Documentación de diseño de taller 2

Taller de Sistemas Operativos

Escuela de Ingeniería Informática

Ignacio López López

Ignacio.lopezl@alumnos.uv.cl

Resumen: *En este artículo se presentará el diseño e implementación para la solución de un problema de procesamiento de multihilos presentado por la asignatura de taller de sistemas operativos, este contiene una actividad planteada, para lo cual a continuación se expondrá el contexto del problema en el que se encuentra inmerso las actividades planteadas, exponiendo el objetivo a realizar y como se debe hacer, posteriormente se presentar la solución mediante un diagrama alto nivel para cada módulo, y así implementar el diseño en el lenguaje C++, posteriormente se realizaran pruebas de desempeño para obtener resultados y saber si la paralelización es mejor que una ejecución secuencial.*

1. Introducción

Como se ha mencionado anteriormente este articulo nace de la necesidad de presentar un diseño para el problema planteado por el "taller 02" de la asignatura taller de sistemas operativos, este consiste crear un programa que este compuesto por dos módulos, el primero debe rellenar un arreglo dinámico con un rango de números random, y el segundo modulo debe sumar dichos números, estos dos módulos se deben paralelizar, esto se diseñara a través de hilos, esto conceptos se detallan más en el marco teórico que se presenta a continuación.

1.1. Marco teórico.

Esta actividad abarca un nuevo ámbito dentro de los conceptos de sistemas operativos, donde se es necesario conocer los siguientes conceptos para entender los diseños que serán planteado posteriormente: programa es conjunto de instrucciones que se ejecutan secuencialmente para realizar una tarea específica, Proceso se define básicamente como un programa en ejecución en donde las líneas de código se ejecutan de manera dinámica [1].

La parte fundamental de este proceso viene en la estructura dentro de este mismo llamado thread (hilos) estos se pueden definir como secuencia de control dentro de un proceso, este ejecuta las instrucciones de forma independiente [1].

La programación multithreading permite la ocurrencia simultánea de varios flujos de control a través de la distribución de distintos subprocesos/actividades a más de un hilo. Cada uno de ellos puede programarse independientemente y realizar un trabajo, distinto, idéntico o complementario, a otros flujos paralelos, es decir, un hilo será un proceso independiente, que se podrá ejecutar paralela o concurrentemente con otros procesos [1]. Este thread podrá trabajar sobre datos distintos o compartidos, y podrá en un momento dado pararse, reiniciarse, sincronizarse o esperar a otros.

2. Diseño.

2.1. Diseño general

En este apartado se busca tener una vista general de como funcionara el diseño, en cuanto a las salidas y entras que tendrán cada modulo para el funcionamiento del mismo bajo los requisitos impuestos en el taller 02, esto se puede observar mediante el diagrama 1.

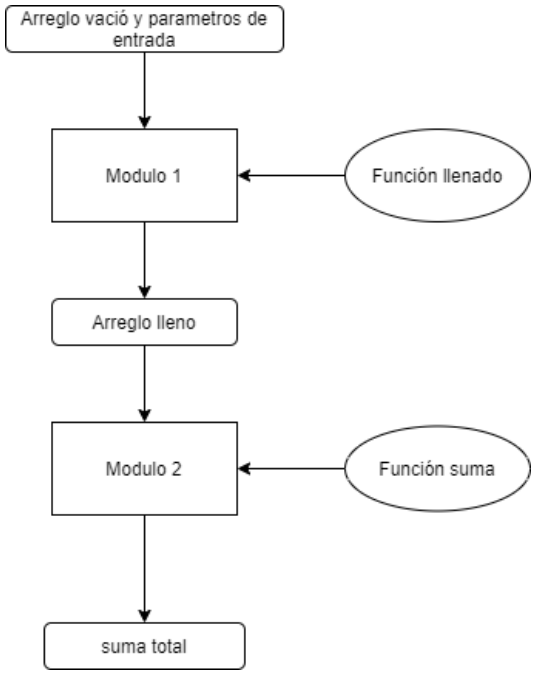

Diagrama 1 – diagrama general

Como se puede observar en el diagrama 1 muestra un panorama general de los módulos a implementar, de esto se puede extraer que su funcionalidad no está ligada entre sí, dado que funcionan como cajas negras, abstrayéndose de lo que sucede alrededor, es importante destacar que los paramentos de entrada que se pueden observar en el modulo 1, son los que se exigen a la hora de ejecutar el código, estos son el largo del arreglo, el numero de hilos y el rango de números aleatorios que se llenara el arreglo, eso se detalla más en el readme.

