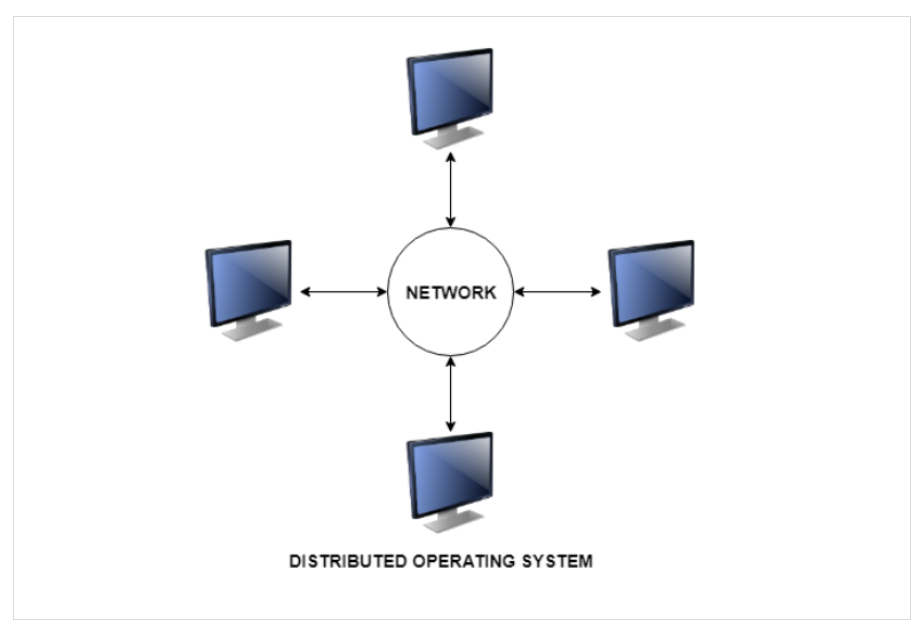
**Distributed Systems**

* A distributed system contains multiple nodes that are physically separate but linked together using the network. All the nodes in this system communicate with each other and handle processes in tandem. Each of these nodes contains a small part of the distributed operating system software.



**Types of Distributed Systems**

The nodes in the distributed systems can be arranged in the form of client/server systems or peer to peer systems. Details about these are as follows −

**Client/Server Systems**

In client server systems, the client requests a resource and the server provides that resource. A server may serve multiple clients at the same time while a client is in contact with only one server. Both the client and server usually communicate via a computer network and so they are a part of distributed systems.

**Peer to Peer Systems**

The peer to peer systems contains nodes that are equal participants in data sharing. All the tasks are equally divided between all the nodes. The nodes interact with each other as required as share resources. This is done with the help of a network.

**Goals :**

* Resource Accessibility
* Distribution Transparency
* Openness
* Scalability

**Resource Accessibility**

* Support user access to remote resources (printers, data files, web pages, CPU cycles) and the fair sharing of the resources
* Economics of sharing expensive resources
* Performance enhancement – due to multiple processors; also due to ease of collaboration and info exchange – access to remote services
* Groupware: tools to support collaboration
* Resource sharing introduces security problems.

**Distribution Transparency**

An external programmer or end user sees a distributed system as a single computational unit rather than as its underlying parts, allowing users to interact with a single logical device rather than being concerned with the system’s architecture.

* Makes the system more user friendly.
* A distributed system that appears to its users & applications to be a single computer system is said to be transparent.
* Users & apps should be able to access remote resources in the same way they access local resources.
* Transparency has several dimensions.

**Openness**

* How it can be extended.
* Definitions are language & machine independent
* Support communication between systems using different OS/programming languages; e.g. a C++ program running on Windows communicates with a Java program running on UNIX
* Communication is usually RPC-based.
* Interoperability: the ability of two different systems or applications to work together
* A process that needs a service should be able to talk to any process that provides the service.
* Multiple implementations of the same service may be provided, as long as the interface is maintained
* Portability: an application designed to run on one distributed system can run on another system which implements the same interface.
* Extensibility: Easy to add new components & features

**Scalability**

* Dimensions that may scale:
* With respect to size
* With respect to geographical distribution
* With respect to the number of administrative organizations spanned
* A scalable system still performs well as it scales up along any of the three dimensions.

