Parallel Programming Exercise 10 - 4

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

A cylindrical hole with diameter d is drilled completely through a cube with edge length s so that the center of the cylindrical hole intersects two opposite comers of the cube. Write a program to determine, with five digits of precision, the volume of the portion of the cube that remains when $s = 2$ and $d = 0.3$.

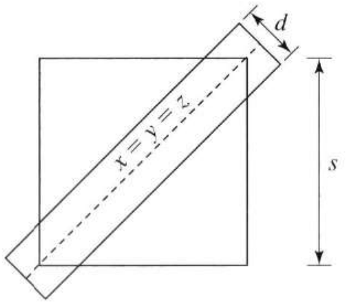

Figure 1: Looking from the side of the cube.

Approach

Use Monte Carlo approach to approximate the answer. Random points in the cube and get the ratio of the cylinder volume over the cube volume. We can use Equation 1 to get the distance from the middle line.

$$\sin(\cos^{-1}(\frac{x_1 + y_1 + z_1}{\sqrt{3}\sqrt{x_1^2 + y_1^2 + z_1^2}}))\sqrt{x_1^2 + y_1^2 + z_1^2} \tag{1}$$

I ran 10^{12} random samples on 國網中心 with 320 processes, and get the volume is approximately 7.7717.

2 Theoretical Analysis Model

For the sequential version, the run time is χn, where χ is the run time for a sample. We can let all the process generate the same number of samples, so the run time of the parallel version is simply $\chi n/p$. Finally, the communication takes $\lambda \log p$ for each round to reduce the average volume.

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

$$\chi n/p + \lambda \log p \tag{2}$$

The time complexity:

$$O(n/p + \lambda \log p) \tag{3}$$

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

$$\begin{cases} \sigma(n) = 0 \\ \phi(n) = \chi n \\ \kappa(n) = \lambda \log p \\ \psi(n,p) \leq \frac{\sigma(n)+\phi(n)}{\sigma(n)+\phi(n)/p+\kappa(n,p)} \end{cases} \tag{4}$$

$$n \geq \log p \tag{5}$$

This method is highly scalable.

3 Performance Benchmark

Use Equation 2 and the real execution time of one processor to get the value of χ. λ is approximately equal to 10^{-6} second. Then, use the same equation again to get all the estimation execution time.

I wrote a script to automatically submit the jobs with $p = [1, 8]$. 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), ($n = 10000000$)

Processors	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Real execution time	0.61545	0.30794	0.20610	0.15486	0.12445	0.10393	0.09045	0.08015
Estimate execution time	0.61545	0.30772	0.20515	0.15386	0.12309	0.10257	0.08792	0.07693
Speedup	1.00000	1.99857	2.98616	3.97414	4.94529	5.92192	6.80438	7.67856
Karp-flatt metrics		0.00071	0.00232	0.00217	0.00277	0.00264	0.00479	0.00598

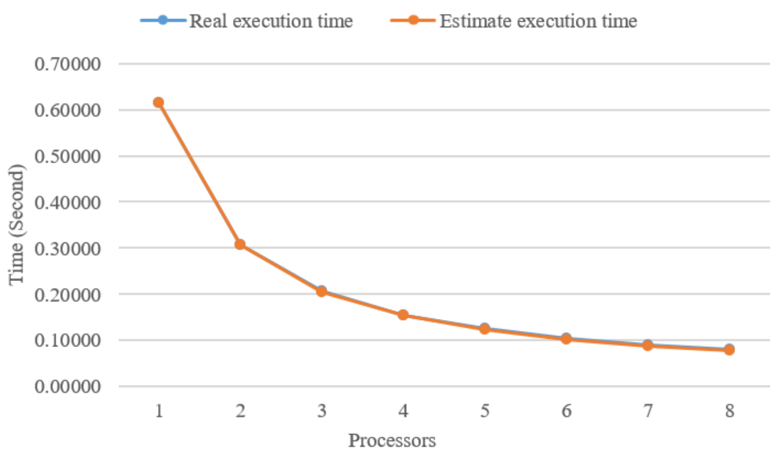

Figure 2: The performance diagram.

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. However, the overhead is small in this problem.

4.2 How can you improve your program furthermore?

Maybe we can try some different random generators to compare the accuracy and the run time.

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

The only communication is to reduce the prime property for each number, so it do not effect out run time too much.

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

The Karp-Flatt metrics values is small because the overhead is pretty small.