2.2.Modulo uno.

En este módulo como se ha dicho con anterioridad se es necesario rellenar un arreglo dinámico de forma paralela, para esto se requería la utilización de hilos, estos hilos se encargaran mejorar la optimización del programa en función del tiempo dado que, la tarea principal se dividirán en subtareas encargadas a cada hilo (diagrama 2), es decir por cada hilo generado se dividirá el arreglo, y se rellenara con numero random, para posteriormente “unir” el arreglo al finalizar cada hilo, este proceso se realizara mediante la función random (random function) esto se detalla en diagrama 3.

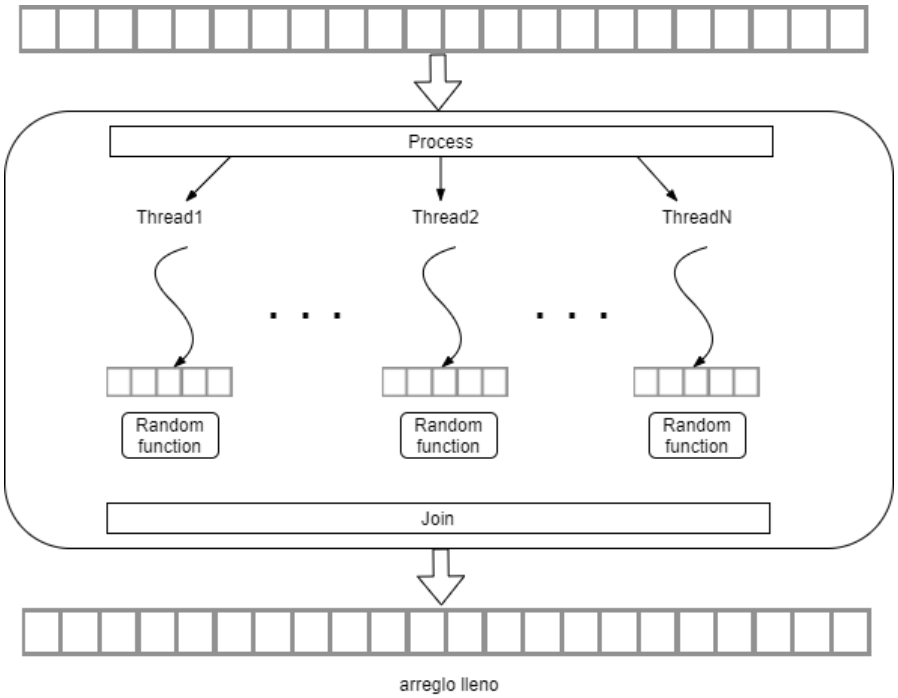

Diagrama 2 – separación modulo uno.

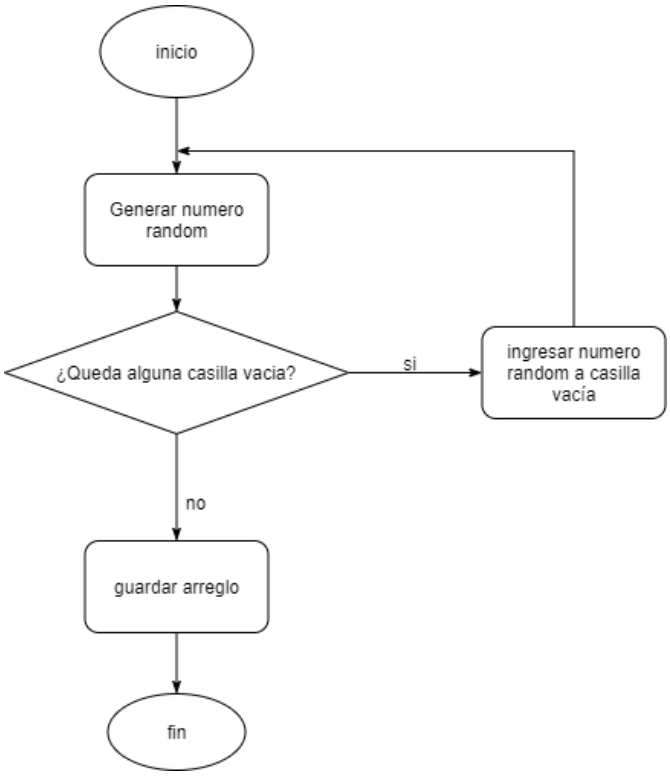

Diagrama 3 – función random.

2.3.Modulo dos.

