NAME: Rayed Muhammad Saeed

ROLL NO: 20i-1822

DEGREE: BSCS

SECTION: F

SUBJECT: Parallel and Distributed Computing

SUBMITTED TO: Sir Aleem

SUBJECT CODE: CS-3006

ASSIGNMENT NO: 1

DATE: 6th February 2023

Report For Assignment 1

Task 1:

In task 1 we had to analyze the execution of the program by dividing the work among the number of cores of our machine and by using only a single core.

The task was to first populate an array of size 2^16 which is equal to 65536, with random numbers and then calculate the square root of this array and replace the array with the updated square root values. Finally, we had to print out this resultant square root array.

Performance analysis:

For this we used affinity and recorded the Gflops/s taken for both using affinity and without using affinity. The results are below:

```
rayed@rayed:~/Documents/PDC/Assignment1$ gcc -o t1 t1.c -lm -pthread
rayed@rayed:~/Documents/PDC/Assignment1$ ./t1 2
Number of CPUs is: 2
The time taken for execution using affinity is:
 Performance in Gflops 0.054 Gflop/s

The time taken for execution without using affinity is:
 Performance in Gflops 0.040 Gflop/s

The resultant array is:

120 121 107 109 125 107 115 108 103 104 96 97 113 101 103 104 125 100 89 113
100 111 93 112 121 123 125 100 118 102 99 115 100 119 101 102 99 124 119 115
105 91 121 126 103 97 103 106 110 107 89 123 93 93 108 91 93 110 101 122
124 111 114 98 103 123 109 115 121 105 103 99 107 97 103 122 107 119 101 126
102 105 125 106 112 107 108 116 126 118 114 124 102 101 99 114 98 121 105 96
99 117 109 119 89 121 115 108 116 91 107 90 107 107 107 94 125 123 119 92
119 107 93 96 120 103 123 90 96 102 97 105 91 119 100 91 116 124 109 106
92 92 107 109 112 89 113 115 90 109 117 119 90 120 124 121 95 121 122 106
96 96 123 101 124 97 106 117 98 124 96 100 93 116 121 115 116 111 103 117
95 96 109 96 125 111 90 98 109 126 117 114 100 113 90 101 120 108 90 93
109 97 104 112 122 102 103 115 125 119 105 93 124 126 100 126 111 105 102 95
108 90 122 117 114 123 94 111 104 95 114 124 106 94 110 105 110 122 92 108
117 108 111 118 111 124 118 97 106 96 102 126 97 97 120 124 93 124 108 111
96 98 109 111 106 94 124 124 93 93 109 120 115 95 112 98 92 107 105 112
116 117 112 123 89 109 123 97 106 105 118 112 115 104 99 93 108 100 93 115
104 112 109 90 118 96 99 123 117 117 109 106 110 97 106 110 118 103 117 97
120 112 121 108 90 92 115 110 106 118 100 119 104 121 123 94 90 99 94 121
124 117 104 108 123 119 90 115 93 121 124 89 106 122 108 107 90 99 92 107
89 107 103 108 101 99 115 102 108 119 99 109 109 113 126 109 104 93 99 111
90 96 111 107 94 95 89 95 108 96 112 109 113 123 91 122 99 117 101 116
109 114 102 93 98 101 112 116 109 123 103 110 96 89 92 101 99 96 106 117
```

The above screenshot shows the performance in Gflops/s for both using affinity and without using affinity.

The above analysis shows that using affinity the Gflops/s is higher and hence the time taken to compute the result is lesser compared to the performance achieved without the use of affinity based on a single core.

Variance:

Below is the mean and sample variance calculated of the two performances:

```
116 119 89 97 112 108 95 121 108 121 122 120 115 123 112 118
The mean of the performances in Gflops/s is: 0.046900
The difference between the two performances in Gflops/s is: 0.013989
The sample variance of the performance in Gflops/s is: 0.000098
rayed@rayed:~/Documents/PDC/Assignment1$
```

The above screenshot shows the mean of the two performances calculated with and without using affinity, the difference between the two and finally the sample variance.

Task 2:

In this task we had to make a dynamic 2d array of size 2^16*2^16, however the maximum allowed was 2^8*2^8. This array was then populated and its sum calculated via different methods divided into versions.

