Reporte Técnico: Taller 02

Taller de Sistemas Operativos
Escuela de Ingeniería Informática

Fernando Del Pino Machuca

Fernando.delpino@alumnos.uv.cl

Resumen. *En este reporte se documenta y plantea, una propuesta de diseño de la solución en base al problema principal del taller, este problema consiste en la utilización del lenguaje "C++" para manejar el llenado de un arreglo de números enteros, los cuales procederán a ser sumados y mostrar la suma total por pantalla. Este proceso consiste en 2 módulos que deben ejecutarse en forma paralela, el llenado de números aleatorios del arreglo y el proceso de suma de los datos ingresados en el arreglo, mediante la implementación de Threads POSIX. Este proceso se debe manejar con el uso de secciones criticas para establecer la concordancia entre los datos y evitar errores de consistencia. El presente reporte se basa a grandes rasgos del modelamiento de una solución posible, para tener mayor claridad en los procesos siguientes relacionados con la codificación de la solución y su uso.*

1. Introducción

1.1. Problema

Este reporte se basa en la implementación de un programa que llene un arreglo de números enteros donde estos se sumen, realizando estas tareas de forma paralela, proceso implementado con "Threads" (Hilos) POSIX. Se debe crear un programa que este compuesto de dos módulos. Uno que llene un arreglo de números enteros aleatorios del tipo "uint32_t"(enteros sin signo de 32 bits) en forma paralela y otro que sume el contenido del arreglo también en forma paralela. Se deben hacer 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 hilos utilizados. Para generar números aleatorios, se debe utilizar funciones que sean "Thread Safe" para que observe una mejora en el desempeño del programa. Junto a esto, el programa debe ser ejecutado con parámetros establecidos (Mostrados en la tabla 1), donde estos parámetros están especificados en la tabla 2.

Tabla 1. Forma de uso:

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

Tabla 2. Parámetros:

-N : tamaño del arreglo

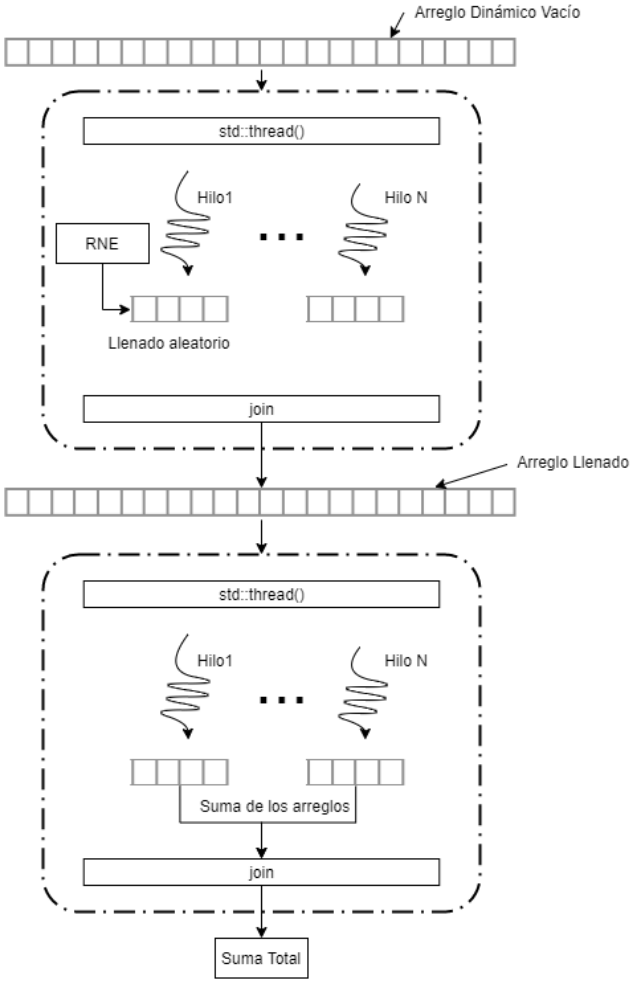
-t : Numero de Threads.
-l : Límite inferior del rango aleatorio
-L : Límite superior del rango aleatorio
[-h] : Muestra la ayuda de uso y termina

Cada parámetro deberá ser utilizado para especificar el funcionamiento del programa, especificando la cantidad de hilos a crear, el tamaño del arreglo y los limites de los números aleatorios que deben ser rellenados sobre el arreglo.

