**CS5168 – Parallel Computing – Assignment 1**

**Chapter 1**

**Part A**

* Question 2: What is the difference between shared memory and message-passing models?

Both shared memory and message-passing models are widely used for parallel programming. Their differences are as follows:

|  |  |
| --- | --- |
| Shared Memory Model | Message-Passing Model |
| Multiple tasks share a single memory area in which we can read and write memory independently and asynchronously.  Easier for the software design – availability of shared variables offers the advantage of not having to clarify the details of communication between tasks.  More difficult to manage data locality, that is managing caches in order to keep data local to the processor that is reading and writing it. | The message passing model is usually applied in cases where each processor has its own memory (referred to as a distributed memory system). Multiple tasks reside on either the same physical processor or distributed on an arbitrary number of interconnected processors.  Software developers using this model are responsible for determining the amount of parallelism, the number of processors and the data exchange that occurs through the explicit sending and receiving messages. |

* Question 3: What is the difference between multithreaded control parallelism and data parallelism?

The differences between multithreaded control parallelism and data parallelism are as follows:

|  |  |
| --- | --- |
| Multithreaded Control Parallelism | Data Parallelism |
| In this form of parallelism, a process can have multiple flows of execution (each flow is referred to as a thread)  Each thread consists of a program counter, a stack, and a set of registers as well as an identifier. When threads operate on shared memory, programmers must prevent multiple threads from updating the shared memory locations in a way that affects the correctness of the computation (i.e., a race condition). Thus, multi-threaded programming can be a complex and error-prone practice. | In this form of parallelism, tasks operate on subsets of the same data structure, usually arrays.  Each task operates on a different portion of data array, but essentially does the same operation, which greatly simplifies the control needed to execute. The simplified control used in the data parallel model helps programmers avoid the race conditions often arising in multi-threaded control model. |

* Question 7: How does Amdahl’s Law differ from Gustafson’s Law?

The differences between Amdahl’s Law and Gustafson’s Law are as follows:

|  |  |
| --- | --- |
| Amdahl’s Law | Gustafson’s Law |
| Provides pessimistic projection for parallel speedup.  States that the maximum speedup that can be achieved is limited by the serial component of the work of any program.  Speedup <= 1 / 1-P , where 1 – P denotes the percentage of the serial component (not parallelized) of a program. | Provides optimistic projection for parallel speedup.  Gustafson's law stands in contrast to Amdahl's law, which assumes that the overall workload of a program does not change with respect to the number of processors. This law states that The faster the parallel system is, the larger the problems that can be solved over the same period of time.  Gustafson's law is parameterized by p, the number of processors, and states the following result about parallel speedup: Speedup(p) = p - α(p-1) |

**Part B**

* Question 11: A program just ran in 0.32 seconds on a core with speed 2 Tflop/sec, how many Gflop does the program perform?

Execution Time = 0.32 s

Speed = 2 Tflop/sec

We know, Execution Time = Work/Speed 🡪 Work = Execution Time \* Speed

= 0.32 \* 2 \* 1012 / 109 Gflops

= 640 **Gflops**

* Question 12: You have to run a program that performs 2000 Tflop, and your core computes at speed 450 Gflop/sec. How long will the program run?

Work = 2000 Tflops

Speed = 450 Gflops/sec

We know, Execution Time = Work/Speed

= 2000 \* 1012 / (450 \* 109) = **4444.44 seconds**

* Question 13: A program that performs 3000 Gflop just ran in 1.5 minutes on a core. What is the core speed in Tflop/sec?

Work = 3000 Gflops

Time = 1.5 minutes = 90s

We know, Execution Time = Work/Speed 🡪 Speed = Work / Execution Time

= 3000 \* 109 / (90 \* 1012) = **0.0333 Tflops/sec**