Taller de Sistemas Operativos, 2020

Taller 03

Escuela de Ingeniería Informática

Rodrigo Montenegro Farias

rodrigo.montenegro@alumnos.uv.cl

Resumen.

El objetivo principal de este taller número tres es solucionar la problemática, la cual consiste en la creación de un programa en "c++" junto con el contenido de manejo de hilos a través de la api openMP, la solución está compuestas por dos módulos, el primero es llenar un arreglo con números enteros random, el segundo suma dichos números, ambas tareas se realizan de forma paralela y serial, para realizar pruebas de desempeños que generen gráficos de datos sobre el comportamiento del tiempo de ejecución.

1. Introducción

Actualmente la informática forma parte de nuestras vidas cotidianas, es complicado encontrar un campo en donde no se utilice esta ciencia, debido al aumento de personas que tienen la posibilidad de obtener un computador, nace la interrogante del consumo de energía, este afecta a nivel de software, en el caso que nuestro diseño no sea el adecuado. A su vez esto perjudica a nivel hardware en el caso de que este se caliente por medio de el flujo de corriente que circula mediante las tareas del procesador. La computación paralela es el uso de múltiples recursos computacionales. Se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente.
La programación paralela, se compone de threads para división de tareas independientes, de este modo se pueden ejecutar tareas simultáneas.
Los threads o hilos son una sección secuencia de tareas que permite descomponer de forma eficiente al mismo tiempo otra tarea.

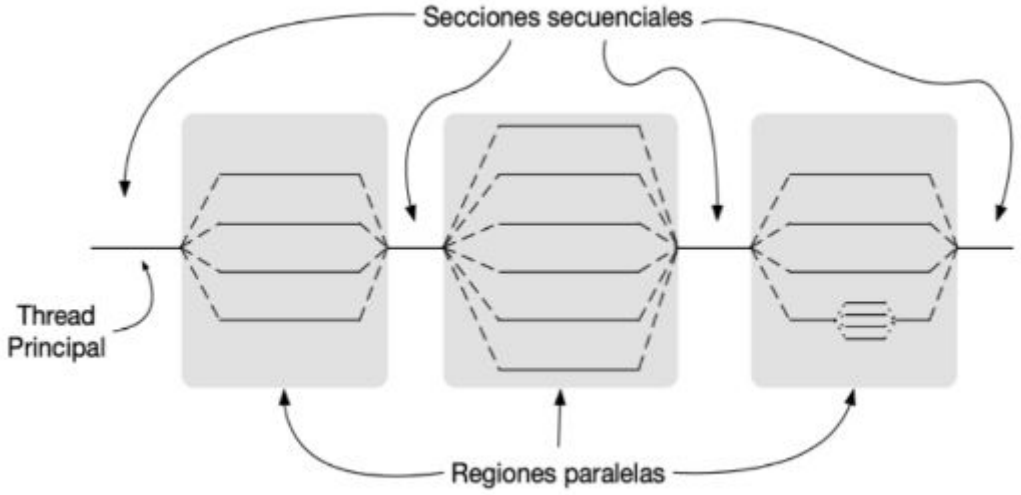

Imagen 1 Funcionamiento de un threads.

1.1 conceptos previos

1.1.1 Paralelismo

Antes de entender que es multiprocesamiento, multithreading y thread debemos entender que paralelismo es una función que realiza el procesador para ejecutar varias tareas al mismo tiempo, en otras palabras, puede realizar varios cálculos simultáneamente, basado en el principio de dividir los problemas grandes para obtener varios problemas pequeños. Al hablar sobre paralelismo surge un concepto fundamental llamado concurrencia, el cual se refiere a la existencia de varias actividades ejecutándose simultáneamente y necesita sincronizarse para ejecutarse de manera conjunta.

1.1.2 Multiprocesamiento

Como su nombre lo indica es un procesamiento el cual permite la ejecución de uno o más hilo de ejecución a la vez. Esto es debido a la existencia de múltiples CPU las cuales pueden ser utilizadas para ejecutar múltiples procesos o múltiple thread dentro de un mismo proceso.

