Module-1.

1) Explain Flynn's taxonomy of parallel computer with suitable example

=> flynn's taxonomy classifies a par-allel computer according to the. nu-mber of instruction streams and the number of the data streams (or data paths) it can simultaneously manage. It categorizes into four. -main types:

i) SISD (single instruction, single data) This architecture processes a single instruction stream and a single data. stream it represent traditional, non-. parallel computer where a single. processor executes instruction seque-ntially on one piece of data at a time.

Ex:- A classic uniprocessor computer like an older PC running a single program without parallel processing features.

ii) SIMD (single instruction, multiple. data) In this architecture, a single. instruction stream operates on multi-ple data streams simultaneously. All processors execute the same instru-

-ction on different data elements at the same time. this particularly efficient tasks involving repetitive operation on large datasets.

Ex:- Graphics processing units used in modern computers for rendering gra -phics or performing scientific computations. coherethe same operation is applied to many pixels concurre -ntly.

iii) MISD (multiple instruction single data)

This architecture involves multiple instruction stream operation on a single data stream.

This is a use common architecture -al type. It implies different proce -ssors performing different operation the same data simultaneously

Ex:- while rare in general purpose computing some specialized fault tolerant systems or pipeline archi -tectures might exhibit MISD cha -racteristics. where different sta -ges of pipline operate on the same data in sequence but with different instructions

iv) MIMD (multiple-instruction, multiple Data) This is the most common type of parallel architecture, a single instruc -tion stream operates where multiple processors can execute different data streams simultaneously this allows for true parallel processing, where independent tasks can run concurre -ntly.

Ex:- modern multi-core processors (eg: Intel, AMD, Ryzen) or large-scal super computers and distributed computing systems like server farms. where multiple independent programs or threads can run in parallel on different data sets.

2) compare and contrast SIMD and MIM -D systems in terms of archite -cture performance and applications

Terms	Feature	SIMD	MIMD
Architecture	Instruction Stream	single instru -ction stream	multiple instru -ction stream
	Data Stream	multiple data stream	multiple data streams

	Control unit	centralized control for all processing units	independent control units for each processor
	Synchronization	highly synchronized	Asynchronous more complex coordination
performance	efficiency	High for uniform tasks	High for diverse tasks
Terms	Aspect	SIMD	MIMD
	Scalability	limited by uniformity operations	highly scalable across heterogeneous workloads
	overhead	low control overhead	higher due to independent instruction
	parallelism	- Data-level parallelism	-> Task-level parallelism
Applications	Image/Video processing	Excellent for pixel-wise operation	less efficient unless tasks vary significantly.

	Scientific computing	Ideal for matrix/vector operations
	machine Learning	used in GPU based training

	Scientific computing	Ideal for matrix/vector operations	Suitable for simulations with varied computations
	machine Learning	used in GPU based training	used in distributed training across nodes

(9) Discuss the architecture and working of a vector processor and working of a vector processor how do vector registers and interleaved memory enhanced performance?

Ans:- Typical recent systems have the following characteristics

4 vector registers:

- These are registers capable of storing a vector of operands & operating simultaneously on their contents

* The vector length is fixed by the system and can range from 4 to 256 64-bit elements.

=) vectorised and pipelined functional units:

* Note that the same operation is applied to each element in the vector, or in the case of operations like addition, the same operation is applied to each pair of corresponding elements in the two vectors. Thus vector operations are SIMD.

=) Vector instructions: These are instructions that operate on vectors rather than scalars. If the vector length is vector length these instructions have the great virtual that a simple loop such as

```
for(i=0; i<n; i++)
    x[i] += y[i],
```

requires only a single load, add and store for each block of vector-length elements while a conventional system requires a load add

=) Interleaved Memory: The memory system consists of multiple "banks" that can be accessing more or less independently. After accessing one bank, there will be a delay before it can be reaccessed, but a different bank can be accessed much sooner.

=) Strided memory access and hardware scatter/gather:
In strided memory access the program accesses elements of a vector located at fixed interval. Scatter/gather (in this context) is writing (scatter) or reading (gather) elements of a vector located at irregular intervals.

4) With the help of a diagram, explain the working of shared-memory and distributed-memory system. Highlight their advantages and limitations.

Shared-memory system:- In a shared memory system, multiple processors share a common memory space.

Each processors can access any memory location, and changes made by one processor are visible to all others.

Working:

1) processors Access shared memory: each processor can access any memory location in the shared memory space.

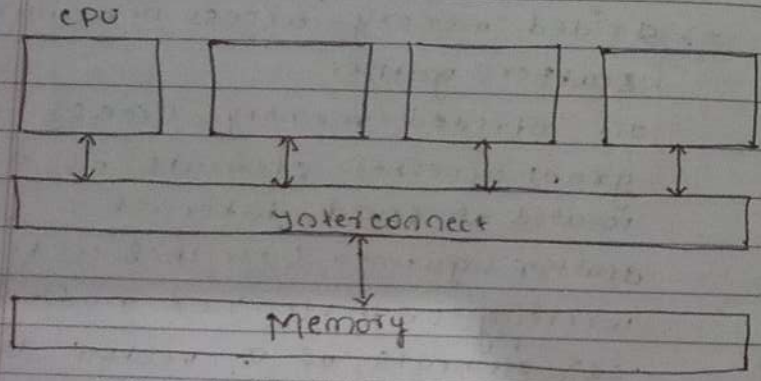

Fig:- A shared memory system.

