Martin Farres - 13495

Arquitectura Distribuida

Trabajo Practico 1 – Paralelismo a nivel de hilos.

Información del Sistema

Para poder tener una imagen y entender el contexto sobre el cual se va a trabajar, se presentan las características de la computadora que ejecutará los siguientes programas:

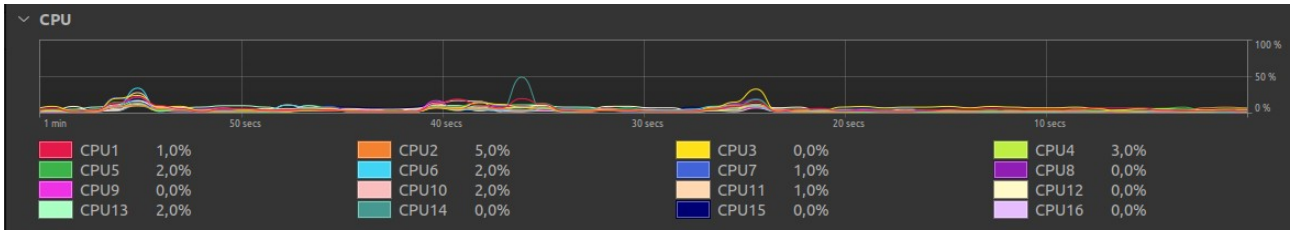
Arquitectura:

- Tipo: x86_64
- Modo(s) de operación de CPU: 32-bit, 64-bit
- Tamaños de dirección: 48 bits físicos, 48 bits virtuales

Detalles del CPU:

- Total de CPUs: 16
- ID del vendedor: AuthenticAMD
- Nombre del modelo: AMD Ryzen 7 5700U con gráficos Radeon
- Hilos por núcleo: 2
- Núcleos por socket: 8
- Socket: 1

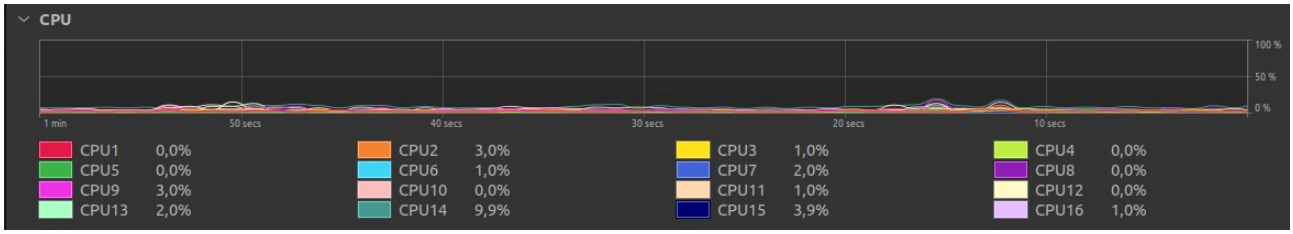
Uso de CPUs como referencia base



Actividad 1 – Serie de Taylor del Logaritmo Natural

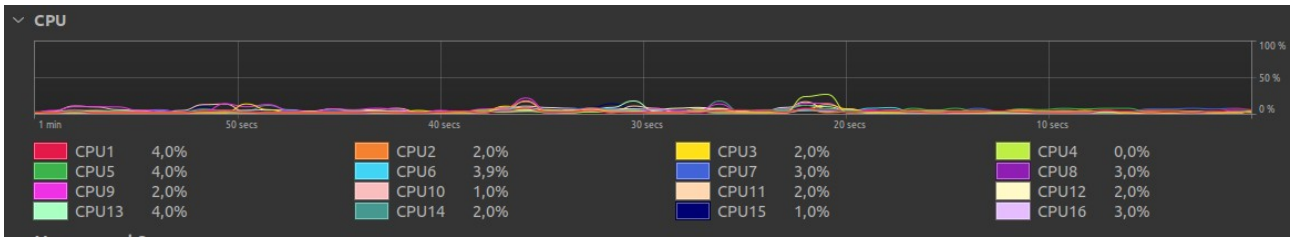
Se realizó un programa que calcula el logaritmo natural mediante la serie de Taylor, empleando diez millones de términos de dicha serie. El objetivo es observar la diferencia entre la ejecución con un hilo y varios hilos, así como brindar una aproximación del speedup obtenido.

