Reporte Técnico – Taller 2

Taller de Sistemas Operativos

Escuela de Ingeniería Informática de la Universidad de Valparaíso

Yian Vera Soto

yian.vera@alumnos.uv.cl

Resumen. *Este documento esta basado en el trabajo a realizar en el taller 2 del ramo de Taller de Sistemas Operativos. El objetivo de este taller es aprender el uso de threads para el procesamiento de tareas específicas. En este caso se busca llenar un arreglo con números generados con una función random y sumarlos así logrando la multitarea logrando un mayor desempeño y eficiencia en el procesamiento y el tiempo.*

1. Introducción

Este reporte técnico está basado en el Taller 2 donde el objetivo de este es implementar un programa que llene un arreglo de números enteros generados con una función random y luego los sume, estas tareas tienen que ser realizadas de forma paralela e implementadas con threads POSIX y el lenguaje de programación 2014 o superior.

Paralelismo es la técnica de computo que permite procesar varios cálculos al mismo tiempo, se basa en dividir un problema grande y trabajar simultáneamente sus partes pequeñas a nivel de bit, instrucciones, datos y tareas. Esta técnica con los años a logrado solucionar problemas que se consideraban muy largos y costosos incluso abarcando áreas de la biología y economía.

La Multitarea en sistemas operativos es la característica que permite que varios procesos se ejecuten aparentemente al mismo tiempo dando servicio a estos procesos para que se puedan ejecutar varios programas al a vez, todo esto es debido a que se realiza una operación llamada cambio de contexto, esta quita un proceso del CPU, ingresa uno nuevo, y luego vuelve a ingresar el proceso que quitó del CPU en una especie de cola de ejecución permitiendo la multitarea

Los Threads o hilos en Informática son simplemente una tarea que puede ser ejecutada al mismo tiempo que otra tarea, estos poseen un estado de ejecución y pueden sincronizarse entre ellos para evitar problemas de compartición de recursos. Generalmente cada hilo tiene una tarea específica y determinada, para aumentar la eficiencia del uso del procesador.

POSIX significa Portable Operating System Interface (for Unix). Es un estándar orientado a facilitar la creación de aplicaciones confiables y portables. La mayoría de las versiones populares de UNIX (Linux, Mac OS X) cumplen este estándar en gran medida. La biblioteca para el manejo de hilos en POSIX es pthread.

Este reporte técnico consta con 3 secciones, la Introducción del Taller 1, la descripción del Problema donde se contextualizará este y se explicaran las tareas a realizar y el Diseño de la solución que explicara la solución de las tareas a través de diagramas de alto nivel y los métodos empleados para la solución.

2. Descripción del Problema

El problema por enfrentar en el Taller 2 es implementar un programa que llene un arreglo de números enteros y luego los sume, estas tareas tienen que realizarse en forma paralela e implementadas con threads bajo el estándar POSIX. Y el lenguaje de programación C++ 2014 o superior.

Este programa tiene que estar compuesto de dos módulos, uno tiene que llenar un arreglo de números enteros aleatorios del tipo **`uint32_t`** en forma paralela y el otro que sume el contenido del arreglo 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 threads utilizados. También se generarán números aleatorios con funciones que sean *thread safe* para observar una mejora en el desempeño del programa

2.1. Forma de uso

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

Parámetros:

```
-N   : tamaño del arreglo.
-t   : número de threads.
-l   : limite inferior rango aleatorio.
-L   : límite superior rango aleatorio.
[-h] : muestra la ayuda de uso y termina.
```

2.2. Ejemplo de Uso

1) Crea un arreglo de 1000000 posiciones, con 4 threads. Los números enteros aleatorios están en el rango [10,50]

```
./sumArray -N 1000000 -t 4 -l 10 -L 50
```

2) Muestra la ayuda y termina

```
./sumArray -h
./sumArray
```

3. Diseño de la Solución

3.1. Solución General

El diagrama de secuencia de la Figura 1 demuestra la solución general al problema al ejecutar el programa y sus relaciones con procesador, la creación y llenado del arreglo y la suma de sus valores, describiendo cómo funciona el proceso que se requiere para cumplir los dos módulos.

