Roll No : 41453

# High Performance Computing : Assignment 3

**Title :**
Parallel Sorting Algorithms

**Problem Statement :**
For Bubble Sort and Merge Sort, based on existing sequential algorithms, design and implement parallel algorithm utilizing all resources available.

**Learning Objectives** :
- Learn parallel decomposition of problem.
- Learn parallel computing using CUDA.

**Learning Outcome:**
Be able to :
- Decompose problem into sub problems, to learn how to use GPUs, to learn to solve sub problem using threads on GPU cores.

**Software and Hardware Requirements :**
- 64 bit CPU
- 4 GB RAM
- Ubuntu 18 OS
- CUDA toolkit
- Nvidia GPU
- NVCC compiler

**Steps to run:**
- Install CUDA for your GPU
- Check NVCC version
- Use NVCC to compile .cu program (nvcc filename.cu -o filename)
- Run .cu program using generated output file (./filename)
- Check runtimes using command nvprof (nvprof ./filename)

**Programmers Perspective:**
S = {s; e; X; Y; $F_{me}$; $F_f$; DD; NDD}
s = start state (start of program)
e = end state (final required output obtained)
X = {X1} where X1 is element of Vector.
Y= Output Set (Sorted Vector)
$F_{me}$ = function to perform parallel sorting operations.
$F_f$ = {f1,f2,f3}
where
f1 = decomposition function

f2 = function to generate vectors
f3 = function to display results
DD = Vector of Elements.
NDD = No non deterministic data

**Theory :**

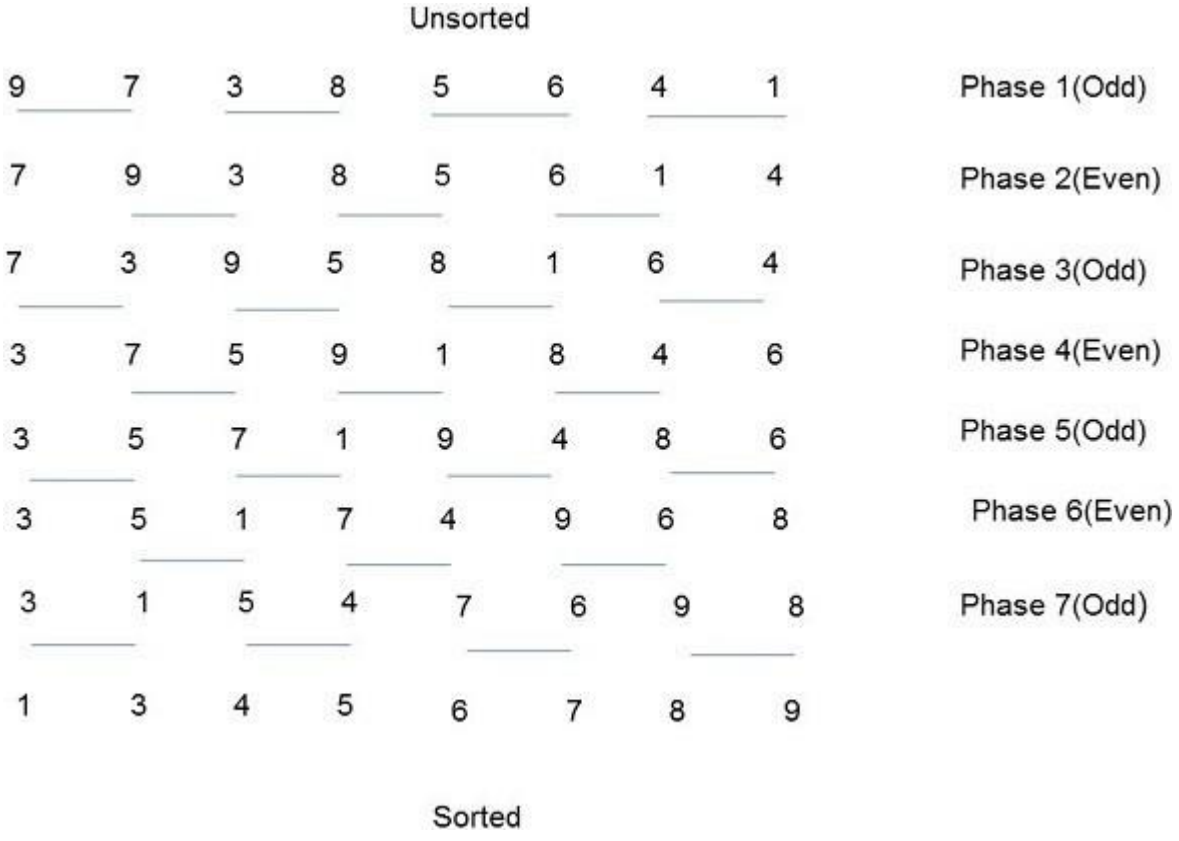
**Bubble Sort :**
Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of O(n^2) where n is the number of items. We observe in algorithm that Bubble Sort compares each pair of array element unless the whole array is completely sorted in an ascending order. This may cause a few complexity issues like what if the array needs no more swapping as all the elements are already ascending.
To ease-out the issue, we use one flag variable swapped which will help us see if any swap has happened or not. If no swap has occurred, i.e. the array requires no more processing to be sorted, it will come out of the loop.