1.2. Análisis

Analizando lo requerido, se busca la implementación de un programa que llene un arreglo de números enteros donde estos números sean sumados, mostrando el resultado total por pantalla. Para lograr la codificación de la solución se debe usar un proceso de llenado y suma concurrente, utilizando métodos de paralelización [1] para proporcionar mejoras de rendimiento de procesado en cuanto a los bucles y trabajos del código. Estos métodos de paralelización abarcan el paradigma de programación paralela, proceso que se basa en realizar varias tareas e instrucciones de forma simultánea, esto gracias al principio de “dividir y vencer”, principio enfocado a tomar grandes tareas y dividirlas en pequeñas tareas, las cuales se entregan a distintos receptores de procesamiento que se encargaran de trabajar simultáneamente, agilizando el procedimiento de computo y ejecución. Para que este proceso de paralelización se lleve correctamente a cabo, se deben disponer de algoritmos de exclusión mutua, alias “mutex” (mutual exclusion), para evitar que mas de uno de estos procesos ingrese a la sección critica(fragmento del código o proceso donde se modifica el recurso compartido), ayudando a si a mantener una consistencia de los datos a la hora de trabajar con información compartida y variables globales. Este proceso del manejo de hilos corresponde a lo establecido para mantener un desarrollo “Thread Safe”(seguridad en hilos) [2], el cual se basa en satisfacer que múltiples hilos puedan acceder a los mismos datos compartidos, manteniendo la integridad de estos datos junto con minimizar el comportamiento inesperado a la hora de ejecución de estos hilos. El entendimiento del problema se ve reflejado en la Fig. 1.

Figura 1 Objetivo General del Problema.



Es posible ver como se debería comportar el arreglo de manera general y su llenado según se comprende. Es necesario destacar que “RNE” (Random Number Engine) es el encargado de generar números aleatorios, para llenar cada arreglo vacío que poseerán los N Hilos y ser sumados en el módulo siguiente por los nuevos N Hilos para mostrar la suma total por pantalla. La cantidad de hilos creados es instanciada a la hora de ejecutar el programa, donde el modulo de llenado y suma deben tener a lo menos 1 hilo cada uno que los ejecute. Mientras mas hilos creados haya, cada tarea de los módulos debe dividirse en pequeñas partes para cada hilo, donde habrá muchos hilos que guarden datos en el mismo arreglo, y muchos hilos que sumen los datos del arreglo. Su relación está basada con respecto a “N:N”, que quiere decir que la cantidad de hilos que guarden datos, serán la misma cantidad de hilos que sumen los datos.

En base a esto se formula la estructura de este primer reporte, abarcando el diseño de la posible solución al problema planteado, junto con su comprensión respectivo al modulo de llenado y el modulo de sumado, analizando el comportamiento para dar claridad al proceso de codificado a realizar en próximos avances.

Antes de profundizar en el procedimiento de diseño, es necesario destacar que todo el proceso de desarrollo se realizara a través de la maquina virtual instanciada en estudios previos, la cual se encuentra establecida con

sistema operativo Ubuntu (distribución de Linux), a la cual se le instalo los paquetes de las aplicaciones respectivas para la compilación y ejecución de programas basados en lenguaje “C++”.

