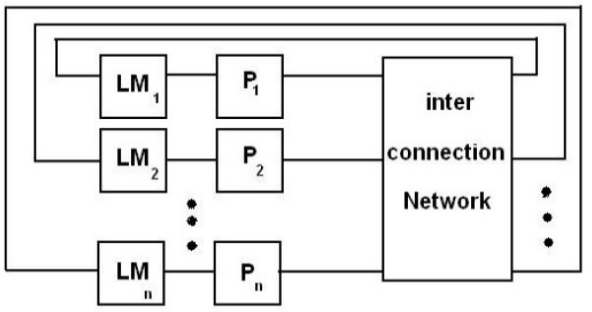
Uniform Memory Access (UMA):

* + Uniform memory access (UMA) is a shared memory architecture used in parallel computers. All the processors in the UMA model share the physical memory uniformly.
  + In a UMA architecture, access time to a memory location is independent of which processor makes the request or which memory chip contains the transferred data.
  + In the UMA architecture, each processor may use a private cache. Peripherals are also shared in some fashion.
  + The UMA model is suitable for general purpose and time sharing applications by multiple users. It can be used to speed up the execution of a single large program in time- critical applications.
  + Tightly Coupled

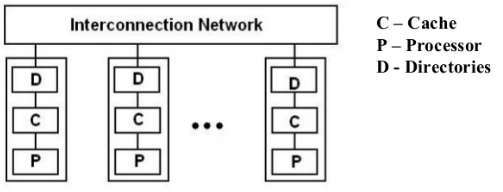


Non-Uniform Memory Access (NUMA):

* Non-uniform memory access (NUMA) is a computer memory design used multiprocessing, where the memory access time depends on the memory location relative to the processor.
* Under NUMA, a processor can access its own local memory faster than non-local memory.
* Access time varies with the location of memory word.
* Shared memory is distributed to all processors.
* Slow access to remote memory attached to other.



Cache Only Memory Access (COMA):

* Data have no specific permanent location (no specific memory address)
* Data partitioning is dynamic and implicit
* There are cache memory instead of local memory known as “attraction memory”
* If Local memory in NUMA model is replaced by a cache memory, then it will be COMA model.
* Easier to program and execution time is less.

1. **Define implicit and explicit parallelism.**

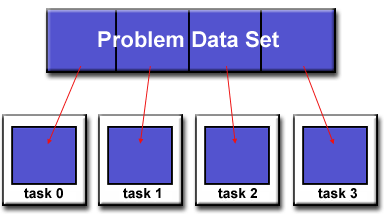
Implicit Parallelism: implicit parallelism is a characteristic of a programming language that allows a compiler or interpreter to automatically exploit the parallelism inherent to the computations expressed by some of the language's constructs. It is visible to programmer.

Explicit Parallelism: Explicit parallelism is a concept of processor-compiler efficiency in which a group of instructions is sent from the compiler to the processor for simultaneous rather than sequential execution. This is not visible to programmer.

1. **How parallel processing is accomplished.**

Parallel processing accomplished by dividing a large task into many smaller tasks and executes the smear tasks concurrently on several nodes. As a result, the larger task completes more quickly. To accomplish parallel processing there are some steps to follow:

Partitioning: One of the first steps in designing a parallel program is to break the problem into discrete "chunks" of work that can be distributed to multiple tasks. This is known as decomposition or partitioning. There are two basic ways to partition computational work among parallel tasks: domain decomposition and functional decomposition.



Communication: A communication between every task is necessary to maintain synchronization. There are a number of important factors to consider when designing your program's inter-task communications:

* Communication Overhead: Communication overhead is the proportion of time spending in communicating with other tasks.
* Latency and Bandwidth: latency is the time it takes to send a minimal (0 byte) message from point A to point B. Commonly expressed as microseconds. Bandwidth is the amount of data that can be communicated per unit of time. Commonly expressed as megabytes/sec or gigabytes/sec.
* Visibility of Communication: With the Message Passing Model, communications are explicit and generally quite visible and under the control of the programmer.
* Scope of Communication: Knowing which tasks must communicate with each other is critical during the design stage of a parallel code. So scope of communication is considered.
* Efficiency of Communication: It depends on network fabric, types of communication etc.

**Synchronization:** Managing the sequence of work and the tasks performing it is a critical design consideration for most parallel programs. It is called synchronization.

**Handle Dependencies:** A dependence exists between program statements when the order of statement execution affects the results of the program.Distributed memory architecture is need to handle communication and shared memory architecture is need to handle I/O operations between dependent tasks.

**Load Balancing:**Load balancing refers to the practice of distributing approximately equal amounts of work among tasks so that all tasks are kept busy all of the time. It can be considered a minimization of task idle time.