1.1.3 Multithreading

Multithreading o multihilo es la capacidad de ejecutar eficientemente múltiples hilos de ejecución. Esta técnica junto con el Multiprocesamiento, descrito en el punto anterior, son los principales métodos para mejorar el rendimiento de una máquina. Un punto fundamental de esta técnica es que se utiliza para dar solución a las instrucciones ejecutadas secuencialmente, ya que al crear más de un hilo de ejecución este puede realizar trabajos de manera paralela.

1.1.4 Thread

Un Thread [2] o hilo, es la unidad básica de ejecución de un Sistema Operativo. Además, puede ser ejecutado al mismo tiempo que otra tarea. Todos los hilos de ejecución comparten el espacio de direccionamiento del proceso y permiten simplificar el diseño de una aplicación que debe llevar a cabo distintas funciones simultáneamente.

2. Descripción del problema

2.1 contexto del programa

Se solicitó crear un programa en lenguaje de programación C++, cuya funcionalidad del programa debe estar desarrollada a partir de dos módulos, uno que llene un arreglo de números enteros aleatorio y luego otro que los sume, se desarrolló a partir de dos formas una en serie y otra en paralelo es implementada a partir de multithreaded de la api openMP, para finalizar se realizaron pruebas de desempeño a partir de parámetros solicitados, las cuales permitirá ver el resultado de los tiempos y sumas totales de ambas formas, junto a esto el programa debe ejecutarse de la siguiente forma como muestra en la tabla 1, dichos números son parámetros que se encuentran especificado en la tabla 2.

./sumArray -N <número> -t <número> -l <número> -L <número> [-h]

tabla 1 forma de uso.

Parámetros	Descripción
-N	Tamaño del arreglo.
-t	Número de threads.
-l	Límite inferior rango aleatorio.
-L	Límite superior rango aleatorio.
[-h]	Muestra la ayuda de uso y termina.

tabla 2 Parámetros de funcionamiento.

2.2 Análisis de datos

Luego de obtener el programa listo, se desarrollan pruebas de desempeño que generen datos que permitan visualizar el comportamiento del tiempo de ejecución de ambos módulos dependiendo del tamaño del problema y de la cantidad de threads utilizados. Esto se hará mediante comandos que permitan medir el tiempo de ejecución de los módulos en la terminal bash. en la figura 1se podrá visualizar el diseño en general.

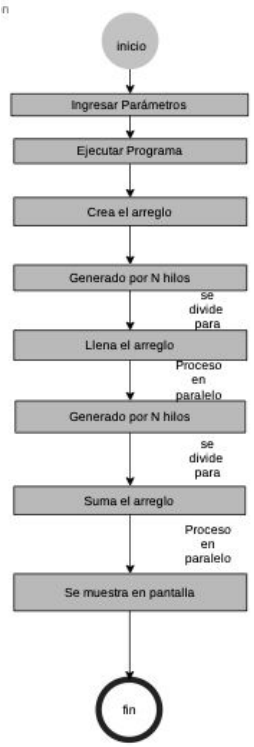

figura 1 muestra el diseño en general.

2.3 Ejemplo de uso

Una vez desarrollado el programa, se ejecutó con los datos descriptivos o parámetros que se encuentran en la tabla 2, para ejecutar este programa se a creado un arreglo con 100000000 elementos, 5 hilos, y el rango de números aleatorios se encuentra entre [20,47] como se puede ver en la siguiente figura 1 .

```
rodrigo@fracasado:~/TSSOO-taller03/TSSOO-taller03$ ./sumArray -N100000000 -t5 -l20 -L47
```

Figura 2 representa la ejecución del programa.

Para mostrar la ayuda de uso del programa es necesario colocar como se muestra en la figura.

```
[rodrigo@fracasado:~/TSSOO-taller03/TSSOO-taller03$ ./sumArray -h
Uso: ./sumArray -N tam_problema -t threads -n num_buscado [-h] Descripción:
        -N   Tamaño del arreglo
        -t   numero de threads
        -l   limite inferior
        -L   limite superior
        -h   ayuda de uso
```

Figura 3 Representa la ejecución sin parámetros sólo la descripción del archivo.

Figura 1, Visualización del diagrama general.