## More about :

Distributed systems are commonly defined by the following key characteristics and features:

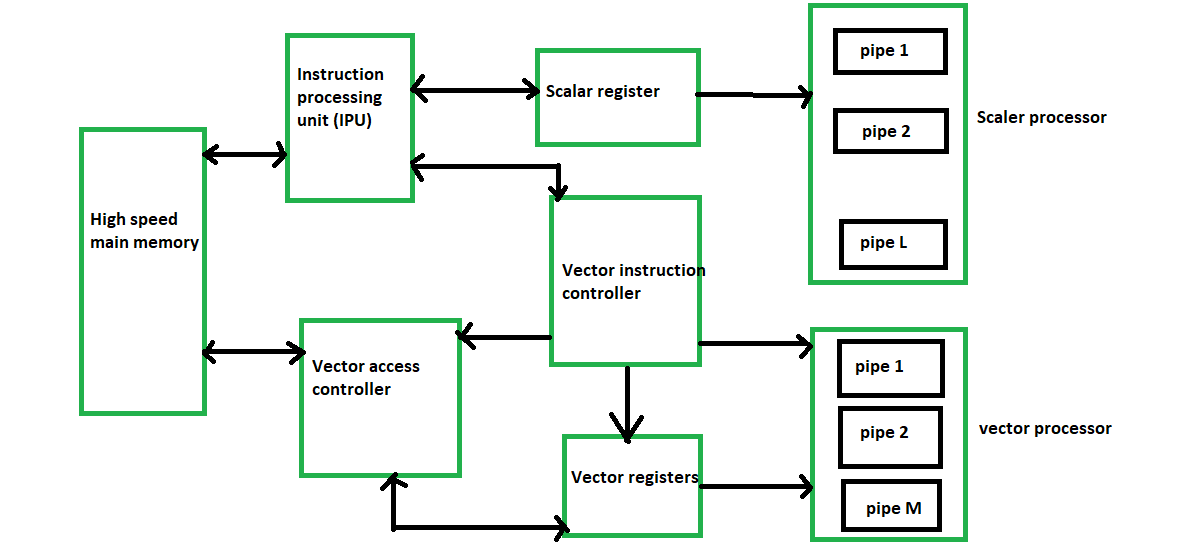
* **Scalability:**The ability to grow as the size of the workload increases is an essential feature of distributed systems, accomplished by adding additional processing units or nodes to the network as needed.
* **Concurrency:** Distributed system components run simultaneously. They’re also characterized by the lack of a “global clock,” when tasks occur out of sequence and at different rates.
* **Availability/fault tolerance:** If one node fails, the remaining nodes can continue to operate without disrupting the overall computation.
* **Transparency:** An external programmer or end user sees a distributed system as a single computational unit rather than as its underlying parts, allowing users to interact with a single logical device rather than being concerned with the system’s architecture.
* **Heterogeneity:** In most distributed systems, the nodes and components are often asynchronous, with different hardware, middleware, software and operating systems. This allows the distributed systems to be extended with the addition of new components.
* **Replication:** Distributed systems enable shared information and messaging, ensuring consistency between redundant resources, such as software or hardware components, improving fault tolerance, reliability and accessibility.

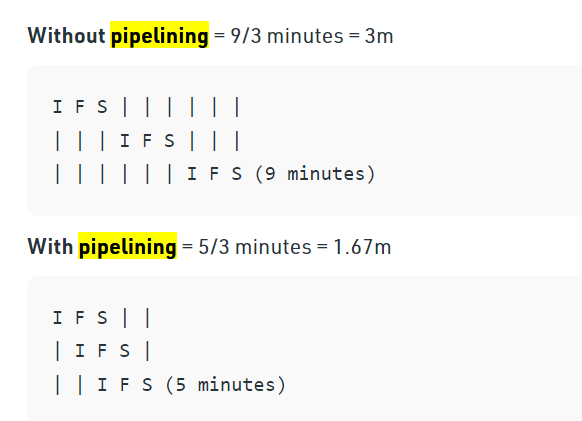
**Parallel Computing Architectures**

* In computers, parallel processing is the processing of program instructions by dividing them among multiple processors with the objective of running a program in less time. In the earliest computers, only one program ran at a time. A computation-intensive program that took one hour to run and a tape copying program that took one hour to run would take a total of two hours to run. An early form of parallel processing allowed the interleaved execution of both programs together. The computer would start an I/O operation, and while it was waiting for the operation to complete, it would execute the processor-intensive program. The total execution time for the two jobs would be a little over one hour.
* The next improvement was multiprogramming. In a multiprogramming system, multiple programs submitted by users were each allowed to use the processor for a short time. To users it appeared that all of the programs were executing at the same time. Problems of resource contention first arose in these systems. Explicit requests for resources led to the problem of the deadlock. Competition for resources on machines with no tie-breaking instructions lead to the critical section routine.