**Granularity:** Granularity (or grain size) of a task is a measure of the amount of work (or computation) which is performed by that task.The most efficient granularity is dependent on the algorithm and the hardware environment in which it runs.

**I/O:**Parallel I/O systems may be immature or not available for all platforms.So proper and effective I/O operation is considered.

**Debugging:**Debugging parallel codes can be incredibly difficult, particularly as codes scale upwards. But there are some tools like MPI,OpenMPetc are available for debugging parallel process.

**Performance Analysis and Tuning:** It is also done with debugging. It improves performance of program.

1. **What are the important issues in parallel processing?**

Important issues in parallel processing are:

* Goal is to reduce execution time
  + computation time
  + idle time - waiting for data from other processors
  + communication time - time the processors take to send and receive messages
* Load Balancing
  + divide the work equally among the available processors
* Minimizing Communication
  + reduce the number of messages passed
  + reduce amount of data passed in messages
* Where possible - overlap communication and computation
* Many problems scale well to only a limited number of processors

1. **Explain general architecture for parallel processing.**
2. **Mention some advantages of parallel processing.**

Advantages of parallel processing are:

* Need to solve larger problems
  + more memory intensive
  + more computation
  + more data intensive
* Parallel programming provides
  + more CPU resources
  + more memory resources
  + solve problems that were not possible with serial program
  + solve problems more quickly

1. **Illustrate and describe briefly the various classes of computer architectures in terms of instruction and data streams introducing by Michael Flynn’s.**

**Or,**

**Explain with diagram the Flynn’s classification of various computer architecture.**

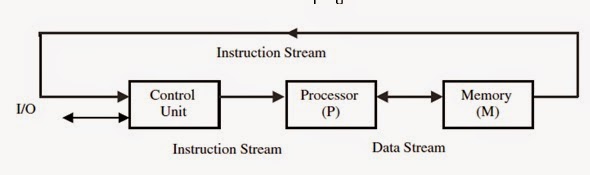
In 1966, Michael Flynn proposed a classification for computer architectures based on the number of instruction streams and data streams. A stream means a sequence of items. The classification of computer architectures based on the number of instruction streams and data streams.

Flynn’s taxonomy:

* SISD: Single Instruction Single Data
* SIMD: Single Instruction Multiple Data
* MISD: Multiple instructions single data
* MIMD: Multiple instructions multiple data

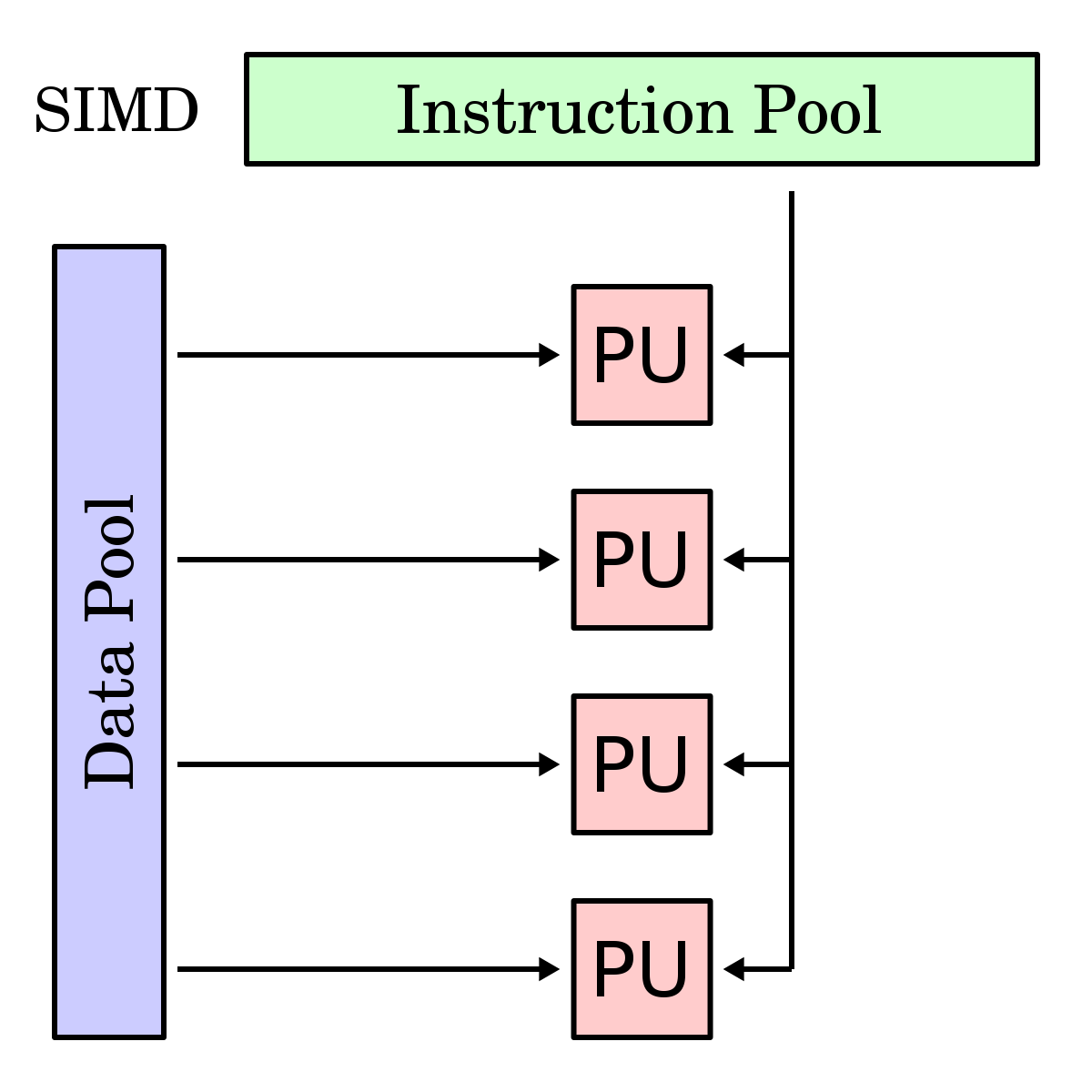
SISD:

* SISD stands for Single Instruction Single data stream
* SISD corresponds to the traditional mono processor.
* A single data stream is being processed by one instruction stream.
* Data stored in single memory.
* Deterministic execution



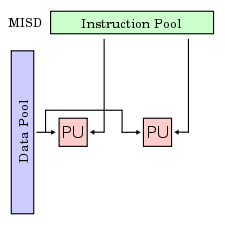
SIMD:

* Stands for Single Instruction Multiple Data Stream
* A type of parallel computer
* Single machine instruction
* Control simultaneous execution
* Each processing element has associated data memory.
* All processing units execute the same instruction at any given clock cycle.
* Each processing unit can operate on a different data element.
* This type of machine typically has an instruction dispatcher, a very high bandwidth internal network and a very large array of very small capacity instruction units.
* Synchronous and deterministic execution.



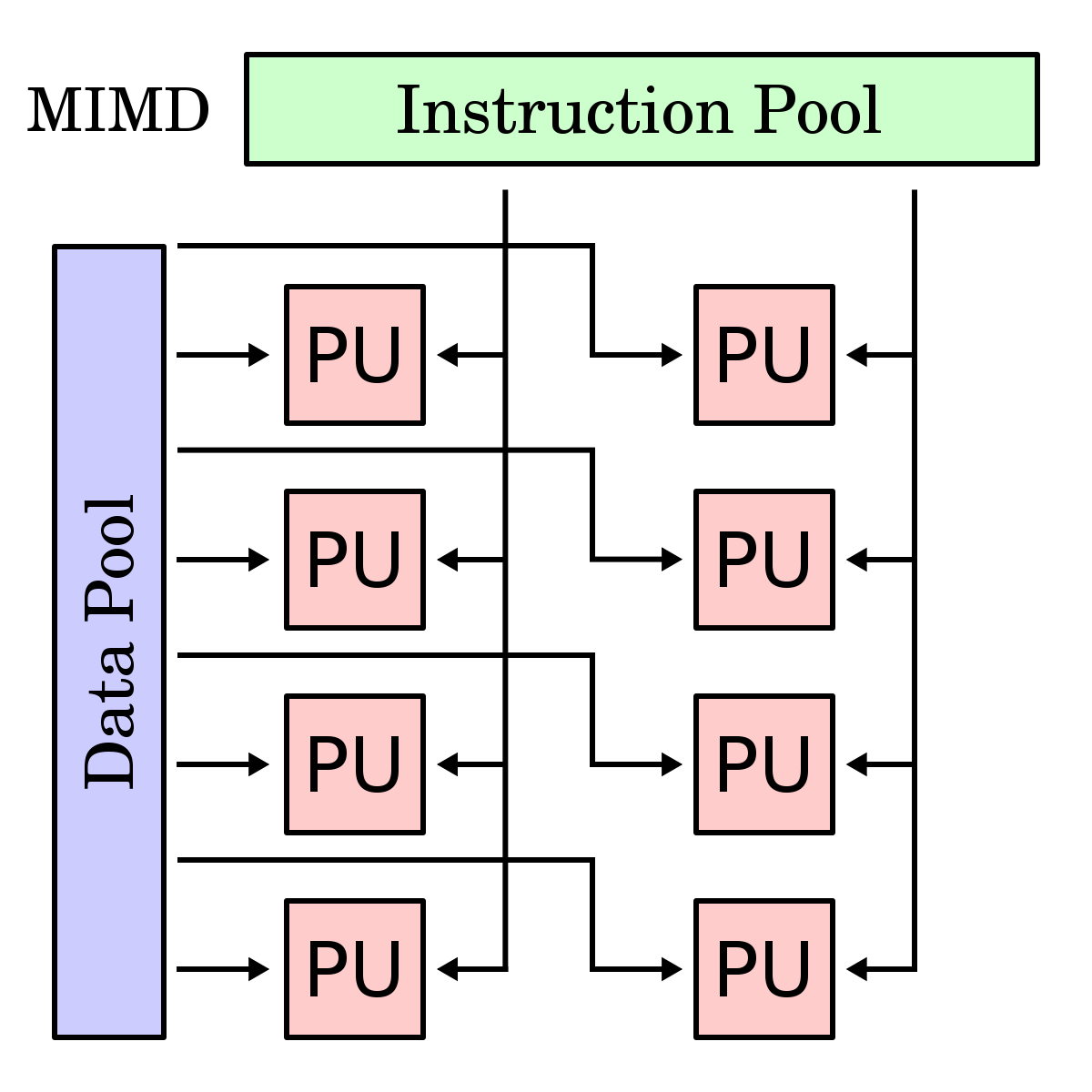
MISD:

* Stands for Multiple Instruction Single Data Stream
* Each processor executes a different sequence of instruction
* Multiple processing units operate on one single data stream.
* Never been implemented



MIMD:

* Stands for Multiple Instruction Multiple Data stream
* Each processor has a separate program
* Each instruction operates on different data.
* Each processor may be working with a different data stream.
* Execution can be synchronous or asynchronous, deterministic or non-deterministic
* Currently most of the super computers are into this category.



1. **Distinguish between shared memory multiprocessor and message passing multiprocessor.**

|  |  |
| --- | --- |
| Shared Memory | Message Passing |
| Implicit communication via memory operations (Load/ Store) | Explicit memory communication via messages |
| Tight coupling of program components. | Loose coupling of program components |
| Communication process execute in non-overlapping lifetimes | Communicating processes have to execute at the same time |
| Faster and cheaper | Slower and expensive |

1. **Define cycle per instruction (CPI) and throughput Rate.**

CPI:CPI (cyles per instruction, or clocks per instruction) is the number of computer clock speed cycles (alternating current pulses) that occur while a computer instruction is being executed (performed by the computer processor). The number of cycles per instruction can be reduced by using pipelining . In some superscalar processors, more than one instruction can be performed during a single clock cycle.Cycles Per Instruction is defined by the following:

Where is the number of instruction for a given instruction type , is the clock-cycles for that instruction and is the total instruction count.

Throughput Rate:How many programs a system can execute per unit time, called the system throughput is often lower than the CPU throughput defined by:

Where is the clock rate, is the instruction count and CPI is clock per second.

1. **Define distributed system.**

A distributed system is a collection of independent computers that appears to its users as a single coherent system. -Tanenbaum and Van Steen

A distributed system is one in which components located at networked computers communicate and co-ordinate their actions only by passing messages. -Coulouris, Kindberg, Dellimore

1. **Write down some goals of the distributed system.**

Four important goals to be achieved by any distributed system are:

* Making Resource Accessible:

Main goal of a distributed system:-

* Make it easy for the users to access remote resources.
* To share them in a controlled and efficient way.
* Distribution Transparency:

Hide the fact that its processes and resources are physically distributed across multiple computers – systems should be transparent.

* Openness:

Offer services according to standard rules that describe the syntax and semantics of those services.

* Scalability:
  + Can easily add more users and resources to the system.
  + Users and resource may lie far apart, but user can have the resource any time.
  + Can be easy to manage even if it spans many independent administrative organizations.

