Assignment HPC-4

Title: Parallel Search Algorithms

Problem Statement

Design and implement parallel algorithm utilising all available resources for

- Binary search for sorted array
- Depth first search (DFS) or Breadth first search (BFS) or Best first search

Objectives

To study and learn about parallel implementation of searching algorithms.

To learn about MPI API in C/C++

Outcome

To be able to learn about parallel searching techniques

To be able to learn about MPI

Software and hardware requirements

Fedora 20/ Ubuntu (64-bit), GCC/C++ compiler, MPICC compiler using OpenMPI, 4GB RAM, 500 GB HDD

Theory related concepts

1. Binary Search

- It is an algorithm that finds the position of the target value within a sorted array.

- It compares the target value with the middle element of an array. If they are not equal, the half in which target element cannot lie is eliminated and the search continues the remaining half.
- If the search ends with remaining half being empty, the target is not the array.

2. Breadth first search

- BFS is the most traversed algorithm
- It starts traversing from the source and travels the graph lengthwise, thus exploring the neighbour nodes first
- A queue is maintained of the neighbour nodes in each layer

Open MPI

- It is a message passing interface library which provides extremely high and competitive performance.
- The OPEN MPI code has 3 modules

1) OMPI : MPI code

2) ORTE : Open Runtime Environment

3) OPAL : Open Portable Access layer

Algorithm

A) Parallel Binary Search

Parallel_binary_search (sorted array)

1. Divide the array into m blocks of size n/m.
2. Apply one step of comparison of the middle element of each block
3. If equality obtained, return address and terminated
4. Otherwise, identify the adjacent blocks and form a new block

starting from the element following the one that signalled (>)
and ending at the element preceding the the that signalled (<)

5. If they are same element, return index
6. Otherwise, parallel_binary_search (new block).

B) Breadth first Search

```
BFS (Graph root Q, source s)
1. enqueue (s)
2. Mark s as visited
3. While(Q is not empty)
      //remove that vertex from Q whose neighbour will be visited now
   3.1 v= deque (Q)
      //processing all the neighbour of v
      // w = neighbour of v
   3.2 if (w is not visited)
      3.2.1 enque (w)
   3.3 endif
4. end while
```

Test Cases

	Input size	Sequential time	Parallel time	Efficiency
Binary search	n = 1024	1.153	1.542	0.747
(key = 54)	n = 2048	1.673	1.236	1.357
	n = 4096	1.075	0.933	1.150
Depth first	n = 1024	0.011	0.007	1.57
search traversal	n = 2048	0.05	0.019	2.63
	n = 4096	0.109	0.026	4.19

Efficiency = WCSA / WCFA

Conclusion

Thus, we understood and successfully implemented parallel searching algorithms i.e. binary search and breadth first search.
traversal