Parallel Programming Exercise 5 – 11

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 create a parallel program using MPI (Message Passing Interface) in C that computes the sum of a harmonic series, defined as:

Sn=1/1+1/2+1/3+...+1/n.

Step1 **Input Handling**: On process 0, prompt the user to enter the values for n and d.
Step2 **Computing the Harmonic Sum**: Data decomposition is used. The harmonic sum among all available processes.
Step3 **Combining Results**: Use MPI_Reduce() to combine the partial results from all processes to compute the total harmonic sum.

2 Theoretical Analysis Model

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

Loop Execution: Every iterations per process ≈ n / p.
MPI_Reduce: Takes O(logp) time.
The total time complexity can be summarized as: O(n / p + logp).
=>To(n, p) = plogp
=> M(n) = n; n >= Cplogp
=> M(f(p)) / p = Cplogp / p = Clogp
=> The system has good scalability

3 Performance Benchmark

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

Table 1. The execution time

Processors	1	2	3	4	5	6	7	8
Real execution time	0.003528	0.001821	0.001253	0.009474	0.007901	0.006759	0.005648	0.004915
Estimate execution	0.003528	0.001764	0.001176	0.008821	0.007057	0.005880	0.005040	0.004410

time								
Speedup	1	1.9370	2.8156	3.7241	4.4654	5.2203	6.2468	7.1782
Karp-flatt metrics		0.0325	0.0274	0.0244	0.0221	0.0212	0.0201	0.0189

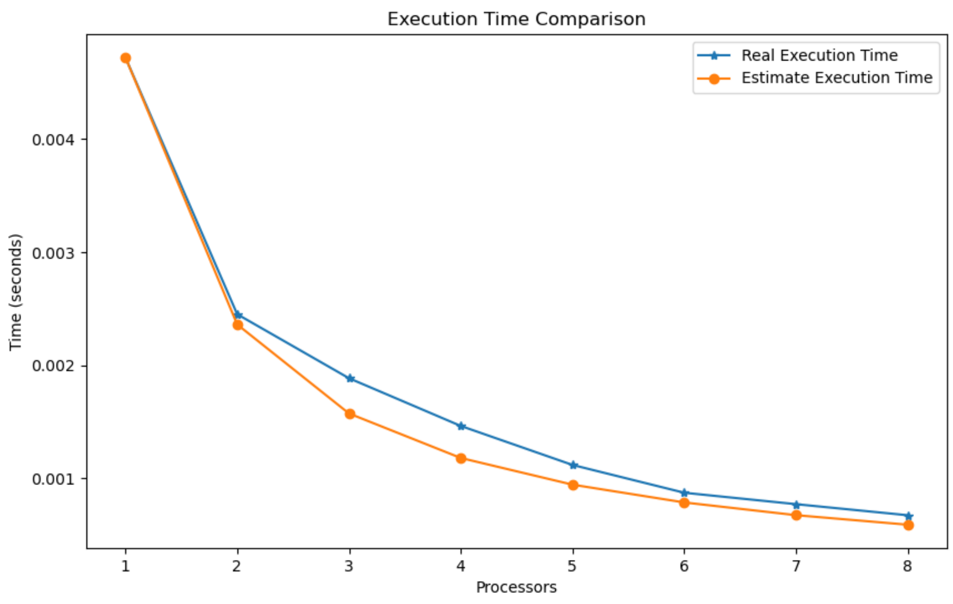

Figure 1. The performance of diagram

4 Conclusion and Discussion

(Discuss the following issues of your program

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

   In this case, these values show the speedup achieved as the number of processors increases and shows that adding more processors continues to improve performance, though the efficiency gains reduce as the processor count grows. Ideally, the speedup would increase linearly with the number of processors, but due to non-parallelizable parts of the code, the actual speedup is sublinear.
2. How can you improve your program further more

   Within each MPI process, you can utilize multi-threading (e.g., OpenMP) to further parallelize the summation across CPU cores in the node. This hybrid parallelism approach can increase computation efficiency, especially on systems with multiple cores per node.
3. How does the communication and cache affect the performance of your program?

   Reduction Operations: The program relies on an MPI_Reduce operation to aggregate the partial sums calculated by each processor. As the number of processors increases, the communication cost also increases due to the need to gather results from multiple nodes. This overhead grows with O(logP), but for large P, it can still become a bottleneck.

Network Latency and Bandwidth: Each communication operation introduces latency, especially in distributed-memory systems (e.g., clusters with multiple nodes). If the interconnect bandwidth is limited, large data transfers (even for double precision floating-point values) can introduce delays, further reducing the parallel efficiency.

Synchronization Delays: During reduction, all processes need to wait for the slowest one to complete its calculation and communicate its result. This can lead to idle time on faster processes, where they're waiting for data from slower ones, impacting the overall speedup.

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

   In this program, if the Karp-Flatt metric starts low and gradually increases with more processors, it would indicate that as processor count grows, the impact of communication and non-parallelizable components (such as the final reduction) becomes more prominent and reveals the increasing impact of non-parallelizable work and communication overhead as processors are added. In this program, a rising Karp-Flatt value with more processors would indicate that further parallelization benefits are limited by these factors. And iso-efficiency Metric suggests that to maintain efficiency, the problem size n should grow at least as fast as logP. If not, the program will experience diminishing returns with added processors due to communication overhead outpacing computational workload.

)

Appendix(optional):

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