En este módulo como se ha dicho con anterioridad se es necesario realizar una suma de los valores ingresados en

el arreglo, y paralelizar este proceso. esto se realizar con la misma estructura que en el modulo 2, es decir hilos (diagrama 4), a cada hilo se le otorgara un segmento del arreglo y se sumaran solo los valores de ese segmento para posteriormente sumarlo como se detalla en el diagrama con la función suma (suma function) esto se detalla en el diagrama 5.

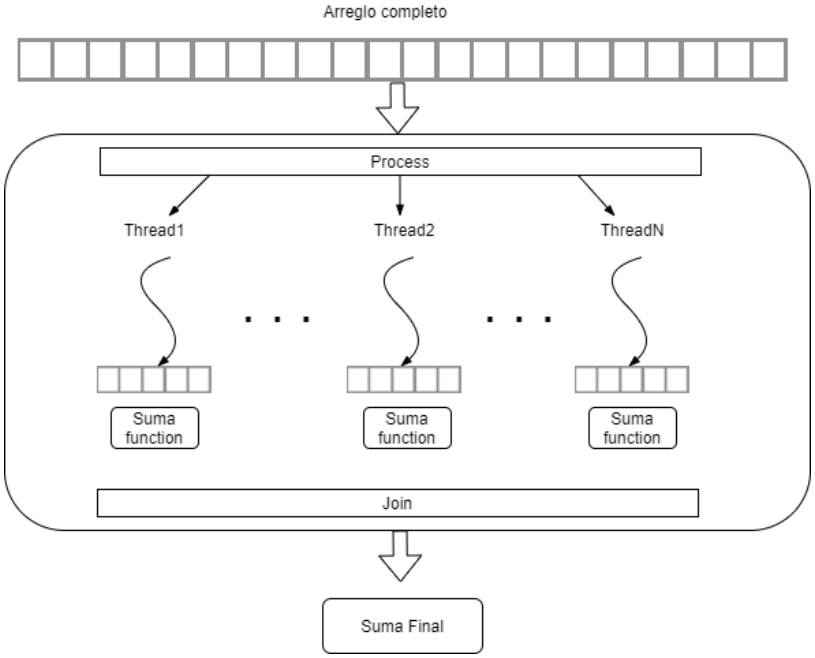

Diagrama 4 – separación modulo dos.

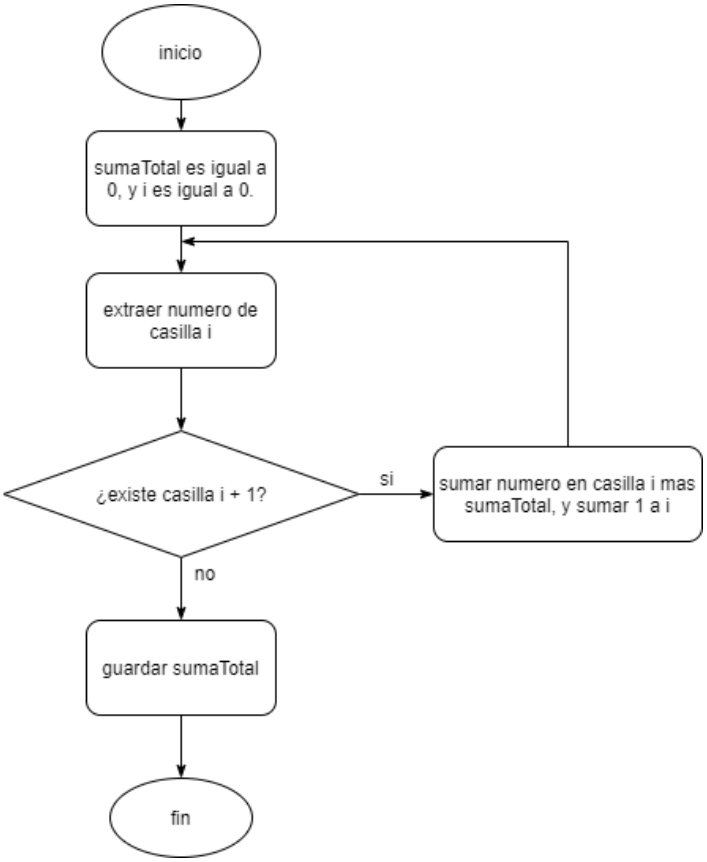

Diagrama 5 – función suma.

