Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Guadalajara

Ingeniería en Robótica y Sistemas Digitales

Clave: TC2036.501

Diseño de sistemas embebidos avanzados

Cálculo del número de PI en paralelo usando threads

Alumno:

Jorge Carrillo Castro - A01634630

Fecha y lugar entrega:
8 de octubre de 2022
Zapopan Jalisco

Cálculo del número de PI en paralelo usando threads

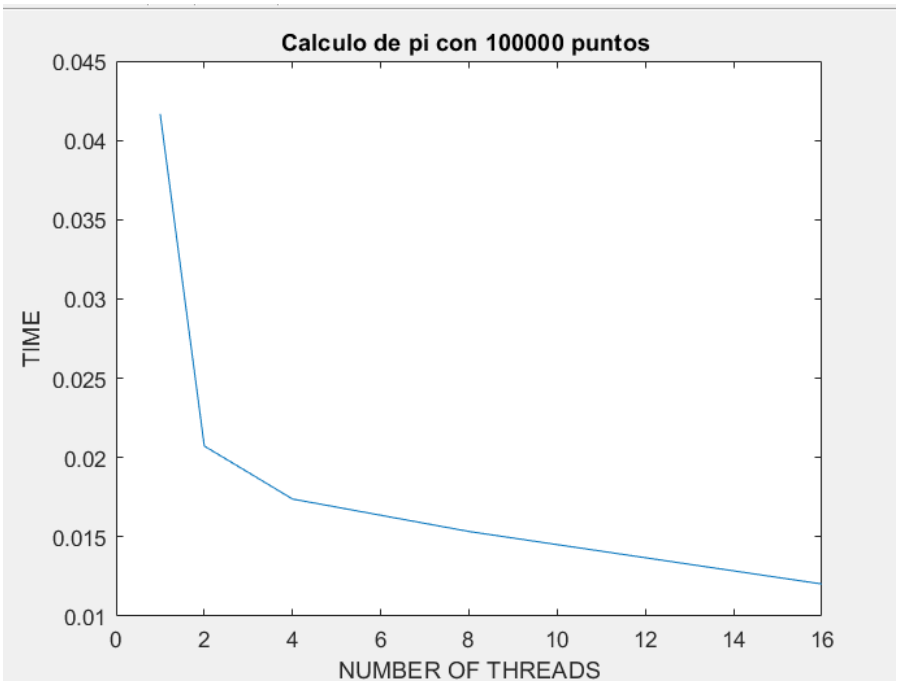
En esta actividad se nos dio la tarea de calcular el valor de pi usando threads. En clase aprendimos sobre el método de Monte Carlo para llegar a la solución del problema. La forma en la que funciona este método es creando puntos aleatorios dentro de un cuadrado, después de haber creado el punto se evalúa si está dentro del círculo unitario o si está fuera. Después se calcula el radio de los puntos que están dentro contra el total.

Una vez que implementé la solución en ubuntu, me percaté que los valores y el tiempo que tomaba, no era constante. Esto me dio curiosidad y decidí hacer más pruebas de las solicitadas. Hice el cálculo con 1, 2, 4, 8 y 16 threads pero volví a hacer los cálculos cambiando el número de puntos. Use 1000, 100000 y 1,000,000 para ver como varían los resultados y los grafiqué en matlab.

