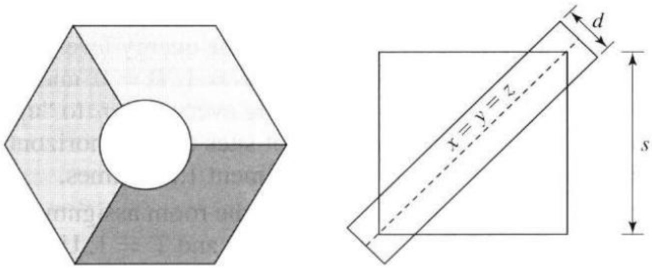
Parallel Programming Exercise 10-4

Author:	李子筠(b06901145@ntu.edu.tw)
Student ID	b06901145
Department	Electrical 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.)



(a) (b)

題目是在 s=2, d=0.3 的條件下，求出立方體扣掉與圓柱重疊的體積，精度是五個位數。

我用 Monte Carlo method， x,y,z 在 [0,2] 的範圍內 sample 多個點，看跟 x = y = z 的距離是否超過圓柱的半徑，統計落在圓柱外的點數目來求體積的近似值。實做上每個 process 平均分到 n/p 個 sample，最後用 reduce 加總全部的個數。

2 Theoretical Analysis Model

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

N: number of sample

Sequential execution time: $O(\chi N)$

Parallel computation time: $O(\chi N/p)$

Parallel communication time: $O(\lambda \log p)$ (reduction)

Parallel execution time: $O(\chi N/p + \lambda \log p) = O(N/p + \log p)$

Isoefficiency metric: $n \geq Cp \log p$

$M(n) = \log n$ (只需要 log n bit 的整數來存 count，所以需要 log n 的記憶體)

Scalability function: $M(f(p))/p = \log (C p \log p)/p = O(\log (p \log p)/p)$

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	8.177	4.261	2.862	2.188	1.752	1.461	1.253	1.097
Estimate execution time	8.177	4.089	2.726	2.044	1.635	1.363	1.168	1.022
Speedup	x	1.919	2.857	3.737	4.667	5.595	6.525	7.454
Karp-flatt metrics	x	0.04	0.02	0.02	0.01	0.01	0.01	0.01

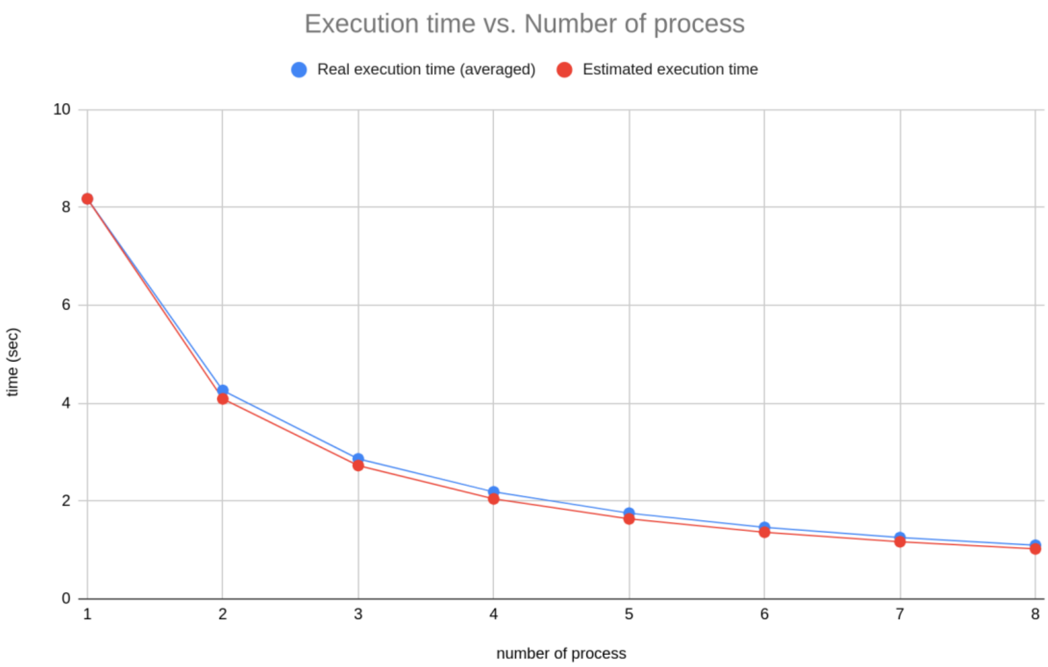


Figure 1. The performance diagram

4 Conclusion and Discussion

(Discuss the following issues of your program

1. What is the speedup respect to the number of processors used?
2. How can you improve your program further more
3. How does the communication and cache affect the performance of your program?
4. How does the Karp-Flatt metrics and Iso-efficiency metrics reveal?

)

1. speedup 大致隨著 processor 數量增加
2. 可以改變距離的算法，課本上的算法是

   $$\sin\left[\cos^{-1}\left(\frac{x_1+y_1+z_1}{\sqrt{3}\sqrt{x_1^2+y_1^2+z_1^2}}\right)\right]\sqrt{x_1^2+y_1^2+z_1^2}$$

   但是如果用外積算會得到距離是 sqrt([(y-z)^2 + (z-x)^2 + (x-y)^2]/3)
   化簡之後可以得到 (y-z)^2 + (z-x)^2 + (x-y)^2 > 3*radius^2 是在圓柱外的條件，這個作法不需要 sin, acos, sqrt，所以會比原本的快，實測大概快了 2.4 倍。

   另外一個加速的方向是直接用體積分算，雖然會比較難寫但是得到的答案應該比較精確。
3. communication 只有 reduction，所以影響不大。
4. Karp-Flatt metrics 大致上維持定值，代表 sequential part 佔了很大一部份。

   Scalability 算出來是 O(log (p log p)/p)，代表有很好的 scalability。

Appendix(optional):

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

程式執行結果：

```
[u1167044@clogin1 10-4]$ mpicc -o volume ./Calculate\ Volume.c -std=c99 -O3
[u1167044@clogin1 10-4]$ mpiexec -n 20 ./volume 1000000000
The volume is 7.771687 (Number of sample = 1000000000), time elapsed = 1.788517 (Number of process = 20)
[u1167044@clogin1 10-4]$ mpiexec -n 20 ./volume 10000000000
The volume is 7.771659 (Number of sample = 10000000000), time elapsed = 17.893537 (Number of process = 20)
```