Parallel Programming Exercise 8 - 10

Author	張凱捷 (r08922054@ntu.edu.tw)
Student ID	R08922054
Department	資訊工程所

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

1 Problem and Proposed Approach

Problem

Write a program that implements matrix-vector multiplication based on a checkerboard block decomposition of the matrix. The program should read the matrix and the vector from an input hie and print the answer to standard output. The names of the hies containing the matrix and the vector should be specified as command-line arguments.

Approach

Used the checkerboard algorithm mentioned in chapter 8 with some minor modifications on the algorithm. I let all the processes $fseek$ to the corresponding position of the file and directly read the required part of matrix. Base on the experiment result, this approach can significantly improve the run time of reading the matrix.

The author did not calculate the run time while reading file because the I/O time is rather difficult to analyze.

2 Theoretical Analysis Model

For the sequential version, the run time is $\chi n * n$, where χ is the run time for checking a factor. We can equally divide the matrix to all the process, so the run time of the parallel version is simply $\chi(n/\lceil\sqrt{p}\rceil)^2$. Finally, the communication takes $\log\lceil\sqrt{p}\rceil \cdot (\lambda + 8\lceil n\sqrt{p}\rceil/\beta)$ for each round to reduce the prime property.

Hence, the expected execution time of the parallel algorithm is approximately:

$$\chi(n/\lceil\sqrt{p}\rceil)^2 + \log\lceil\sqrt{p}\rceil \cdot (\lambda + 8\lceil n/\sqrt{p}\rceil/\beta) \tag{1}$$

The time complexity:

$$O(n^2/p + n\log p/\sqrt{p}) \tag{2}$$

We can further formalize the speedup related to processors by the isoefficiency relation:

$$\begin{cases} \sigma(n) = 0 \\ \phi(n) = \chi n^2 \\ \kappa(n) = \log\lceil\sqrt{p}\rceil \cdot (\lambda + 8\lceil n\sqrt{p}\rceil/\beta) \\ \psi(n,p) \le \frac{\sigma(n)+\phi(n)}{\sigma(n)+\phi(n)/p+\kappa(n,p)} \end{cases} \tag{3}$$

$$n^2 \ge Cn\log p/\sqrt{p} \tag{4}$$

$$n \ge C\log p/\sqrt{p} \tag{5}$$

This method is highly scalable.

3 Performance Benchmark

Use Equation 1 and the real execution time of one processor to get the value of χ. λ is approximately equal to 10^{-5} second. β is approximately equal to $4 * 10^7$. Then, use the same equation again to get all the estimation execution time.

I wrote a script to automatically submit the jobs with $p = \{1, 4, 9, 16\}$. For each job, I ran the main program for 20 times, and eliminate the smallest / largest 5 record. The value in the table is the average of the rest ten records.

Table 1: The execution time (second), ($n = 3000$)

Processors	**1**	**4**	**9**	**16**
Real execution time	0.02673	0.00723	0.00569	0.00216
Estimate execution time	0.02673	0.00699	0.00339	0.00199
Speedup	1.00000	3.69538	4.69492	12.39647
Karp-flatt metrics		0.02748	0.11462	0.01938

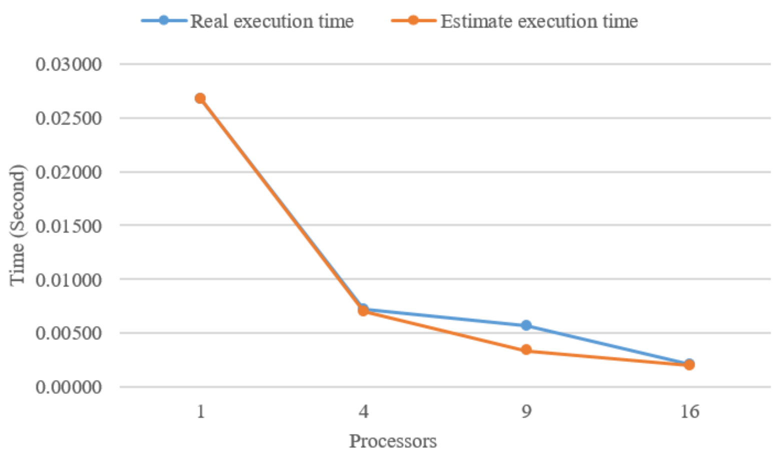

Figure 1: The performance diagram.

4 Conclusion and Discussion

4.1 What is the speedup respect to the number of processors used?

The speedup is less than p since there exist some overhead while doing parallel programming. The run time of $p = 9$ is greater than the expectation is because of the implementation issue of MPI in my opinion. The run time is abnormal for those p which is not the power of 2.

4.2 How can you improve your program furthermore?

Maybe there are some other efficient ways to deliver the matrix and vector.

4.3 How do the communication and cache affect the performance of your program?

It affected a lot in my experiment. If I only consider the matrix multiplication part for the run time, it is nearly the same as the expectation.

4.4 How do the Karp-Flatt metrics and Iso-efficiency metrics reveal?

The Karp-Flatt metric value is quite large when $p = 9$ because there exist some significant overhead as I mentioned in first problem.