19CS8708 – MULTI-CORE ARCHITECTURES AND PROGRAMMING
Question bank

Part – A

1. List the restrictions to sections constructs.

 Orphaned section directives are prohibited. That is, the section directives must appear within the sections construct and must not be encountered elsewhere in the sections region.
 The code enclosed in a sections construct must be a structured block.
 Only a single no wait clause can appear on a sections directive.

2. Brief about Simple lock routines.

The type omp_lock_t is a data type capable of representing a simple lock. For the following routines, a simple lock variable must be of omp_lock_type. All simple lock routines require an argument that is a pointer to a variable of type omp_lock_t. The simple lock routines are as follows:
 The omp_init_lock routine initializes a simple lock.
 The omp_destroy_lock routine uninitializes a simple lock.
 The omp_set_lock routine waits until a simple lock is available, and then sets it.
 The omp_unset_lock routine unsets a simple lock.
 The omp_test_lock routine tests a simple lock, and sets it if it is available.

3. List out the different datatype constructors?

 MPI_TYPE_CONTIGUOUS
 MPI_TYPE_VECTOR
 MPI_TYPE_CREATE_HVECTOR
 MPI_TYPE_INDEXED

4. What is the purpose of wrapper script?

A wrapper script is a script whose main purpose is to run some program. In this case, the program is the C compiler. However, the wrapper simplifies the running of the compiler by telling it where to find the necessary header files and which libraries to link with the object file.

5. Define the term linear speedup.

The ideal value for S(n, p) is p. If S(n, p)= p, then our parallel program with comm_sz = p processes is running p times faster than the serial program. In practice, this speedup, sometimes called linear speedup, is rarely achieved. Our matrix-vector multiplication program got the speedups.

6. List the functions of group assessors.

 MPI_GROUP_SIZE(group, size)
 MPI_GROUP_RANK(group,rank)
 MPI_GROUP_TRANSLATE_RANKS(group1,n,ranks1,group2,ranks2)
 MPI_GROUP_COMPARE(group1,group2,result)

7. Distinguish between MPI_Pack and MPI_Unpack.

MPI_Pack	MPI_Unpack
Packing data into a buffer of contiguous memory	Unpacking data from a buffer of contiguous memory
Takes the data in data_to_be_packed and packs it into contig_buf	Takes the data in contig_buf and unpacks it into unpacked_data

8. Brief about pthread_mutex_trylock.

Pthreads provides a nonblocking alternative to pthread_mutex_lock called pthread_mutex_trylock:

```
int pthread_mutex_trylock(
          pthread_mutex_t*   mutex_p    /* in/out */);
```

This function attempts to acquire mutex_p. However, if it's locked, instead of waiting, it returns immediately.

9. **How can you demonstrate a graph?**

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

10. **Write about Fulfill_request functions.**
If a process has enough work so that it can usefully split its stack, it calls Fulfill request. Fulfill request uses MPI_probe to check for a request for work from another process. If there is a request, it receives it, splits its stack, and sends work to the requesting process.

11. Brief about pragma.
A compiler directive in C or C++ is called a pragma. The word pragma is short for ―pragmatic information.‖ A pragma is a way to communicate information to the compiler. The information is nonessential in the sense that the compiler may ignore the information and still produce a correct object program. However, the information provided by the pragma can help the compiler optimize the program.
Like other lines that provide information to the preprocessor, a pragma begins with the # character. A pragma in C or C++ has this syntax:
#pragma omp <rest of pragma>.

12. Write short notes on collective communication.
Collective communication is defined as communication that involves a group of processes. The functions of this type provided by MPI are the following:
 Barrier synchronization across all group members
 Broadcast from one member to all members of a group
 Gather data from all group members to one member
 Scatter data from one member to all members of a group
 A variation on Gather where all members of the group receive the result
 Scatter/Gather data from all members to all members of a group
 Global reduction operations such as sum, max, min, or user-defined functions, where the result is returned to all group members and a variation where the result is returned to only one member
 A combined reduction and scatter operation
 Scan across all members of a group

13. Brief about strongly and weakly scalable.
Programs that can maintain a constant efficiency without increasing the problem size are sometimes said to be strongly scalable. Programs that can maintain a constant efficiency if the problem size increases at the same rate as the number of processes are sometimes said to be weakly scalable**.**

Big Questions

1. How to handle loops in OpenMP? Explain in detail.
2. List out the OpenMP Library functions in detail.
3. In parallel programming, it‘s common for the processes to be identified by nonnegative integer ranks. So, if there are p processes, the processes will have ranks, how can you demonstrate the execution of MPI programs?
4. For executing MPI programming, the process MPI_Send and MPI_Receive plays a vital role. How can you execute the process of MPI_Send and MPI_Receive?
5. Explain in detail about the following collective communication mechanisms.
i) Tree-structured communication
ii) Broadcast
6. In virtually all distributed-memory systems, communication can be much more expensive than local computation. For example, sending a double from one node to another will take far longer than adding two doubles stored in the local memory of a node. Explain in detail about various MPI Derived Datatypes.
7. Parallelizing the two n-body solvers using Pthreads is very similar to parallelizing them using OpenMP. Explain in detail about how to parallelize the solvers using Pthreads and MPI. (13 marks)
Parallelizing the solvers using pthreads
Parallelizing the basic solver using MPI
8. Explain the following tree search mechanisms.
i) Recursive depth-first search (7 marks)
ii) Non-recursive depth-first search (6 marks)
9. Discuss briefly about the data structures and performance of the serial implementations. (13 marks)
Data structures for the serial implementations
Performance of the serial implementations

10. Explain briefly about the implementation of tree search using MPI and static partitioning (13 marks)
11. All the processes in the communicator must call the same collective function. For example, a program that attempts to match a call to MPI_Reduce on one process with a call to MPI_Recv on another process is erroneous, and, in all likelihood, the program will hang or crash. Discuss elaborately about the differences of collective and point-to-point communications.
12. The local storage required for the MPI version is substantially less than the local storage required for the OpenMP version. So, for a fixed number of processes or threads, we should be able to run much larger simulations with the MPI version than the OpenMP version. Of course, because of hardware considerations, likely to be able to use many more MPI processes than OpenMP threads, so the size of the largest possible MPI simulations should be much greater than the size of the largest possible OpenMP simulations. The MPI version of the reduced solver is much more scalable than any of the other versions, and the "ring pass" algorithm provides a genuine breakthrough in the design of n-body solvers. Explain in detail about performance of the MPI Solvers and Pseudocode for the MPI version of the basic n-body solver.