VETRI VINAYAHA COLLEGE OF ENGINEERING AND TECHNOLOGY
THOTTIAM – 621215
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
B.E EIGHTH SEMESTER

CS6801- MULTI-CORE ARCHITECTURES AND PROGRAMMING
(Regulation 2013)

UNIT I
MULTI-CORE PROCESSORS

Single core to Multi-core architectures - SIMD and MIMD systems - Interconnection networks - Symmetric and Distributed Shared Memory Architectures - Cache coherence – Performance Issues – Parallel program design

PART - A

1. Define Single core processor?
- A computing component having only on central processing unit (CPU).
- All programs or software are executed on only one core.

2. What is Multi-core processor?
- A computing component which is composed of two or more processors that is called as “cores”
- Cores read and execute programs in an most efficient way than single core.

3. Compare Data stream with Instruction stream.
- Data stream is a Sequence of data on which the operations are performed.
- Instruction stream is a Sequence of machine instructions read from memory.

4. Mention the types of MIMD systems.
- Shared memory systems
- Distributed memory systems

5. What is UMA System?
- UMA – Uniform memory access system
- Interconnect can connect all the processors directly to main memory.
- Time to access all the memory locations will be the same for all the cores.

6. What is NUMA System?
- NUMA – Non-Uniform memory access system
- Interconnect can connect each processor directly to a block of main memory.
- Processors can access each other’s blocks of main memory through special hardware built into the processors.

7. Define Bus. State the Key characteristics of Bus.
- A collection of parallel communication wires together with some hardware that controls access to the bus.
- Communication wires are shared by the devices that are connected to it.
- Low cost and flexibility
- Multiple devices can be connected to a bus with little additional cost.

8. Name the measure used to describe the performance of an interconnect.

1. Latency
2. Bandwidth
3. Message transmission time

9. Define Cache Coherence.

Consistency of shared resource data that ends up stored in multiple local caches.

10. Give the idea of Snooping cache coherence.

Also referred to as a bus-snooping

Idea

- Based on the bus-based systems.
- When the cores share a bus, any signal transmitted on the bus can be "seen" by all the cores connected to the bus.

11. Give the limitations of snooping cache coherence.

- Expensive for large networks broadcasts
- Requires a broadcast every time a variable is updated.
- Not scalable for larger systems
- Cause performance to degrade.

12. Specify the metrics used for performance evaluation.

1. Speedup and efficiency
2. Amdahl's law
3. Scalability
4. Taking timings

13. Define Speedup.

Ratio of the serial runtime of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve same problem on a p processors.

$$S = \frac{T_{serial}}{T_{parallel}}$$

14. What is Scalability?

A technology is Scalable if it can handle ever-increasing problem sizes.

15. Compare Weakly and Strongly scalable.

<u>Strongly scalable</u>

If when the number of processes/threads is increased, the efficiency is kept fixed without increasing the problem size.

<u>Weakly scalable</u>

If the efficiency is kept fixed by increasing the problem size at the same rate as the number of processes/threads is increased.

16. List the step involved in Foster's methodology.

1. Partitioning
2. Communication
3. Agglomeration or aggregation
4. Mapping

PART- B

1) Explain in detailed about SIMD and MIMD systems.

2) Explain briefly on Interconnection networks.

3) Discuss in detail about the cache coherence.

4) Describe briefly on the performance issues of Multi-core processors.

5) Compare the single-core and multi-core processor.

6) Design the parallel program and explain in detail.

7) Explain briefly on Symmetric Memory Architecture and Distributed Memory Architecture.

UNIT II
PARALLEL PROGRAM CHALLENGES

Performance – Scalability – Synchronization and data sharing – Data races – Synchronization primitives (mutexes, locks, semaphores, barriers) – deadlocks and livelocks – communication between threads (condition variables, signals, message queues and pipes).

PART – A

1. Specify the reason for serial performance is important for parallel applications.

- ➢ Each thread in a parallel application is a serial stream of instructions
- ➢ Making these instructions execute as fast as possible will lead to better performance for the entire application.
- ➢ Rare to find a parallel application that contains no serial code
- ➢ As the number of threads increases, it is the performance of the serial sections of code that will ultimately determine how fast the application runs.

2. What is Items per unit time?

Ability of the system to complete tasks rather than on the duration of each individual task measure of bandwidth

Example:

SPEC Java Application Server benchmark
Linpack benchmark

3. Define Time per item.

Measure of the time to complete a single task.
Measure of latency or response time.

Example:

SPEC CPU benchmark suite

4. What is Quality of Service (QOS)?

Specify the expectations of the users of the system as well as penalties if the system fails to meet these expectations.

