**Parallel Programming Exercise 10 – 4**

|  |  |
| --- | --- |
| **Author:** | 蕭彧 ([r11922138@ntu.edu.tw](mailto:r11922138@ntu.edu.tw)) |
| **Student ID** | R11922138 |
| **Department** | Computer Science |

# Problem and Proposed Approach

This programming exercise is to find the volume of a portion of a cube. There’s a hint that offer a formula to calculate the distance between the point (x1 , y1 , z1) and the line x = y = z. I put the center of the cube on the origin, so that I can apply this formula to determine whether a point is inside the cylindrical hole or not. I use Monte Carlo simulation to randomly sample points inside the cube. The volume can then be calculated by the following formula: points not in cylindrical hole \* cube’s volume / total sample points. The volume calculate by Monte Carlo is around 7.77187.

# Theoretical Analysis Model

Assume we sample n points, using p processors

χ is time for sample a point

λ is the message latency

β is network bandwidth

Sequential execution time: n\*χ

Reduction time: (*λ* + 4/β) ⎡log *p*⎤

Expected execution time: χ ⎡*n*/*p*⎤ + (*λ* + 4/β) ⎡log *p*⎤

# Performance Benchmark

The message latency is λ = 0.001521 sec, and the network bandwidth is β = 1658 MB/sec

The total sampling points are fixed at 10,000,000.

Table 1. The execution time

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Processors | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| Real execution time | 7.73257 | 3.86961 | 1.9391 | 0.97445 | 0.49181 | 0.2494 | 0.13519 | 0.07214 |
| Estimate execution time | 7.7325 | 3.8657 | 1.9384 | 0.97398 | 0.49284 | 0.2487 | 0.13481 | 0.07168 |
| Speedup | 1 | 2 | 3.99 | 7.94 | 15.72 | 31 | 57.2 | 107.19 |
| Karp-flatt metrics | - | 0 | 0.0008 | 0.0011 | 0.0012 | 0.001 | 0.0019 | 0.0015 |

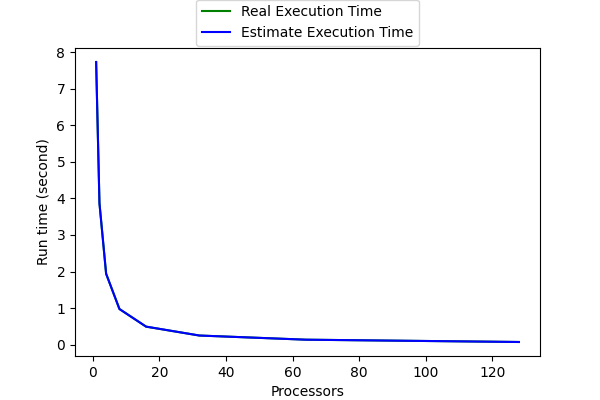


Figure 1. The performance of diagram

# Conclusion and Discussion

Table 1 shows that this program is almost perfectly parallelized. That’s because the program doesn’t introduce too much parallel overhead. The only parallel overhead is one reduction. There’s even no sequential part in this program. Every processor can sample their points independently. In conclusion. Monte Carlo method shows a great parallelism, which is then a perfect algorithm for multi processors/threads coding.