5.4 CLUSTERS AND WAREHOUSE SCALE COMPUTERS

Warehouse-scale computers form the foundation of internet services. The present days WSCs act as one giant machine. The main parts of a WSC are the building with the electrical and cooling infrastructure, the networking equipment and the servers.

> ***A cluster is a collection of desktop computers or servers connected together by a local area network to act as a single larger computer. A warehouse-scale computer (WSC) is a cluster comprised of tens of thousands of servers***

WSCs as Servers

The following features of WSCs that makes it work as servers:

- **Cost-performance**: Because of the scalability, the cost-performance becomes very critical. Even small savings can amount to a large amount of money.
- **Energy efficiency:** Since large numbers of systems are clustered, lot of money is invested in power distribution and for heat dissipation. Work done per joule is critical for both WSCs and servers because of the high cost of building the power and mechanical infrastructure for a warehouse of computers and for the monthly utility bills to power servers. If servers are not energy-efficient they will increase
  - cost ofelectricity
  - cost of infrastructure to provide electricity
  - cost of infrastructure to cool the servers.
- **Dependability via redundancy:** The hardware and software in a WSC must collectively provide at least 99.99% availability, while individual servers are much less reliable. Redundancy is the key to dependability for both WSCs and servers. WSC architects rely on multiple cost-effective servers connected by a low cost network and redundancy managed by software. Multiple WSCs may be needed to handle faults in whole WSCs. Multiple WSCs also reduce latency for services that are widely deployed.
- **Network I/O:** Networking is needed to interface to the public as well as to keep data consistent between multiple WSCs.

Interactive and batch-processing workloads: Search and social networks are interactive and require fast response times. At the same time, indexing, big data analytics etc. create a lot of batch processing workloads also. The WSC workloads must be designed to tolerate large numbers of component faults without affecting the overall performance and availability.

Differences between WSCs and data centers

Data Centers	WSCs
Datacentreshostsservicesformultiple providers.	WSCsarerunbyonlyone client.
Therewillbelittlecommonalitybetween Homogenous software.	hardwareandsoftware hardwareand management.
Thirdpartysoftwaresolutions.	In-house middleware.

WSC are not servers:

The following features of WSCs make them different from servers:

Ample parallelism:

Servers need not to worry about the parallelism available in applications to justify the amount of parallel hardware.

But in WSCs most jobs are totally independent and exploit request-level parallelism.

Request-Level parallelism (RLP) is a way of representing tasks which are set of requests which are to be to run in parallel.

Interactive internet service applications, the workload consists of independent requests of millions of users.

Also, the data of many batch applications can be processed in independent chunks, exploiting data-level parallelism.

Operational costs count:

Server architects normally designsystems for peak performance withina cost budget.

Power concerns are not too much as long as the cooling requirements are

maintained. The operational costs are ignored.

- WSCs, however, have a longer life times and the building, electrical and cooling costs are very high.
- So, the operational costs cannot be ignored. A
- ll these add up to more than 30% of the costs of a WSC in 10 years.
- Power consumption is a primary, not secondary constraint when designing the WSC system.
- **Scale and its opportunities and problems**:
- The WSCs are massive internally, so it gets volume discounts and economy of scale, even if there are not too many WSCs.
- On the other hand, customized hardware for WSCs can be very expensive, particularly if only small numbers are manufactured.
- The economies of scale lead to cloud computing, since the lower per-unit costs of WSCs lead to lower rental rates.
- Even if a server had a Mean Time To Failure (MTTF) of twenty five years, the WSC architect should design for five server failures per day.

Architecture of WSC

The height of the servers is measured by **rack units.** A typical rack is 42 rack units. But the standard dimension to hold the servers is 48.26 cm.

1 rack unit (U)=1.75 inches or 44.45 mm.

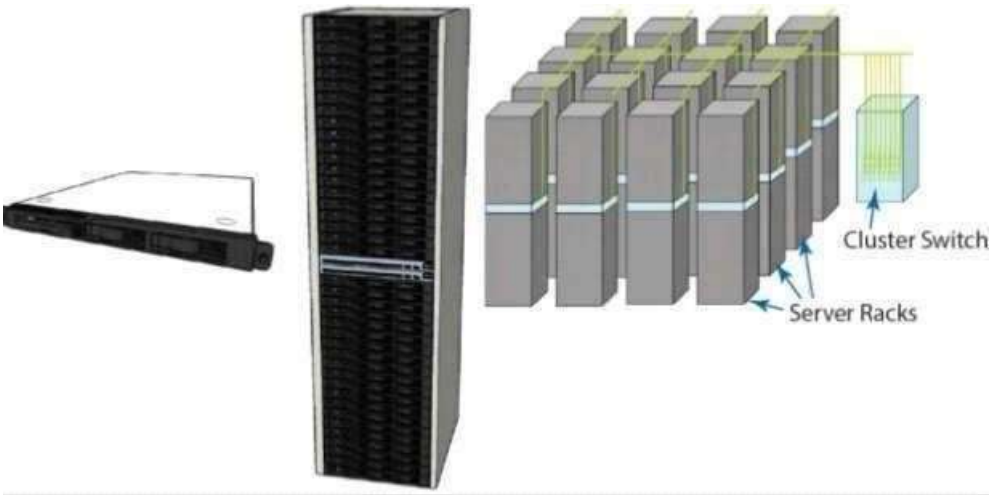

Fig 1: Architecture of WSCs

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

The fig 1 shows a WSC system with 1Unit server, 7 inch rack with an Ethernet switch. This figure shows a high end server. But low end servers are of 1U size mounted within a rack and connected with Ethernet switch. These rack level switches use 1 or 10 Gbps links with a number of uplink connections to cluster level switches. The second level switching can span more than 10,000 individual servers.

Programming model for WSC

There is a high variability in performance between the different WSC servers because of:

- varying load on servers
- file may or may not be in a file cache
- distance over network can vary
- hardware anomalies

A WSC will start backup executions on other nodes when tasks have not yet completed and take the result that finishes first. Rely on data replication to help with read performance and availability. A WSC also has to cope with variability in load. Often WSC services are performed with in-house software to reduce costs and optimize for performance.

Storage of WSC

- A WSC uses local disks inside the servers as opposed to network attached storage (NAS). The Google file system (GFS) uses local disks and maintains at least three replicas to improve dependability by covering not only disk failures, but also power failures to a rack or a cluster of racks by placing the replicas on different clusters.
- A read is serviced by one of the three replicas, but a write has to go to all three replicas.
- Google uses a relaxed consistency model in that all three replicas have to eventually match, but not all at the same time.

WSC networking

- A WSC uses a hierarchy of networks for interconnection.
- The standard rack holds 48 servers connected by a 48-port Ethernet switch. A rack switch has 2 to 8 uplinks to a higher switch.
- So the bandwidth leaving the rack is 6 (48/8) to 24 (48/2) times less than the bandwidth within a rack.
- There are array switches that are more expensive to allow higher connectivity.
- There may also be Layer 3 routers to connect the arrays together and to the Internet.
- The goal of the software is to maximize locality of communication relative to the rack.

Performance

Power Utilization Effectiveness (PUE) is widely used metric to estimate the performance of WSCs.

$$\mathbf{PUE} = \frac{Total\_utility\_power}{IT\_equipment\_power}$$

Bandwidth is an important metric as there may be many simultaneous user requests or metadata generation batch jobs. Latency is also equally important metric as it is seen by users when they make requests. Users will use a search engine less as the response time increases. Also users are more productive in responding to interactive information when the system response time is faster as they are less distracted.

