Universidad de Granada

Doble Grado en Ingeniería Informática y Matemáticas

Sistemas Concurrentes y Distribuidos

Seminario 1

Autor:
Jesús Muñoz Velasco

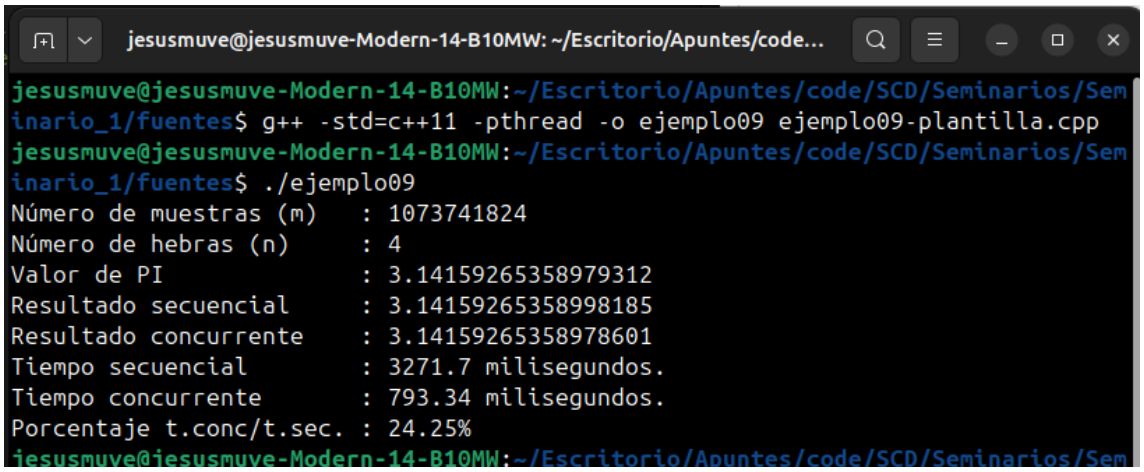
Curso 2024-2025

1. Seminario 1. Actividad propuesta.

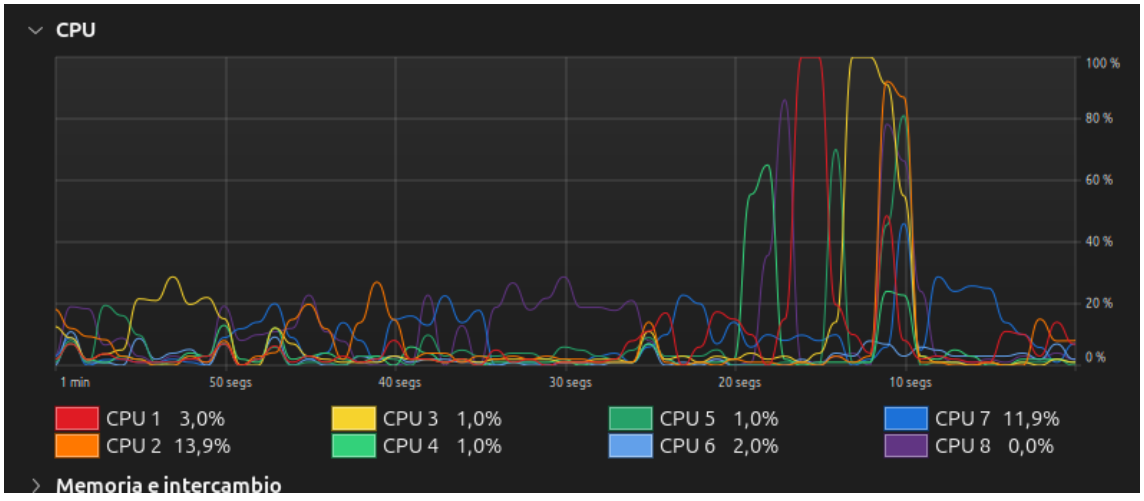
Como se puede observar en el archivo adjuntado se ha optado por la ditribución de trabajo a las hebras de forma cíclica (ya que empíricamente ofrecía los mejores resultados) su implementación en código ha sido la siguiente:

```
double funcion_hebra( long i )
{
    double suma = 0.0 ;

    // Cíclico
    for(int j=i; j<m; j+=n) {
        suma += f((j+0.5)/m );
    }

    return suma;
}
```

Tras compilar y ejecutar el programa obtenemos la siguiente salida:



Además, monitorizando los recursos del sistema (en concreto la CPU) podemos observar un pico general en 4 hebras al ejecutar el programa:



Además, en el código se han dejado 2 resoluciones más comentadas en el código: la repartición por bloques contiguos (vista en clase) y esta última pero reduciendo el número de divisiones (ya que es la operación más costosa). Sin embargo, por la precisión de mi ordenador no he podido obtener un resultado válido con este último método ya que aproxima la variable $paso = 1/m$ a 0, no avanzando en el bucle y llegando a una condición de error (bucle infinito). Sin embargo, con un ordenador con una precisión mayor en la representación de números de coma flotante, esta debería ser la opción mejor.

Voy a explicar de forma razonada la obtención de este código. En primer lugar, el código para bloques contiguos es el siguiente:

```
for(int j=(i*(m/n)); j<((i+1)*(m/n)); j++){
    suma += f( (j+0.5)/m );
}
```

El código "mejorado" es el siguiente:

```
const long double medio=0.5/m;
const long double paso=1/m;

for(long double j=(i/n); j<((i+1)/n); j+=paso){
    suma += f( j+medio );
}
```

Como se puede ver se ha sustituido en primer lugar j por $\frac{j}{m}$ obteniendo de la asignación inicial

$$\frac{j}{m} = \frac{i * (m/n)}{m} = (i/n)$$

y de la condición.

$$\frac{j}{m} < \frac{((i+1) * (m/n))}{m} = ((i+1)/n)$$

Además en el paso se deberá dividir también por m, siendo este valor constante y pudiendo ser calculado una sola vez (valor de la variable *paso*).

En la ejecución de f se han hecho los cambios pertinentes. Si llamamos a la j actual j' y a la anterior j la dejamos como j (siendo $j' = \frac{j}{m}$), podemos escribir $j = m * j'$, y sustituyendo esto en la línea de la ejecución de f obtenemos

$$f((j+0{,}5)/m) = f((m * j' + 0{,}5)/m) = f(((m * j')/m) + (0{,}5/m)) = f(j' + (0{,}5/m))$$

De esta forma podemos eliminar la división por m en cada iteración definiendo la variable *medio* con el valor $0{,}5/m$ que es constante. Con estos cambios habríamos reducido en cada iteración 2 productos (de la definición del bucle) y 1 división (de la ejecución de f). Sabiendo que m que es el número de iteraciones es muy elevado esto supondría una gran mejora en la eficiencia y en la concurrencia. Esta misma solución se podría aplicar sobre la distribución cíclica pero obtendríamos el mismo problema con la precisión.