**Vector processing**

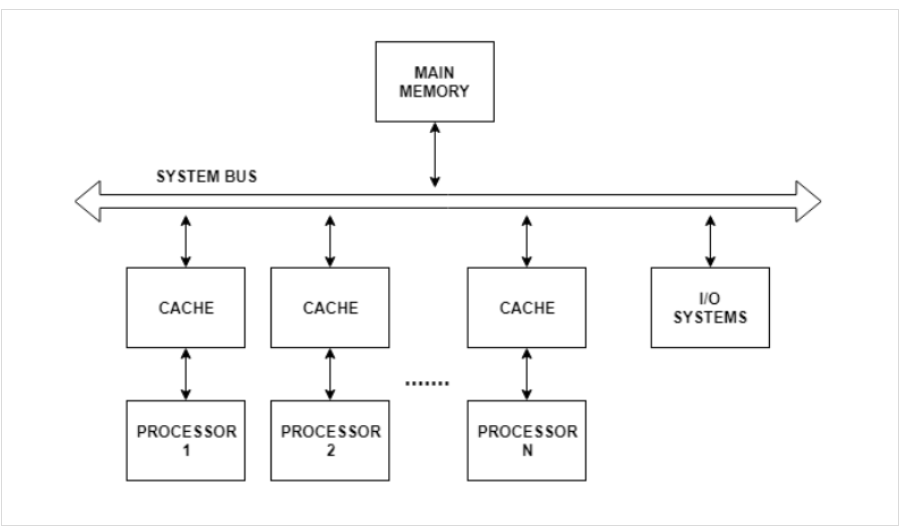
* Normal computer system
* ALU --- integer ALU1, ALU2, FPU--- SCALAR Execution units
* ADD R1,R2,R3 --- R1<-- R2+R3 SCALAR instruction R1,R2…R32
* Also called an Array Processor.
* Runs multiple mathematical operations on multiple data elements simultaneously.
* Common in supercomputers of the 1970’s 80’s and 90’s.
* Today most CPU designs contains at least some vector processing instructions, typically referred to as SIMD.
* Vector processing was another attempt to increase performance by doing more than one thing at a time.
* A Vector is a collection of scalar variables which is similar to array of variables of same type.
* These vector elements are stored at contiguous locations. A vector processor consists of hardware resources like vector registers, functional pipelines, processing elements & register counters for performing vector operations.
* Vector processing occurs when arithmetic or logical operators need to be performed on vectors.
* In this case, capabilities were added to machines to allow a single instruction to add (or subtract, or multiply, or otherwise manipulate) two arrays of numbers.



* Ability to process vectors, and related data structures such as matrices and multi-dimensional arrays, much faster than conventional computers
* Vector Processors may also be pipelined
* Typically operate on a few vectors elements per clock cycle in a pipeline.
* **Explain about pipelining**
* **Pipelining :** Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased. Simultaneous execution of more than one instruction takes place in a pipelined processor.
* Let us see a real life example that works on the concept of pipelined operation. Consider a water bottle packaging plant. Let there be 3 stages that a bottle should pass through, Inserting the bottle(**I**), Filling water in the bottle(**F**), and Sealing the bottle(**S**). Let us consider these stages as stage 1, stage 2 and stage 3 respectively. Let each stage take 1 minute to complete its operation.
* Now, in a non pipelined operation, a bottle is first inserted in the plant, after 1 minute it is moved to stage 2 where water is filled. Now, in stage 1 nothing is happening. Similarly, when the bottle moves to stage 3, both stage 1 and stage 2 are idle. But in pipelined operation, when the bottle is in stage 2, another bottle can be loaded at stage 1. Similarly, when the bottle is in stage 3, there can be one bottle each in stage 1 and stage 2. So, after each minute, we get a new bottle at the end of stage 3. Hence, the average time taken to manufacture 1 bottle is:
* ****

**Symmetric Multiple Processing**

* The next step in parallel processing was the introduction of multiprocessing. In these systems, the tasks are executed parallelly with a single processor. One processor (the master) was programmed to be responsible for all of the work in the system; the other (the slave) performed only those tasks it was assigned by the master.
* Two or more (64 today) processors
* Each processor is equally capable
* All processors share all the global resources available
* Single copy of the OS runs on these systems
* Issue : Scalability



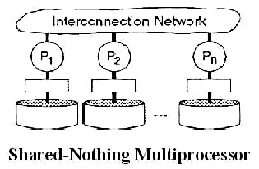
* Symmetric multiprocessing is also known as tightly coupled multiprocessing as all the CPU’s are connected at the bus level and have access to a shared memory.
* All the parallel processors in symmetric multiprocessing have their private cache memory to decrease system bus traffic and also reduce the data access time. Symmetric multiprocessing systems allow a processor to execute any process no matter where its data is located in memory. The only stipulation is that a process should not be executing on two or more processors at the same time.
* In general, the symmetric multiprocessing system does not exceed 16 processors as this amount can be comfortably handled by the operating system.