Example:

Number of transactions of latency greater than some threshold
Amount of time that the system is unavailable called downtime or availability

5. Tell the techniques that reduce the latency.

1. Out-of –order execution
2. Hardware Prefetching
3. Software Prefetching

6. Define False Sharing.

Situation where multiple threads are accessing items of data held on a single cache line

7. What is Pipeline Resource Starvation?

If a single thread is running on a core, its performance is limited by the rate at which the core can issue instructions and the number of stalls that the thread encounters.

8. What is Data Races?
Situations where multiple threads are updating the same data in an unsafe way.

9. Name any two Tools to Detect Data Races.

1. Helgrind
2. Thread Analyzer

10. How to avoid Data Races?

- Place a synchronization lock around all accesses to that variable
- Ensure the thread must acquire the lock before referencing the variable

11. When the Barriers is used?

- Used when a number of threads have to all complete their work before any of the threads can start on the next task.
- Where the threads will wait all are present.

12. Define Atomic operation.

Operation that must be performed entirely or not performed at all Used often to enable the writing of lock-free code.

13. What is Lock Free Code? Tell its use.
Code without synchronization primitives such as mutexes also known as lockless code
Does not rely on a mutex lock to protect access.
Used to achieve more complex operations.

14. Define Deadlocks.

A situation in which two threads sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs accessing to function.

15. What is Livelocks?

A condition that occurs when two or more threads continually change their stain response to changes in the other threads.

16. What is Pipe?
Used to pass data from one process to another
Example
Command ls – Lists all the files in a directory
Command wc – Counts the number of lines, words, and characters in the input
Command ls is piped into the wc command – count the number of files in the directory

17. Define Named Pipes.
File like objects that are given a specific name that can be shared between processes.

PART- B

1) Outline about necessity of Structure reflects in performance.

2) Brief about Superliner scaling.

3) Write in detail about Hardware constraints applicable to improve scaling.

4) Discuss about Operating System constraints to scaling.

5) Explain in detail about Data races and to overcome it.

6) Write short note on Mutexes, Semaphore, Spin lock and Reader – Writer locks.

7) Write in detail about Conditional variables in communication between threads.

UNIT III
SHARED MEMORY PROGRAMMING WITH OpenMP

OpenMP Execution Model – Memory Model – OpenMP Directives – Work-Sharing Constructs – Library functions – Handling Data and Functional Parallelism – Handling Loops – Performance Considerations.

PART A

1. What is OpenMP?
An APIs for shared-memory parallel programming.
'MP' stands for Multiprocessing.
Designed for systems in which each thread or process can potentially have access to all available memory.

2. Specify the Components of OpenMP.
- Compiler Directives.
- Runtime Library Routines.
- Environment Variables.

3. Give the syntax to compile and execute the OpenMP program.
Compilation
$ gcc-g-wall-fopenmp-o omp_hello omp_hello.c
Execution
To run, the number of threads is specified on the command line
$./omp_hello4

4. Give the syntax of strol( ) function.

Long strol(		/*		*/,
Const char*	Number_p		In	

Char**	End_p	/*	Out	*/,
int	base	/*	in	*/);

5. What is Structured Blocks?

- An OpenMP construct is defined to be a complier directive plus a block of code.
- A single statement or a component statement with a single entry at the top and a single exit at the bottom.
- Branching in or out of a structured block is not allowed.

6. Give the syntax of pragma.

#pragma omp directive –name[clause[clause]…] new-line

Where

#pragma	Required for all OpeMP C/C++ directives
Directive-name	A valid OpenMP directive Must appear after the pragma and before any clauses.
[clause[clause]….]	Optional Clauses can be in any order, and repeated as necessary unless otherwise restricted
New-line	Required Precedes the structured block which is enclosed by this directive

7. Define Parallel Constructs.

- Defines a parallel.
- Code that will be executed by multiple threads in parallel.

8. Give the Syntax for Parallel Constructs.

```
#pragma omp parallel [clause….] new-line
  Structured-block
```

9. List the OpenMP clauses supported by the parallel directive.

```
if(scalar-expression)
num_threads(integer-expression)
default (shared|none)
private(list)
firstprivate(list)
shared(list)
copyin(list)
reduction(operator:list)
```

10. What is Work-sharing constructs?

- Distributes the execution of the enclosed code region among the members of the team that encounter it.
- Do not launch new threads.

11. Name the types of Work-sharing Constructs.

1. For
2. Sections
3. Single

12. What is the use of master directive?

Specifies that only the master threshold execute a section of the program

Syntax

```
    #pragma omp master new-line
        Structured-block
```

13. Tell the OpenMP control variables and its use.

nthreads-var

Stores the number of threads requested for future parallel regions.

dyn-var

Controls whether dynamic adjustment of the number of threads to be used in future parallel regions is enabled.

