

Taller 02

Taller de Sistemas Operativos
Escuela de Ingeniería Informática

Jorge Rodríguez Antiquera

Jorge.rodriqueza@alumnos.uv.cl

Resumen. *En este informe se presenta la descripción del problema, los datos y variables a utilizar, y el diseño de la solución, en este caso se usó un diagrama de componentes para tener una mejor vista de la implementación a desarrollar. Donde las tareas que se crearan deben realizarse de forma paralela e implementados con thread POSIX.*

1. Introducción

La computación paralela [1] es donde se usan diversos recursos del computador para resolver un problema, la diferencia que se tiene con la computación secuencial, es que las operaciones pueden ocurrir de forma simultánea para resolver el problema. Las ventajas de utilizar el paralelismo es resolver problemas en un tiempo acotado, en comparación con la programación secuencial. La ejecución del código es más rápida. Los resultados se obtienen en un tiempo menor. Permite la ejecución de varias instrucciones en simultáneo.

En la programación paralela se utilizan threads o hilos[2], esto quiere decir que al tener una cantidad de hilos específicos se pueden realizar varias tareas, ya que cada hilo estará a cargo de una tarea en específico. Al tener procesos con diversas tareas independientes estas pueden terminar antes si estas tareas se encuentran en hilos diferentes, por lo que ahorraría tiempo de ejecución. La creación de hilo requiere menos recurso que un proceso y el tiempo de crearla es menor.

Los threads POSIX[3], es un modelo de ejecución que no depende del lenguaje, sino que es un modelo de ejecución para el paralelismo. En un programa se encuentran diversos controles de flujo de trabajo, donde en cada flujo de trabajo se encuentra un thread. La utilización de POSIX principalmente es por ser un estándar que facilita la creación de aplicaciones por su portabilidad y confiabilidad.

En el presente documento se podrá observar la descripción del problema, los módulos que se deberán crear y que deben encontrarse en forma paralela para el desarrollo de la implementación, de los datos y variables que se utilizarán al crear el código. Finalmente, se podrá observar un diagrama de componentes que da la vista de la implementación, desde el llenado, las sumas parciales, los threads, hasta mostrar los resultados y el desempeño de estos.

2. Descripción del problema

El problema consiste en implementar un programa que conste con dos módulos. Uno que llene un arreglo con números enteros aleatorios de tipo `uint32_t` en forma paralela y otro que sume el contenido del arreglo en forma paralela. Se deben realizar 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.

Los datos a utilizar serán `uint32_t`, esto quiere decir que se utilizarán números enteros sin signo que serán generados de forma aleatorias. Las variables a utilizar serán los parámetros, estos son “-N” que tiene el tamaño de arreglo que el usuario ingresará, “-t” este es el número de thread a utilizar, “-I” y “-L” son los límites inferior y superior del rango aleatorio

3. Diseño de la solución

En la figura 1 se puede observar el diseño de la solución con el cual se realizará la implementación para la solución del problema que fue mencionado previamente. Se visualiza una vista de procesos [4] que muestra la perspectiva que tendrá el programador para la implementación.

La metodología que se utilizará es crear un módulo de llenado que por medio de los parámetros ingresados del usuario se podrá determinar el rango de los posibles valores que tomarán los números aleatorios que serán ingresados al arreglo, luego se realizará una suma serial y calcular su tiempo, luego al llegar a la sección de los thread estos separan el trabajo de la suma en sumas parciales dependiendo de la cantidad de thread que hay en el parámetro asignado, se sincronizan y se suman los resultados de las sumas parciales para finalmente calcular su tiempo. En la última parte se muestran los resultados con su eficiencia respectiva.