3. Resultados.

Una vez habiendo implementado el diseño de la solución del problema planteado, es necesario evaluar la funcionalidad y eficiencia del diseño y/o código por medio de pruebas, estas se hicieron dentro de un espacio virtualizado de Ubuntu Server, como ya se mencionó anteriormente a la hora de ejecutar el código se deben asignar una serie de parámetros. Para las pruebas se trabajó con los siguientes parámetros de entrada: un largo del arreglo de cien millones con un rango de llenado de uno a diez millones, de modo que el parámetro que varia es la cantidad de hilos con un rango de uno diez y seis hilos en 5 instancias diferentes para luego promediarlas y así obtener un dato mas fidedigno descartando datos atípicos, estos resultados fuero insertados en el grafico 1 y grafico 2 que se muestra a continuación, cabe destacar que el eje X que especifica los hilos utilizados solo afecta al llenado/suma paralelo dentro de los gráficos, la línea serial que se muestra en los gráficos, se utilizan más como una representación para ver el margen de mejora que existe en ambos.

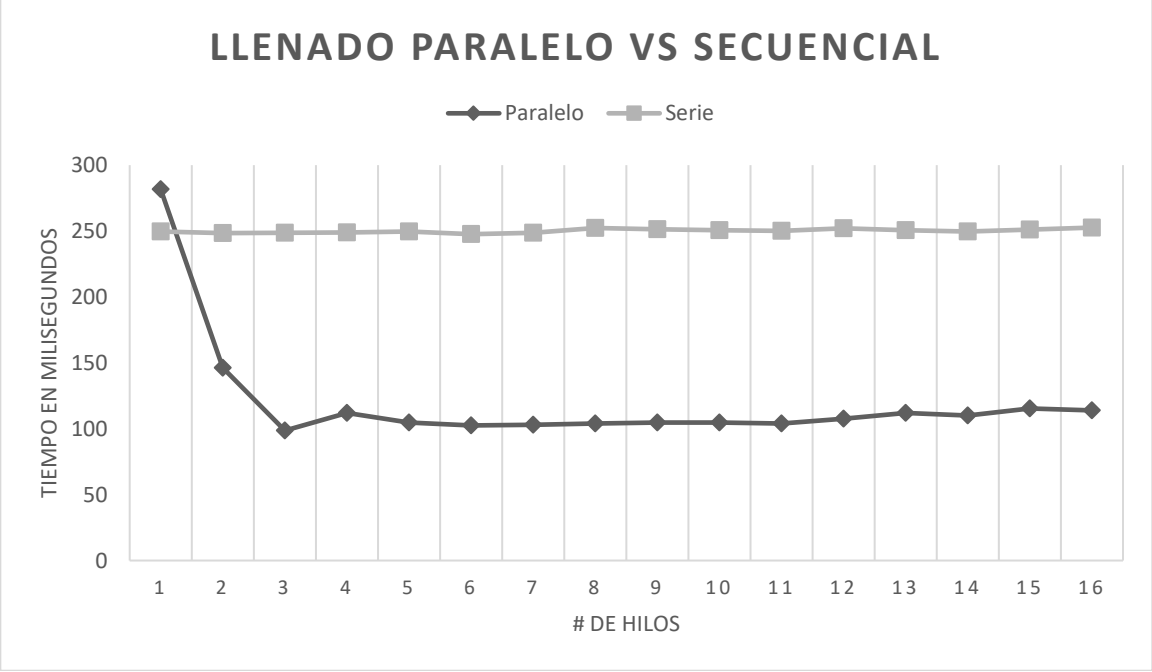

Gráfico 1 – Llenado paralelo vs secuencial

Como se puede observa en el grafico 1 este describe el tiempo de llenado de manera secuencial y paralela, de esto se puede desprender las siguientes observaciones, siendo la primera de estas en cuanto al llenado secuencial en primera instancia es peor el tiempo de ejecución secuencial, una observación de por que sucede esto, es dado a la complejidad a la hora de utilización de los hilos a diferencia de su contraparte secuencial, pero a partir del segundo hilo la mejor es considerable aproximadamente el doble en cuanto a tiempo de

ejecución, posteriormente en una tercera instancia es decir al usar tres hilos se encuentra el mejor tiempo, para esta implementación del problema, dado que posteriormente al aumentar los hilos se mantiene constante con un máximo de ciento quince coma 3, en cuanto al tiempo de ejecución, de esta observación se puede afirmar que la mejora al ocupar hilo está limitada en cuanto a el hardware donde se ejecuta, dado que existe una cierta cantidad de hilos que se puede pueden ocupar en un Core a la vez, dado que en la maquina virtual tiene 2 core se desprende que la mejora máxima es de 4 hilos en esta instancia, esto se puede observar en el gráfico.

