CX 4220/CSE 6220 High Performance Computing
PA 1 Report
Submitted January 29, 2025

Group Member: Xuzheng Tian & Honglin Liu

1. Analysis of processors effect for $n = 10^9$:

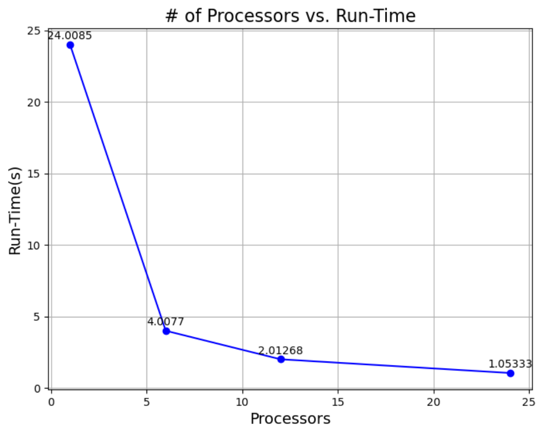

Figure 1: Relationship between number of processors and runtimes.

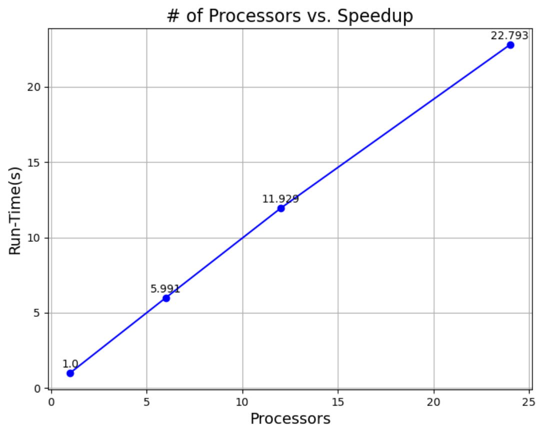

Figure 2: Relationship between number of processors and speedups.

According to the images, our analysis reveals that the runtimes are inverse proportional to the number of processors, while a linear relationship between speedup and processors is observed.

2. Theoretical analysis on code:
   Codes are already submitted to PA1-Code section.
   For our code, here are several subsections:

   (a) Create p processors, assume a tree based initialize which requires $\Theta(\log p)$ complexity
   (b) Total task allocation: divide tasks evenly to subgroups (contain $\frac{n}{p}$ or $\frac{n}{p}+1$ points) for each processor, takes $\Theta(1)$ complexity.
   (c) Generate random position for each points, takes $\Theta(\frac{n}{p})$ complexity.
   (d) Count the number of points inside the circle locally, takes $\Theta(\frac{n}{p})$.
   (e) For parallel run, sum result over all processors, with complexity $\Theta(\log p)$

   Then for serial code $p = 1$:
   $$T(n,1) = \Theta(n)$$
   For parallel code (with small p):
   $$T(n,p) = \Theta(\log p) + \Theta(1) + \Theta(\frac{n}{p}) + \Theta(\frac{n}{p}) + \Theta(\log p) = \Theta(\frac{n}{p})$$
   Therefore speedup is:
   $$S(n,p) = \frac{T(n,1)}{T(n,p)} = \Theta(p)$$

3. Other questions:

   (a) Why is using MPI Reduce better than using MPI Gather in such cases?
   The MPI-Reduce function is a tree-based algorithm with a time complexity of $\Theta(\log p)$, while the MPI-Gather algorithm takes $\Theta(p)$
   (b) Why is it necessary to have a random generator function for generating points? Why is necessary to have different seeds for the random generator in every processor?
   For this monte carlo method, we need to ensure the random sampling, so that we can avoid bias.
   Since we are making a parallel algorithm, we need to achieve randomize among the total sample space rather than only a local data space for one processor. The same random seed assign to two different processor will generate exactly the same results, which introduce bias and a waste of computing resource.
   (c) Explain the functioning of every MPI instruction that you used in your program (pi.h and pi.cpp), also including the MPI setup instructions (eg. MPI Init, etc).
      i. MPI_Init, to initialize the MPI environment.
      ii. MPI_Comm_size, to get the total number of processors.
      iii. MPI_Comm_rank, to get the rank for the current processor.
      iv. MPI_About, to terminate all processors.

v. MPI_Finalize, after all the processors finish their work, shut down the MPI environment.

vi. MPI_Wtime, record current time, normally used to count the time cost for some procedure.

vii. MPI_Reduce, sum the results obtained from all processors in a tree-based manner.