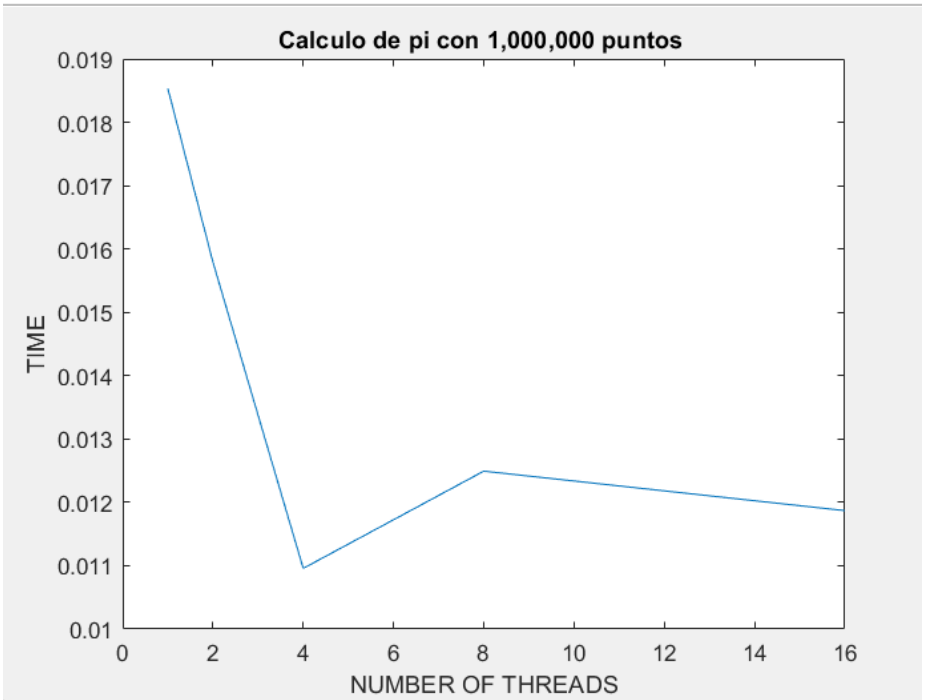
Link a repositorio de Github:
https://github.com/jorgais1234/parallel-programming-ITESM/tree/main/Practice_1/prueba_1.c

Gráficas de comparación

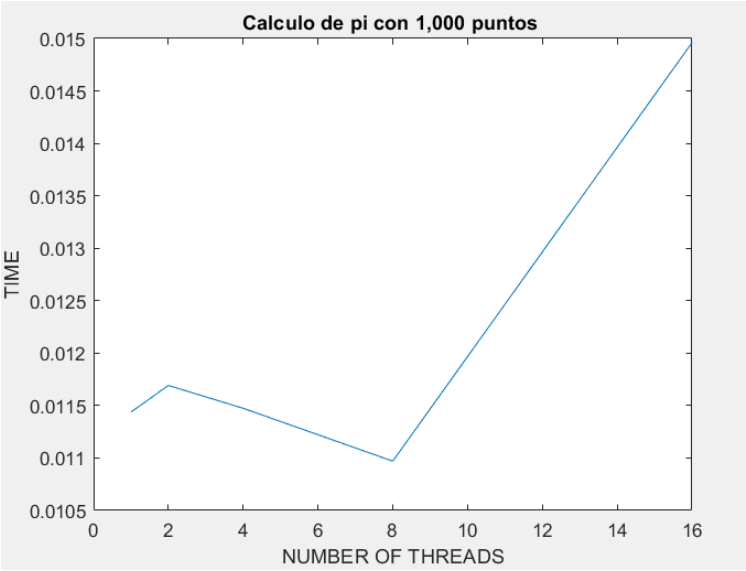
```
threads=[1 2 4 8 16];
time = [0.041666 0.020726 0.017385 0.015338 0.012027];
plot (threads, time)
ylabel("TIME")
xlabel("NUMBER OF THREADS")
title("Calculo de pi con 100000 puntos ")
```



