Threads y paralelismo

Taller de Sistemas Operativos
Escuela de Ingeniería Informática

Raúl Valenzuela Cinto

Raul.valenzuela@alumnos.uv.cl

Resumen. *Este informe tecnico consta de la creación de un programa el cual consta de dos modulos, el primero deberá crear y rellenar un arreglo con números randomicos en forma paralela y el segundo modulo deberá tomar este arreglo y sumarlo de forma paralela, además se deberá realizar pruebas de desempeño para visualizar el desempeño del programa.*

1 Introducción

Los Threads (hilos) son considerados una agrupación en un programa junto un conjunto de registros del procesador que utilizan además de una pila de maquina, estos Threads son simpliemente tareas que se pueden ejecutarse al mismo tiempo que otra tarea, si estos se ejecutan al mismo tiempo y comparten los mismos recursos se pueden llamar un proceso, este proceso seguirá funcionando ya hayan muchos threads o al menos uno siga activo en ejecución.

Debido a esto el paralelismo que constan en varios calculos que pueden realizarse simulteaneamente esto ayuda a dividir problemas grandes para obtener varios pequeños, que luego serán solucionados en paralelo. Existen 2 tipos de paralelismo: Single program y Multiple Data las cuales son tecnicas para empleadas para que se logre el paralelismo.

Para completar este taller se utilizarán todo lo antes mencionado además de utilizar un lenguaje de programación C++ que permite agrupación de instrucciónes y la separación de un programa en modulos que admiten una compilación independiente, permitiendo así cumplir con el objetivo del taller el cual consistirá en realizar un programa con dos modulos, uno que llene un arreglo de numeros enteros aleatorios de forma paralela y otro en el que sume los numeros del arreglo de forma paralela.Además se tiene que tener en cuenta que para la realización de pruebas de desempeño lo mejor es comparar cosas, por lo que en este caso también se hará los mismos pasos anteriores pero de forma secuencial para luego compararlos entre si con la forma paralela y así obtener mejores resultados de desempeño.

2 Descripción

Se deberá implementar un programa que este compuesto por 2 modulos, uno en el cual se llenará un arreglo de números aleatorios enteros de tipo uint32_t de forma paralela, el otro modulo tendrá que sumar el contenido del arreglo de forma paralela.

Además se deberán hacer pruebas de desempeño las cuales generen datos que permitan visualizar el comportamiento del tiempo de ejecución en ambos modulos, todo dependiendo del tamaño del problema y la cantidad de threads utilizados.

Al terminar los modulos la forma de ejecutar el programa se dará a cabo de la siguiente Tabla1.

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

Tabla1 Forma de uso.

Y sus parametros estan explicados en la Tabla2.

-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.

Tabla 2 Parametros.

En la Tabla 3 Se puede vizualizar un ejemplo de utilización de los datos.

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

Tabla 3 Ejemplo de utilización.

3 Diseño de la solución

Primero que nada para entender el problema a resolver, se debe tener en cuenta que este consiste en la realización de un codigo en c++, el cual contendrá dos modulos,antes de empezar con los modulos se deberá tener en cuenta que el codigo tiene que solicitar ciertos argumentos de entrada, para definir el tamaño del arreglo,el numero de hilos a crear y el rango de los numeros aleatorios sindo definidos por un limitite inferior y uno superior, luego se creará un arreglo de tamaño dinamico, este será llenado con numeros aleatorios de forma paralela, luego tendremos el modulo de suma, que consistirá en tomar los datos guardados y sumarlos de forma paralela, siendo guardados en una variable para hacer pruebas de desempeño proximamente.

En la Figura *1* se puede analizar la forma del modulo de etapa de llenado en la cual se llena el arreglo con numeros random usando threads.