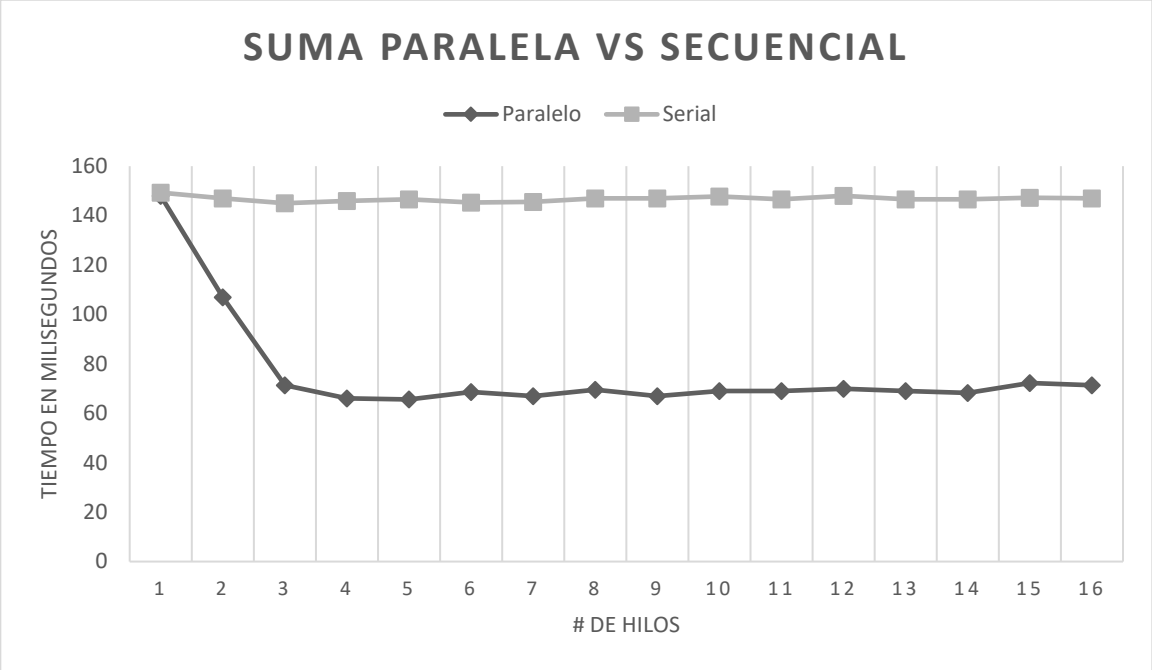

Gráfico 2 – Suma Paralela vs Secuencial

De igual forma que en el gráfico 1 en este grafico se puede observar la evolución del tiempo de ejecución en paralelo como en secuencial pero orientado a la suma de los datos del arreglo, una diferencia apreciable a siempre vista, es que ambos secuencial como paralelo, tiene un tiempo similar cuando existe 1 hilo en ejecución, pero de la misma forma que en el gráfico pasado existe una mejora considerable a partir del segundo hilo en la suma paralela, aproximadamente del cincuenta porciento mejor. Y observando los otros puntos del gráfico a media aumentan los hilos, el tiempo se estabiliza a partir del hilo 4, afirmando la observación extraída partir del grafico anterior en cuanto a la limitación del hardware para la paralelización, dado que no aumentan pese a aumentar los hilos.

4. Conclusiones.

En este reporte se presentó el diseño e implementación del taller 02 del ramo de taller de sistemas operativos, donde el principal objetivo es entender cómo funciona los hilos en un sistema operativo especifico y evaluar si existe una mejora al compáralo con su contraparte serial. Para eso fue necesario presentar el marco teórico en el

cual se presenta el concepto principal en cual se desarrolla el diseño, es decir hilos, mediante este concepto se pudo presentar el diseño de los dos módulos y su posterior implementación en el lenguaje C++, al haber analizado los resultados de las pruebas hechas a la implementación se puedo concluir que cuando se implementan hilos para la paralelización de procesos puede existir una mejora contundente en cuanto al tiempo de ejecución del mismo, pero esto depende de dos factores importantes dentro paralelización el primero es el hardware por el cual se ejecutara el código dado que existen limitaciones físicas en cuanto a los hilos que se pueden ejecutar a la vez en un Core, y el segundo factor es la complejidad del código a ejecutar, esto se pudo observar al hacer las pruebas y asignar un rango bajo al arreglo y/o al número aleatorio, dado que el tiempo que se obtenía era el mismo al hacer la paralelización que al hacerlo de forma secuencial.

5. Referencias.

[1] sistemas operativos aspectos internos y principios de diseño, william stalling,ppt 79-136.