3. Descripción de solución

Mediante el anterior diagrama general tenemos una nocion de como es nuestro programa y cómo se comporta en base a su resultado final, nuestro programa da inicio a calcular el primer módulo, el cual llenar un arreglo de números enteros random y de forma concurrente el segundo módulo calcula la suma del arreglo generando la suma total, luego calcula el desempeño y resultado final en función del tiempo para finalizar mostrándolo por pantalla. En la siguiente figura se podrá visualizar el diagrama de alto nivel que aborda el problema.

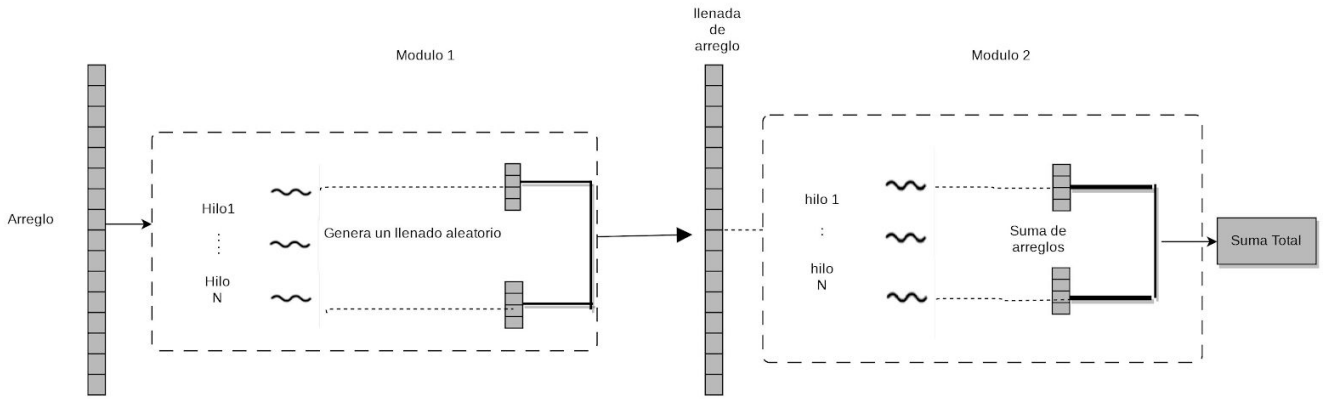

Figura 4, Visualización del diagrama de alto nivel.

El módulo 1, tiene como función llenar un arreglo de números random , el cual divide o troza la tarea en n números de hilos, los cuales son aleatorios y establecidos por parámetros de rangos.

El módulo 2. tiene como función sumar el arreglo de forma paralela y medite en hilos, luego se mostrará en pantalla la suma general.

3.1 Pruebas del programa

Tras ejecutar 12 veces nuestro programa con los siguientes números ingresados, se puede divisar en la figura número 6, el gráfico creado a partir de estas ejecuciones mencionadas antes, se puede comprobar que el tiempo total en paralelo está por debajo que la de serie.

```
rodrigo@fracasado:~/TSSOO-taller03/TSSOO-taller03$ ./sumArray -N100000000 -t5 -l20 -L47
Datos Ingresados
Tamaño del Arreglo: 100000000
Numero de Nhilos : 5
Numero de limite inferior: 20
Numero de limite superior  : 47
Resultados de sumas OMP:
Valor total de suma paralelo OMP :2300000000
Valor total de suma serial OMP :2300000000
Tiempos de llenado OMP:
Tiempo llenado paralelo OMP  :198[ms]
Tiempo llenado serial OMP  :17979[ms]
Tiempos de sumas OMP:
Tiempo suma paralela OMP  :46[ms]
Tiempo suma serial OMP :25784[ms]
Tiempos suma total :
Tiempo total de suma paralela OMP  :244[ms]
Tiempo total de suma serial OMP:51568[ms]
Desempeñpos
rapidez :0.00473162[ms]
eficiencia:0.995291[ms]
```

Figura 5 muestra la ejecución del programa con sus parámetros.

Este gráfico demuestra el tiempo total tanto en serie como paralelo, y los respectivos hilos utilizados.