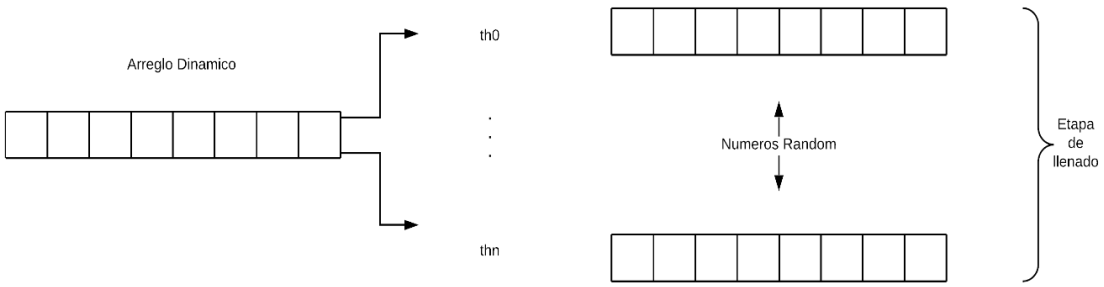

Figura 1: Modulo de llenado

En la Figura *2* se puede ver la forma que toma los datos de suma de cada thread y estas se vuelven a sumar para obtener la suma total.

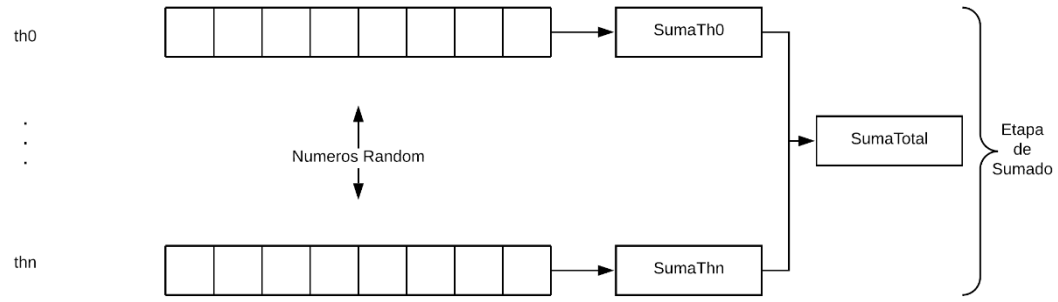

Figura 2: Modulo de Sumado

4 Pruebas

Primeramente se tiene que ver si los parametros ingresados son correctos, si no lo son esto debe mostrar la forma de uso como lo muestra la Figura *3*

```
ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$ ./sumArray -R 29282
./sumArray: invalid option -- 'R'
Uso: ./sumArray -N <nro> -t <nro> -l <nro> -L <nro> [-h] Parametros:
        -N   tamaño del arreglo.
        -t   número de threads
        -l   limite inferior rango aleatorio
        -L   limite superior rango aleatorio
        -h   muestra esta ayuda y termina

ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$
```

Figura 3: Muestra de forma de uso.

Como se puede ver en la Figura *4* tanto como en la Figura *5* y Figura *6* se muestra la forma de ejecución del codigo con distintos parametros, además de las distintas muestras de resultados que serán analizados más adelante, ya que se harán mas pruebas de las que se muestran a continuación para una mejor visualización en resultados.

```
ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$ ./sumArray -N 100000000 -t 2 -l 1 -L 50000
Elementos: 100000000
Threads  : 2
limite inferior: 1
limite Superior: 50000
==========Suma Total================
Suma en Secuencial: 2500214372475
Suma en Paralelo: 2500214372475
==========Tiempos de Llenado=========
TiempodeLLenadoSecuencial: 3682[ms]
TiempodeLLenadoParalelo: 1860.83[ms]
SpeedUp Etapa de Llenado: 1.97869
==========Tiempos de Sumado==========
TiempoDeSumadoSecuencial: 2741[ms]
TiempoDeSumadoParalelo: 80.0636[ms]
SpeedUp Etapa de Suma: 34.2353
ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$
```