5.3 GRAPHICS PROCESSING UNITS

GPU is designed to lessen the work of the CPU and produce faster video and graphics. GPU can be thought as an extension of CPU with thousands of cores. A GPU is extensively used in a PC on a video card or motherboard, mobile phones, display adapters, workstations and game consoles. They are mainly used for offloading computation intensive application. This is also known as a **visual processing unit (VPU).**

A Graphics Processing Unit (GPU) is a single-chip processor primarily used to manage and boost the performance of video and graphics. It is a dedicated parallel processor for accelerating graphical and deeper computations.

Differences between CPU and GPU

GPU	CPU
They facilitate highly parallel operations.	This supports serial execution of programs.
This has more number of cores(in thousands).	This has less number of cores.
They need special faster interfaces to facilitate faster data transfers.	No such special interface are required.
They have deeper pipelines	They have comparatively shallow.

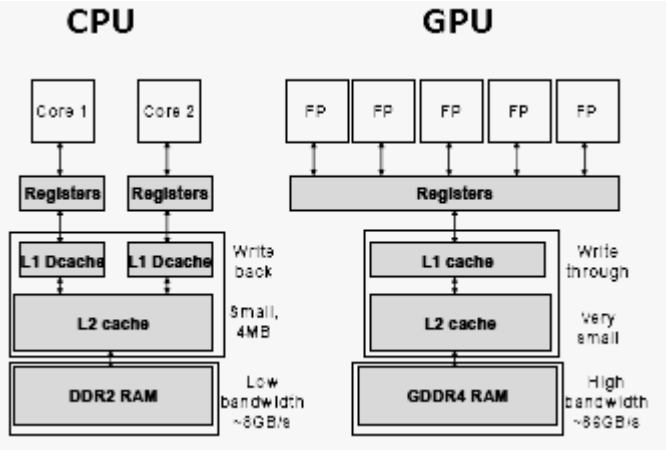

Fig 1: CPU vs GPU architecture

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

GPU features

The following are prominent features of GPU:

- 2-D or 3-Dgraphics
- Digital output to flat panel display monitors
- Texture mapping
- Application support for high-intensity graphics software such as AutoCAD
- Rendering polygons
- Support for YUVcolor space
- Hardware overlays
- MPEG decoding

Development of GPU

- The first GPU was developed by NVidia in 1999 and named as GeForce 256.
- This GPU model could process 10 million polygons per second and had more than 22 million transistors.
- This is a single-chip processor with integrated transform, drawing and BitBLT support, lighting effects, triangle setup /clipping and rendering engines.

- The GPU is connected to the CPU and is completely separate from the motherboard.
- The RAM is connected through the Accelerated Graphics Port (AGP) or the PCI express bus.
- Sometimes, GPUs are integrated into the north bridge on the motherboard and use the main memory as a digital storage area, but these GPUs are slower and have poorer performance.
- The accelerated memory in GPU is used for mapping vertices and can also supports programmable shade implementing textures, mathematical vertices and accurate color formats.
- Applications such as Computer-Aided Design (CAD) can process over 200 billion operations per second and deliver up to 17 million polygons per second.
- The main configurations of GPU processor are: Graphics coprocessor which is independentof CPU and Graphics accelerator that is based oncommands from CPU.

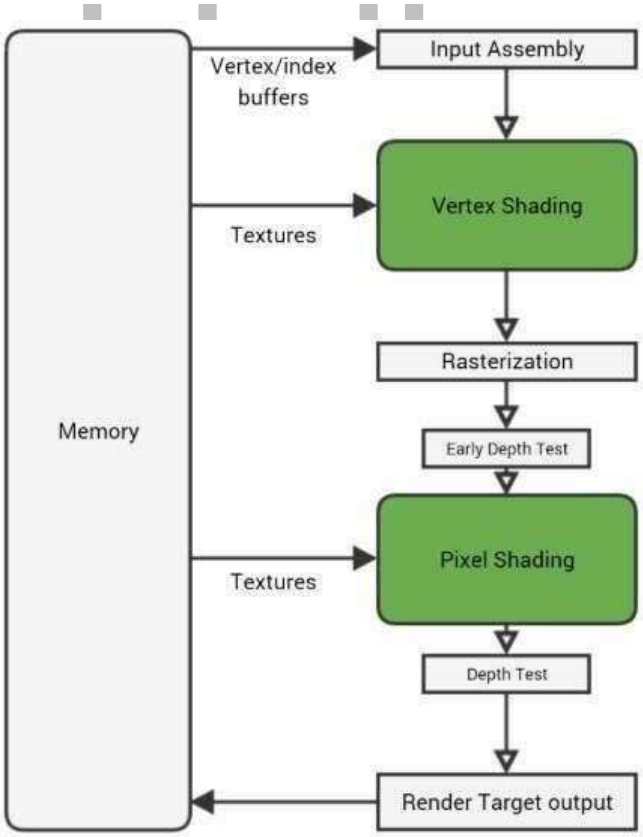

Fig 2: GPU Pipeline

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

Input Assembler stage

- This stage is the communication bridge between the CPU and GPU.
- It receives commands from the CPU and also pulls geometry information from system memory.
- It outputs a streamof vertices in object space with all their associated information.

Vertex Processing

- This processes vertices performing operations like transformation, skinning and lighting.
- A vertex shade takes a single input vertex and produces a single output vertex.

Pixel Processing

- Each pixel provided by triangle setup is fed into pixel processing as a set of attributes which are used to compute the final color for this pixel.
- The computations taking place here include texture mapping and math operations

Output Merger Stage

- The output-merger stage combines various types of output data to generate the final pipeline result.

5.5 MULTIPROCESSOR NETWORK TOPOLOGIES

Multiprocessor system consists of multiple processing units connected via some interconnection network plus the software needed to make the processing units work together. There are two major factors used to categorize such systems:

- The processing units
- The interconnectionnetwork

A number of communication styles exist for multiprocessing networks. These can be broadly classified according to the communication model as shared memory (single address space) versus message passing (multiple address spaces).

Design Issues of Interconnection Networks

The important issue in the design of multiprocessor systems is how to cope with the problemofanadequate designofthe interconnection network in order to achieve the desired performance at low cost. The choice of the interconnection network may affect several characteristics of the system such as node complexity, scalability and cost etc. The following are the issues which should be considered while designing an interconnection network.

- **Dimension and size of network:** It should be decided how many processing element are there in the network and what the dimensionality of the network is i.e. with how many neighbors, each processor is connected.
- **Symmetry of the network:** It is important to consider whether the network is symmetric or not i.e., whether all processors are connected with same number of processing elements or the processing elements of corners or edges have different number of adjacent elements.
- **Message Size:** Message size is dependent on the amount of data that can be transferred in one unit time.
- **Data transfer Time:** The time taken for a message to reach to another processor, Whether this time is a function of link distance between two processors or it depends upon the number of nodes coming in between are chief factors

- **Startup Time:** It is the time of initiation of the process.

Performance parameters

- **Number of nodes (N):** The number of nodes in a multiprocessor network plays a dynamic role by virtue of which the performance of the system is evaluated. Higher number of nodes means higher complexity but higher is the system performance. Therefore, number of processors should be optimal.
- **Node degree (D):** The node degree of the network is defined as the number of edges connected with the nodes. It is the connectivity among different nodes in a network. The connectivity of the nodes determines the complexity of the network. The greater number of links in the network means greater is the complexity. If the edge carries data fromthe node, it is called out degree and if this carries data into the node then it is called in degree.
- **Diameter (D):** The network diameter is defined as the maximum shortest path between the source and destination node. The path length is measured by the number of links traversed. This virtue is important in determining the distance involved in communication and hence the performance of parallel systems. The low diameter is always better because the diameter puts a lower bound on the complexity of parallel algorithms requiringcommunication between arbitrary pairs ofnodes.
- **Cost (C):** It is defined as the product of the diameter and the degree of the node for a symmetric network.