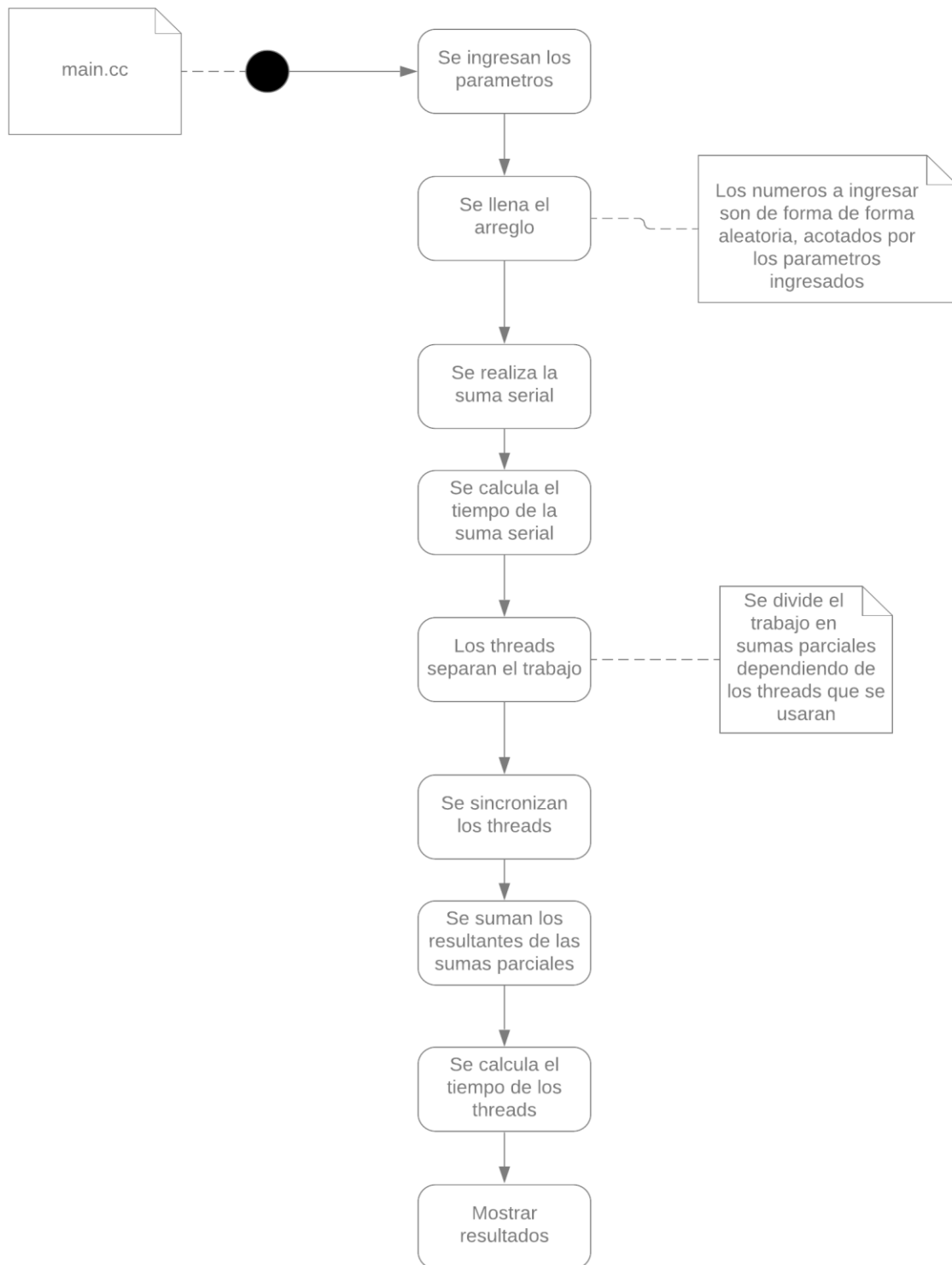


Figura 1. Diagrama de actividades.

4. Resultados

Los resultados obtenidos serán mostrados por pantalla, se podrá ver los parámetros que fueron ingresados, la suma serial y su tiempo, también la suma total de las sumas parciales hechas por los hilos y su tiempo, por último se mostrará el speedUp y la eficiencia del programa.

5. Conclusiones

Se puede concluir que el diseño de alto nivel que se muestra en el documento permite visualizar la estructura que tendrá la implementación para satisfacer los requerimientos del problema, de cómo se debe trabajar en él. También la importancia que tiene la utilización de los threads para hacer programación paralela, en este caso para el llenado del arreglo y la suma de estos por medio de los diversos hilos que el usuario ingresará la cantidad de estos.

Referencias

- [1] Github, http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela_teor%C3%ADa/index.html.
- [2] Wikipedia, [https://es.wikipedia.org/wiki/Hilo_\(inform%C3%A1tica\)#Funcionalidad_de_los_hilos](https://es.wikipedia.org/wiki/Hilo_(inform%C3%A1tica)#Funcionalidad_de_los_hilos).
- [3] Utsm, http://profesores.elo.utfsm.cl/~agv/elo330/2s08/lectures/POSIX_Threads.html.
- [4] Lucidchart, <https://www.lucidchart.com/pages/es/tutorial-diagrama-de-actividades-uml>.