**Shared Memory : UMA vs. NUMA**

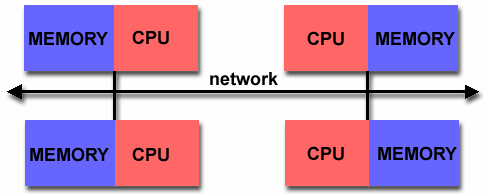
* Uniform Memory Access (UMA):
* Most commonly represented today by Symmetric Multiprocessor (SMP) machines
* Identical processors
* Equal access and access times to memory
* Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache cohérence 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

**Massively parallel processors**

* A large parallel processing system with a shared-nothing architecture



* Consist of several hundred nodes with a high-speed interconnection network/switch
* Each node consists of a main memory & one or more processors
* Runs a separate copy of the OS
* advantages: scalability, price
* 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.



* **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

There are several types of MPP database architectures, each with their own benefits:

* **Grid computing**– uses multiple computers in distributed networks. This type of architecture uses use resources opportunistically based on their availability. This architecture reduces costs for server space, but also limits bandwidth and capacity at peak times or when there are too many requests.
* **Computer clustering** – links the available power into nodes that can connect with each other to handle multiple tasks at once.

**Grid Computing**

* Similar to clusters but processors are more loosely coupled, tend to be heterogeneous, and are not all in a central location.
* Can handle workloads similar to those on supercomputers, but grid computers connect over a network (Internet?) and supercomputers’ CPUs connect to a high-speed internal bus/network
* Problems are broken up into parts and distributed across multiple computers in the grid – less communication between parts than in clusters.
* Grid Computing can be defined as a network of computers working together to perform a task that would rather be difficult for a single machine. All machines on that network work under the same protocol to act like a virtual supercomputer. The task that they work on may include analysing huge datasets or simulating situations which require high computing power. Computers on the network contribute resources like processing power and storage capacity to the network.
* Grid Computing is a subset of distributed computing, where a virtual super computer comprises of machines on a network connected by some bus, mostly Ethernet or sometimes the Internet. It can also be seen as a form of [Parallel Computing](https://www.geeksforgeeks.org/introduction-to-parallel-computing/) where instead of many CPU cores on a single machine, it contains multiple cores spread across various locations. The concept of grid computing isn’t new, but it is not yet perfected as there are no standard rules and protocols established and accepted by people.

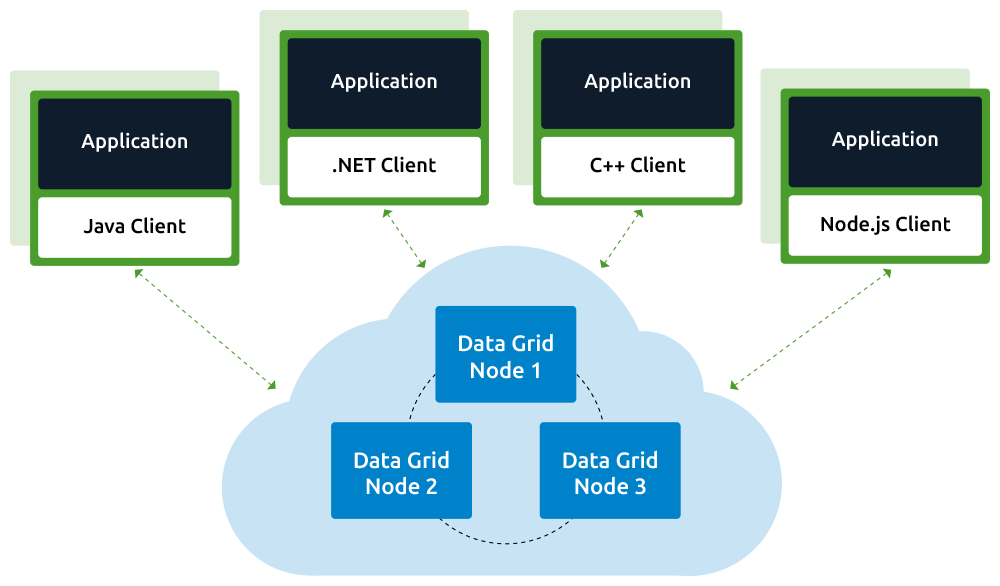
**Working:**   
A Grid computing network mainly consists of these three types of machines

1. **Control Node:**   
   A computer, usually a server or a group of servers which administrates the whole network and keeps the account of the resources in the network pool.
2. **Provider:**   
   The computer which contributes it’s resources in the network resource pool.
3. **User:**   
   The computer that uses the resources on the network.

