Universidad de Investigación de Tecnología Experimental Yachay Tech

School of Mathematics and Computer sciences

Program : Undergraduate	Semester : VII
Subject : HPC	Professor : Saravana Prakash T
Date : 23/05/2019	Time:
Semester: Jan-May-2019	
Student's Surname, Name : HERNANDEZ MARCOS	
Student Reg.No: 1804727194	

Duration: 40 minutes *Take-home/Quiz/exercises* Maximum Marks: 100

Neatly show all of your work. If you need to use extra paper, please indicate so and attach to this exam.

Mode of submission : Upload your answers in D2L or hard-copy at my desk
Turn down the quiz /submit : Thursday (28/02/2019), at 8:00 PM.

SPRING SIMULATION

1. Introducction

   This code is a representation of predicting the motion of a spring. To demonstrate it there were implemented two methods Runge-Kutta4 ODE and Runge-Kutta2 to solve this problem. The equations that describe the change in the location and velocity as a function of time for the spring are shown below.

$$m\frac{dv_x}{dt} = -\mu v_x - kx$$

$$\frac{dx}{dt} = v_x$$

   In order to compute the motion of the spring, it is necessary to create a structure that represents the spring motion ODEs.Therfore the structure can be reused in the whole code. The first thing the SpringODE structure does is to declare fields that represent spring-specific data such as the spring constant and damping coefficient. Since the motion of the spring will be computed as a function of time, the independent variable for this problem is time.
   The a iteration is implemented to increment the time. Here we call either of each method RK2 or RK4 then it return the correct the structure with the velocity and position.

2. Implementation using MPI and OpenMP

Question 2 continues on the next page...

The problem was implemented using MPI and openMP to see the performance. The figures below represent the number of iteration, which represent the increment of time, vs the execution time.