Version 1 was for blockwise distribution where we had to make 3 different models, one with blocksize 2^2, second with 2^4, and last with 2^8.

Version 2 was for cyclic distribution where each thread was to compute 4 rows.

For all of these 4 models we calculated the performance of the models in Gflops/s. Below is the analysis shown:

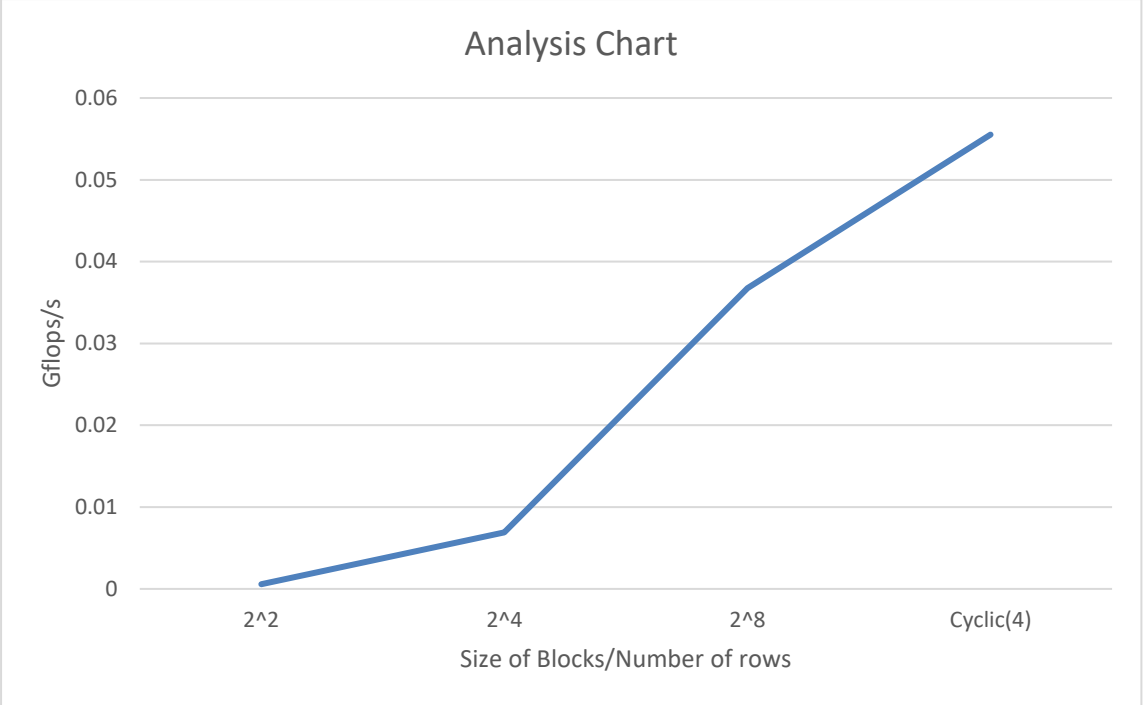
```
rayed@rayed:~/Documents/PDC$ gcc -o t2 t2.c -lm -pthread
rayed@rayed:~/Documents/PDC$ ./t2
The sum of array is: 442591209
The performance for blockwise 2^2 is:
 Performance in Gflops 0.00057209 Gflop/s

The sum of array is: 1134190728
The performance for blockwise 2^4 is:
 Performance in Gflops 0.00690470 Gflop/s

The sum of array is: 1914646308
The performance for blockwise 2^8 is:
 Performance in Gflops 0.03677150 Gflop/s

The sum of array is: 1914646308
The performance for cyclic is:
 Performance in Gflops 0.05552722 Gflop/s

rayed@rayed:~/Documents/PDC$
```



The first 3 values of the x axis (2^2,2^4,2^8) are the block sizes for the block wise distribution. And the last distribution cyclic, we had divided the threads into computing 4 rows each and there were 2 threads created. The Gflops/s values of each are graphed above.

The above graph shows that increasing the number of block size increases the performance as is shown by the increasing trend. The graph also shows that using cyclic gives better performance than all the other block wise distributions for the array 2^8*2^8.