COL380 - Parallel and Distributed Programming : Assignment - 1

Param Khakhar - 2018CS10362

24 February 2021

Introduction

The following is a writeup for the Assignment-1, Parallel and Distributed Programming (COL380). There are sections corresponding to each of the four questions and within each section, there are sub-sections for each sub-part.

Task -1 : Parallelized Versions

Two different parallelized versions of the summation are implemented.

- In the first approach, nested parallelism is used, and a local variable corresponding to each thread stores the total sum for that thread, and finally the local sums from the all the threads are added in the global variable sum.
- The second approach is an implementation of the tree based approach as discussed in the second lecture. In this approach, pairs of elements are added and the size of the array reduces by half at each stage. The height, of the tree so formed would be **O(logN)**.

Task -2 Performance Analysis

2.1 First Approach

The following execution times for the entire programs and the parallelized parts was observed for the first approach on various threads and inputs sizes.

Limit|Threads	**1**	**2**	**4**	**8**
1e03	0.041	0.290	0.197	8.811
1e05	3.813	3.636	2.338	5.791
1e07	122.026	95.856	87.911	85.502

Table 1: The Time Values are in milliseconds

The following *speedup* and *efficiency* values are observed for different number of threads and limits for summation for the first approach.

Size	**p**	**1**	**2**	**4**	**8**
1e03	S	1.0	0.1415	0.2086	0.0047
	E	1.0	0.0707	0.0521	0.0005
1e05	S	1.0	1.0487	1.6310	0.6584
	E	1.0	0.5243	0.4078	0.0823
1e07	S	1.0	1.273	1.3881	1.4271
	E	1.0	0.6365	0.347	0.1789

Table 2: S = Speedup, E = Efficiency

2.2 Second Approach

The following execution times for the entire programs and the parallelized parts was observed for the second approach on various threads and inputs sizes.

Limit|Threads	**1**	**2**	**4**	**8**
1e03	0.041	0.134	0.123	0.110
1e05	3.813	4.648	3.951	2.56
1e07	122.026	125.327	123.177	112.395

Table 3: The Time Values are in milliseconds

The following *speedup* and *efficiency* values are observed for different number of threads and limits for summation for the first approach.

Size	**p**	**1**	**2**	**4**	**8**
1e03	S	1.0	0.3068	0.3324	0.3715
	E	1.0	0.1534	0.0831	0.0464
1e05	S	1.0	0.8204	0.9652	1.4897
	E	1.0	0.4102	0.2413	0.1862
1e07	S	1.0	0.9737	0.9907	1.0857
	E	1.0	0.4868	0.2477	0.1357

Table 4: S = Speedup, E = Efficiency

Task -3 Remarks

- From the above observations, we can conclude that **Amdahl's Law** is approximately followed, as the **efficiency values** are **decreasing consistently** for both the approaches, which implies that the added advantage on increasing the threads is decreasing and ultimately the overheads for parallelizing the code would be more as compared to the advantage on the computation time as is the case for 8 threads in the first approach for the input size of 1e03, and 1e05.

- During the experimentation, significant variations were present in the above data, and the values mentioned above are the ones which were occurring most frequently during multiple executions.

- The *Speedup* values are increasing mostly along a row as well as along a column.

- The *Efficiency* values are decreasing mostly along a row but increasing along a column.
- The *Serial* approach performs reasonably well, which can be attributed to the simplicity of the task as well as caching of the entries by the memory.