14. Tell the OpenMP environmental variables and its use.

OMP_NUM_THREADS

This environment variable will be used to initialize nthreads-var

OMP_DYNAMIC

Setting this variable to value to true allows the OpenMP implementation to adjust the number of threads to use in parallel regions.

OMP_SCHEDULE

Sets default scheduling mechanism and chunk size.

15. What is the use of Data Scope Attribute Clauses?

- Also called Data-sharing Attribute Clauses
- Apply only to variables whose names are visible in the construct on which the clause appears.

16. Specify the use of Data Copying Clauses.

Support the copying of data values from private or thread –private variables on one implicit task or thread to the corresponding variables on their implicit tasks or threads in the team.

17. List the common loop transformation in OpenMP.

1. Loop unrolling
2. Unroll and jam
3. Loop interchange
4. Loop fusion
5. Loop fission
6. Loop tilling/blocking

PART B

1) Illustrate an OpenMP execution model with an example.

2) Explain briefly about internal control variables and Array sections in directives.

3) Explain the conceptual idea about work-sharing constructs.

4) Write short notes on General data parallelism.

5) Brief short notes on Functional parallelism.

6) Illustrate the execution environment routines.

7) Describe briefly about Parallel construct and Canonical loop form in directives.

UNIT IV
DISTRIBUTED MEMORY PROGRAMMING WITH MPI

MPI program execution – MPI constructs – libraries – MPI send and receive – Ponit-to –Point and Collective communication – MPI derived data types – Performance evaluation.

PART A

1. Classify the MIMD computers.

Distributed-memory systems
Shared –memory systems

2. What is a Distributed –memory system?

- Consists of core-memory pairs connected by a network.
- Memory associated with a core.

3. What is a Shared –memory system?

- Consists of a collection of cores connected to a globally accessible memory.
- Each core can have access to any memory location.

4. Define the term MPI.

Message Passing Interface is abbreviated as MPI. Program running on one core-memory pair is called process. Two processes can communicate each other by using calling functions one process calls a Send function and other process calls a Receive function.

5. Tell the use of MPI_Int and its syntax.

Tells the MPI system to do all the necessary setup.
No MPI functions first calls MPI_int before any other function to be called
Syntax

```
Int MPI_Int(
Int* argc_p           /* in/out*,
Char***  argv_p       /* in/out*/);
```

Where

argc_p and argv_p are pointers to the arguments
MPI_Init returns an int error code

6. Tell the use of MPI_Finalize and its syntax.

- MPI_Finalize release the resources allocated for MPI
- No other MPI functions should be called after the calling the MPI_Finalize()

  Syntax

  ```
  Int MPI_Finalize(void);
  ```

7. Present the basic outline of the MPI program.

```
#include<mpi.h>
….
int main(int argc,char* argv[])
{
        ……..
```

```
        /* No MPI calls before this */
        MPI Init(&argc, &argv);
        …….
        MPI Finalize();
        /*No MPI calls after this*/
        ……..
        Return 0;
    }
```

8. What are SPMD programs?

Single program is written so that different processes can carry out different action achieved by the having the processes branch on the basis of their process rank.

9. Name the MPI Predefined Data types.

MPI CHAR
MPI SHORT
MPI INT
MPI LONG
MPI LONG LONG
MPI FLOAT
MPI DOUBLE
MPI BYTE
MPI PACKED

10. What is Collective communication?

Involve communication among all processes in a process group.

11. Give the examples of Collective communication.

MPI_Bcast() - Broadcast from root to all processes
MPI_Gather() – Gather values for group of processes
MPI_Reduce()- Combine values on all processes to single value.

12. Specify the types of Communication Modes.

Blocking Mode – Program will not continue until the communication is completed.
Non-Blocking Mode – Program will continue, without waiting for the communication to be completed.

13. Mention the Predefined Reduction Operations in MPI.

Operation Value	Meaning
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product

14. What is MPI_Allreduce?

A variant of MPI_Reduce
Store the result on all the processes in the communicator.

Useful in which all the processes need the result of a global sum in order to complete some larger computation.

15. Define Elapsed parallel time.
Returns the number of seconds that have elapsed since some time in the past.

16. Define Elapsed serial time.
Returns time in microseconds elapsed from some point in the past.

PART B

1) Illustrate an MPI program execution with an example.

2) Explain briefly about MPI Constructs of distributed memory.

3) Describe briefly about libraries for Groups of processes and Virtual topologies.