Cost (C) = Diameter * Degree = D * d

Greater number of nodes means greater the cost of the network. It is good creation to measure the hardware cost and the performance of the multiprocessor network and gives more insight to design a cost-effective parallel system.

- **Extensibility**

It is virtue which facilitates large sized system out of small ones with minimum changes in the configuration of the nodes. It is the smallest increment by which the system can be expanded in a useful way. A network with large number of links or a large node degree tends to increase the hardware cost. Expandability is an important

parameter to evaluate the performance of a multiprocessor system. The feasibility to extend a system while retaining its topological characteristics enables to design large scale parallelsystems.

Network Topologies

The multiprocessor networks are classified in two broad categories based on their topological properties. These are given below:

- Cube based network
- Linearly Extensible Network

Cube Based Network

- The cube based architectures are widely used networks in parallel systems. They have good topological properties such as symmetry, scalability and possess a rich interconnection topology. The types of cube based networks are:
- **Binary hypercube or n-cube:**
- This is a loosely coupled parallel multiprocessor based on the binary n-cube network.
- An n-dimensional hypercube contains 2^n nodes and has n edges per node.
- In hypercube, the number of communication links for each node is a logarithmic function of the total number of nodes.

The hypercube organization has low diameter and high bisection width at the expense of the numberofedges pernode andthe lengthofthe longestedge.

- The length of the longest edge in a hypercube network increases as the number of nodes in the network increases.
- The node degree increases exponentially with respect to the dimension, making it difficult to consider the hypercube a scalable architecture.
- The major drawback of the hypercube is the increase in the number of communication links for each node with the increase in the total number of nodes.
- The bellow Fig 1 represents the Hypercube.

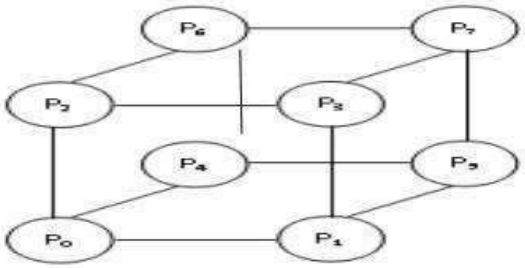

Fig 1: Hypercube

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

Cube Connected Cycle (CCC)

- The CCC architecture is an attractive parallel computation network suitable for VLSI implementation while preserving all the desired features of hypercube.
- The CCC is constructed from the n- dimensional hypercube by replacing each node in hypercube with a ring containing n node.
- Each node in a ring then connects to a distinct node to one of the n dimensions.
- The advantage of the cube- connected cycles is that nodes degree is always 3, independent of the value of n. This architecture is modified from hypercube i.e. a 3-cube is modified to form a 3-cube-connected cycles (CCC) restricted the node degree to 3.
- The idea is to replace the corner nodes (vertices) of the 3-cube with a ring of 3-nodes.
- In general one can construct k-cube-connected cycles from a k-cube with $n=2^k$ rings nodes.

Folded Hyper Cube (FHC)

- The FHC is the variation of the hypercube network and constructed by introducing some extra links to the hypercube.
- Halved diameter, better average distance, shorter delay in communication links, less message traffic density, lower cost make it very promising.
- The hardware overhead is almost 1/n, n being the dimensionality of the hypercube,

which is negligible for large n.

- Optimal routing algorithms are developed are developed and proven to be remarkably more efficient than those of the conventional n-cube.
- A folded hypercube of dimension n is called FHC (n).
- The FHC (n) is a regular network of node connectivity (n+1)and the hypercube of degree 3 is converted to FHC (n) network. The bellow Fig 2 represents Folded Hypercube.
- Extended versions of FHC (n) is called Extended Folded Cube (EFC). The EFC has better properties than the other variations of basic hypercube in terms of parameter.
- It has constant node degree, smaller diameter, and lower cost and also it maintains several numerous desirable characteristics including symmetry, hierarchical, expansive, recursive.

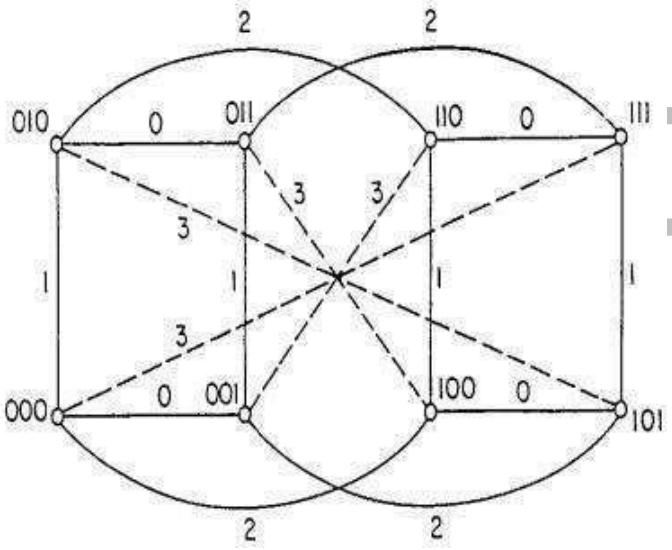

Fig 2: Folded Hypercube

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

Crossed Cube

- The Crossed Cube (CC) has the same node and link complexity as the hypercube and has most of its desirable properties including regularity, recursive structure, partition

ability, strongconnectivityandability to simulate otherarchitectures.

- Its diameter is only half of the diameter of the hypercube.
- Mean distance between vertices is smaller and it can simulate a hypercube through dilation 2 embedding.
- The basic properties of the CC, optimal routing and broadcasting algorithms are developed. The below fig 3 shows crossed cube.
- The CC is derived from a hypercube by changing the way of connection of some hypercube links.
- The diameter of CC is almost half of that of its corresponding hypercube.

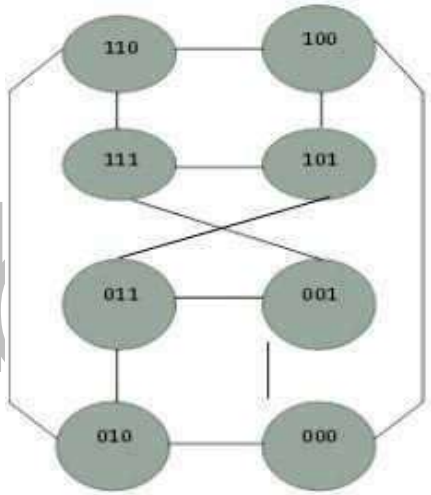

Fig 3: Crossed Cube

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

Reduced Hypercube (RHC)

- The RH (k, m) is obtained from the n- dimensional hypercube by reducing node edges in hypercube by following rules where k+2m= n.
- The lower VLS) complexity of RCs permit the construction of systems with more processing elements than are found in conventional hypercube.
- There are clusters and each cluster is a conventional k- dimensional hypercube.

Of the higher n-k=2m dimensions, a node has only one direct connection is decided by the leftmost mbits in the k-bits field, i.e., the (2i + k) dimension, where i is the value of the

m-bit binary number.