2. Diseño

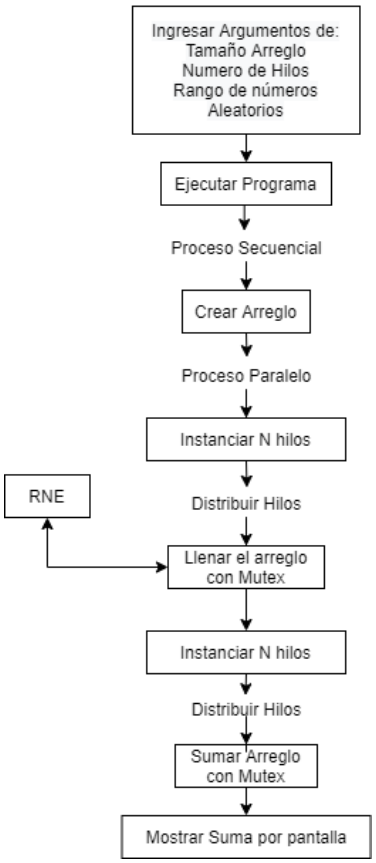
2.1. Metodología:

La metodología para abarcar las tareas propuestas por cada módulo fue analizar cada requerimiento, diseñar un modelo general del funcionamiento del programa, donde se vea reflejado un orden de procesos a realizar antes, para lograr el llenado y suma de los arreglos. Una vez analizado el funcionamiento general del programa, se debe crear un modelo de diseño específico para cada módulo del taller, eso quiere decir, que se debe idear un diseño de comportamiento para cada módulo, ya que cada módulo presenta una estructura y modo de ejecutar cada tarea de manera distinta, por lo que una vez especificado, se procederá a utilizar como ejemplo para su implementación en código en entregas futuras.

2.2. Descripción General:

Para lograr un correcto entendimiento del funcionamiento del programa, es necesario, que se entienda el proceso de forma general, ya que se analizar más adelante como será el llenado del arreglo en el primero hilo, además de entender como es el flujo de los datos hacia el segundo hilo para luego ser sumados. Este modelo general se ve reflejado en la Fig. 2.

Figura 2. Diseño general de la solución

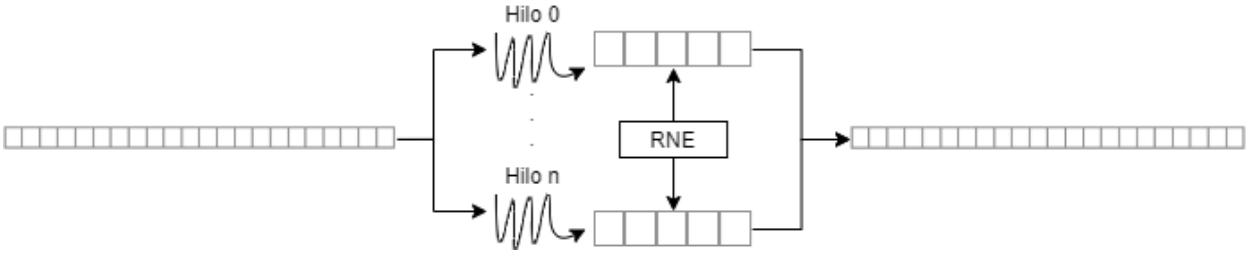


Como se puede observar, se exige que los argumentos de entrada a la hora de ejecutar el programa entreguen los parámetros para definir el tamaño del arreglo, numero de hilos a crear y el rango de los números aleatorios que se pueden presentar (este rango ingresa como limite inferior y limite superior). Para esto se puede analizar que es conveniente instanciar que los parámetros entregados son correctos, es decir, la línea de comando para la ejecución del código debe cumplir con los parámetros que requiere el programa. Una vez ejecutado el programa este debe comenzar secuencialmente a crear el arreglo vacío con el tamaño establecido, para luego crear la cantidad respectiva de hilos que ayudaran a los diferentes módulos. En base a esto se debe disponer de un numero equilibrado de hilos entre modulo 1 y modulo 2, para que completen las tareas de llenado y suma del arreglo exigido. Para crear a los hilos respectivos se deberá profundizar en condicionales que cumplan la función de crear los respectivos hilos exigidos. Una vez realizado este proceso se debe determinar cuantos hilos se usarán para el modulo 1 y asignarles las tareas de llenado del arreglo por las partes que se estimen convenientes. Este llenado se debe hacer mediante el encargado de generar números aleatorios en base a los limites propuestos a la hora de ejecutar el programa. Tras realizar el llenado del arreglo, se debe ejecutar el modulo de suma del arreglo compuesto de la otra mitad de hilos instanciados. Se sugiere que como mínimo se creen 2 hilos (Hilo 1 e Hilo 2), para que cada uno realice un modulo y así entregar la información respectiva a la suma total de los datos.