Para un hilo se observan los siguientes resultados:




```
Ingrese un número mayor a 1500000: 99999999999
Ingrese el número de hilos: 1
ln(200375420520689664) = 17.3884584963201
Tiempo de ejecución: 0.04166 segundos
```

Para 10 hilos:



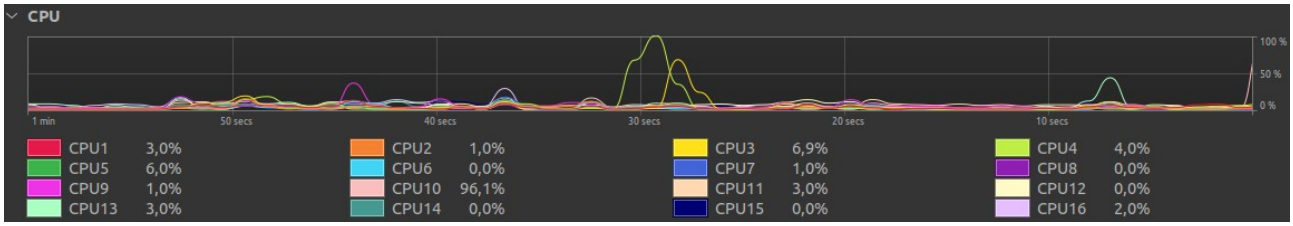

```
Ingrese un número mayor a 1500000: 99999999999
Ingrese el número de hilos: 10
ln(200375420520689664) = 17.3884584963991
Tiempo de ejecución: 0.025923 segundos
```

Analizando ambos casos, vemos que para un solo hilo el CPU14 se dispara a casi 10%, mientras que con 10 hilos se observa un leve incremento en varios CPUs hasta 4% en algunos casos. La comparación más clara es el tiempo de ejecución, siendo casi la mitad del tiempo empleado con 10 hilos para el mismo problema, retornando así un speedup de 1.607.

Actividad 2 – Búsqueda de Patrones

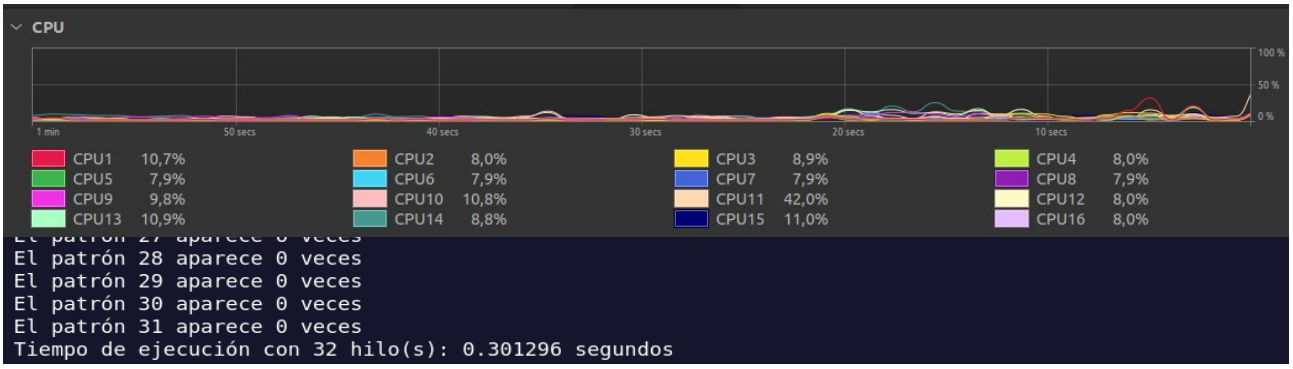
En el siguiente apartado se pidió realizar un reconocimiento de patrones en un texto dado. Para ello, se diseñó una búsqueda directa de subcadenas, siendo esta la más ineficiente, para tener una mejor comparativa a la hora de implementar hilos.