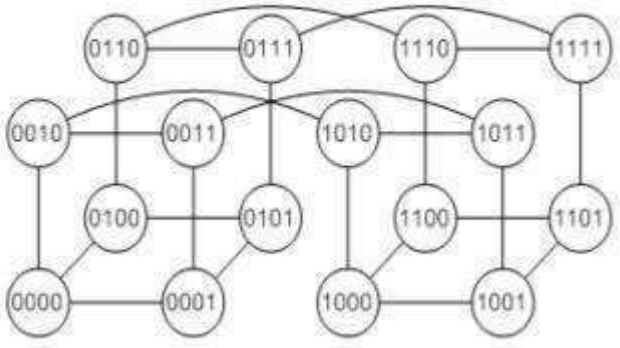

Fig 4: Reduced Hypercube Hierarchal Cube Network (HCN)

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

- The Hierarchical Cube Network (HCN) is interconnection network for large-scale distributed memory multiprocessors. The above fig 4 represents the Reduced Hypercube Hierarchal Cube Network(HCN).
- HCN has about three-fourths the diameter of a comparable hypercube, although it uses about half as many links per node-a fact that has positive ramifications on the implementation of HCN-connected systems.
- The HCN (n, n)has 2n clusters, where each cluster is an n-cube.
- Each node in the HCN (n, n) has n+1 links connected to it. n links are used inside the cluster. The additional links are used to connect nodes among clusters.
- The advantage of HCN is that the number of links required is reduced approximately to half as many links per node and the diameter is reduced to about three-fourth of a corresponding hypercube. The below Fig 5 shows the Hierarchical Cube Network.

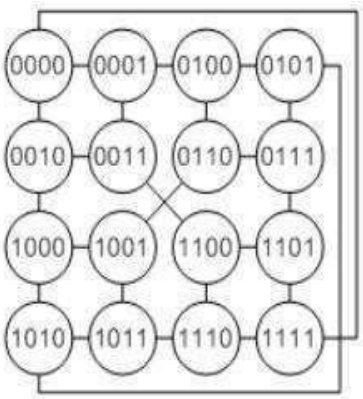

Fig 5: Hierarchical Cube Network

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

Dual Cube (DC)

- The DC is a new interconnection topology for large-scale distributed memory Multiprocessors that reduces the problem of increasing number of links in the large-scale hypercube network.
- This preserves most of the topological properties of the hypercube network.
- The DC shares the desired properties of the hypercube, however increases tremendously the total number of nodes in the systemwith limited links per node.
- The key properties of hypercube are also true in the dual-cube: each node can be represented by unique binary number such that two nodes are connected by an edge if and only if the two binary numbers differ in one bit only.
- However, the size of the dual-cube can be as large as eight thousands with up to eight links pernode.
- A dual-cube uses binary hypercube as basic components. Each such hypercube component is referred to as a **cluster.**
- Assume that the number of nodes in a cluster is 2^m. In a dual cube, there are two
- Classes with each class consisting of 2^m clusters.
- The total number of nodes is 2^m or 2^{m+1}. Therefore, the nodes address has 2m+1 bits

- The leftmost bit is used to indicate the type of the class (class 0 and class 1).
- For the class 0, the rightmost m bits are used as the node ID within the cluster.
- Each node in cluster of class 0 has one and only one extra connection to a node in a cluster of class 1.

Meta Cube (MC)

- The MC is an interconnection topology for a very large parallel system. Meta cube network has two level cube structures. An MC (k, m) network can connect $2k+m^{2k}$ nodes with (k+m) links per node where k is the dimension of the high-level cubes (classes) and m is the dimension of the low-level cubes (clusters).

In this network, the number of nodes is much larger than the hypercube with a small number of links per node

- An MC network is a symmetric network with short diameter, easy and efficient routing.
- Similar to that of the hypercube.
- The meta cube has tremendous potential to be used as an interconnection network for very large scale parallel computers since the meta cube can connect hundreds of millions nodes with up to six links per node and it keeps some desired properties of the hypercube that are useful efficient communication among the nodes.

Folded Dual Cube (FDC)

- The FDC is a new cube based Interconnection topology for parallel systems with reduced diameter, cost and constructed from DC and FHC.
- The FDC is a graph Fr (V, E), where V represents a set of vertices and E represent a set of links.
- The FDC is to be slightly greater than Dual cube but quite less than HC and FHC.
- Diameter of FDC is found to be smaller than that of Dual cube and with the comparison of Dual cube, HC and FHC.

FDC exhibits quite a good improvement in broadcast time over its parent networks

with millions ofnodes.

- The cost of the FDC topology is found to be less. The FDC will help to speed up the overall operation of large scale parallel systems.

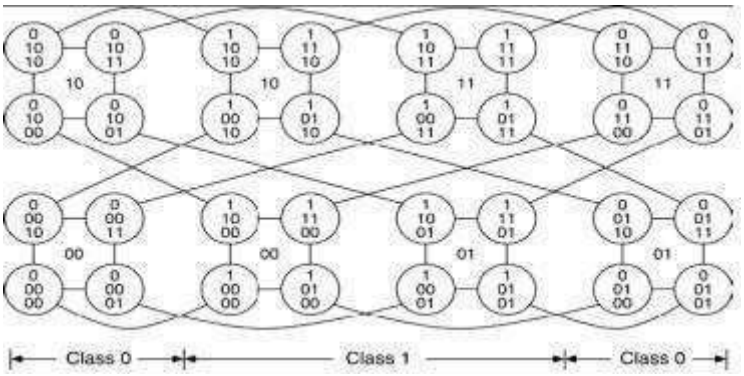

Fig 6: Folded Dual Cube

Folded Meta cube (FMC)

- The FMC is an efficient large scale parallel interconnection topology with better features such as reduced diameter, cost, improved broadcast time and constructed from MC. The above fig 6 shows Folded Dual Cube.
- The FMC is a graph G (V, E), where V represents a set of vertices and E represent a set of links.
- The FMC is to be slightly greater than Meta cube but quite less than HC and FHC.
- Diameter of FMC is found to be smaller than that of Meta cube.
- FMC exhibits quite a good improvement in broadcast time over its parent network while connecting millions of nodes.
- The cost of the FMC is found to be less and will help to speed the overall operation of large scale parallelsystems.

Necklace Hypercube (NH)

- NH is an array of processors attached to each two adjacent nodes of the hypercube network.
- It is highly scalable architecture while preserving most of the desirable properties of hypercube such as logarithmic diameter, fault tolerance etc.

- It has also some other properties such as hardware scalability and efficient VLSI layout that make it more attractive than an equivalent hypercube network.
- The Necklace-Hypercube is an undirected graph which has a necklace of processors to each edge of hypercube. The below Fig 7 shows the Necklace Hypercube.
- The necklace length may be fixed or variable for different edge necklaces.

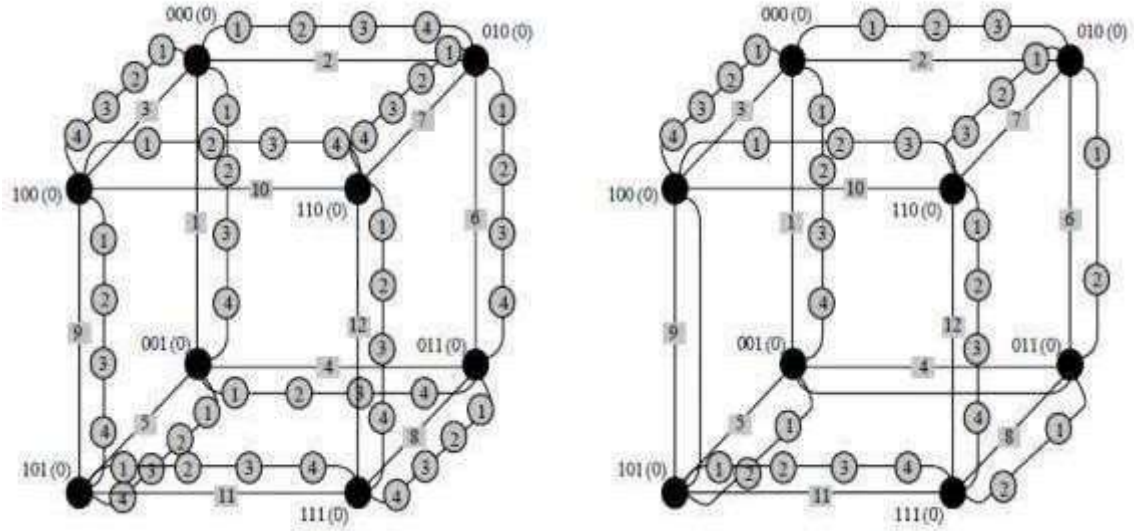

Fig 7: Necklace Hypercube

Source: Miles J. Murdocca and Vincent P. Heuring, — "Computer Architecture and Organization: An Integrated approach"

Linearly Extensible Network

The Linearly Extensible Networks is another class of multiprocessor architectures which reduces some of the drawbacks of HC architectures. The complexity of these networks is lesser as they do not have exponential expansion. Besides the scalability, other parameters to evaluate the performance of such networks are degree, number of nodes, diameter, bisection width and fault tolerance. Selection of a better interconnection network may have several applications with lesser complexities and improved power-efficiency.