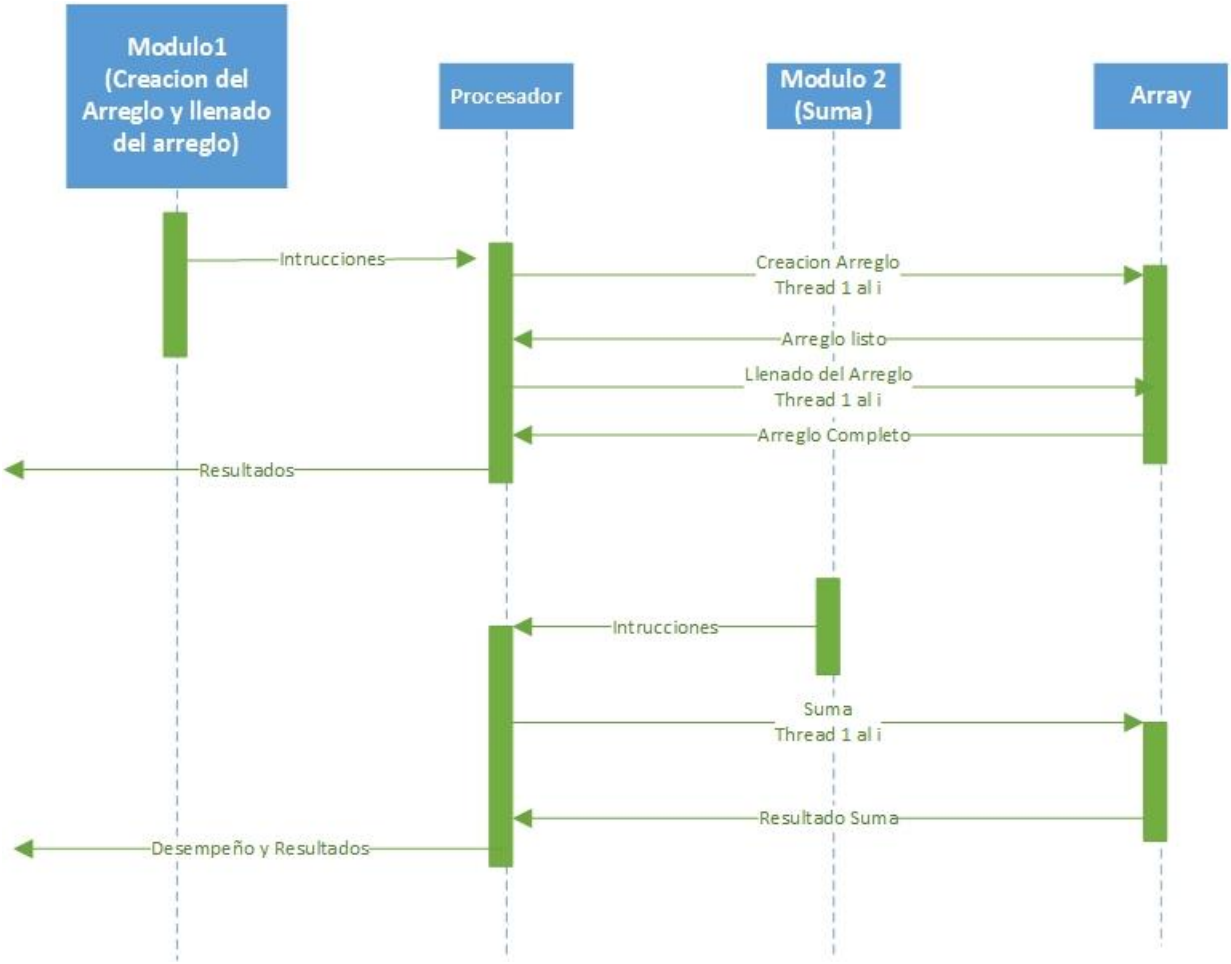

Figura 1. Diagrama de Secuencia solución general

3.2. Modulo 1

El diagrama de flujo de la Figura 2 representa a la función **fillArray** la cual se encargara de llenar el arreglo global **g_numbers[].** Esta función cuenta con dos parámetros **beginIdx** y **endIdx** que indican el inicio y el fin respectivamente del índice, así permitiendo la llamada de la función de forma secuencial o parcial por los hilos. También cuenta con un generador de números aleatorios para el llenado del arreglo.

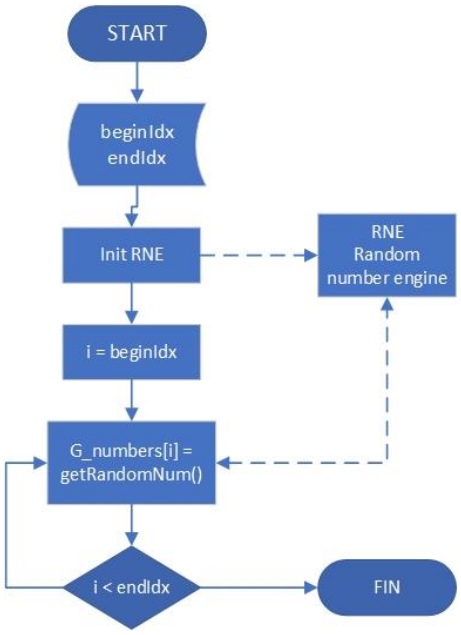

Figura 2. Función fillArray

En la Figura 3 se demuestra a través de un diagrama de secuencia el proceso de llenado del arreglo con los threads, la función `Main()` se encargara de generar el arreglo, crear los threads y matarlos cuando estos hayan cumplido su tarea la que es junto a la función `random()` que generara los números aleatorios llenar el arreglo.