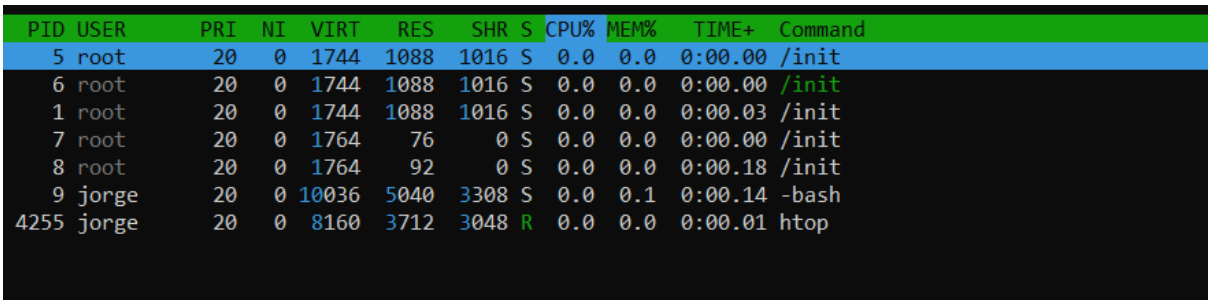
```
threads=[1 2 4 8 16];
time = [0.018535 0.015792 0.010958 0.012493 0.011870];
plot (threads, time)
ylabel("TIME")
xlabel("NUMBER OF THREADS")
title("Calculo de pi con 1,000,000 puntos ")
```

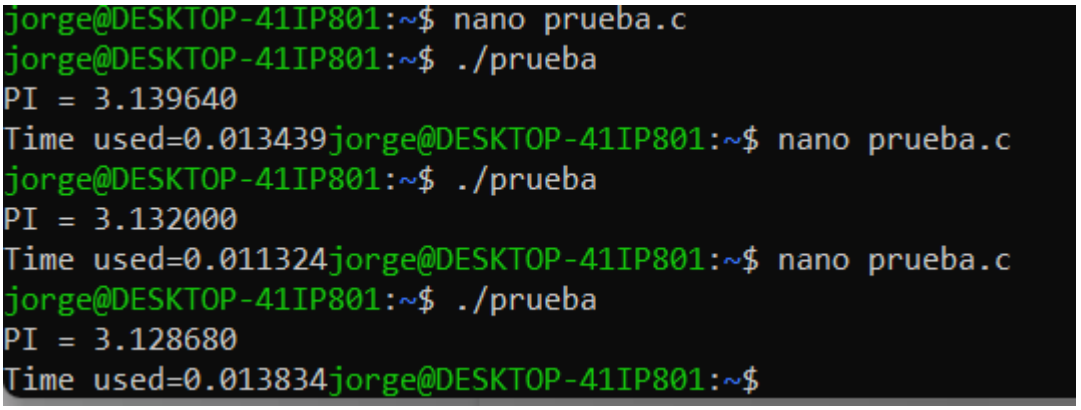



```
threads=[1 2 4 8 16];
time = [0.011438 0.011692 0.011474 0.010970 0.014960];
plot (threads, time)
ylabel("TIME")
xlabel("NUMBER OF THREADS")
title("Calculo de pi con 1,000 puntos ")
```



The graph of the CPU utilization using HTOP





Para concluir con esta actividad analice todas las gráficas que obtuve. La primera con 100000 puntos me hizo mucho sentido ya que entre más threads agregaba, menos tiempo tomaba. Esto es debido a que entre más threads existan, cada una tiene que generar menos puntos y es más eficiente. Algo que llamo mi atención es que esto no es algo constante, ya que la misma lógica debería aplicar para el resto de casos (cuando cambio el número de puntos) pero no es así. Como se puede ver cuando hice la prueba con 1,000,000 de puntos tomó más tiempo hacer la tarea con 8 threads que con 4 y en mi última prueba con solo 1,000 puntos tardó más cuando había 16 threads. Decidí hacer una prueba definitiva con 16 threads (constante en las 3) y cambiar la cantidad de puntos. Cuando el número de puntos era 1000, el resultado de pi fue más exacto pero el tiempo de ejecución fue mayor. Cuando eran 100000 puntos, el tiempo fue menor y el resultado no está tan alejado del esperado. Cuando la cantidad de puntos es 1,000,000 el resultado fue menos preciso y el tiempo fue el mayor de todos. Esto claramente puede variar porque la generación de los puntos es aleatoria entonces habrá casos en los que el número de puntos dentro del círculo sea mayor que el de afuera. Debido a eso los resultados no son constantes.