Documentación de diseño de taller 3

Taller de Sistemas Operativos

Escuela de Ingeniería Informática

Ignacio López López

Ignacio.lopezl@alumnos.uv.cl

Resumen: *En este artículo se presentará el diseño para un problema de procesamiento de multihilos, presentado por la asignatura de taller de sistemas operativos, donde expondrá el diseño del problema, y posterior implementación para la paralelización mediante OpenMP. Posteriormente se presentara la actividad planteada, para lo cual a 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++ usando OpenMP, posteriormente se realizaran pruebas de desempeño para obtener resultados y saber si la paralelización es mejor que una ejecución secuencial y hacer una comparación de rendimiento con el taller previo.*

1. Introducción

Como se ha mencionado anteriormente este articulo nace de la necesidad de presentar un diseño para el problema planteado por el "taller 03" 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, estos hilos se implementaran mediante la api de programación paralela OpenMP, 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, el Programa: se define como un conjunto de instrucciones que se ejecutan secuencialmente para realizar una tarea específica, este se ve ligado al concepto de proceso, este 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.

OpenMP (Open multiprocessing) : este se puede definir como un modelo de programación paralelo basado en directrices, este se proporciona un interfaz de programación de aplicaciones (API) que permite la programación multiprocesamiento de memoria compartida.

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 03, esto se puede observar mediante el diagrama 1.

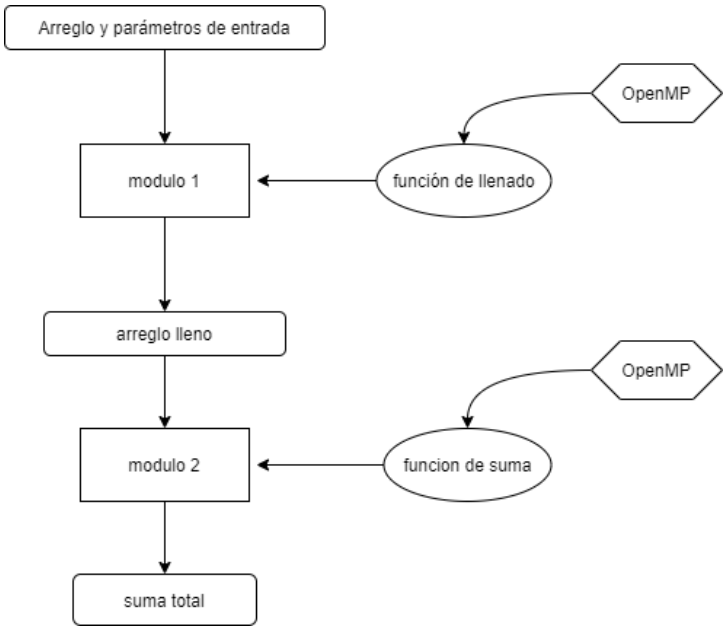

Diagrama 1 – diagrama general

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 llenar un arreglo dinámico de forma paralela, para esto se requería la utilización de hilos, estos 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 1), es decir por cada hilo generado se dividirá el arreglo, y se llenara con numero random, para posteriormente “unir” el arreglo al finalizar cada hilo, este proceso se realizara mediante la función “llenado OMP” en esta se implementara la api OpenMP para su paralelización.