4) Describe briefly about libraries for attribute caching and context of communication.

5) Explain the functioning of MPI_Send and MPI_Recv.

6) Draft an outline about the point-to-point communication with their functions.

7) Inscribe about the collective communication with their functions.

UNIT V
PARALLEL PROGRAM DEVELOPMENT

Case studies – n-Body Solvers – Tree Search – OpenMP and MPI implementations and comparison.

PART A

1. **State *n*-body problem.**
   To find the positions and velocities of a collection of interacting particles over a period ot time.

2. **Give the examples of *n*-body problem.**
   - ➢ An astrophysicist might want to know the positions and velocities of a collection of stars
   - ➢ A chemist might want to know the positions and velocities of a collection of molecules or atoms
3. ***Define n*-body solver.**
   A program that finds the solution to an *n*-body problems by simulating the behavior of the particles

4. **Give the Pseudocode for the serial program in parallelizing the basic solver using Open MP.**

```
for each timestep {
        if (timestep output) Print positions and velocities of particles;
for each particle q
        Compute total force on q;
for each particle q
```

```
        Compute position and velocity of q;
    }
```
Note: two inner loops are both iterating over particles

5. How is *Acceleration defined?*
- Assume the position and the velocity are found
- Acceleration of particle *q* is given by

$$a_q\ (t) = s_q{}^{n}\ (t)=F_q(t)/m_q$$

6. Give the formula to compute Forces.
- Let s_q (t) be the position of the particle *q* at time *t*
- Let s_k (t) be the position of the particle *k* at time *t*
- Force on particle q exerted by particle *k* is given by

$$f_{qk}(t)= \frac{Gm_q m_k}{|s_q(t)-s_k(t)|^3}\ [s_q(t)-s_k(t)]$$

Where
- G – gravitational constant ($60673*10^{-11}m^3/(kg.s^2)$)
- m_g – masses of particles *q*
- m_k - masses of particles *k*
- $[s_q(t)-s_k(t)]$ - Distance from particle *k* to particle *q*

7. State Newton's third law of motion.
- For every action there is an equal and opposite reaction
- Used to halve the total number of calculations required for the forces

8. What is graph?
A graph is a collection of vertices and edges or line segments joining pairs of vertices, G(V,E).

9. What is directed graph of digraph?
In a directed graph or digraph, the edges are oriented --- one end of each edge is the tail and the other is the head.

10. Why digraph is used in Travelling Salesperson Problem?
The vertices of the digraph correspond to the cities in an instance of the Travelling Salesperson Problem, the edges correspond to routes between the cities, and the labels on the edges correspond to the costs of the routes.

11. How to find a least cost tour in TSP?
One of the most commonly used is called depth-first search. In depth first search, probe as deeply as can into the tree. After either reached a leaf or found a tree node that can't possibly lead to a least-cost, back up to the deepest "ancestor" tree node with unvisited children, and probe one of its children as deeply as possible.

12. What are the global variables for Recursive depth first search?
- N: the total number of cities in th**e** problem
- Digraph: a data structure representing the input digraph
- How town: a data structure representing vertex or city 0, the salesperson's hometown

- Best tour: a data structure representing the best tour

13. Define the term P threads or POSIX Threads.

POSIX Threads, usually referred to as Pthreads, is an execution model that exists independently from a language, as well as a parallel execution model. It allows a program to control multiple different flows of work that overlap in time. Each flow of work is referred to as a thread, and creation and control over these flows is achieved by making calls to the POSIX Threads API. Pthreads defines a set of C programming languages types, functions and constants. It is implemented with a pthread.h header and a thread library.

14. What are the different categories of pthreads?

- Threads management- creating, joining threads etc.
- Mutexes
- Condition variables
- Synchronization between threads using read/write lochs and barriers

15. What are the reasons for parameter threads_in_cond_wait used in Tree search?

- When it’s less than threads_count, it tells us how many threads are waiting.
- When it’s equal to threads_count, it tells us that all the threads are out of work, and it’s time to quit.

16. What are the modes of message passing interfaces for send and its functions?

MPI provides four modes for sends: standard (MPI_Send), synchronous (MPI_Ssend), read (MPI_Rsend), and buffered (MPI_Bsend).

PART B

1) Describe briefly about basic and reduced solver using OpenMP.

2) Explain in detail about basic and reduced solver using MPI.

3) Discuss in detail about evaluating the OpenMP codes.

4) Discuss in detail about performance of MPI Solvers.

5) Explain in detail about Recursive and Non recursive depth-first search with explain.

6) Describe about parallelizing tree search using pthreads.

7) Explain briefly about tree search with static and dynamic partitioning.