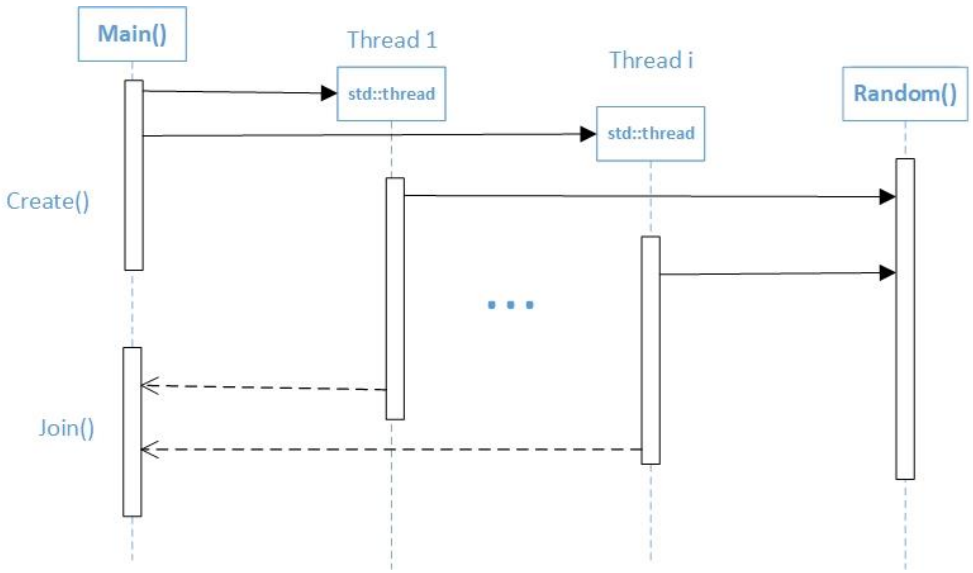

Figura 3. Funcionamiento de los threads para el llenado del arreglo

3.3. Modulo 2

4. Pruebas

Se realizaron una serie de pruebas sobre el programa construido el cual se reflejarán a través de capturas de pantalla sobre lo que nos arroje este, estas pruebas se realizaron sobre la maquina virtual construido en los inicios de este taller.

La primera prueba corresponde a las formas de uso, en la Figura 4.

```
suvin@fracaso:~/Talleres/U2/TSSOO2$
suvin@fracaso:~/Talleres/U2/TSSOO2$ ./sumArray
Uso: ./sumArray -N tam_problema -t threads -l limite_inferior -L limite_superior[-h] Descripción:
        -N   Tamaño del problema
        -t   Cantidad de threads a utilizar
        -l   Limite inferior del rango de numeros randomicos
        -L   Limite superior del rango de numeros randomicos
        -h   Muestra esta ayuda y termina

suvin@fracaso:~/Talleres/U2/TSSOO2$
```

Figura 4. Formas de uso

En la segunda prueba se pone a prueba el programa, con valores bajos para cada parámetro, se puede apreciar que los números son tan bajos que prácticamente el tiempo fue despreciable, Figura 5.

```
suvin@fracaso:~/Talleres/U2/TSSOO2$ ./sumArray -N 100 -t 1 -l 1 -L 50
Total de Elementos: 100
Total de Hilos  : 1
===Llenado del arreglo===
Tiempo de llenado :0[ms]
==========Sumas==========
Suma Serial:2469
Suma Paralela: 2469
=====Tiempos de Suma=====
Tiempo serial :0[ms]
Tiempo paralelo :0[ms]
suvin@fracaso:~/Talleres/U2/TSSOO2$
```

Figura 5. Prueba 2 con valores mínimos

En la tercera prueba se mantiene con un hilo, pero se aumentan sus valores, Figura 6.

```
suvin@fracaso:~/Talleres/U2/TSSOO2$ ./sumArray -N 1000000 -t 1 -l 1 -L 50000
Total de Elementos: 1000000
Total de Hilos  : 1
===Llenado del arreglo===
Tiempo de llenado :57[ms]
==========Sumas==========
Suma Serial:3518169526
Suma Paralela: 3518169526
=====Tiempos de Suma=====
Tiempo serial :21[ms]
Tiempo paralelo :21[ms]
suvin@fracaso:~/Talleres/U2/TSSOO2$
```