1. **Differentiate parallel processing and distributed processing.**

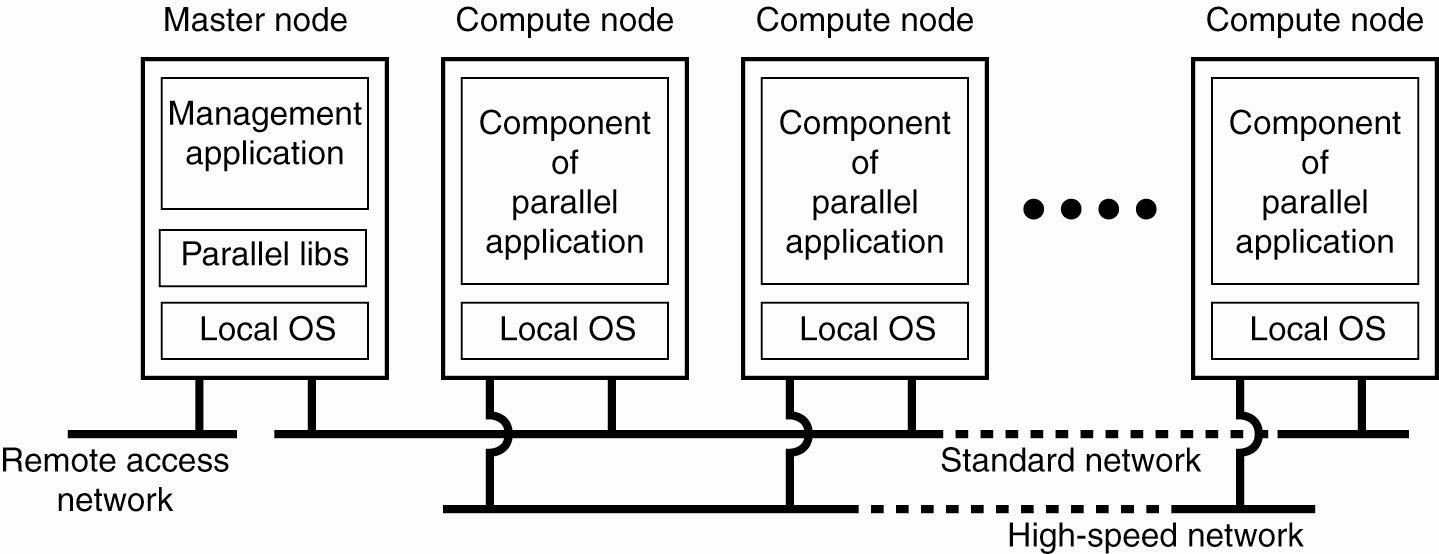
|  |  |
| --- | --- |
| Parallel Processing | Distributed Processing |
| Tightly coupled system. | Weakly coupled system. |
| Share a memory | Each processing unit has it’s own memory. |
| Has global clock control. | Has no global clock control. |
| Program run time comparatively Lower. | Program run time comparatively higher. |
| Amount of overhead is low. | Amount of overhead is high. |
| Interaction between I/O devices is high. | Interaction between I/O devices is very low. |
| Processor has unmapped local memory. | There are no unmapped local memory. |
| More expensive. | Less expensive. |
| Maintaining consistency is easier. | Maintaining consistency is difficult. |
| Difficult to expand. | Easier to expand. |

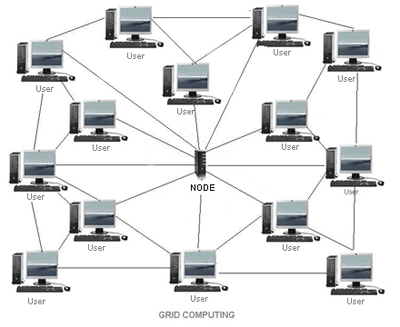
1. **What are the main types of distributed system? Explain with necessary diagram.**

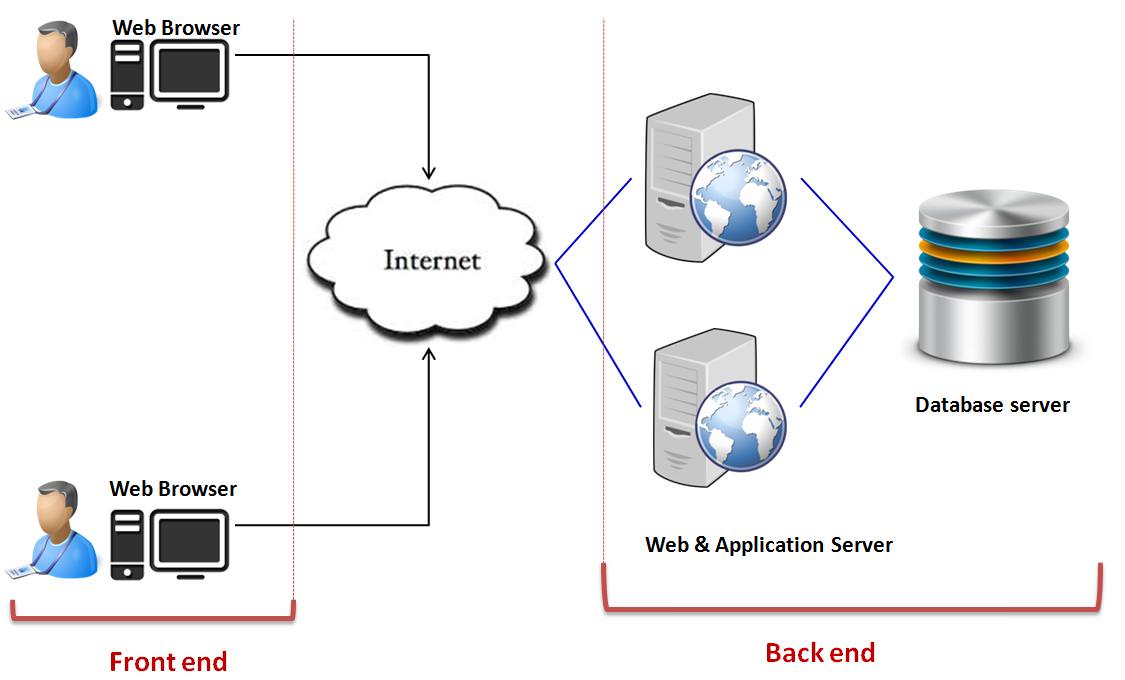
There are mainly three types of distributed system:

1. Distributed computing system
   1. Clusters
   2. Grids
   3. Clouds
2. Distributed information systems
3. Distributed embedded system

* Cluster Computing: A collection of similar processors (PCs, workstations) running the same operating system, connected by a high-speed LAN. They are several types of clusters. High performance clusters, Load balancing clusters, High availability clusters.



* Grid Computing: This is similar to clusters but processors are more loosely coupled, tend to be heterogeneous, and are not all in a central location. Here Problems are broken up into parts and distributed across multiple computers in the grid. It is highly heterogeneous with respect to hardware, software, networks, security policies, etc. 
* Cloud Computing: Cloud computing is the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a local server or a personal computer.



Distributed Information System: A set of information systems physically distributed over multiple sites, which are connected with some kind of communication network. Remote processes called clients access the servers to manipulate the information. Different models are used to communicate. The most usual are RPC and the object oriented RMI.

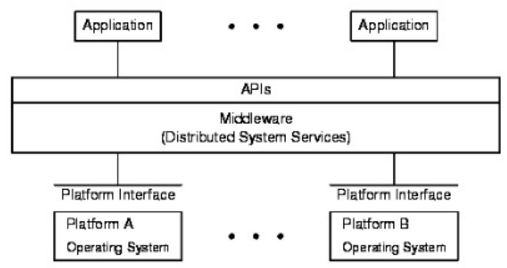
Distributed Embedded System: These are the distributed systems involving mobile and embedded computer devices. These systems characterized by their instability when compared to more traditional distributed systems.

1. **What are the roles of middleware in a distributed system?**

Middleware is computer software that provides services to software applications beyond those available from the operating system. Middleware makes it easier for software developers to perform communication and input/output, so they can focus on the specific purpose of their application. Middleware is the software that connects software components or enterprise applications. Typically, it supports complex, distributed business software applications.

So in short middleware is:

* Layer between OS and distributed applications.
* Hides complexity and heterogeneity of distributed system.
* Software that functions as a conversion or translation layer.
* Provides common programming abstraction and infrastructure for distributed applications.



* In the context of distributed system middleware is a software that provides services beyond those provided by the operating system to enable the various components of a distributed system to communicate and manage data.
* Middleware supports and simplifies complex distributed applications.
* It includes web servers, application servers, messaging and similar tools that support application development and delivery.

1. **Describe with block diagram the architectural evolution of parallel/ vector processors and parallel computers.**
2. **Write short notes on performance factors and throughput rate.**

Performance factors: here are five performance factors (Ic, p, m, k, t) which are influenced by four system attributes:

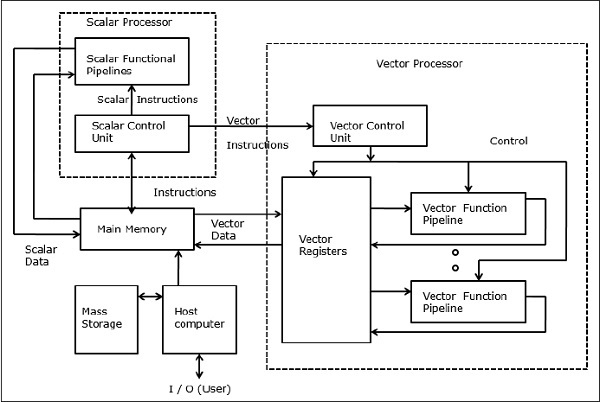
* MIPS: Millions of Instructions Per Second.
* FLOPS: FLoating point Operations Per Second.
* SPECint: SPEC (Standard Performance Evaluation Corporation) benchmarks that evaluate processor performance on integer arithmetic (1992).
* SPECfp: SPEC benchmarks that evaluate processor performance on floating point operations (2000).
* Whetstone: synthetic benchmarks to assess processor performance on floating point operations (1972).
* Dhrystone: synthetic benchmarks to asses processor performance on integer arithmetic (1984).

