INFORME DE TRABAJO FINAL

MPI y OpenMP

ALUMNOS:
Arreguez, Germán Rodrigo.
Ventura, Gino.

INSTITUCIÓN: Universidad Blas Pascal

CARRERA: Ingeniería Informática

ASIGNATURA: Programación Concurrente

DOCENTE: Maximiliano A. Eschoyez

FECHA DE ENTREGA: 08/02/2023

UBP UNIVERSIDAD Blas Pascal

CONSIGNA

RESUMEN

El objetivo de este trabajo final es realizar un software para obtener números primos utilizando técnicas de programación paralela. Se deberá implementar el software utilizando la Interfaz de paso de mensajes – MPI y la interfaz de hilos paralelos OpenMP. Además, se debe realizar una serie de mediciones de tiempo de ejecución para comparar el programa secuencial, el comportamiento del programa paralelizado con múltiples hilos y con múltiples procesos.

CONSIGNA PRINCIPAL

En este trabajo se deben desarrollar diversas versiones de un programa, tanto para su implementación con la biblioteca OpenMP y con MPI, para realizar la búsqueda de los primeros n números primos. La cantidad la indica el usuario como argumento al programa.

El objetivo es dividir el espacio de búsqueda entre todos los procesos/hilos que se vayan a ejecutar (no considerar los valores pares porque no son primos, excepto el 2). De esta forma, la cantidad de procesos/hilos dependerá de los recursos disponibles o la configuración de inicio, y cada proceso buscará sólo en el rango que le corresponde.

La primera versión del programa debe ejecutar la búsqueda de los números primos de forma secuencial sobre los números candidato.

Implementar una versión en la que todos se sincronizan (barrera) antes de avanzar sobre el siguiente número y otra donde trabajan de forma independiente. Comparar los tiempos de ejecución.

La segunda versión debe proveer algún mecanismo para que los hilos/procesos tomen el próximo valor a verificar. De esta forma, el proceso o hilo no debería estar ocioso esperando a que otros terminen. Sin embargo, se necesita sincronización o comunicación (según corresponda) para poder avanzar.

Compare los tiempos de ejecución contra los observados para la primera versión.

Finalmente, busque en Internet algoritmos eficientes, implemente el mejor en forma secuencial y en los programas anteriores. Compare los resultados de tiempo entre todas las versiones de los programas utilizados.

PRESENTACIÓN DEL TRABAJO FINAL

GRUPOS DE TRABAJO

El trabajo se podrá presentar en forma individual o en grupo de dos integrantes, prefiriéndose la modalidad grupal.

CÓDIGO FUENTE

El código fuente y la versión digital del informe en PDF deben entregarse a través del enlace correspondiente en la plataforma miUBP del examen final. En dicho enlace se deberá subir un único archivo en formato ZIP conteniendo todos los códigos fuente que se requieran para la realización del trabajo final.

INFORME ESCRITO

Se entregará al profesor un informe escrito en versión digital donde se debe describir la problemática abordada en el trabajo final, el desarrollo de la solución propuesta, los resultados de las mediciones de tiempo y una conclusión. El texto deberá ser conciso y con descripciones apropiadas. No se debe incluir el código fuente, si no los textos necesarios para realizar las explicaciones pertinentes. El formato de entrega es PDF.

EXPLICACIÓN DE LAS VERSIONES

- ***Primos_Secuencial.c:*** esta versión muestra de forma secuencial como calcular los números primos con un algoritmo muy eficiente. Básicamente hicimos una función de tipo ***bool*** que recibe como parámetro un número entero, que es el número el cual se va a verificar si es primo o no para posteriormente imprimirlo en consola si lo es. Este parámetro entero se incrementa mediante un ***for***.

- ***Primos_OMP_independiente.c:*** en la primera versión utilizando la librería OpenMP para calcular los números primos utilizamos el mismo algoritmo que en la versión secuencial. Pero esta vez el ***for*** se divide entre todos los hilos disponibles. Para ello utilizamos las directivas preprocesador ***#pragma omp parallel*** y luego ***#pragma omp for***.

- ***Primos_OMP_barrera.c:*** en la segunda versión de OpenMP pasamos de utilizar un loop ***for*** a utilizar un loop ***while***. Utilizamos de vuelta la directiva ***#pragma omp parallel*** pero en este caso el número a verificar si es primo se convierte en una variable compartida entre todos los hilos. Luego utilizamos la directiva ***#pragma omp critical*** para que solo un hilo a la vez pueda verificar si el valor contenido por la variable compartida es primo e imprimirlo en consola si lo es.