Figura 4: Muestra de datos N°1

```
ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$ ./sumArray -N 2000000 -t 4 -l 1 -L 6000
Elementos: 2000000
Threads  : 4
limite inferior: 1
limite Superior: 6000
==========Suma Total================
Suma en Secuencial: 6000774866
Suma en Paralelo: 6000774866
==========Tiempos de Llenado=========
TiempodeLLenadoSecuencial: 77[ms]
TiempodeLLenadoParalelo: 26.3761[ms]
SpeedUp Etapa de Llenado: 2.91931
==========Tiempos de Sumado==========
TiempoDeSumadoSecuencial: 51[ms]
TiempoDeSumadoParalelo: 1.6962[ms]
SpeedUp Etapa de Suma: 30.0672
ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$
```

Figura 5: Muestra de datos N° 2

```
ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$ ./sumArray -N 40000000 -t 5 -l 1 -L 8000
Elementos: 40000000
Threads  : 5
limite inferior: 1
limite Superior: 8000
==========Suma Total================
Suma en Secuencial: 160017377330
Suma en Paralelo: 160017377330
==========Tiempos de Llenado=========
TiempodeLLenadoSecuencial: 1421[ms]
TiempodeLLenadoParalelo: 434.017[ms]
SpeedUp Etapa de Llenado: 3.27407
==========Tiempos de Sumado==========
TiempoDeSumadoSecuencial: 1073[ms]
TiempoDeSumadoParalelo: 24.6829[ms]
SpeedUp Etapa de Suma: 43.4714
ignaciov@DESKTOP-HHF3GNH:/mnt/c/Users/Nasho/Desktop/U2-master/taller02$
```

Figura 6: Muestra de datos N° 3

5 Resultados

Luego de ciertas pruebas se pudo analizar que para el mismo numero de elementos y limites inferior, superior, al aumentar el numero de threads se puede verificar que hasta los threads disminuian su tiempo de llenado respecto a la forma secuencial(1 thread) como se muestra en la **¡Error! No se encuentra el origen de la referencia.**. Primeramente se hicieron las pruebas basadas en el computador usuario para después comparar con el sistema invitado.

Forma de uso para las pruebas : ./sumArray -N 40000000 -t 1 -l 1 -L 8000 cambiando el numero de threads para ver el cambio.

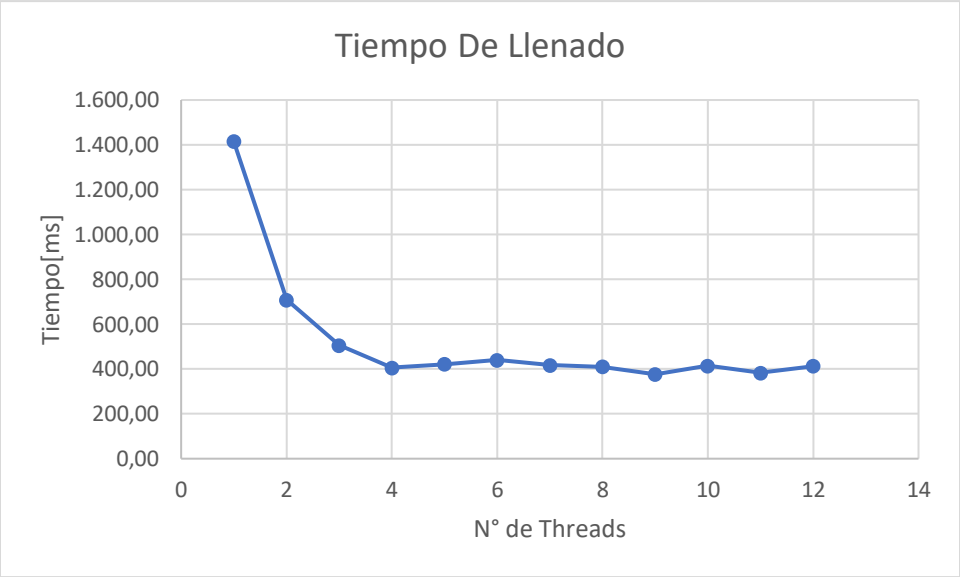

Figura 7:Gráfico N°Threads vs Tiempo de ejecución

Analizando los resultados en el grafico se puede evidenciar que para el llenado de la solución, el tiempo que demora paralelamente, es más eficiente siempre que sea hasta 4 threads ya que después el cambio es minimo.