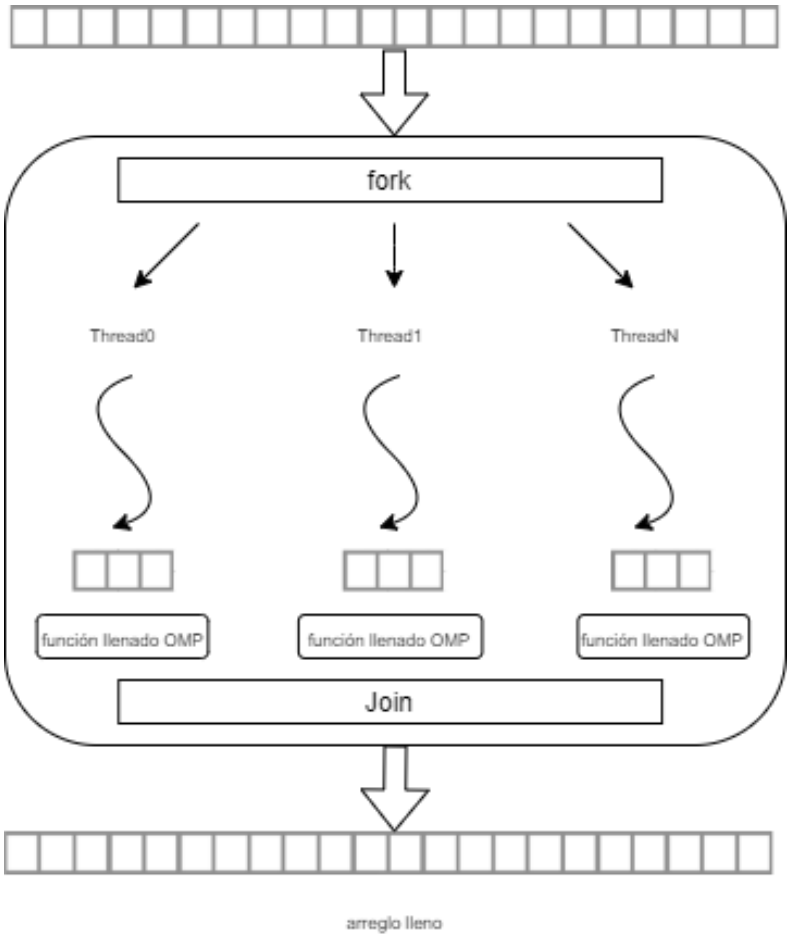

Diagrama 1 – separación modulo uno.

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 módulo 1, es decir hilos (diagrama 2), a cada hilo se le otorgara un segmento del arreglo y se sumaran solo los valores de ese segmento para posteriormente sumarlo esta acción se hace mediante la función “suma OMP”, en donde se implementa OpenMP para su paralelización.

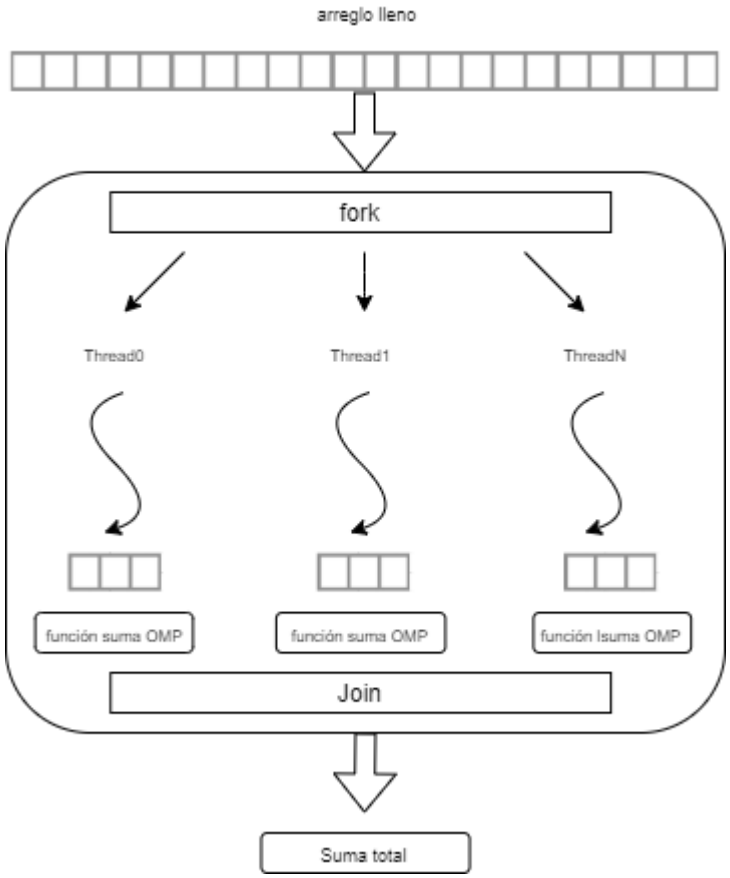

Diagrama 2 – separación modulo dos.

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 setenta millones con un rango de llenado de uno a cien mil, 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, en estos gráficos se muestran dos implementaciones de paralelismo siendo la correspondiente a este taller OpenMP, y al correspondiente al taller dos, esto con la finalidad de hacer una comparación de rendimiento, cabe destacar que los dos gráficos se han probado con las mismas entradas es decir cantidad de datos por lo cual la salidas y/o resultados se pueden comparar y se mostraran 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(para todos los gráficos).