Linear Array (LA)

- It is one dimensional network having the simplest topology with n-nodes having n-1

communication links.

- The internal nodes have degree 2 and the termination nodes have degree1.
- The diameter is n-1, which is long for large n and the bisection width is 1.
- It is asymmetric network. Linear array are the simplest connection topology.
- As the diameter increases linearly with respect to n, it should not be used for large n.

For every small n, it is rather economical to implement a linear array.

Binary Tree (BT)

- A binary tree is either empty or consists of node called the root together with two binary trees called left sub tree and the right sub tree.
- When h is equal to height of a binary tree then maximum leaves are equal to 2^h and maximum nodes are $2^{h+1}-1$.
- In a binary tree network there is only one path between any two nodes.
- The binary tree is scalable architecture with a constant node degree and constant bisection width. In general, an n-level, complexity balanced binary tree should have N=2n-1 nodes.
- The maximum node degree is 3 and the diameter is 2(n-1). But has a poor bisection width of 1.

Ring (R)

- This is a simple linear array where the end nodes are connected. It is equivalent to mesh with wrap around connections.
- The data transfer in a ring is normal one direction. A ring is obtained by connecting the two terminal nodes of a linear array with one extra link.
- A ring network can be uni-or bidirectional and it is symmetric with a constant.
- It has a constant node degree of d=2, the diameter is N/2for a bidirectional ring and N for unidirectionalring.
- A ring network has a constant width 2.

Linearly Extensible Tree (LET)

- The Linearly Extensible Tree (LET) architecture exhibits better connectivity, lesser number of nodes over cube based networks.
- The LET network has low diameter, hence reduce the average path length traveled by all message and contains a constant degree per node. The below Fig 8 shows Linearly Extensible Tree.
- The LET network grows linearly in a binary tree like shape.
- In a binary tree the number of nodes at level n is 2n whereas in LET network the number is (n+1).

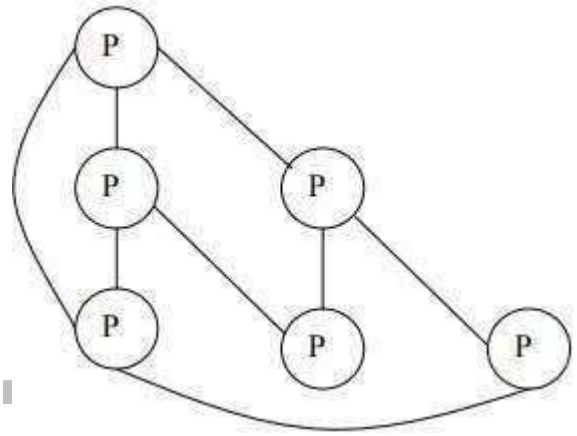

Fig 8: **Linearly Extensible Tree**

5.2 MULTICORE AND SHARED MEMORY MULTIPROCESSORS

A multi-core processor is a single computing component with two or more independent processing units called cores, which read and execute program instructions. A shared-memory multiprocessor is a computer system composed of multiple independent processors that execute different instruction streams.

Multi-core is usually the term used to describe two or more CPUs working together on the same chip. It is a type of architecture where a single physical processor contains the core logic of two or more processors.

Shared Memory Processor (SMP) follows multiple-instruction multiple-data (MIMD) architecture.

The processors share a common memory address space and communicate with each other via memory. All the processors will have dedicated cache memory.

In a 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 a data item simply using load and store operations, and process communication is through shared memory.

It is the hardware that makes all CPUs access and use the same main memory.

Since all CPUs share the address space, only a single instance of the operating system is required.

When a process terminates or goes into a wait state for whichever reason, the O.S. can look in the process table for another process to be dispatched to the idle CPU.

On the contrary, in systems with no shared memory, each CPU must have its own copy of the operating system, and processes can only communicate through message passing.

The basic issue in shared memory multiprocessor systems is memory itself, since the larger the number of processors involved, the more difficult to work on memory efficiently.

All modern OS support symmetric multiprocessing, with a scheduler running on every

processor the ready to run processes can be inserted into a single queue, that can be accessed by every scheduler, alternatively there can be a ┼ ready to run┼ queue for each processor.

When a scheduler is activated in a processor, it chooses one of the ready to run processes and dispatches it on its processor.

Load Balancing:

A distinct feature in multiprocessor systems is load balancing.

It is useless having many CPUs in a system, if processes are not distributed evenly among thecores.

With a single ready-to-run queue, load balancing is usually automatic: if a processor is idle, its scheduler will pick a process from the shared queue and will start it on that processor.

Modern OSs designed for SMP often have a separate queue for each processor to avoid the problems associated with a single queue.

There is an explicit mechanism for load balancing, by which a process on the wait list of anoverloaded processor is moved to the queue ofanother, less loaded processor.

Types of shared memory multiprocessors

There are three types of shared memory multiprocessors:

Uniform Memory Access (UMA)

Non Uniform Memory Access (NUMA)

Cache Only Memory Access (COMA)

Uniform Memory Access (UMA)

Here, all the processors share the physical memory in a centralized manner with equal access time to all the memory words.

Each processor may have a private cache memory. Same rule is followed for peripheral devices.

- When all the processors have equal access to all the peripheral devices, the system is called a **symmetric multiprocessor.**
- When only one or a few processors can access the peripheral devices, the system is called an **asymmetric multiprocessor.**
- When a CPU wants to access a memory location, it checks if the bus is free, then it sends the request to the memory interface module and waits for the requested data to be available on the bus.

Multicore processors are small UMA multiprocessor systems, where the first shared cache is actually the communication channel. Fig 1 shows the uniform memory access mmodel.

- Shared memory can quickly become a bottleneck for system performances, since all processors must synchronize on the single bus and memory access.

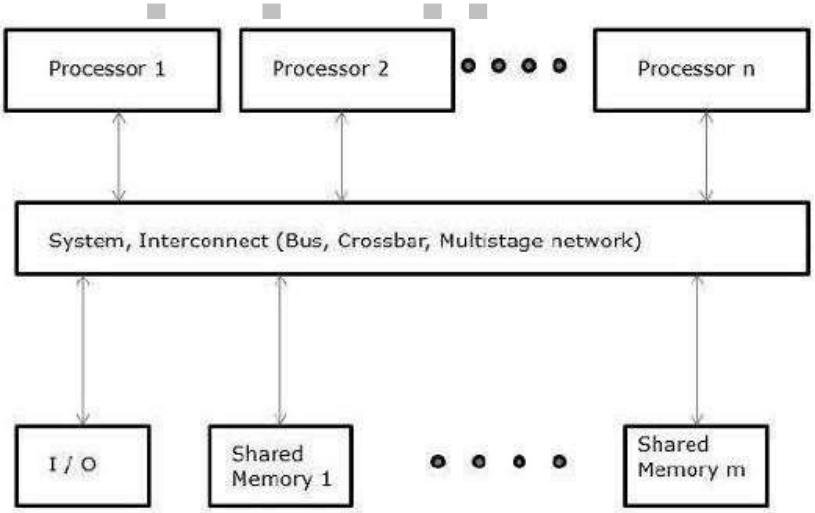

Fig 1: Uniform memory access model

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

- **Non-uniform Memory Access (NUMA)**

In NUMA multiprocessor model, the access time varies with the location of the memory word.