Para 1 hilo tenemos que el CPU10 llega a casi un 100% de uso de sus recursos, tomando 2.53124 segundos en total:




```
Ingrese la cantidad de hilos (1 a 32): 1
El patrón 0 aparece 14 veces
El patrón 1 aparece 3 veces
El patrón 2 aparece 0 veces
El patrón 3 aparece 0 veces
El patrón 4 aparece 0 veces
El patrón 5 aparece 0 veces
El patrón 6 aparece 4 veces
El patrón 7 aparece 0 veces
El patrón 8 aparece 0 veces
El patrón 9 aparece 3622 veces
El patrón 10 aparece 0 veces
El patrón 11 aparece 2 veces
El patrón 12 aparece 0 veces
El patrón 13 aparece 6 veces
El patrón 14 aparece 0 veces
El patrón 15 aparece 0 veces
El patrón 16 aparece 2 veces
El patrón 17 aparece 0 veces
El patrón 18 aparece 6 veces
El patrón 19 aparece 0 veces
El patrón 20 aparece 0 veces
El patrón 21 aparece 2 veces
El patrón 22 aparece 0 veces
El patrón 23 aparece 0 veces
El patrón 24 aparece 0 veces
El patrón 25 aparece 0 veces
El patrón 26 aparece 0 veces
El patrón 27 aparece 6 veces
El patrón 28 aparece 0 veces
El patrón 29 aparece 0 veces
El patrón 30 aparece 0 veces
El patrón 31 aparece 0 veces
Tiempo de ejecución con 1 hilo(s): 2.53124 segundos
```

Mientras que para 32 hilos:

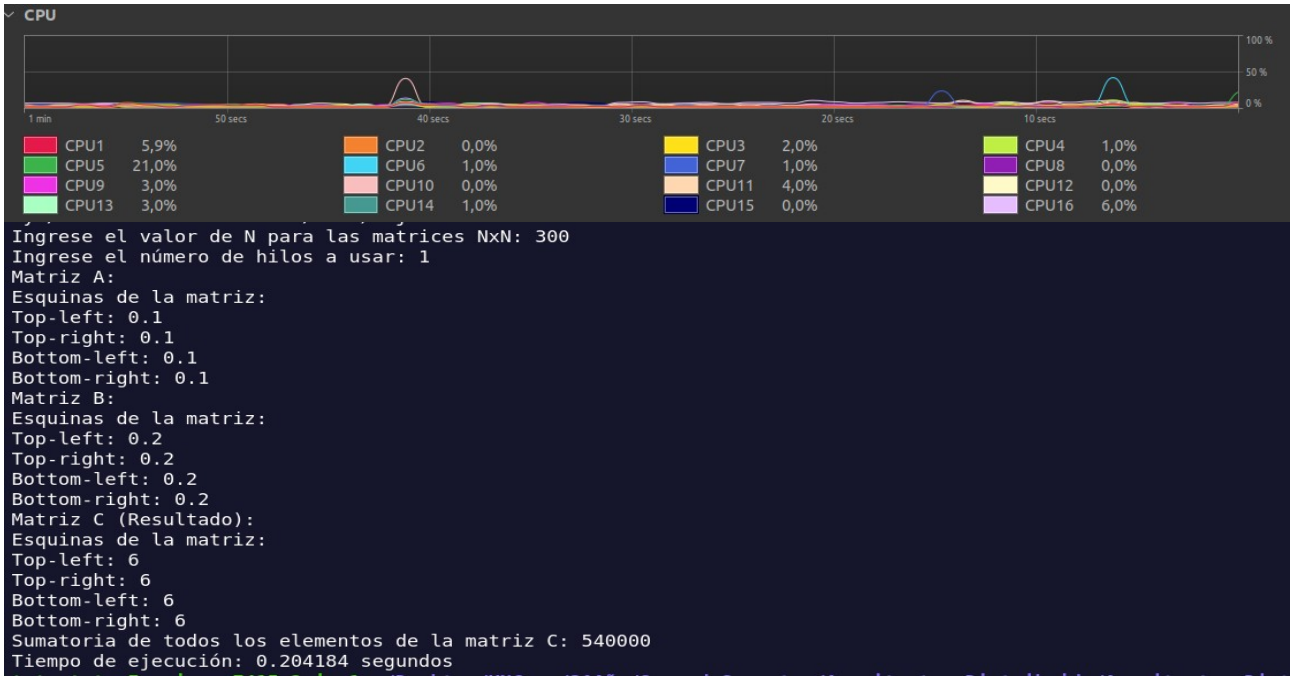


Se observa una mejor distribución del uso de los recursos, llegando en el peor caso a un 42% de uso del CPU11. Sin embargo, en el tiempo de ejecución, vemos una notable disminución, llegando a 0.301296 segundos.

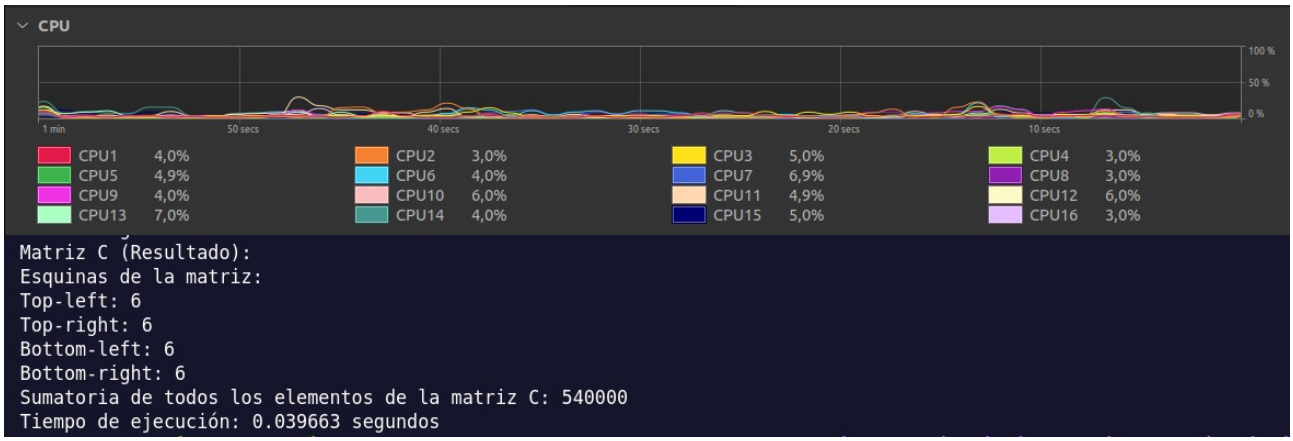
Entonces, analizando los resultados, tenemos que, incluso para un algoritmo de búsqueda ineficiente, la correcta aplicación de hilos nos retorna una mejora sustentable. Siendo el speedup para este caso de 8.401.

Actividad 3 – Multiplicación de matrices N*N

En esta actividad se desarrolló un programa que multiplica matrices de tamaño N. Para la implementación con hilos, se subdividieron las filas de la matriz A según la cantidad de hilos y se realizaron los cálculos asociados a dichas filas, para luego unificar los resultados en la matriz C. Para estos resultados se mantuvo el estándar de 0.1 para todos los valores de A y 0.2 para B.
Para la ejecución con 1 hilo:



Para 32 hilos:

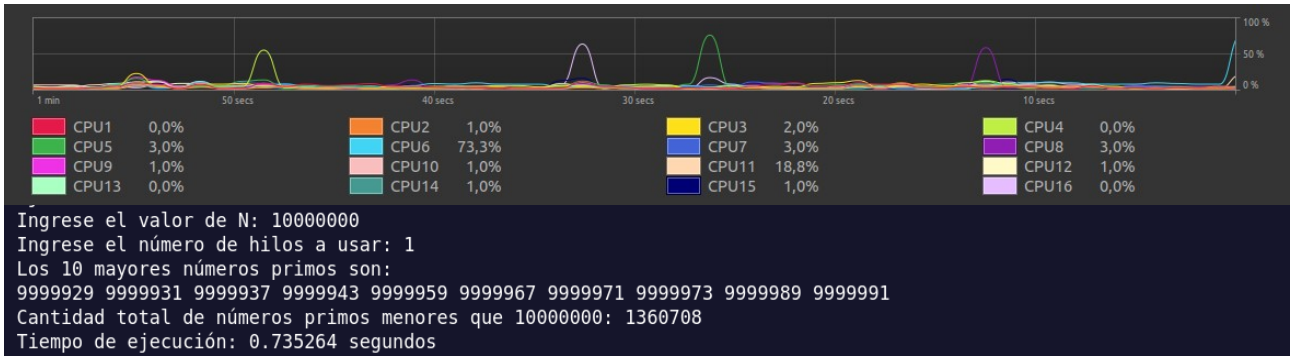


Al igual que las pruebas anteriores, es notoria una mejor distribución de los recursos al utilizar hilos y, nuevamente, una gran diferencia en velocidad. El speedup es aproximadamente 5.14797. Aunque en este caso la diferencia no es tan apreciable, para matrices de 3000*3000, con un hilo no se llegó a un resultado luego de 10 minutos; pero, con 32 hilos, en 66 segundos se logró un resultado, utilizando el 100% de todos los recursos disponibles.

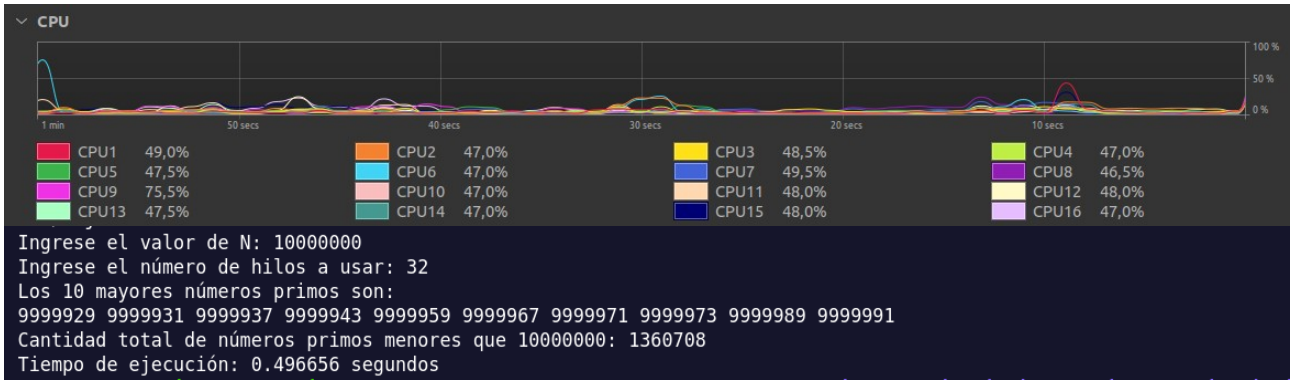
Actividad 4 – Cálculo de números primos

Para esta actividad se necesitaba calcular números primos utilizando hilos. Primero, se generan los primos base hasta la raíz cuadrada de N en un solo hilo. Luego, los hilos se distribuyen en diferentes rangos del espacio de búsqueda para identificar primos, usando los primos base para eliminar múltiplos. Cada hilo agrega sus resultados de forma segura a la lista global de primos usando un mutex.

Resultados con 1 solo hilo:



Y para 32 hilos:



Al igual que en las actividades anteriores, se observa un comportamiento similar. Al trabajar con varios hilos, la carga se distribuye equitativamente en varios CPUs y el tiempo de ejecución baja notablemente. Sin embargo, en este caso vemos que el speedup obtenido, 1.4749, no es tan alto como en los anteriores. Además, se llegó a ocupar casi el 50% de todos los CPUs trabajando con hilos; mientras que, con un solo hilo, se ocupó el 75% de un CPU.