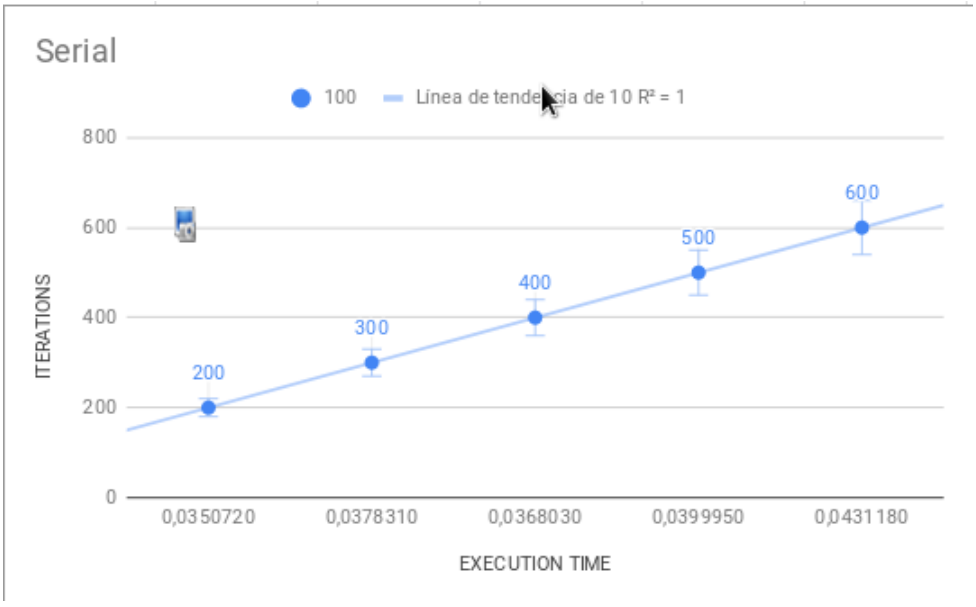

Figura 1: Execution time serial

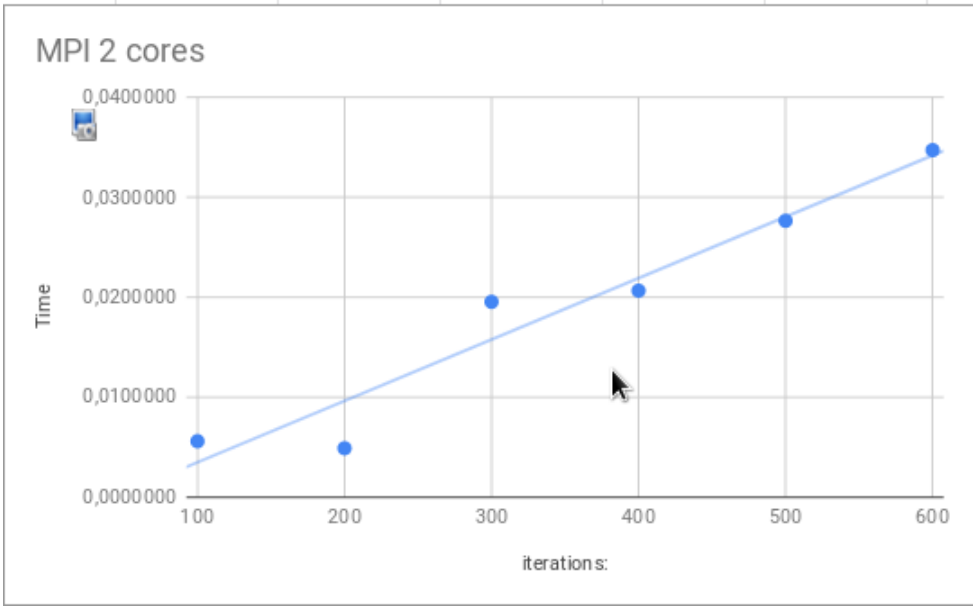

Figura 2: Execution time MPI using 2 cores

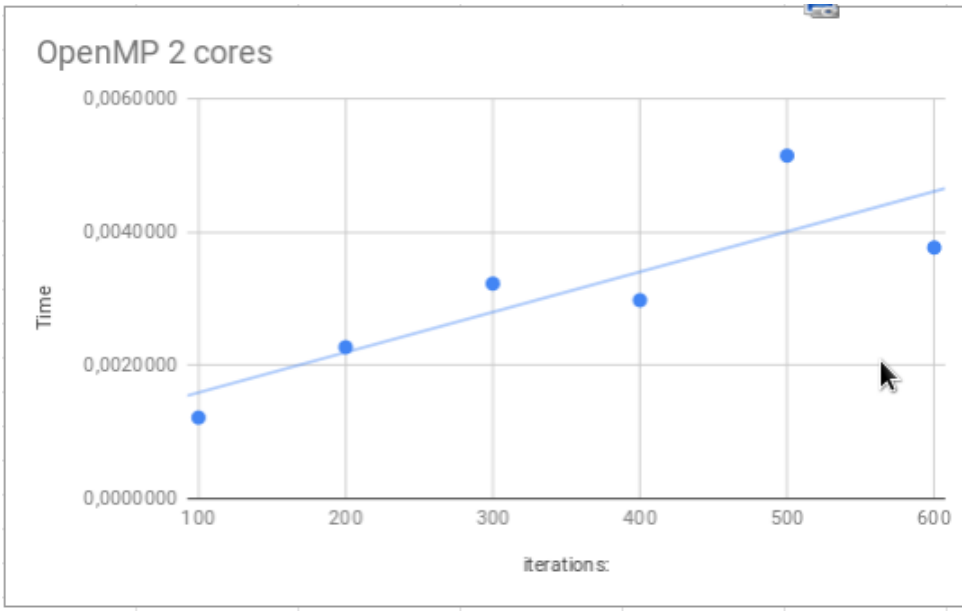

Figura 3: Execution time OpenMP using 2 cores

Figura 4: Execution time MPI using 4 cores

Figura 5: Execution time OpenMP using 4 cores

3. CONCLUSION

As we can see the execution time is reduced depending on the number of thread or cores. The performance is much better than a serial code.To program in parallel, is to change the way of thinking a little. The same program runs on several cores at the same time, but each maintains a different domain of values for the same variables. This feature and many others commented throughout the memory, make programming in parallel is somewhat complicated. However, with relatively little learning time, it is possible to carry out programs that really have a great application in the scientific field.

Signature of the student