**CSYE7105 HW1**

**1. Give several reasons to use parallel computing**

1. Save time/money
2. Solve Larger/More Complex problems like Web search engines/databases processing millions of transactions every second
3. Provide Concurrency, for example Collaborative Networks provide a global venue where people from around the world can meet and conduct work virtually
4. Take Advantage of Non – local Resources that is using compute resources on a wide area network, or even the internet when local compute resources are scarce or insufficient
5. Make better use of Underlying Parallel Hardware

**2. On the CPU architecture, how many parallel programming models are there based on memory access methods? List them and explain briefly.**

Parallel Programming Models

There are several parallel programming models in common use:

* Shared Memory (without threads)
* Threads
* Distributed Memory / Message Passing
* Data Parallel
* Hybrid
* SPMD and MPMD

**Shared Memory Model (without threads)**

* In this programming model, processes/tasks share a common address space, which they read and write to asynchronously.
* Various mechanisms such as locks / semaphores are used to control access to the shared memory, resolve contentions and to prevent race conditions and deadlocks.
* An advantage of this model from the programmer's point of view is that the notion of data "ownership" is lacking, so there is no need to specify explicitly the communication of data between tasks.

**Threads Model**

* This programming model is a type of shared memory programming.
* In the threads model of parallel programming, a single "heavy weight" process can have multiple "light weight", concurrent execution paths.
* Unrelated standardization efforts have resulted in two very different implementations of threads: POSIX Threads and OpenMP.
* POSIX Threads
  + Specified by the IEEE POSIX 1003.1c standard (1995). C Language only.
  + Part of Unix/Linux operating systems
  + Library based
* OpenMP
  + Industry standard jointly defined and endorsed by a group of major computer hardware and software vendors, organizations and individuals.
  + Compiler directive based
  + Portable / multi-platform, including Unix and Windows platforms

**Distributed Memory / Message Passing Model**

* A set of tasks that use their own local memory during computation. Multiple tasks can reside on the same physical machine and/or across an arbitrary number of machines.
* Tasks exchange data through communications by sending and receiving messages.
* Data transfer usually requires cooperative operations to be performed by each process. For example, a send operation must have a matching receive operation.

## Data Parallel Model

* May also be referred to as the Partitioned Global Address Space (PGAS) model.
* Address space is treated globally
* Most of the parallel work focuses on performing operations on a data set. The data set is typically organized into a common structure, such as an array or cube.

## Hybrid Model

|  |
| --- |
| * A hybrid model combines more than one of the previously described programming models. * Currently, a common example of a hybrid model is the combination of the message passing model (MPI) with the threads model (OpenMP).   + Threads perform computationally intensive kernels using local, on-node data   + Communications between processes on different nodes occurs over the network using MPI * This hybrid model lends itself well to the most popular hardware environment of clustered multi/many-core machines.   **SPMD and MPMD**   * Single Program Multiple Data (SPMD):   + SPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.   + SINGLE PROGRAM: All tasks execute their copy of the same program simultaneously. This program can be threads, message passing, data parallel or hybrid.   + MULTIPLE DATA: All tasks may use different data   + SPMD programs usually have the necessary logic programmed into them to allow different tasks to branch or conditionally execute only those parts of the program they are designed to execute. That is, tasks do not necessarily have to execute the entire program - perhaps only a portion of it.   + The SPMD model, using message passing or hybrid programming, is probably the most commonly used parallel programming model for multi-node clusters. * Multiple Program Multiple Data (MPMD):   + Like SPMD, MPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.   + MULTIPLE PROGRAM: Tasks may execute different programs simultaneously. The programs can be threads, message passing, data parallel or hybrid.   + MULTIPLE DATA: All tasks may use different data |

**3.What is Flynn’s Taxonomy? Please explain it.**

Flynn's Classical Taxonomy

There are different ways to classify parallel computers. One of the more widely used classifications, in use since 1966, is called Flynn's Taxonomy.

Flynn's taxonomy distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of Instruction Stream and Data Stream. Each of these dimensions can have only one of two possible states: Single or Multiple.

The 4 possible classifications according to Flynn:

1. Single Instruction, Single Data (SISD)
2. Single Instruction, Multiple Data (SIMD)
3. Multiple Instruction, Single Data (MISD)
4. Multiple Instruction, Multiple Data (MIMD)

**4.What kind of classification in Flynn classification does GPU computing belong to? Please explain.**

In Flynn classification GPU computing belong to Single Instruction, Multiple Data (SIMD). A type of parallel computer. Single Instruction: All processing units execute the same instruction at any given clock cycle. Multiple Data: Each processing unit can operate on a different data element. Most modern computers, particularly those with graphics processor units (GPUs) employ SIMD instructions and execution units.

**5.What is the embarrassingly parallel?**

Solving many similar, but independent tasks simultaneously; little to no need for coordination between the tasks.

**6.Give 3 popular math libraries for high performance computing**

1. BLAS basic Linear Algebra Subprograms
2. LAPACK Linear Algebra PACKage
3. ScaLAPACK Scalable Linear Algebra PACKage

**7.How can we evaluate the speedup of parallel computing comparing with serial computing? Please explain in detail.**

Amdahl's Law states that potential program speedup is defined by the fraction of code (P) that can be parallelized:

The number of processors performing the parallel fraction of work, the relationship can be modeled by:

where P = parallel fraction, N = number of processors and S = serial fraction.

**8.Provide 6 MPI necessary functions for MPI parallel computing. Only write the function name, no parameters. C is recommended here.**

1. MPI\_INIT - Initiate an MPI computation
2. MPI\_FINALIZE - Terminate a computation
3. MPI\_COMM\_SIZE - Determine number of processes
4. MPI\_COMM\_RANK - Determine my process identifier.
5. MPI\_SEND - Send a message.
6. MPI\_RECV - Receive a message.

**9.What are strong scaling and weak scaling? Or what difference between them**

**Strong scaling:**

* The total problem size stays fixed as more processors are added
* Goal is to run the same problem size faster
* Perfect scaling means problem is solved in 1/P time (compared to serial)

**Weak scaling:**

* The problem size per processor stays fixed as more processors are added. The total problem size is proportional to the number of processors used
* Goal is to run larger problem in same amount of time
* Perfect scaling means problem Px runs in same time as single processor run

**10.Give several possible reasons why sometimes the computation time on multi-CPU on multiple nodes is slower than that of multi-CPU on a single node.**

* Communications between processes on different nodes add the extra time
* Process slowdown is known as Parallel slowdown. As more processors are added each processor spends more time doing communication, as compared to processing.
* After a particular point when more processing nodes are added, the communication overhead created by adding these nodes surpasses the processing power and the Parallel slowdown occurs.