When a computer makes a request for resources to the control node, control node gives the user access to the resources available on the network. When it is not in use it should ideally contribute it’s resources to the network. Hence a normal computer on the node can swing in between being a user or a provider based on it’s needs. The nodes may consist of machines with similar platforms using same OS called homogeneous networks, else machines with different platforms running on various different OS called heterogeneous networks. This is the distinguishing part of grid computing from other distributed computing architectures.

For controlling the network and it’s resources a software/networking protocol is used generally known as **Middleware**. This is responsible for administrating the network and the control nodes are merely it’s executors. As a grid computing system should use only unused resources of a computer, it is the job of the control node that any provider is not overloaded with tasks.

Another job of the middleware is to authorize any process that is being executed on the network. In a grid computing system, a provider gives permission to the user to run anything on it’s computer, hence it is a huge security threat for the network. Hence a middleware should ensure that there is no unwanted task being executed on the network.



**Cluster Computing**

* A computer cluster is a group of linked computers, working together closely thus in many respects forming a single computer. The components of a cluster are connected to each other through fast local area networks.
* Cluster computing is a form of computing in which a group of computers are linked together so that they can act like a single entity. It is the technique of linking two or more computers into a network (usually through a local area network) in order to take advantage of the parallel processing power of those computers.
* A node – Either a single or a multiprocessor network having memory, input and output functions and an operating system.

**Cluster classification :**

* High performance (HP) clusters:
  + With a high performance computing cluster, multiple computers, or nodes, are linked together through a local-area network (LAN) to create an HPC cluster architecture. HP clusters use computer clusters and supercomputers to solve advance computational problems.
  + They are designed to take benefit of the parallel processing power of several nodes.
* Load-balancing clusters:
  + Here workload is equally distributed across multiple installed servers in the cluster network. Incoming requests are distributed for resources among several nodes running similar programs or having similar content. This type of distribution is generally used in a web-hosting environment.
* High Availability (HA) Clusters :
  + A group of clusters which ensure to maintain very high availability. This are very much reliable.
  + High-availability clusters (also known as Failover Clusters) are implemented for the purpose of improving the availability of services which the cluster provides. It provides redundancy and eliminate single points of failure

Single System Image (SSI)

A single system image is the illusion created by software or hardware, that presents a collection of resources as one, (single)more powerful resource.

SSI makes the cluster appear like a single machine to the user, to applications, and to the network.

A cluster without a SSI is not a cluster

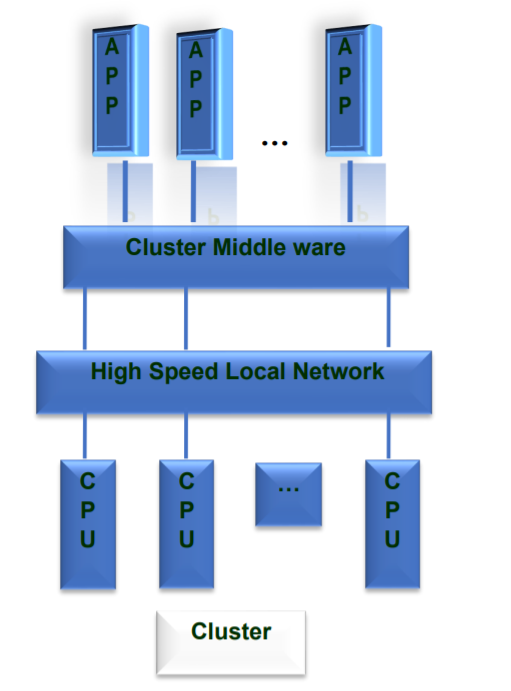
Cluster consists of:

Nodes

Network

OS

Cluster middleware



Advantages of Cluster Computing:

1. High Performance: The systems offer better and enhanced performance than that of mainframe computer networks.

2. Easy to manage: Cluster Computing is manageable and easy to implement.

3. Scalable: Resources can be added to the clusters accordingly.

4. Expandability: Computer clusters can be expanded easily by adding additional computers to the network. Cluster computing is capable of combining several additional resources or the networks to the existing computer system.

5. Availability: The other nodes will be active when one node gets failed and will function as a proxy for the failed node. This makes sure for enhanced availability.

6. Flexibility: It can be upgraded to the superior specification or additional nodes can be added.