- Here, the shared memory is physically distributed among all the processors, called local memories.
- The collection of all local memories forms a global address space which can be accessed by all the processors.
- NUMA systems also share CPUs and the address space, but each processor has a local memory, visible to all other processors.
- In NUMA systems access to local memory blocks is quicker than access to remote memory blocks.
- Programs written for UMA systems run with no change in NUMA ones, possibly with differentperformances because ofslower access times to remote memoryblocks.
- Single bus UMA systems are limited in the number of processors, and costly hardware is necessary to connect more processors.

• Current technology prevents building UMA systems with more than 256 processors.

• To build larger processors, a compromise is mandatory: not all memory blocks can have the same access time with respect to each CPU.

• Since all NUMA systems have a single logical address space shared by all CPUs, while physical memory is distributed among processors, there are two types of memories: local and remote memory. Fig 2 represents the Non-uniform Memory Access Model.

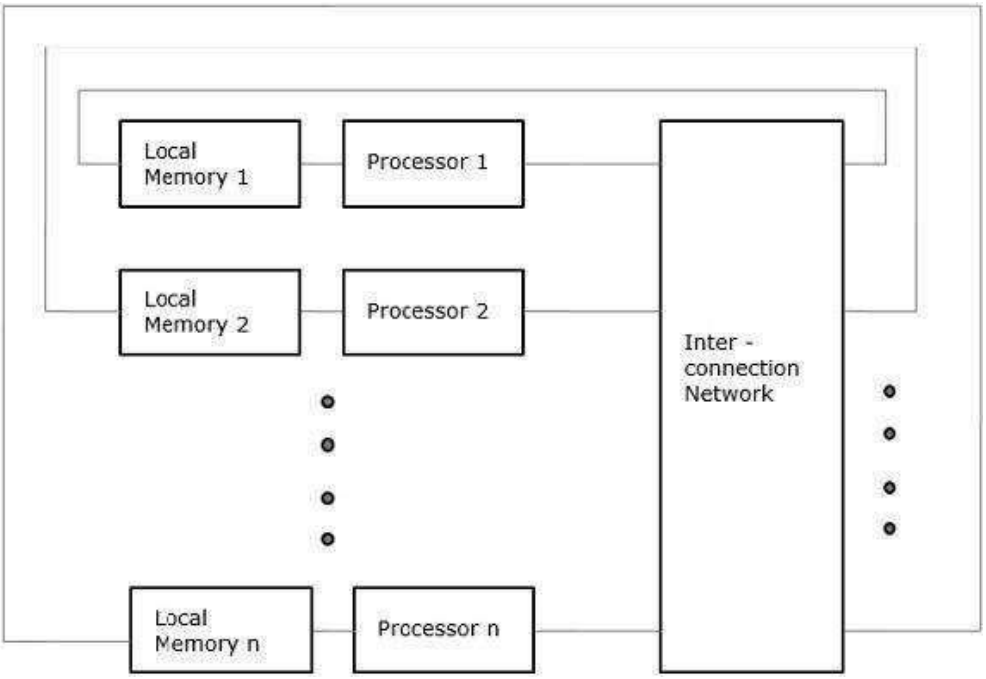

Fig 2:Non-uniform Memory Access model

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

There are two types of NUMA systems: Non-Caching NUMA (NC-NUMA) Cache-Coherent NUMA (CC-NUMA).

Non-Caching NUMA (NC-NUMA):

In a NC-NUMA system, processors have no local cache. Each memory access is managed with a modified MMU, which controls if the request is for a local or for a remote block; in the latter case, the request is forwarded to the node containing the requested data.

Obviously, programs using remote data will run much slower than what they would, if the data were stored in the local memory. In NC-NUMA systems there is no cache coherency problem, because there is no caching at all: each memory item is in a single location. Fig 3 represents Non-Caching NUMA Cache-Coherent NUMA (CC-NUMA)

Remote memory access is however very inefficient. For this reason, NC-NUMA

systems can resort to special software that relocates memory pages from one block to another, just to maximize performances.

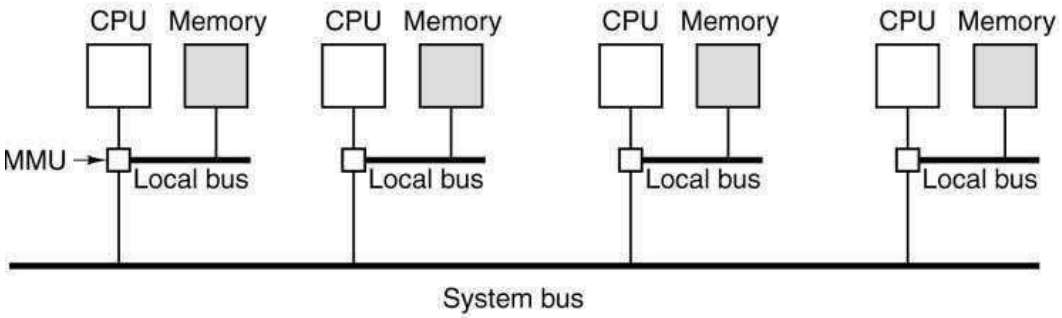

Fig 3:Non-Caching NUMA Cache-Coherent NUMA (CC-NUMA)

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

- Caching can alleviate the problem due to remote data access, but brings the cache coherency issue.
- A method to enforce coherency is obviously bus snooping, but this techniques gets too expensive beyond a certain number of CPUs, and it is much too difficult to implement in systems that do not rely on bus-based interconnections.
- The common approach in CC-NUMA systems with many CPUs to enforce cache coherency is the directory-based protocol.
- The basic idea is to associate each node in the system with a directory for its RAM blocks: a database stating in whichcache is located a block, andwhat is its state.
- When a block of memory is addressed, the directory in the node where the block is located is queried, to know if the block is in any cache and, if so, if it has been changed respect to the copy in RAM.
- Since a directory is queried at each access by an instruction to the corresponding memory block, it must be implemented with very quick hardware, as an instance with an associative cache, or at least with static RAM.
- Fig 4 represents the Cache-Coherent NUMA

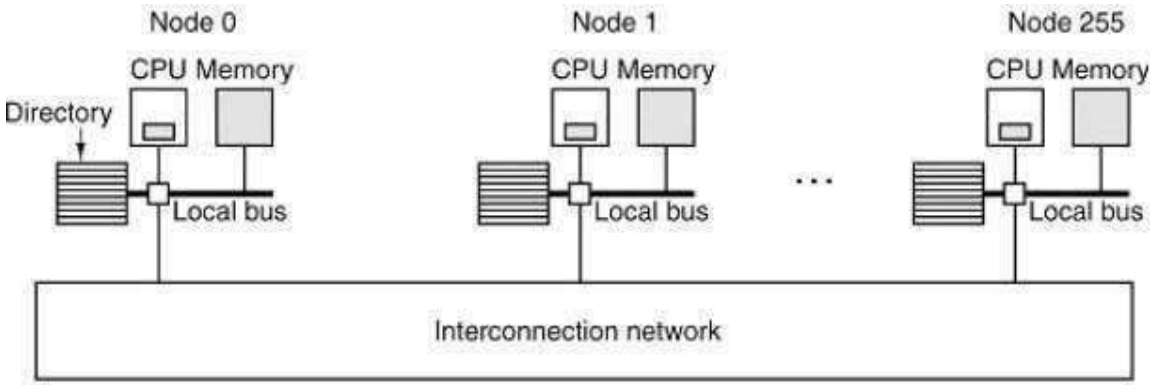

Fig 4: Cache-Coherent NUMA

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

iii) Cache Only Memory Access (COMA)

The COMA model is a special case of the NUMA model. Here, all the distributed main memories are converted to cache memories.

