**FAST National University of Computer and Emerging Sciences**



**Research Topic:** To what extent can parallelism improve time efficiency of Bubble Sort Algorithm over a large dataset?

**Instructor:** Nadeem Kafi

**Group Members:**

1. Aahil Halani (20K-0308)
2. Ali Ahmed (20K-0140)
3. Faizan Hakim (20K-0290)

**Date:** 19th October, 2022

**Description:**

Bubble sort algorithm is one of the most commonly used comparative sort algorithm that can be used to sort a collection of data elements in either ascending or descending order. It has a space complexity of O (1), making it a viable option when memory is limited. However, in time bound situations, this algorithm could perform poorly over large data set as average time complexity of the algorithm is calculated as Ꝋ (). This implies that the algorithm has a tendency to consume large amount of CPU resources which in result degrades system’s overall performance. Hence, to overcome this effect, parallelism techniques could be utilized to improve the time and overall CPU utilization of the algorithm. The techniques of parallelism that are going to be utilized in this project lie in the domain of Open-MP and MPI libraries.

Open-MP is a high-level approach to parallelism based on threads sharing a memory space. Tasks and relevant data sets are distributed amongst implicitly created threads with in a single system and results from each thread can be communicated within threads using a common shared memory. Threads can be tracked using unique thread ids and inclusion of many highly functional, user-friendly clauses make this approach easy to implement.

MPI on the other hand, is a message passing based approach relying on a distributed memory scheme. Tasks and memory sets are distributed to number of processes working in a single MPI communication network sometimes referred MPI\_COMM\_WORLD. Each process within this communication world has an ability to send, receive, or gather data elements from all other processes lying in the same communication world by utilization many message passing calls.

Both approaches have a common technique to distribute independent tasks and gather results from each distribution to combine into a single solution, saving time and CPU resources. Therefore, this study aims to investigate the highest gain in performance that can be achieved by using either of the two parallelism models. This information can hopefully come in hand for developers when designing and implementing a real time, time bound systems.

**Methodology:**

1. A data set containing n random values would be created. (n size may vary to 1K to 100K).
2. Bubble sort algorithm will be tailored and implemented on data set using each parallelism module (Open-MP and MPI) separately.
3. Performance of each module will be recorded upon each task completion. The time stamp with number of thread or process used will be stored in a file; separate files for Open-MP and MPI.
4. Comparison in performance will be made according to time taken to complete the task against varying number of process and threads.

**Tools and Technique:**

* **Programming Language**: C-Language
* **Operating System**: Ubuntu (Linux) Version 16
* **Libraries**: omp.h, mpi.h