Throughput rate: Throughput Rate is indicates a number of programs a system can execute per unit time. It is often specified as programs/second. Throughput can be further measured separately for the system (Ws) and for the processor (Wp). The processor throughput is given as

1. **Describe architecture of vector supercomputer.**

In a vector computer, a vector processor is attached to the scalar processor as an optional feature. The host computer first loads program and data to the main memory. Then the scalar control unit decodes all the instructions. If the decoded instructions are scalar operations or program operations, the scalar processor executes those operations using scalar functional pipelines.

On the other hand, if the decoded instructions are vector operations then the instructions will be sent to vector control unit.



1. **What are the types of data dependency? Describe in brief.**

A dependence exists between program statements when the order of statement execution affects the results of the program. A data dependence results from multiple use of the same location(s) in storage by different tasks. Dependencies are important to parallel programming because they are one of the primary inhibitors to parallelism. There are three types of data dependencies:

* Flow dependency
* Anti-dependency
* Output dependency

**Flow dependency:** A Flow dependency, also known as a data dependency or true dependency or read-after-write (RAW), occurs when an instruction depends on the result of a previous instruction. For example:

i. A=3

ii. B=A

iii. C=B

Here instruction 3 is truly dependent upon instruction 2 and instruction 2 is truly dependent on instruction 1, therefore instruction 3 is also truly dependent on instruction 1.

**Anti dependency:** An anti-dependency, also known as write-after-read (WAR), occurs when an instruction requires a value that is later updated.In the following example, instruction 2 anti-depends on instruction 3 — the ordering of these instructions cannot be changed, nor can they be executed in parallel (possibly changing the instruction ordering), as this would affect the final value of A.

i. B=3

ii. A=B+1

iii. B=7

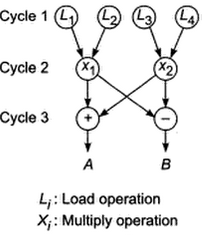
**Output dependency**: An output dependency, also known as write-after-write (WAW), occurs when the ordering of instructions will affect the final output value of a variable. In the example below, there is an output dependency between instructions 3 and 1 — changing the ordering of instructions in this example will change the final value of A, thus these instructions cannot be executed in parallel.

i. B=3

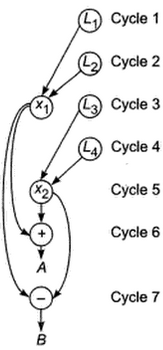
ii. A=B+1

iii. B=7

1. **Explain with an example the mismatch between software parallelism and hardware parallelism.**



Consider the example program graph above. There are eight instruction (four loads and four arithmetic operations) to be executed in three consecutive machine cycles. Four load operations are performed in the first cycle. Therefore the parallelism varies from 4 to 2 in three cycles. The average software parallelism is equal to 8/3=2.67 instructions per cycle in this example program.



Now consider execution of the same program by a two-issue processor which can execute one memory access (load or write) and one arithmetic (add, subtract, multiply etc) operation simultaneously. With this hardware restriction, the program must execute in seven machine cycles as shown above. Therefore, the hardware parallelism displays average value of 8/7=1.14 instructions executed per cycle. This demonstrates a mismatch between the software parallelism and hardware parallelism.

1. **How mismatch problem can be solved?**

To solve the mismatch problem between software parallelism and hardware parallelism, one approach is to develop compilation support, and the other is through hardware redesign for more efficient exploitation of parallelism. These two approaches must cooperate with each other to produce the best result.

1. **What are the different program flow mechanism? Define and also differentiate them.**

There are three different program flow mechanism.

a) Data flow

b) Control flow

c) Reduction

Conventional computers are based on a control flow mechanism by which the order execution is explicitly stated in the user programs. In control flow, tasks require completion before moving to next task.

Dataflow computers are based on a data driven mechanism which allows the execution of any instruction to be driven by data availability. In data flow, one component will not wait for other component to finish, all of them will work together in processing and managing data in streaming way.

In a string reduction model, each demander gets a separate copy of the expression for its own evaluation. A long string expression is reduced to a single value in a recursive fashion.