In a mono processor architecture and in shared memory architectures each block and each line are located in a single, precise position of the logical address space, and have therefore an address called home address.

When a processor accesses a data item, its logical address is translated into the physical address, and the content of the memory location containing the data is copied into the cache of the processor, where it canbe read and/or modified.

In the last case, the copy in RAM will be eventually overwritten with the updated copy present in the cache of the processor that modified it.

This property turns the relationship between processors and memory into a critical one, both in UMA and in NUMA systems:

In NUMA systems, distributed memory can generate a high number of messages to move data from one CPU to another, and to maintain coherency in home address values. Remote memory references are much slower than local memory ones.

In CC-NUMA systems, this effect is partially hidden by the caches.

In UMA systems, centralized memory causes a bottleneck, and limit its the

interconnection between CPU and memory, and its scalability.

- In COMA, there is no longer a home address, and the entire physical address space is considered a huge, single cache.

Data can migrate within the whole system, from a memory bank to another, according to the request of a specific CPU, that requires that data.

5.1 PARALLEL PROCESSING ARCHITECTURES

Parallel computing architectures breaks the job into discrete parts that can be executed concurrently. Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.

> ***Flynn's taxonomy is a specific classification of parallel computer architectures that are based on the number of concurrent instruction (single or multiple) and data streams (single or multiple) available in the architecture.***

Parallel processing architectures and challenges, Hardware multithreading, Multicore and shared memory multiprocessors, Introduction to Graphics Processing Units, clusters and Warehouse Scale Computers - Introduction to Multiprocessor network topologies.

Introduction:

Multiprocessor: A Computer system with atleast two processors.

Job – level parallelism (or) process – level parallelism:

Utilizing multiple processors by running independent programs simultaneously.

Parallel Processing Program:

A single program that runs on multiple processor simultaneously.

Multicore microprocessor:

A microprocessor containing multiple processors("Cores") in a single integrated circuit.

Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized. The crux of parallel processing are the CPUs.

Parallelism in computer architecture is explained used Flynn's taxonomy. This

classification is based on the number of instruction and data streams used in the architecture. The machine structure is explained using streams which are sequence of items. The four categories in Flynn's taxonomy based on the number of instruction streams and data streams are the following:

- (SISD) single instruction, single data
- (MISD) multiple instruction, single data
- (SIMD) single instruction, multiple data
- (MIMD) multiple instruction, multiple data

SISD (Single Instruction, Single Data stream)

Single Instruction, Single Data (SISD) refers to an Instruction Set Architecture in which a single processor (one CPU) executes exactly one instruction streamatatime. Fig 5.1 shows SISD and SIMD.

It also fetches or stores one item of data at a time to operate on data stored in a single memory unit.

Most of the CPU design is based on the von Neumann architecture and the follow SISD. The SISD model is a non-pipelined architecture with general-purpose registers, Program Counter (PC), the Instruction Register (IR), Memory Address Registers (MAR) and Memory Data Registers (MDR).

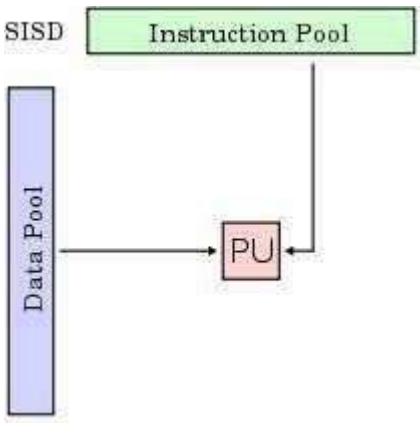

Fig 1: Single Instruction, Single Data Stream SIMD (Single Instruction, Multiple Data streams

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

Single Instruction, Multiple Data (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.

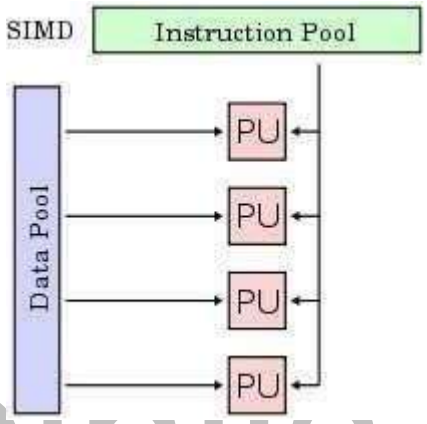

Fig 2: Single Instruction, Multiple Data streams MISD (Multiple Instruction, Single Data stream

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

The SIMD architecture is capable of achieving data level parallelism.

Multiple Instruction, Single Data (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. The above Fig 2 shows SIMD streams MISD.

This type of architecture is common mainly in the fault-tolerant computers executing the same instructions redundantly in order to detect and mask errors. The bellow Fig 3 represent the MISD streams.

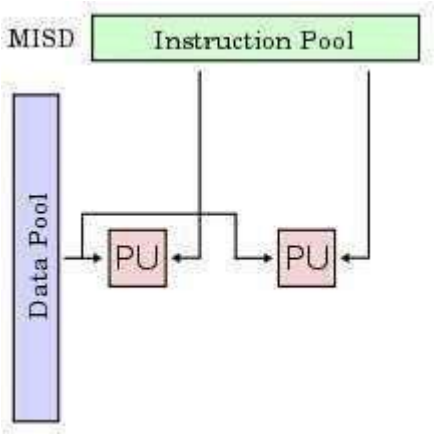

Fig 3: Multiple Instruction, Single Data stream

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

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 asynchronously different set of the instructions independently on the different set of data units.

The MIMD based computer systems can used 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, modelling, and communication switches etc. The bellow Fig 4 represents MIMD streams.

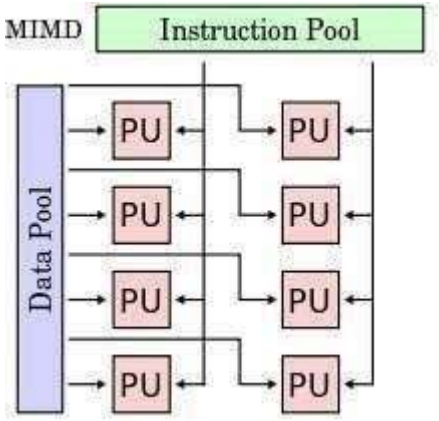

Fig 4 multiple instruction, multiple data streams

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"

5.1.1 Challenges in Parallelism

Thefollowingarethedesignchallenges inparallelism: Available parallelism. Load balance: Some processors work while others wait due to insufficient parallelism or unequal size tasks. Extra work. Managing parallelism Redundant computation Communication

5.1.2 HARDWARE MULTITHREADING

Multithreading enables the processing of multiple threads at one time, rather than multiple processes. Since threads are smaller, more basic instructions than processes, multithreading may occur within processes. Threads are instruction stream with state (registers and memory). The register state is also called thread context. Threads could be part of the same process or from different programs. Threads in the same program share the same address space and hence consume fewer resources. Fig 5 shows the hardware multithreading.

The terms multithreading, multiprocessing and multitasking are used interchangeably. But each has its unique meaning:

Multitasking: It is the process of executing multiple tasks simultaneously. In multitasking, when a new thread needs to be executed, old thread's context in hardware written back to memory and new thread's context loaded.

- **Multiprocessing:** It is using two or more CPUs within a single computer system.
- **Multithreading:** It is executing several parts of a program in parallel by dividing the specific operations within a single application into individual threads.
- **Granularity:** The threads are categorized based on the amount of work done by the thread. This is known as granularity. When the hardware executes from the hardware contexts determines the granularity of multithreading.

