COMS3008A Assignment – Report

Sahil Dinanath, Darshan Singh

Put your submission date here

1 Problem 1: Parallel Scan

- Given a set of elements, $[a_0, a_1, \cdots, a_{n-1}]$, the scan operation associated with addition operator for this input is the output set $[a_0, (a_0 + a_1), \cdots, (a_0 + a_1 + \cdots + a_{n-1})]$.
- For example, the input set is $[2, 1, 4, 0, 3, 7, 6, 3]$, then the scan with addition operator of this input is $[2, 3, 7, 7, 10, 17, 23, 26]$.

Approach

1.1 Serial

1.2 OpenMP

1.3 MPI

2 Problem 2: Parallel Bitonic Sort

Approach

2.1 Serial

2.1.1 Algorithm

The main code that implements the Bitonic Sort algorithm is shown in Listing 1.

```
// Code snippet of the Bitonic Sort algorithm
// Function definitions for compAndSwap, bitonicMerge, and bitonicSort

int main(int argc, char *argv[]) {
  // Initialization and setup code
  // Read input file and convert characters to integers
  // Start timing
  // Call bitonicSort to sort the input array
  // Stop timing
  // Print sorted array
}
```

Listing 1: Bitonic Sort Algorithm

2.1.2 Testing

The code begins by reading the input file and converting the characters to integers. This section of the code consists of the following functions:

- `getFileSize`: This function determines the size of the input file by seeking to the end of the file, retrieving the current position (which represents the file size), and then resetting the file position to the beginning.
- `readFile`: This function reads the contents of the input file into a character array `line` using the `fgets` function. It also closes the file after reading.
- `convertCharToIntArray`: This function converts the character array `fileCharacters` into an array of long integers `input` by subtracting the ASCII value of `'0'` from each character.

2.1.3 Results

The Bitonic Sort algorithm is then applied to the input array using the `bitonicSort` function. The execution time of the sorting process is measured using the `omp_get_wtime` function. Finally, the sorted array is printed using the `printArray` function, and the total execution time is displayed.

Table 1: Bitonic sort with different input sizes

No of characters	2^3	2^{16}	2^{23}	2^{26}	2^{28}
Serial	0.1	0.2	0.3	0.4	0.5
Parallel					
Speedup	2	3	4	5	6

2.2 OpenMP

2.3 MPI

3 Problem 3: Parallel Graph Algorithm

Approach

3.1 Serial

3.2 OpenMP

3.3 MPI

An example figure is given in Figure 1.

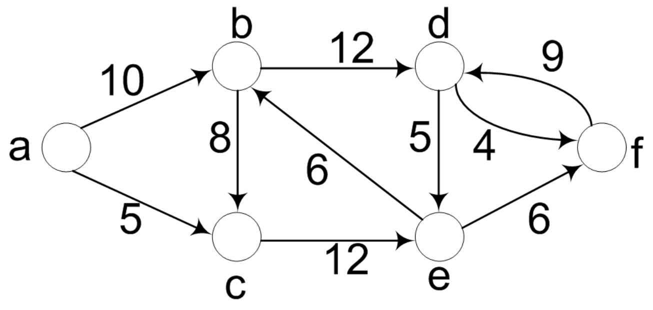

Figure 1: A directed graph

An example of table is given Table 2.

Table 2: An example of a table

No of vertices	64	128	256	384	512
Serial	0.1	0.2	0.3	0.4	0.5
Parallel					
Sppedup	2	3	4	5	6