Difference between them are as following:

|  |  |  |  |
| --- | --- | --- | --- |
| Machine Model | Control flow | Dataflow | Reduction |
| Basic Definition | Conventional computer; token of control indicates when a statement should be executed. | Eager evaluation; statements are executed when all of their operands are available. | Lazy evaluation; statements are executed only when their result is required for another computation. |
| Advantages | Full control  The most successful model for commercial products. | Very high potential for parallelism. | Only required instructions are executed. |
| Complex data and control structures are easily implemented. | High throughput | High degree of parallelism. |
| Free from side effects. | Easy manipulation of data. |
| Disadvantages | In theory preventing run-time errors. | High control overhead. | Time needed to propagate demand tokens. |
| Difficult in preventing run-time errors. | Difficult in manipulating data structures. |

1. **Write down the advantage & disadvantage of distributed system over centralized system.**

Advantages of distributed system over centralized ones:

* Incremental growth: Computing power can be added din small increments.
* Reliability: If one machine crashes, the system as a whole can still survive where centralized system can not survive if one machine crashes.
* Speed: A distributed system may have more total computing power than a centralized system.
* Open system: This is the most important point and the most characteristic point of a distributed system. Since it is open system it is always ready to communicate with other systems. An open system that scales has an advantage over a perfectly closed and self-contained system.
* Economic: Distributed system’s microprocessors offer a better price/performance than centralized system’s microprocessor.

Disadvantages of distributed systems over centralized ones:

* Security: Centralized system is more secure than distributed system.
* Networking: If the networking gets saturated then problems with transmission will surface. Centralized system
* Software: There is currently very little software support for distributed system where centralized system has more support.
* Troubleshooting: Troubleshooting and diagnosing problems in a distributed system can also become more difficult, because the analysis may require connecting to remote nodes or inspecting communication between nodes. This may not possible. But in centralized system, it is very easy to troubleshoot.

1. **Describe briefly the different design issue of a distributed system.**

1. Openness

The openness of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways.The openness of distributed systems is determined primarily by the degree to which new resource-sharing services can be added and be made available for use by a variety of client programs.

2. Security

Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users.Their security is therefore of considerable importance. Security for information resources has three components: confidentiality, integrity, and availability.

3. Scalability

Distributed systems operate effectively and efficiently at many different scales, ranging from a small intranet to the Internet. A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users.

4. Failure handling

Computer systems sometimes fail. When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the intended computation. Failures in a distributed system are partial – that is, some components fail while others continue to function. Therefore the handling of failures is particularly difficult.

5. Concurrency

Both services and applications provide resources that can be shared by clients in a distributed system. There is therefore a possibility that several clients will attempt to access a shared resource at the same time. Object that represents a shared resource in a distributed system must be responsible for ensuring that it operates correctly in a concurrent environment. This applies not only to servers but also to objects in applications. Therefore any programmer who takes an implementation of an object that was not intended for use in a distributed system must do whatever is necessary to make it safe in a concurrent environment.

6. Transparency

Transparency can be achieved at two different levels. Easiest to do is to hide the distribution from the users. The concept of transparency can be applied to several aspects of a distributed system.

a) Location transparency: The users cannot tell where resources are located

b) Migration transparency: Resources can move at will without changing their names

c) Replication transparency: The users cannot tell how many copies exist.

d) Concurrency transparency: Multiple users can share resources automatically.

e) Parallelism transparency: Activities can happen in parallel without users knowing.

7. Reliability

One of the original goals of building distributed systems was to make them more reliable than single-processor systems. The idea is that if a machine goes down, some other machine takes over the job. A highly reliable system must be highly available, but that is not enough. Data entrusted to the system must not be lost or garbled in any way, and if files are stored redundantly on multiple servers, all the copies must be kept consistent. In general, the more copies that are kept, the better the availability, but the greater the chance that they will be inconsistent, especially if updates are frequent.

8. Performance

Always the hidden data in the background is the issue of performance. Building a transparent, flexible, reliable distributed system, more important lies in its performance. In particular, when running a particular application on a distributed system, it should not be appreciably worse than running the same application on a single processor. Unfortunately, achieving this is easier said than done.

1. **What do you mean by RPC?**
2. **Describe the basic RPC operation for parameter passing with necessary diagram.**
3. **Sketch the basic client-server model with different layer.**
4. **Describe with diagram the principle operation of a vector super computer.**
5. **How a baseline network can be generated recursively?**
6. **Draw and explain “Illiac mesh”, “Torus” and “CCC” network.**
7. **Write down the characteristics of CISC and RISC architectures.**
8. **Describe the architecture of VAX 8600 processor with figure.**
9. **What are the disadvantages of RISC processor?**
10. **Describe the concept of pipelining of superscalar processor of degree m=3.**