Parallel Programming Exercise　8 – 10

Author:	邱亮茗 (r12942159@ntu.edu.tw)
Student ID	R12942159
Department	Graduate Institute of Communication Engineering

(If you and your team member contribute equally, you can use (co-first author), after each name.)

1　Problem and Proposed Approach

(Brief your problem, and give your idea or concept of how you design your program.)

The task is to implement a matrix-vector multiplication. The matrix is decomposed into smaller checkerboard blocks that are distributed across multiple processors. The vector is shared among processors as needed to perform the multiplication.

(1) Input Handling: The root processor (rank 0) reads the matrix and vector from the input files and the matrix is partitioned into checkerboard blocks, and the vector is partitioned as needed.
(2) Distribution of Data: catter matrix blocks to their respective processors using MPI_Scatter or MPI_Scatterv. And broadcast the vector to all processes along the corresponding rows or columns using MPI_Bcast.
(3) Local Computation: Each processor multiplies its local block of the matrix by the relevant portion of the vector.
(4) Result Gathering: Collect partial results from all processors using MPI_Gather or MPI_Reduce and combine these to form the final result vector on the root processor.
(5) Output: The root processor prints the resulting vector.

2　Theoretical Analysis Model

(Try to give the time complexity of the algorithm, and analyze your program with iso-efficiency metrics)

Matrix size: N×N
Vector size: N
Number of processes: P, organized into a sqrt(P)×sqrt(P) grid.
Computational time: $\chi \lceil n/\sqrt{p} \rceil \lceil n/\sqrt{p} \rceil$
Send/Recv: $\lambda + 8 \lceil n/\sqrt{p} \rceil / \beta$
Broadcast: $\log \sqrt{p} \, (\lambda + 8 \lceil n/\sqrt{p} \rceil / \beta)$
⇨ Time complexity is given by $\log \sqrt{p} \, (\lambda + 8 \lceil n/\sqrt{p} \rceil / \beta)$

And block size per process: N/sqrt(P) × N/sqrt(P) .
⇨ T(N,1) = O(N^2)

⇨ T(N,P) = O(P * (N / sqrt(P))*log(P))

⇨ N^2 >= C * P * (N / sqrt(P))*log(P)

⇨ iso-efficiency metrics: M(C * sqrt(P) * log(P)) / P = C^2 * log^2 (P)

i.e Good scalable algorithm.

3 Performance Benchmark

(Give your idea or concept of how you design your program.)

Table 1. The execution time with N = 3000

Processors	1	2	3	4	5	6	7	8
Real execution time	0.032495	0.017070	0.011478	0.009172	0.009799	0.009207	0.008131	0.010878
Estimate execution time	0.032495	0.016248	0.010832	0.008124	0.006499	0.005416	0.004642	0.004062
Speedup	1	1.90	2.83	3.54	3.32	3.53	4.00	2.99
Karp-flatt metrics		0.0506	0.0298	0.0430	0.1269	0.1400	0.1253	0.2397

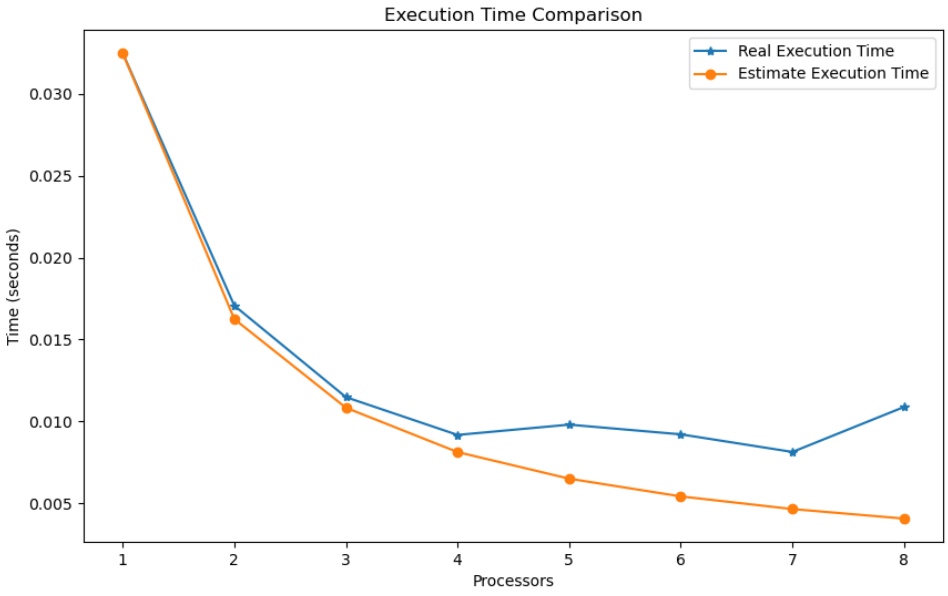

Figure 1. The performance of diagram

4 Conclusion and Discussion

(Discuss the following issues of your program

1. What is the speedup respect to the number of processors used?

   The speedup values reflect how much the execution time is reduced when more processors are used. It increases as more processors are added up to 7

processors, then drops slightly with 8 processors, likely due to communication overhead or inefficiencies.

2. How can you improve your program further more?

   Minimize the frequency and volume of data communication between processors by optimizing data partitioning or merging multiple communications and ensure an even distribution of work across all processors to prevent bottlenecks caused by imbalanced workloads.

3. How does the communication and cache affect the performance of your program?

   Communication: As the number of processors increases, communication overhead can significantly affect performance, especially if frequent synchronization is needed.

   Cache: Poor memory access patterns or large data sizes that exceed the cache capacity can lead to frequent cache misses, slowing down memory access and overall execution.

4. How does the Karp-Flatt metrics and Iso-efficiency metrics reveal?

   Karp-Flatt metrics: This metric indicates the serial fraction and overhead of the parallel computation. When it increases as more processors are added, it suggests that communication and synchronization overheads are becoming more significant.

   Isoefficiency metrics: This metric reveals the scalability of a parallel system. It shows how much the problem size must grow to maintain efficiency as more processors are added. Higher isoefficiency values indicate less scalability, implying that larger problem sizes are needed to effectively utilize additional processors.

)

Appendix(optional):

(If something else you want to append in this file, like picture of life game)