Assignment

- Title : Parallel searching Algorithm
- Problem statement:
  Design and implement parallel algorithm utilizing all available resources for
  - Binary search for sorted array
  - Depth First search or Breadth first search or Best first search
- Objective:
  We will be able to:
  - learn about parallel searching technique
  - Learn about MPI
- Software and Hardware Requirements:
  - OS: Fedora/Ubuntu (64bit)
  - GCC compiler
  - Editor: gedit
  - MPICC compiler using OpenMPI
  - RAM : 4 GB
  - HDD: 500 GB

- Theory:

A.] Binary search:

- Binary search, also known as logarithmic search is an algorithm that finds the position of the target within a sorted array
- It compares the target value with the middle element of array. If they are not equal, the half in which target element cannot lie is eliminated and search continues on remaining half.
- If search ends with remaining half being empty, the target is not array
- Binary search runs the logarithmic time in the worst case, making O(logn) comparisons where n = size array

B.] Breadth first search:

- BFS is the most common graph transversal algorithm
- It starts traversing from source and traverses the graph layerwise, thus exploring the neighbor nodes first
- In sequential implementation a queue is maintained of neighbor nodes in each layer

C.] OpenMPI:

- It is a Message Passing Interface library which provides extremely high and competitive performance
- The OpenMPI code has 3 major code modules
  1. OMPI - MPI code
  2. ORTE - Open Runtime Env.
  3. OPAL - Open Portable Access Layer
- mpicc compiler is used to compile the C/C++ codes embedded with open MPI.

- Algorithms:

A.] Parallel binary Search (sorted array)

1.) Divide array into M blocks of size n/M
2.) Apply one step of comparisons to the middle element of each block
3.) If equal, address is returned & terminate
4.) Otherwise, identify adj blocks & form a new block starting from element foll. the one that signalled (>) & ending at element preceding signalled (<)
5.) If they are same element, return index
6.) Otherwise, parallel binary search (new block)

B.) Breadth first search (graph root G, source S)

1.) Enque (S)
2.) mark S as visited
3.) while (Φ is not empty)
    //remove the vertex from Φ whose neighbor will be visited now
    3.1.) v = deque(Φ)
    //processing all neighbors of v
    // w = neighbor of v neighbor
    3.2.) if (w is not visited)
        3.2.1) enque (w)
    3.3.) end if
4.) end while

- Conclusion:

Thus we have successfully implemented parallel binary search and breadth first search using OpenMPI.