2.3. Módulo de Llenado:

Una vez realizado la estructuración del diseño general de la solución, se puede diseñar el módulo respectivo a la etapa de llenado del arreglo. El diseño de este módulo se puede ver estructurado en Fig. 3.

Figura 3. Diseño modulo llenado del arreglo.

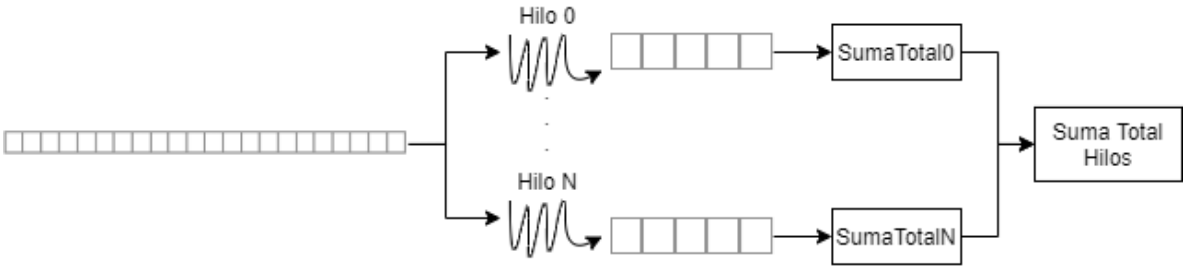


En primer lugar, el arreglo debe separarse en pequeños arreglos para cada hilo instanciado, una vez realizado esto, cada hilo debe rellenar cada fragmento del arreglo entregado, con números aleatorios dentro de los rangos establecidos por los parámetros de entrada (RNE). Una vez realizado esto por cada hilo, se debe reagrupar todos los pedazos del arreglo en uno solo para ser usado luego por el siguiente modulo.

2.4. Módulo de Suma:

Tras rellenar el arreglo, se debe instanciar nuevamente cada hilo especificado, para que realice el proceso de suma del contenido del arreglo. El módulo de suma se ve representado en la Fig. 4.

Figura 4. Diseño modulo suma del arreglo.



El diseño se basa en que cada hilo instanciado, tome una parte del arreglo rellenado, considerado que cada hilo posee pedazos del arreglo original, continuando con el respectivo proceso de suma de cada elemento de estos sub-arreglos. Finalmente, cada una de estas sumas, serán nuevamente sumadas para obtener la suma total de cada hilo.

2.5. Conclusión.

En base a cada diseño es posible implementar de manera mas clara, el código que resuelva la problemática del taller en futuras entregas, dado esto, cada diagrama de solución se considera sujeto a cambios frente a posibles

errores de interpretación, como de implementación de características del código, aunque si bien es cierto, se considera a grandes rasgos, cada esquema como punto base para el proceso de realización del programa, donde se procederá a su implementación. Sin mas que detallar o decir, se da por finalizado este reporte de avance sobre la problemática planteada en el taller 2.

3. Referencias.

[1] Introducción a los Sistemas Operativos, Concurrencia y paralelismos, Marisa Gil, pp.10-19.

[2] An Introduction to Programming with Threads, Andrew D. Birrell, 1989, pp.03-14.