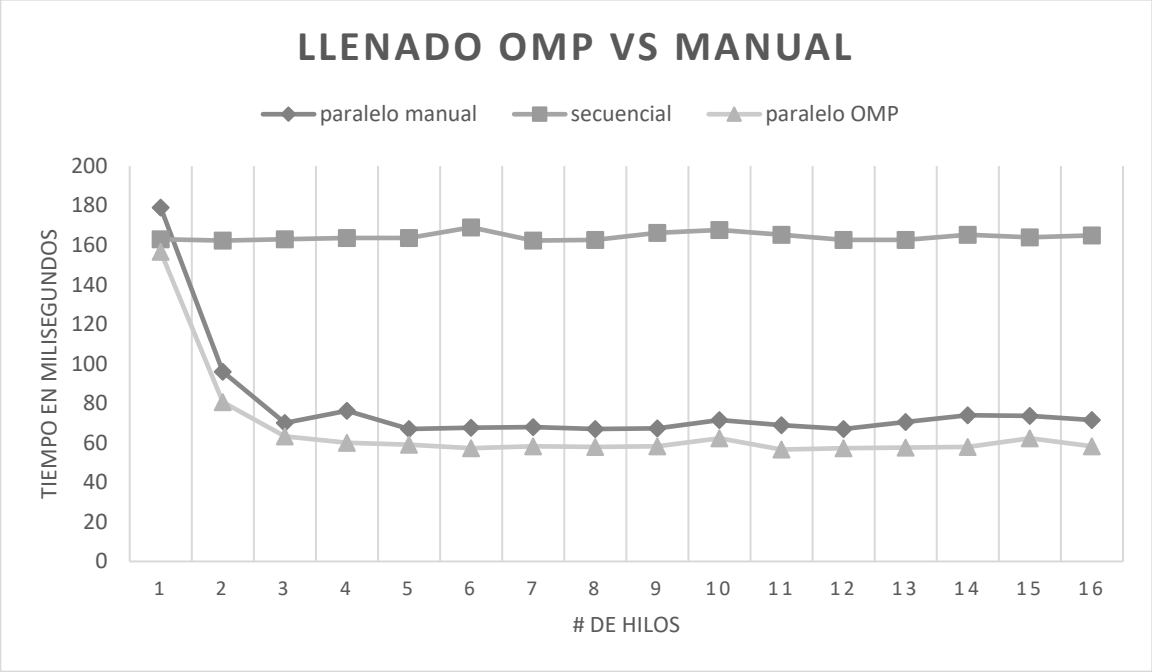

Gráfico 1 – llenado OMP paralelo vs secuencial

Como se puede observar en el grafico 1 este describe el tiempo de llenado de manera secuencial que sirve como valor de referencia para observar cuan grande es la mejor y se hay una, además existe una paralelización OMP haciendo referencia al taller tres y finalmente una paralelización manual haciendo referencia al taller dos, por temas de terminologías a la implementación de taller dos se le llamara manual. En primera instancia del grafico 1 se puede desprender que en el hilo uno tanto para la paralelización OMP como manual no hay una diferencia significativa, y esto debe ser así dado que trabajaría de la misma manera que el secuencial, a partir del hilo dos se ve una mejora significativa de prácticamente del doble en comparación con el hilo uno, pero observando que la implementación OMP es menor en todo momento, esto se da para este caso de implementación manual, y posteriormente se puede observar que a partir del hilo cuatro y/o cinco en ambas implementaciones se estabiliza la mejora temporal, pero de igual forma siendo más eficiente la implementación OMP, de esta información se puede decir que al paralelizar puede existir una mejora hasta cierto punto limitado por los hilos, para ser mas precisos estos son limitados por el hardware del dispositivo, en este caso se puede observar que después de 4 hilos no existe una mejor sino una variación del rango, esto se confirma al saber que la máquina virtual en la que fueron efectuado las pruebas tiene dual core con dos hilos por core.