Hardware vs Software multithreading

Software Multithreading	Hardware Multithreading
Execution of concurrent threads is supported by OS.	Execution of concurrent threads is supported by CPU.
Large number of threads can be span.	Very limited number of threads can span.
Context switching is heavy. switching with more operations.	It involves Light/immediate context limited operations.

Hardware multithreading is having multiple threads contexts to span in same processor. This is supported by the CPU.

The following are the objectives of hardware multithreading:

- To tolerate latency of memory operations, dependent instructions, branch resolution by utilizing processing resources more efficiently. When one thread encounters a long- latency operation, the processor can execute a useful operation from another thread.
- To improve system throughput By exploiting thread-level parallelism by improving superscalar processor utilization
- To reduce context switch penalty

Advantages of hardware multithreading:

- Latency tolerance
- Better hardware utilization

- Reduced context switch penalty

Cost of hardware multithreading:

- Requires multiple thread contexts to be implemented in hardware.
- Usually reduced single-thread performance
- Resource sharing, contention
- Switching penalty (can be reduced with additional hardware)

Types of hardware multithreading

The hardware multithreading is classified based on the granularity of the threads as:

- Fine grained
- Coarse grained
- Simultaneous

Fine Grained Multithreading

- Here, the CPU switch to another thread at every cycle such that no two instructions from the thread are in the pipeline at the same time. Hence it is also known as **interleaved multithreading.**

> ***Fine grained multithreading is a mechanism in which switching among threads happen despite the cache miss or stall caused by the thread instruction.***

- The threads are executed in a round-robin fashion in consecutive cycles.
- The CPU checks every cycle if the current thread is stalled or not.
- If stalled, a hardware scheduler will change execution to another thread that is ready to run.
- Since the hardware is checking every cycle for stalls, all stall types can be dealt with, even single cycle stalls.
- This improves pipeline utilization by taking advantage of multiple threads
- It tolerates the control and data dependency latencies by overlapping the latency with useful work from other threads
- Fine-grained parallelism is best exploited in architectures which support fast communication.

- Shared memory architecture which has a low communication overhead is most suitable for fine-grained parallelism.
- This requires more threads to keep the CPU busy.

Advantages:

- No need for dependency checking between instructions since only one instruction in pipeline from a single thread.
- No need for branch prediction logic.
- The bubble cycles used for executing useful instructions from different threads.
- Improved system throughput, latency tolerance, utilization.

Disadvantages:

- Extra hardware complexity because of implementation of multiple hardware contexts and thread selection logic.
- Reduced single thread performance as one instruction fetched every N cycles.
- Resource contention between threads in caches and memory.
- Dependency checking logic between threads remains.

Coarse grained multithreading

- In this type, the instructions of other threads are executed successively until an event incurrent execution thread cause latency. This delay event induces a context switch.

Coarse grained multithreading is a mechanism in which the switch only happens when the thread in execution causes a stall, thus wasting a clock cycle.

- When a thread is stalled due to some event, the CPU switch to a different hardware context. This is known as Switch-on-event multithreading or **blocked** multithreading.
- This is less efficient that fine grained multithreading but requires only few threads to improve CPU utilization.
- The events that causes latency or stalls are: Cache misses, Synchronization events and floating point operations.
- Resource sharing in space and time always requires fairness considerations. This is implemented by considering how much progress each thread makes.

- The time allocated to each thread affects both fairness and system throughput. The allocation strategies depends on the answers to the following questions:
  - When do we switch?
  - For how long do we switch?
  - When do we switch back?
  - How does the hardware scheduler interact with the software scheduler for fairness?
  - What is the switching overhead vs. benefit?
  - Where do we store the contexts?
- A trade off must be done between fairness and system throughput: Switch not only on miss, but also on data return. This has a severe problem because switching has performance overhead as it requires flushing of pipeline and window; reduced locality and increased resource contention.
- One possible solution is to estimate the slowdown of each thread compared to when run alone. Then enforce switching when slowdowns become significantly unbalanced.

Advantages:

- Simpler to implement, can eliminate dependency checking and branch prediction logic completely
- Switching need not have any performance overhead.
- Higher performance overhead with deep pipelines and large windows

Disadvantages

- Low single thread performance: each thread gets 1/Nth of the bandwidth of the pipeline

Simultaneous Multithreading (SMT)

- Here instructions can be issued from multiple threads in any given cycle.
- Instructions are simultaneously issued from multiple threads to the execution units of a superscalar processor. Thus, the wide superscalar instruction issue is combined with the multiple-context approach.

- In fine-grained and coarse-grained architectures, multithreading can start execution of instructions from only a single thread at a given cycle. Execution unit or pipeline stage utilization can be low if there are not enough instructions from a thread to dispatch in one cycle
- Unused instruction slots, which arise from latencies during the pipelined execution of single-threaded programs by a microprocessor, are filled by instructions of other threads within a multithreaded processor. The executions units are multiplexed among those thread contexts that are loaded in the register sets.
- Underutilization of a superscalar processor due to missing instruction-level parallelism can be overcome by simultaneous multithreading, where a processor can issue multiple instructions from multiple threads in each cycle.

- Simultaneous multithreaded processors combine the multithreadingtechniquewitha wide-issue superscalar processor to utilize a larger part of the issue bandwidth by issuing instructions from different threads simultaneously.

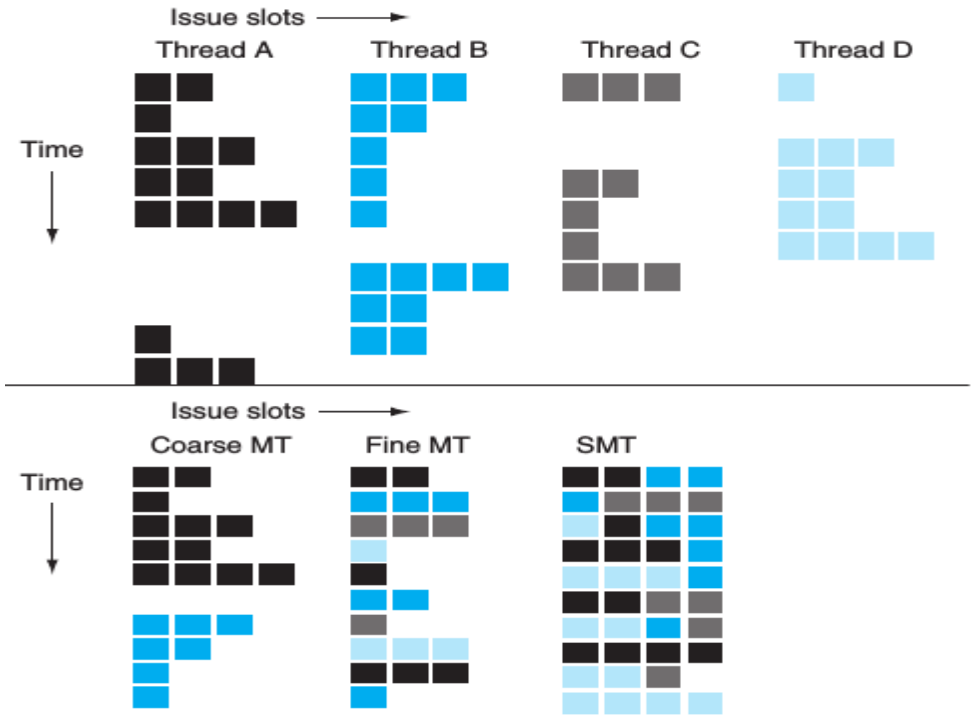

Fig 5: Hardware multithreading

Source: David A. Patterson and John L. Hennessey, — "Computer Organization and Design"