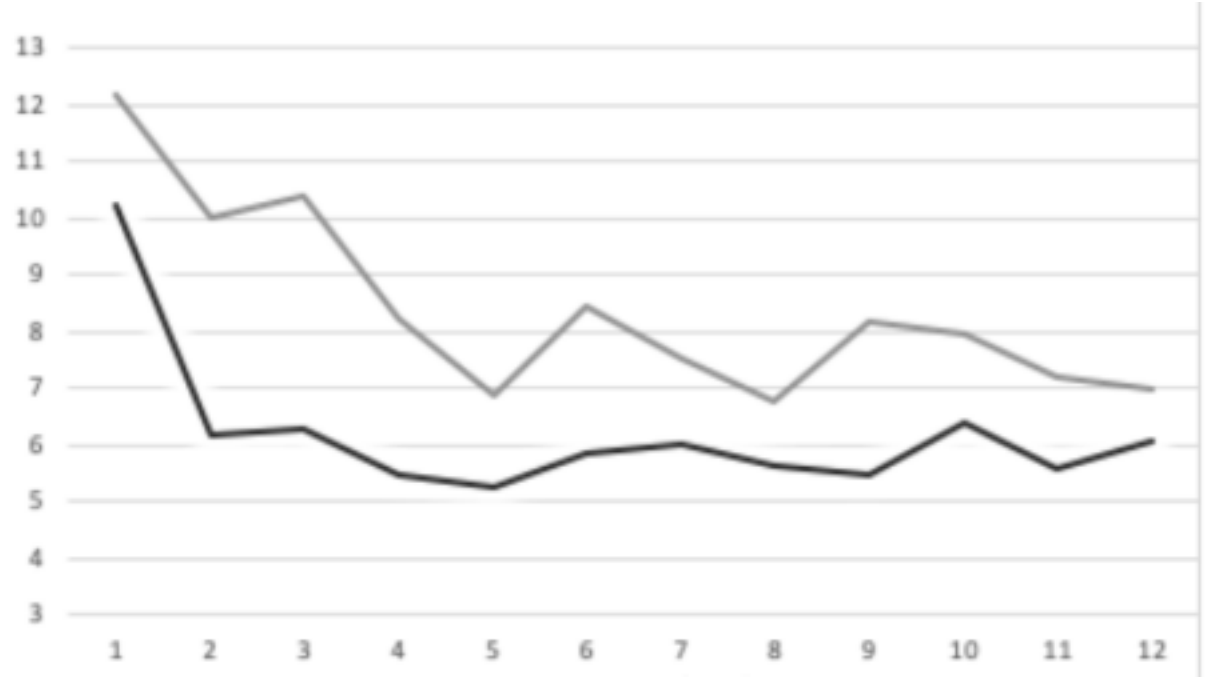

figura 6 Gráfico de tiempo total en serie y paralelo, en función de un número de hilos.

Tras analizar estos gráficos nos damos cuenta que la diferencia es abismante en el caso de el uso de la api con openMP al trabajar de una mejor forma a diferencia del taller entregado antes.

4. Referencias

[1] Programación “Que es c++” (2015). Retrieved from https://openwebinars.net/blog/que-es-cpp/ .

[2] Thread safe “Que es thread safe” (2000). Retrieved from https://esacademic.com/dic.nsf/eswiki/1149762

[3] OpenMP “Que es un hilo, OpenMP” (2000) Retrieved from https://¿queesopenM?/P1149762

5. Conclusión

Como conclusión la programación multihilo, sin duda tiene mucho campo de aplicación, desde los sistemas operativos hasta en las tecnologías actuales que ocupamos cotidianamente, como por ejemplo; celulares y cajeros automáticos.
En la elaboración de este trabajo obtuvimos conceptos que nos darán la fluidez para desarrollar aplicaciones más contundente en c++ [1] u otro lenguaje similar.
Siempre que utilizamos hilos es con la finalidad de poder realizar más de una tarea a la vez es por eso que se utiliza el openMP[3], para administrar los recursos de la mejor forma posible y generar un programa mucho más rápido de lo habitual, aprendí a desarrollar un programa, el cual, está compuesto por hilos, acorde a la dificultad, este código era la necesidad de compartir su trabajo de lo contrario tardaría mucho, así que este método funciona cuando tenemos muchos procedimiento y podemos realizarlos de forma concurrente.