**Merge Sort :**
Merge sort is one of the most efficient sorting algorithms. It works on the principle of Divide and Conquer. Merge sort repeatedly breaks down a list into several sublists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list.
The top-down merge sort approach is the methodology which uses recursion mechanism. It starts at the Top and proceeds downwards, with each recursive turn asking the same question such as "What is required to be done to sort the array?" and having the answer as, "split the array into two, make a recursive call, and merge the results.", until one gets to the bottom of the array-tree.

**Message Passing Interface :**
Message Passing Interface (MPI) is a specification for a standard library for message passing that was defined by the MPI Forum, a broadly based group of parallel computer vendors, library writers, and application specialists(William Gropp). MPI is a specification not an implementation. The main purpose of MPI is to develop a widely used standard for writing message passing programs and achieving practical, portable, efficient and flexible standard for message passing. MPI is a mixture of functions, subroutines, and methods and also have different syntax from other languages.
In parallel programming the problem will be divided into subproblems or processes and assign to different processors. The rule of MPI is to establish communication among the processes, hence MPI is used for distributed memory not for the shared memory. This means that the system works only for sending and receiving the data with their own memory.

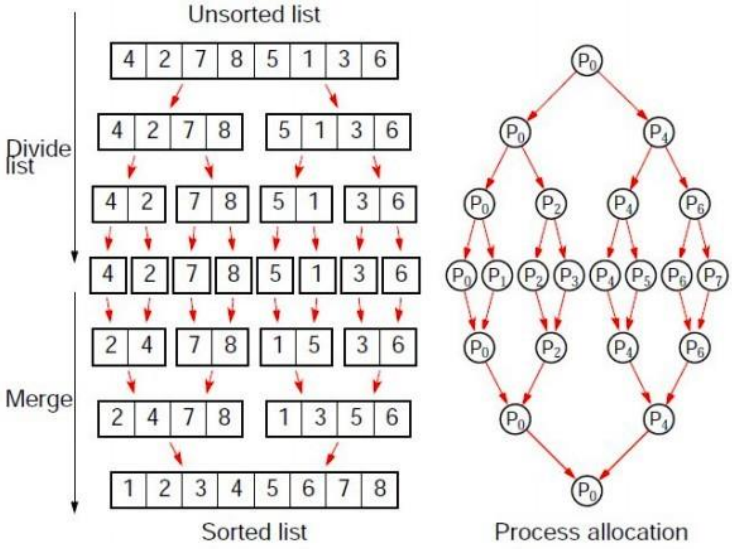
**Parallel Bubble Sort :**

Odd-Even Transposition Sort is based on the Bubble Sort technique. It compares two adjacent numbers and switches them, if the first number is greater than the second number to get an ascending order list. The opposite case applies for a descending order series. Odd-Even transposition sort operates in two phases – odd phase and even phase. In both the phases, processes exchange numbers with their adjacent number in the right.




**Parallel Merge Sort :**

The idea is to take advantage of the tree structure of the algorithm to assign work to processes.



If we ignore communication time, computations still only occur when merging the sublists. But now, in the worst case, it takes 2s − 1 steps to merge all sublists (two by two) of size s in a merging step. Since in total there are log(n) merging steps, it takes log X (n) i=1 (2 i − 1) steps to obtain the final sorted list in a parallel implementation, which corresponds to a time complexity of O(n).

**Conclusion :**
Successfully implement parallel bubble and merge sort algorithms using openmp interface.