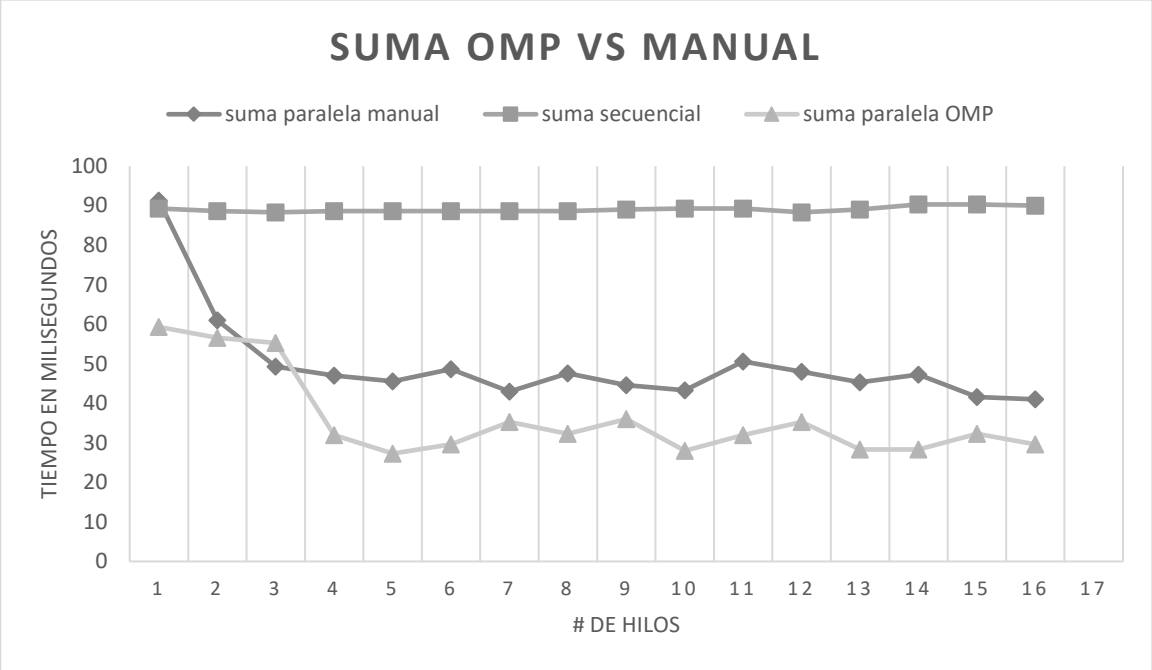

Gráfico 2 – suma OMP Paralelo vs Secuencial.

De igual forma que en el gráfico 1, pero en el ámbito de la suma de los arreglos en el grafico 2 se presenta un análisis de los datos tanto de la paralelización manual como la OMP, de esto se puede decir, que en primera instancia en el hilo uno, la implementación OMP es considerablemente mejor a la implementación secuencial e incluso a la serial, esto sucede dado que la implementación secuencial se hizo de forma manual con funciones, por lo que no fue una implementación eficiente, a diferencia de como lo implementa OMP con un hilo, posteriormente al subir la cantidad de hilos a dos, en ambos casos existe una mejor, pero aun así ya existía una diferencia considerable entre la implementación OMP y manual, al subir a tres hilos , vuelve a bajar los tiempos en ambos casos pero de forma más drástica en la implementación OMP, y posteriormente al cuarto hilo se estabilizan ambos, pero como se ha dicho anteriormente OMP fue implementado de mejor manera, por lo que los tiempos de estabilidad de las dos implementaciones varían entre cincuenta y treinta milisegundos, además de igual forma que en el grafico 1 existe una mejor hasta al hilo cuatro por lo que se confirma lo planteado con anterioridad.

4. Conclusiones.

En este reporte se presentó el diseño e implementación del taller 03 del ramo de taller de sistemas operativos, donde el principal objetivo es entender cómo funciona los hilos para una implementación de OpenMP en un sistema operativo especifico y evaluar si existe una mejora al compáralo con su contraparte serial, además se hizo una comparación de los resultados de los gráficos obtenidos en el taller dos, para tener una referencia temporal, a la hora de la utilización de hilos con OMP, de esto se pudo concluir en primera instancia tanto para el grafico 1

como el grafico 2 existe una mejora considerable al utilizar hilos de forma “manual”, pero al compararla con la versión OMP existe una mejora considerable en el caso de la suma aproximadamente de cincuenta a treinta milisegundos, en el caso de el llenado se puede observar una mejora en la implementación OMP pero no tan grande como en el caso de la suma, y como se dijo en los resultados existe una mejora hasta cierta cantidad de hilos dependiendo del hardware de dispositivo en este caso cuatro hilos por eso existe una mejora hasta ese punto, también se puede comentar al realizar las pruebas que la implementación OMP en la suma, los resultados posterior a cuatro hilos son mas variables entre un rango de 20 milisegundos, Para finalizar un también se puede agregar que la paralelización depende de la complejidad de lo que se quiera paralelizar dado que, para este caso con valores de arreglo inferiores a cien mil, la diferencia temporal entre paralelizado y secuencial es nula.

5. Referencias.

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