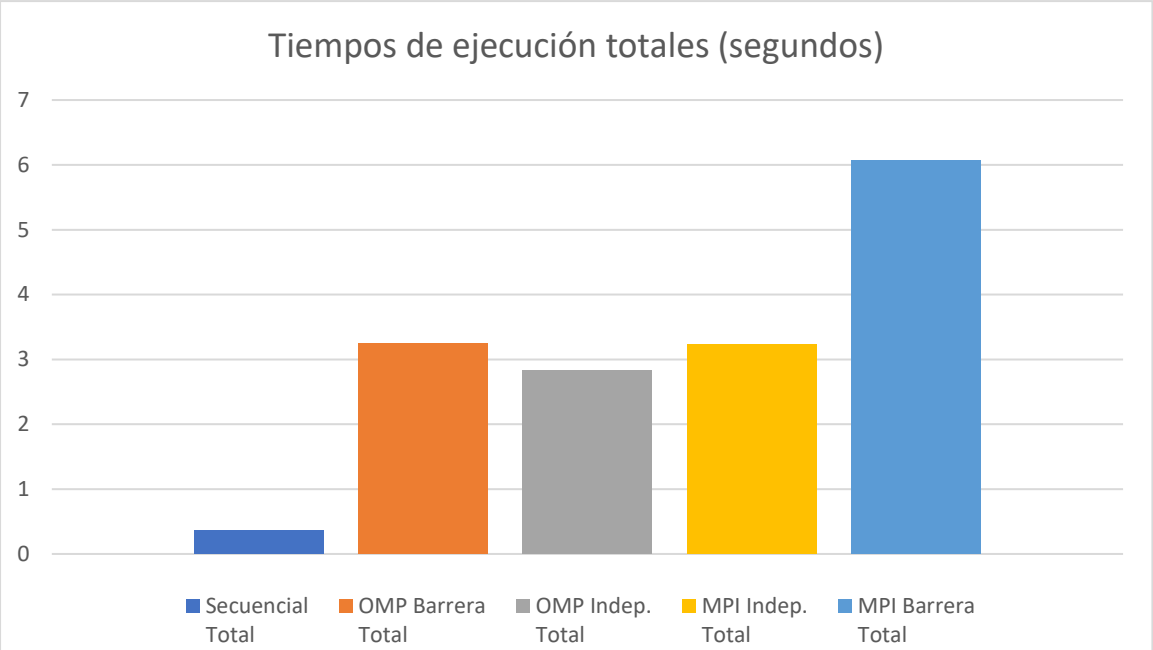
- ***Primos_MPI_independiente.c:*** en la primera versión utilizando la librería MPI utilizamos nuevamente el algoritmo eficiente de la versión secuencial. Para sincronizar los distintos procesos inicializamos el entorno de ejecución MPI con ***MPI_Init()*** y a la hora de ejecutar especificamos el numero de procesos, en este caso, 8 procesos. El proceso 0 se encarga de inicializar y finalizar el reloj para contabilizar el tiempo, además de calcular los números primos en su rango correspondiente. Los demás calculan el resto e imprimen en consola los números primos encontrados. Se utiliza la directiva ***MPI_Barrier()*** para que cuando todos los procesos estén listos, comiencen y finalicen al mismo tiempo. Al terminar los cálculos se cierra el entorno con ***MPI_Finalize().***