En la **¡Error! No se encuentra el origen de la referencia.** se puede ver la comparación de tiempo respecto al uso de threads, teniendo en cuenta los mismos parametros de entrada de la parte de llenado.

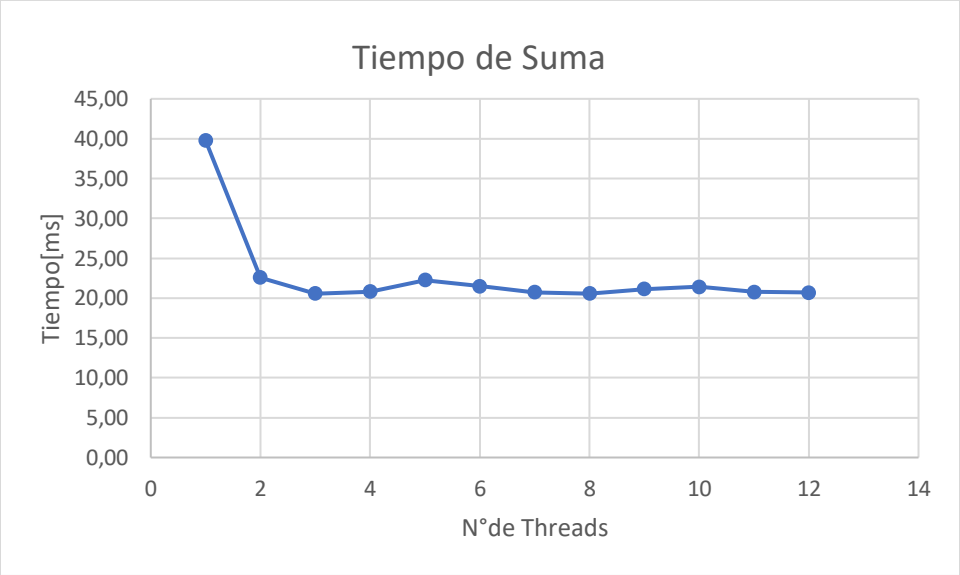

Figura 8:Gráfico N° Threads vs tiempo de ejecucion

Luego de analizar el grafico se puede comprender que luego de 3 threads este ya no se vuelve tan eficiente.

Para la segunda parte de los resultados se hará las mismas pruebas en el sistema invitado como se puede ver en la Figura *9* la nueva comparación pero en el sistema invitado.

