Universidad Politécnica de Yucatán

Correa Castro Cesar Orlando

Data Engineer 7B

High-Performance Computing

Didier Gamboa

4/19/2024

E3 Parallel Programming with Python

Introduction

The number π is a fundamental mathematical constant known to be the ratio between the circumference of a circle and its diameter. Although π is an irrational number and its decimal value is infinite, there are various techniques to calculate approximations of π with high precision.

One of these techniques is numerical integration, specifically through the method of Riemann sums. By integrating the function sqrt (1- x^2) over the interval 0 to 1, we can calculate the area of a quarter unit circle. If we multiply this number by 4, we will have the unit value.

In this task, different ways to solve this problem using Python are done. Initially, we will implement a sequential algorithm that does not use parallelization techniques. Later, we will incorporate parallelization through two different methods: using the multiprocessing library to run calculations in parallel in multiple processes and using mpi4py for distributed parallel computing.

Homework Highlights

Mpi4py:

- Approximate pi: 3.1415946524138114
- Execution Time: 0.276759 seconds

The use of mpi4py allowed for efficient distributed execution across multiple nodes, resulting in the shortest execution time among the tested methods.

Multiprocessing:

- Approximate pi: 3.141594652413769
- Execution Time: 0.438602 seconds

Although the execution time was longer than that observed with mpi4py, multiprocessing proved to be a robust tool for parallelization on a single system, facilitating the use of multiple processors to improve calculation performance.

Without Parallelization (sequential):

- Approximate pi: 3.1415946524138207
- Execution Time: 0.256106 seconds

This method had the lowest execution time of the three.

The sequential approach, in this case, provided an excellent result, possibly due to the simplicity and lower overhead of process management or communication between nodes.

Here is going to be attached a link to the colab code, this because the multiprocessing couldn't be done in Visual Studio, due to the running time.

https://colab.research.google.com/drive/1hAV65086_r5KLzTUiPsUljHcPbH-F8aE?usp=sharing

Also, I am going to attach the link were you can find the codes implemented, where the sequential and mpi4py was able to be run in Visual Studio.

https://github.com/Cesar0ngas/E3-Parallel-Programming-with-Python

References

1. *Find the area of hysteresis loops*. (2020, October 28). https://community.ptc.com/t5/Mathcad/Find-the-area-of-hysteresis-loops/td-p/695220
2. Wikipedia contributors. (2024, April 19). *Approximations of π*. Wikipedia. https://en.wikipedia.org/wiki/Approximations_of_%CF%80
3. Python, R. (2023, October 21). *Python 3.12: Cool new features for you to try*. https://realpython.com/python312-new-features/