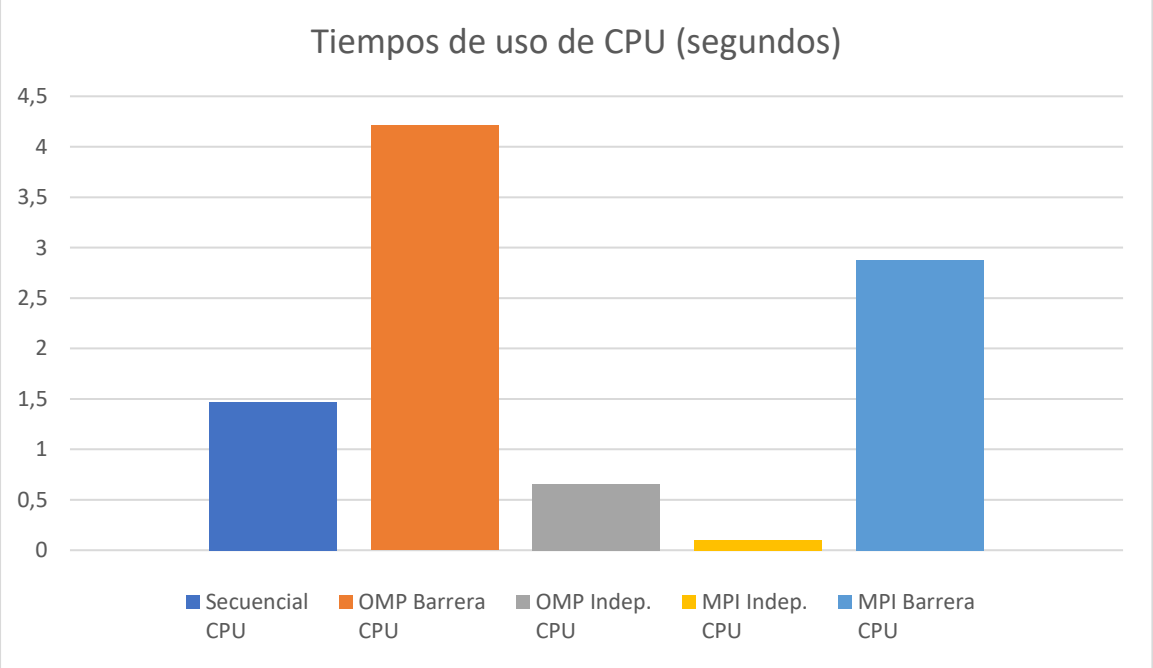
- ***Primos_MPI_barrera.c:*** en la segunda versión utilizando MPI, no pudimos lograr que se implemente correctamente la barrera, calcula los números primos pero el funcionamiento no es el esperado.

PRUEBAS DE EJECUCIÓN:

Para las pruebas de todas las versiones utilizamos una cantidad de números de 1.000.000, en el cual se encuentran 78.498 números primos. Se hicieron un total de 3 pruebas en cada versión y estos son los resultados:

Version / N° Prueba	Secuencial Total	Secuencial CPU	OMP Barrera Total	OMP Barrera CPU	OMP Indep. Total	OMP Indep. CPU	MPI Indep. Total	MPI Indep. CPU	MPI Barrera Total	MPI Barrera CPU
1	0,363171s	1.494917s	3.187624s	4.247699s	2.768035s	0.676459s	3.616453s	0.098332s	6.052248s	2.646380s
2	0,348410s	1.383967s	3.291477s	4.188370s	3.065789s	0.652873s	3.378241s	0.099941s	6.313566s	3.559666s
3	0,378472s	1.555076s	3.241105s	4.197713s	2.656038s	0.646532s	2.717110s	0.114074s	5.821322s	2.424085s
Promedio	0,363351s	1,46948s	3,24006s	4,21126s	2,82995s	0,65862s	3,23726s	0,10411s	6,06237s	2,87671s





DEFINICIONES:

- ***Tiempo de ejecución:*** es el intervalo de tiempo en el que un programa de computadora se ejecuta en un sistema operativo según lo medido por un reloj ordinario. Este tiempo se inicia con la puesta en memoria principal del programa, por lo que el sistema operativo comienza a ejecutar sus instrucciones. El intervalo finaliza en el momento en que este envía al sistema operativo la señal de terminación.

- ***Tiempo de uso de CPU:*** es la cantidad de tiempo en que la unidad central de proceso fue usada para procesar las instrucciones de un programa de computadora. El tiempo CPU es a menudo medido en impulsos del reloj o como un porcentaje de la capacidad del CPU.

CONCLUSIÓN

Como se puede observar en el grafico el tiempo de ejecución de la versión secuencial es menor, a las versiones utilizando OpenMP y MPI. Esto se debe a que al ser un calculo bastante sencillo sumado al algoritmo eficiente el procesador lo puede realizar con hilo/procesador. Si agregamos hilos/procesos en paralelo el tiempo de ejecución es mayor ya que el procesador debe asignarle las distintas tareas a cada uno y recolectar los distintos resultados.