Figura 6. Prueba 3 se aumentan algunos valores

En la prueba 4 se aumentaron los hilos y se colocaron números muy grandes, un evidente suceso es que el tiempo paralelo fue muchas veces superior al serial, Figura 7.

```
suvin@fracaso:~/Talleres/U2/TSSOO2$ ./sumArray -N 72316123 -t 6 -l 343424 -L 2141325
Total de Elementos: 72316123
Total de Hilos  : 6
===Llenado del arreglo===
Tiempo de llenado :4345[ms]
==========Sumas==========
Suma Serial:3814067767
Suma Paralela: 3814067767
=====Tiempos de Suma=====
Tiempo serial :1602[ms]
Tiempo paralelo :9004[ms]
suvin@fracaso:~/Talleres/U2/TSSOO2$
```

Figura 8. Prueba 4 se aumentan demasiado los valores.

5. Desempeño

Después de realizar distintas pruebas para ver el funcionamiento del programa, realizaremos pruebas de desempeño que harán una medición sobre el tiempo y el numero hilos utilizados para sobre el modulo de sumas en paralelo

Los parámetros utilizados para estas pruebas serán treinta millones de elementos, el rango inferior será uno y el mayor cien mil y un thread el cual ira aumentando en uno en cada medición

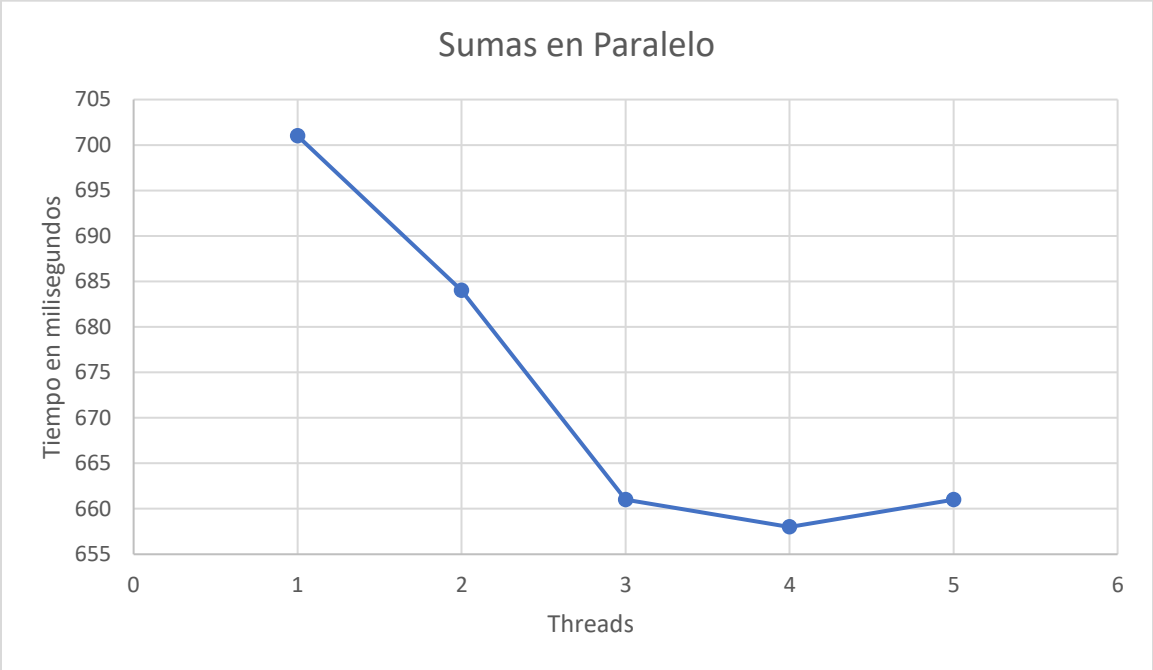

Figura 10. Tiempo de suma en paralelo por thread

6. Conclusión

Finalizando este taller se concluye completando los objetivos correspondientes a la creación de los dos módulos que no pedían la creación y llenado de un arreglo con números aleatorios y sumar todos estos números de forma serial y paralela. En conclusión, al utilizar threads para las sumas se puede optimizar el tiempo pero después del tercero ya no

es útil mayor cantidad de threads y respecto a las primeras pruebas si es mucho el calculo a realizar en sumado en serie es mucho mas eficiente.