2) cache coherence: To maintain data consistency, cache coherence protocols are used to ensure that change made by one processor are reflected in the cache of other processors

3) Synchronisation: processors use Synchronisation mechanisms such as locks or semaphores to cordinate access to shared resources.

Advantages:

-> shared memory system are relatively easy to program.

-> Since processors can access shared memory directly, communication overhead is relatively low.

Limitations:

-> Shared memory systems can be limited in scalability.

-> The shared memory can be come a bottle neck, limiting the performance of the system.

Distributed-memory system:- Each processor has its own private memory space processors communicate with each other by exchanging message over an inter connect network

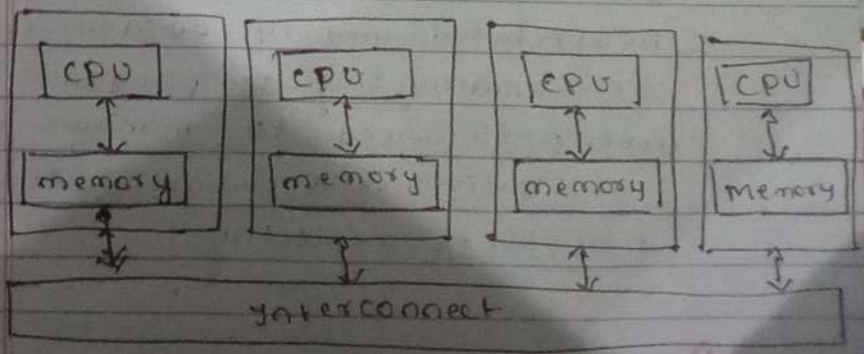

Fig:- A distributed memory system

Working:-
1> processors communicate via message: processors exchange message over the interconnect nlw to communicate with each other

2> Data distribution: Data is distributed across processors and each processor operates on its local data

3> Message passing: processors use message-passing protocols to exchange data and coordinate their acctions

Advantages

-> Distributed memory system ~~scheduler~~ Scale to a large no of processors.

-> Distributed memory system can be designed to accommodate different types of processors of interconnect nlws

Limitations

-> Distributed memory system can be more challenging to program as developers need to manage data distribution and communication blw processors explicitly.

-> The communication overhead can be significant especially for application with complex communication patterns

5> Describe various interconnection network used in parallel systems. Compare bus, crossbar, ring, mesh and hypercube topologies with respect to scalability & performance.

Ans:- interconnection nlw in parallel systems enable communication blw processors memory and other components

The various interconnection nlw's

-> cross bar network: allows any-to-any connections blw processors and memory high performance but costly with p^2 switches for p processors.

-> Omega Network: A multistage nlw using 2x2 switches. Less expensive than crossbar with $2p\log_2(p)$ switches.

-> Mesh network: processors connected in a grid good for local communication patterns.

-> Hypercube network: connects processors with $\log_2(p)$ neighbours

for p processors useful for cer
-tain parallel algorithms.

-> Bus network: A shared channel for communication, simple but can be a bottleneck.

-> Ring Network: - processor connected in a loop moderate scalability and performance.

Comparision of bus crossbar, ring, mesh and hypercube topologies regarding sca
-lability and performance in parallel systems

i) Bus

- Scalability: low, limited by bandwidth and contention
- performance: low for many processors due to shared channel

ii) Crossbar

- Scalability: - limited due to p² swit
-ches for p processors.
- performance: High any to any connections possible without conten
-tion.

iii) Ring:

Scalability: moderate, easy to add nodes but performance degrades with size

Performance: Moderate, data must traverse node to reach destination

mesh

Scalability: Good for 2D/3D structu
-res, local connections scale well

- performance: Good for local commu
-nication patterns. otherwise, late
-ncy increases with distance

Hyper cube :-

Scalability: - High for certain algori
-thms leveraging its connectivity.

performance: - High for algorithms matching hypercube structure, log p hops for p processors.

Q6) Explain the cache coherance problem in shared memory systems? Discu
-ss snooping and directory-based cache coherence protocols with ex.

Cache coherence problem :-

In shared memory systems, multiple processorsor cores access a common memory space each processor type

-cany has its own cache to improve performance by reducing memory access latency. However, when multiple processors cache the same memory location, inconsistencies can arise if one processor modifies its cached copy. This leads to the cache Cohere -nce problem.

key issues:-

1) Data inconsistency :- when one pro -cessor updates a cache copy of memory location, other processors might still have outdated copies in their caches.

