UNIVERSITE JOSEPH KI-ZERBO (UJKZ)
****

VICE-PRESIDENCE CHARGEE DE LA RECHERCHE ET DE LA COOPERATION INTERNATIONALE (VP/RCI)

****

ECOLE DOCTORALE INFORMATIQUE ET CHANGEMENTS CLIMATIQUES (ED/ICC)

MASTER RESEARCH PROGRAM- INFORMATICS FOR CLIMATE CHANGE

High Performance Computing System

Lecturers:

Dr Tégawendé BISSIYANDE

Abdoul Kader KABORE

By Group 1:

Folashade **MOLADE**

Abderahim **TOGUYENI**

Abdramane **DOUMBIA**

Crespin **MOUTO**

Contents

I. Introduction: 3
II. Objective: 3
III. Methodology: 3
Part I : π Estimation 4
1. Main functionalities: 4
2. Libraries used: 5
Part 2: Parallel programming on python 6
1. Main functionalities: 6
2. Libraries used 6
a) Numpy library 6
b) Pandas library 6
c) Parallel and delayed libraries 6
d) Multiprocessing library 7
e) Matplotlib.pyplot 7
f) Seaborn 7
3. Lessons learned 7
4. Difficulties: 8
IV. Conclusion 8

Report on of Evaluation on High Performance Computing System

I. Introduction:

High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.
Parallel programming, in very simple terms, is the use of multiple resources, in this case, processors, to solve a problem. This type of programming takes a problem, breaks it down into a series of smaller steps, delivers instructions, and processors execute the solutions at the same time. It is also a form of programming that offers the same results as concurrent programming but, in less time, and with more efficiency. Many computers, such as laptops and personal desktops, use this programming in their hardware to ensure that tasks are quickly completed in the background
At the end of the High-Performance Computing course, it has been given to us the project on parallel programming in C++ and python.

II. Objectives:

- To understand pi estimation better by generating random points within a square and see how many points falls within an enclosed circle using implementation in C++
- To learn to overcome challenges in pi-estimation in C++ and parallel programming
- Using python in parallel programming implementation
- To strengthen and enhance individual knowledge on each programming style

III. Methodology:

- The editor used for this project are : sublime text and jupyter-notebook
- The compiler used to compile the π estimator is : GNU make
- The library used to the parallel programming of π estimator in C++ is pthread
- The datasets used for parallel programing in python is the temperature records around the world since 1750.
- The library used to do the parallel programming in python is multiprocessing.
- The dataset has been divided in 8 sub datasets, each thread will compute a sub data and them combine the output of each thread at the end.

Part I : π Estimation

One method to estimate the value of π (3.141592...) is by using a Monte Carlo method. This method consists of drawing on a canvas a square with an inner circle. We then generate a large number of random points within the square and count how many falls in the enclosed circle.

1. Main functionalities:

The program " Main version 1.0" is a software designed to illustrate the parallel programming notions in C++. It uses the Monte Carlo method to estimate the value of π in parallel programming.

The program is written in two separates files : main.cpp and helpers.cpp.

The main.cpp file contains the core of the program and the helpers contains the functions which will be used in main program.

```
                                    ----------------------------------------
                                    ----------------------------------------

                                    |  Welcome to the program : π ESTIMATION    |

                                              by Group1
                                    ----------------------------------------
                                    ----------------------------------------
  Thread id : 6 Circle account : 783723721       Starting point : 750000000      Ending point : 875000000
 Thread id : 4 Circle account : 784540816        Starting point : 500000000      Ending point : 625000000
 Thread id : 5 Circle account : 784574278        Starting point : 625000000      Ending point : 750000000
 Thread id : 7 Circle account : 784582757        Starting point : 875000000      Ending point : 1000000000
 Thread id : 2 Circle account : 784887220        Starting point : 250000000      Ending point : 375000000
 Thread id : 3 Circle account : 784935439        Starting point : 375000000      Ending point : 500000000
 Thread id : 0 Circle account : 785365645        Starting point : 0      Ending point : 125000000
 Thread id : 1 Circle account : 785406025        Starting point : 125000000      Ending point : 250000000
Num  of points : 1000000000      PI = 3.14162410000000003051923158636782318353 7
In circle : 785406025
Maximum thread : 8

real    8m32.609s
user    13m36.466s
sys     43m25.484s
```

2. Libraries used:

The libraries used in this program are :

Name	Description
iostream	provides basic input and output services for C++ programs
time	The C date and time functions are a group of functions in the standard library of the C programming language implementing date and time manipulation operations.
pthread	Used to do parallel programming
stdio	Input and Output operations can also be performed in **C++** using the C Standard Input and Output Library (cstdio, known as **stdio**. **h** in the C language).
omp	Used to gather informatiom of the maximum of the thread the computer can create

Part 2: Parallel programming on python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

For the illustration of the parallel programming in python, the date used is the temperature records around the world since 1750.

1. **Main functionalities:**

The program calculates the average temperature for each country. We also compute the maximum temperature of each country. All of these operations are done in parallel programming using appropriate libraries.

2. **Libraries used**

a) NumPy library

In this project, it is imported as np. We have used it in our project because being fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today. it offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more. In other hand, NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries. Furthermore, NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

b) Pandas library

In the project, it is imported as pd. pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. It is designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python.

c) Parallel and delayed libraries

These libraries are sub libraries under Joblib library. Joblib is a set of tools to provide lightweight pipelining in Python. In particular: transparent disk-caching of functions and lazy re-evaluation

(memorize pattern). The vision is to provide tools to easily achieve better performance and reproducibility when working with long running jobs.

d) Multiprocessing library

Multiprocessing is a package that supports spawning processes using an API similar to the threading module. The multiprocessing package offers both local and remote concurrency, effectively side-stepping the Global Interpreter Lock by using subprocesses instead of threads. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine. It runs on both Unix and Windows

e) Matplotlib.pyplot

In this project, this library is imported as plt. Matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with label. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the *axes* part of a figure and not the strict mathematical term for more than one axis).

f) Seaborn

In this project, this library is imported as sns . Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3. Lessons learned

During this project, we have put in application, all the concepts we learned during the course. We have used the knowledge we got in parallels programming in python and in C++. So now, we can manipulate easily the parallel programing tools to do many other projects in the future on C++ and python. Specifically, we learned

- How to divide a specific work into sub work in order to do parallel programming
- Learn how to manage or modify a variable between many processors in python

4. Difficulties:

Although, Python can be said to have a relatively easy to understand programming style but we still encountered some challenges while trying to implement parallel programming I were; managing data distributions, managing inter- processor communication, balancing the computational load when some processors work while others wait due to insufficient **parallelism** or unequal size tasks. In sum these are the main difficulties we faced :

- Manipulate a variable in the parallel programming: It was difficult to fix out the issue. But by using the method **multiprocessing.Manager() ,** it has been possible.
- Choose between the method : process and map in multiprocessing library.
- Dividing the database into different database in order to allow all thread works without repetition and gather all those sub data to compute

IV. Conclusion

At the end of this project, we have been able to learn and practicalize the implementation of pi estimation using C++ and the implementation of parallel programming using python through the jupyter environment. This project has also broadened our understanding on managing internal processors, managing data distributions and balancing the computational loads.