CENG 342 – Parallel Programming I
Spring 2023

HW3 REPORT

Parallel Programming Assignment, MPI

18050111056
Koray ÇELİK

03/04/2023

Brief Explanation

This program creates a square matrix and a vector. Gives the result of vector-matrix multiplication in parallel. Parallelism is provided by usage of MPI.

First, empty matrix, empty vector and empty result array is created. Then, matrix and vector is filled with random double numbers. Matrix is divided by the cores provided from the input from the command 'mpirun -n 2 ./hellomake'. Each part of the matrix is multiplied with the vector and summed up in the main thread, where my_rank = 0.

Results

2 cores run

```
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 2 ./hellomake 1056 156 output.txt
Elapsed time is 0.001740 seconds for parallel mxv with 2 processes (large matrix)
Elapsed time is 0.000048 seconds for parallel mxv with 2 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 2 ./hellomake 1056 156 output.txt
Elapsed time is 0.001687 seconds for parallel mxv with 2 processes (large matrix)
Elapsed time is 0.000050 seconds for parallel mxv with 2 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 2 ./hellomake 1056 156 output.txt
Elapsed time is 0.001684 seconds for parallel mxv with 2 processes (large matrix)
Elapsed time is 0.000048 seconds for parallel mxv with 2 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 2 ./hellomake 1056 156 output.txt
Elapsed time is 0.001716 seconds for parallel mxv with 2 processes (large matrix)
Elapsed time is 0.000047 seconds for parallel mxv with 2 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 2 ./hellomake 1056 156 output.txt
Elapsed time is 0.001734 seconds for parallel mxv with 2 processes (large matrix)
Elapsed time is 0.000043 seconds for parallel mxv with 2 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$
```

1 core run

```
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 1 ./hellomake 1056 156 output.txt
Elapsed time is 0.003311 seconds for parallel mxv with 1 processes (large matrix)
Elapsed time is 0.000073 seconds for parallel mxv with 1 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 1 ./hellomake 1056 156 output.txt
Elapsed time is 0.003307 seconds for parallel mxv with 1 processes (large matrix)
Elapsed time is 0.000072 seconds for parallel mxv with 1 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 1 ./hellomake 1056 156 output.txt
Elapsed time is 0.003307 seconds for parallel mxv with 1 processes (large matrix)
Elapsed time is 0.000072 seconds for parallel mxv with 1 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 1 ./hellomake 1056 156 output.txt
Elapsed time is 0.003311 seconds for parallel mxv with 1 processes (large matrix)
Elapsed time is 0.000074 seconds for parallel mxv with 1 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$ mpirun -n 1 ./hellomake 1056 156 output.txt
Elapsed time is 0.003316 seconds for parallel mxv with 1 processes (large matrix)
Elapsed time is 0.000072 seconds for parallel mxv with 1 processes (small matrix)
kryc1@kryc1-F800-NOTEBOOK-DISCRETE:~/CENG342/tests$
```

Large Matrix Comparison Table

	1 core run elapsed time	2 cores run elapsed time	Improvement
Run 1	0.003311	0.001740	47.41%
Run 2	0.003307	0.001687	49.04%
Run 3	0.003307	0.001648	47.39%
Run 4	0.003311	0.001716	48.21%
Run 5	0.003316	0.001734	47.72%
average	0.0033104	0.001705	48.50%

Small Matrix Comparison Table

	1 core run elapsed time	2 cores run elapsed time	Improvement
Run 1	0.000073	0.000048	34.25%
Run 2	0.000072	0.000050	30.56%
Run 3	0.000072	0.000048	33.33%
Run 4	0.000074	0.000047	36.49%
Run 5	0.000072	0.000043	40.28%
average	0.000073	0.0000455	37.67%

Conclusion

As we see from the results, as the matrix size enlarges, we gain more improvement in parallel execution over single core execution. So, it makes sense to parallelize our programs if we are working with large datasets.