2) state Data: processors might read state data from their cache instead of the updated value

Snooping cache coherence protocol

In snooping protocols, each cache monitors the shared bus for memory transactions. when a cache sees a write to a location it has cached, it updates or invalidates it copy accordingly

Example :-

1) Initially, both processor 1 (P1) and processor 2 (P2) cache the value of x as 10.

2) P1 updates x to 20 in its cache and broadcasts this change on th bus.

3) P2's cache snoops the bus, see -s the updates, and invalidates its copy of x.

Directory Based cache coherence proto -col In directory-based protocols a centralized directory keeps track of which caches have copi -es of each memory line and their states.

Ex:-

1) the directory tracks that both P1 and P2 have cached x

2) when P1 wants to write to x, it stands a request to the directory

3) the directory sends invalidate mess -ages to P2, which then invalidate its copy of x.

Module - 2

1) Explain the concept of GPU programming how does it differe from traditional CPU programming? Discuss challenges such as branching and scheduling in GPU threads

Ans:- GPU (Graphics processing unit) programming involves using GPUs to perform general purpose computing tasks beyond Graphics rendering GPUs are designed for parallel processing with thousands of cores that can handle many threads simultaneously

-> Differences from CPU programming

~~How does it differe from traditional CPU prog~~

1) Parallelism : GPUs are optimized for massive parallelism, where as CPU are designed for serial execution

2) Core Architecture : GPU cores are simpler and more numerous than CPU cores

3) Memory hierachy :- CPUs have a different memory hierachy, with a focus on high-bandwith memory

Challenges in GPU Threads

-> Branching

Brach divergence : In a GPU, threads are executed in groups called warps. when threads within a warp take different branches diver-gence. This reduce performance as the GPU must execute each branch path separately leading to idle threads

Impact : Branch divergence can signi-ficantly Reduce GPU performance especially if the branches are complex or have many divergant paths.

Mitigation : Minimize branching within warps or use techniques like pre-diction where each thread conditi-onally executes instruction based on a predicate

Scheduling

- Thread scheduling : GPUs use a SIMT (single instruction, multiple thread) execution model efficient Scheduling of the treads is crucial to maximize performance

Occupancy: occupancy refers to the no of active warps per multipr -ocessor, high occupancy is essential for hiding memory latency and maxi -mizing throughput.

Mitigation: optimize thread block sizes, minimize thread divergence and ensure efficient memory acce -ss patterns to improve occupancy and overall performance.

(8) what are hybrid programming model in parallel computing? explain with an example how shared-memory and distributed-memory APIs can be combined

Ans:- Hybrid programming models in parallel computing combine multiple parallel programming paradigms to achieve better performance scalability and efficiency. These models integrate different programming models, such as shared-memory and distributed memory models, to leverage their strengths Types of Hybrid models-

1) shared-memory and distributed memory: combined models like open mp (shared-memory) with mpi (dis -tributed-memory) to parallelize computations within and across no -des

2) apu and cpu: use models like CUDA (for NVIDIA APUs) or open (for multiple platforms) in conju -nction with cpu-based parallel pro -gramming models like openmp or mpi

combining shared memory and distrib -uted memory APIs

-> lets consider an example using openmp (shared memory) and mpi (distributed memory) to parallelize a computation

Example: parallelizing a loop

suppose we have a large loop that needs to be parallelized across multi -ple nodes in a cluster with each node having multiple cores.

1) mpi (distributed memory): use mpi to divide the loop iteratio -ns among multiple nodes, each node will process a portion of the iterations.

2) openmp (shared memory): within each node, use openmp to parallelize the loop iterations across multiple cores.

```
#include <mpi.h>
#include <omp.h>
int main (int argc, char **argv){
MPI_Init (&argc, &argv);
int rank, size;
mpi_comm_rank(mpi_comm_world, &rank);
MPI_comm_size (mpi_comm_world, &size);
int num_iterations = 100;
int iterations_per_node = num_iterations / size;
int start = rank * iteration_per_node;
int end = (rank+1) * iteration_per_node;
# programa omp parallel for
for(int i = start; i< end; i++){
    // perform computation
}
mpi_finalize();
return 0;
}
```

(5) Define and compare speedup and efficiency in MIMD systems. Illustrate with suitable numberical examples.

-> Speedup and efficiency in MIMD System:

Speedup

Definition: Speedup is the ratio of the execution time of a sequential algorithm to the execution time of parallel algorithm

formula: $S = T_{sub}sequential / T_{sub}parallel$

Efficiency:-

Definition: Efficiency measure the effectiveness of parallelization by comparing the speedup to the no of processors used

formula: $E = S/p$ where p is the no of processors

Speedup vs Efficiency: Speedup focuses on the performance gain, while efficiency focuses on the effective utilization of resources.

numerical example:

=> suppose a sequential algorithm

tasks 100 seconds to execute. A parallel version of the algorithm using 4 processors takes 25 seconds

1. Speed up calculation.
$S = T_{sequential} / T_{parallel} = 100/25 = 4$.

2. efficiency calculation
$E = S/P = 4/4 = 1$ (or 100%)

In this example, the parallel algorithm achieves a speedup of 4 an efficiency of 100% with 4 processors.