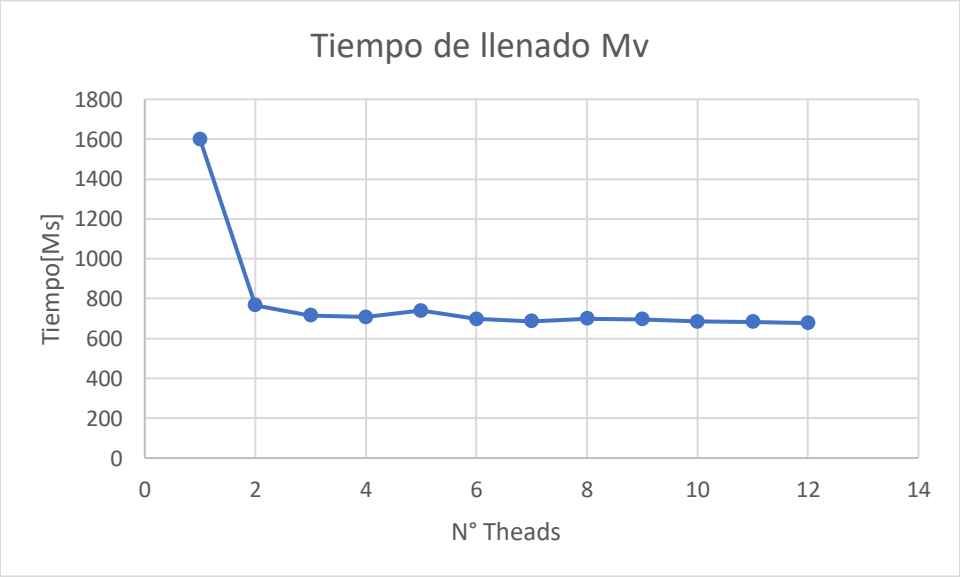

Figura 9:Gráfico N°Threads vs Tiempo de ejecución Mv

Se puede concluir que al igual que en el sistema host la utilidad de los threads es eficiente hasta cierto punto, que en este caso sería de 2 threads. Además no es tanto la eficiencia en el sistema invitado ya que este tiene menos recursos que el sistema host.

Para el caso de la suma en el sistema invitado se puede ver la *Figura 10*

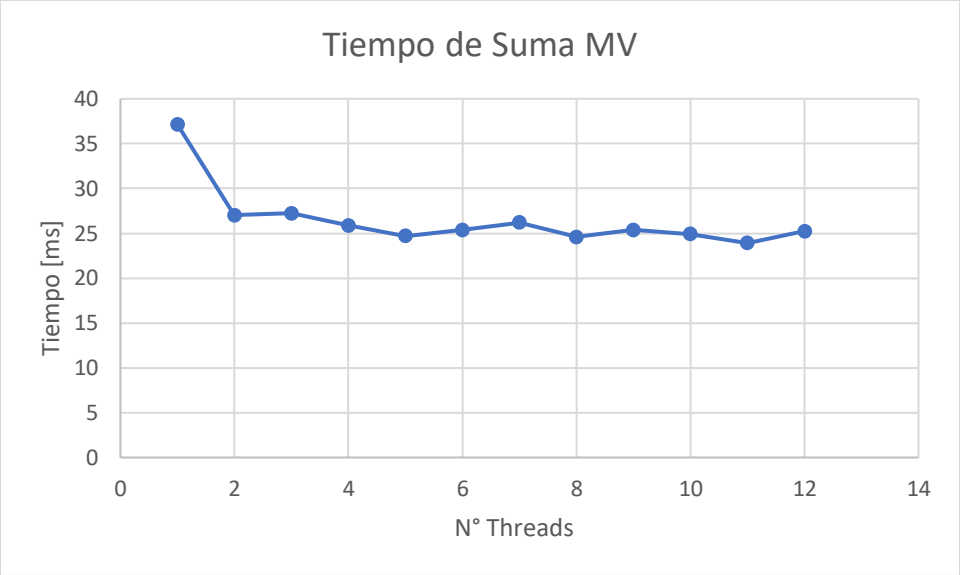

Figura 10: Gráfico N°Threads vs Tiempo de ejecución Mv

En el caso de la suma en el sistema virtual en comparacion al sistema host, se puede ver que es similar la forma de eficiencia respecto a cuando los threads incrementan, pero también se observa que luego del segundo thread no se vuelve tan eficiente el uso de estos.

6 Conclusión

Al termino de este taller se cumple con el objetivo de realizar los dos modulos de llenado y suma paralelamente, y al haber analizado los resultados, se puede concluir que los threads hasta cierto punto pueden llegar a ser más efectivos de forma paralela respecto a lo secuencial, pero no necesariamente en todos los casos ya que la forma secuencial puede ser mejor para otros tipos de soluciones y ser mejores que los threads, por lo que para el taller en si solo se requiere usar entre 2-4 threads para una eficiencia satisfactoria ya que si se usan más no es notorio el cambio. Además se pudo comprender mejor que el uso de threads no necesariamente es mejor que usar la forma secuencial.