Práctica 1: Suma Paralela

Fecha de entrega: Sábado 20 de Marzo del 2021.

Profesora: María de Luz Gasca Soto
Ayudante laboratorio: Muñiz Patiño, Andrea Fernanda

1. Objetivo

Implementar la suma de los primeros 1 000 000 000 números naturales, eventualmente realizar un análisis de como afecta el uso de los procesadores en el rendimiento del algoritmo.

2. Actividad para realizar en laboratorio

Suma de los primeros *1 000 000 000* números naturales de forma paralela, además de medir el tiempo de ejecución para realizar la suma.

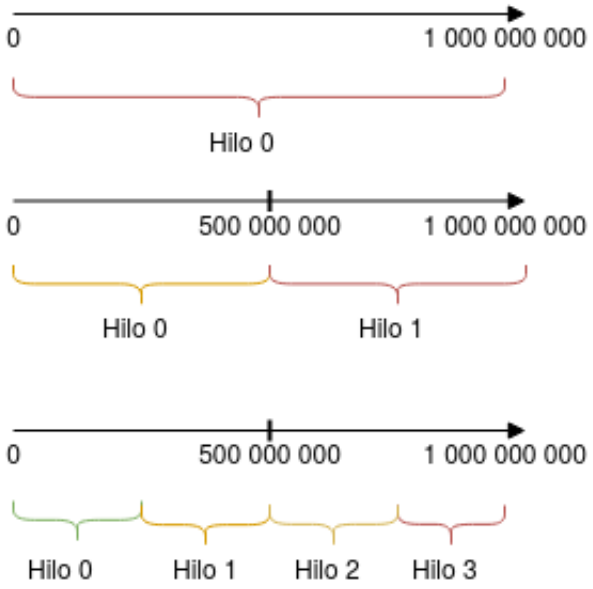


Figura 1: Esquema de suma paralela de los primeros 1 000 000 000 números

Como se muestra en la Figura 1, queremos que cada hilo haga la suma en paralelo de un intervalo $[n, m]$ el cual va a estar determinado por el *identificador* del hilo que está trabajando. Por ello es necesario que cada hilo tenga acceso a una variable común llamada *sumaTotal*.

Un punto importante ha recordar de los algoritmos paralelizados es que el desempeño computacional dependerá de los parámetros de entrada y el número de procesadores.

3. Parámetros y tiempo en C

Queremos que al momento de ejecutar nuestro algoritmo, este reciba como parámetro la cantidad de hilos que llevarán acabo la tarea. Esto con la finalidad de realizar de la siguiente manera la ejecución de nuestra actividad.

```
umm@usuario:~$ ./a.out   n
```

Donde **n** es el número de hilos que queremos pasar como parámetro.

Esto se logra con el siguiente código.

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

int main(int argc, char** argv){

//verifica el numero de hilos que se paso como parametro
        if (argc < 2){
                printf("por favor especifique el numero de hilos\n");
                exit(1);
        }

}
```

Recuerda no aceptar un número de hilos inválido. Por ejemplo no tiene sentido tener $-n$ hilos, donde n es un número entero.

Para obtener el tiempo de ejecución del programa podemos hacerlo con *struct* timeval, así como se muestra en el siguiente código.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <omp.h>
/* Codigo para obtener el tiempo que toma la ejecucion de un programa
*       Struct timeval
*/
int main(int argc, char** argv){
        int suma;
        struct timeval inicio, fin;//nos permiten medir el tiempo de ejecucion

        gettimeofday(&inicio, NULL);//guarda el tiempo al inicio del programa
        int tiempo;

        for(int i =0; i < 10; i++){
                suma = suma +1;
        }

        gettimeofday(&fin, NULL); //guarda el tiempo al final del programa
        tiempo = (fin.tv_sec - inicio.tv_sec)* 1000000 + (fin.tv_usec - inicio.
            tv_usec);
        printf("suma: %i  \ntiempo de ejecucion: %i microsegundos\n",suma, tiempo);
            //imprime resultados
}
```

4. Actividad para el alumno

Con ayuda del código anterior, deberás implementar la suma de los primeros 1 000 000 000 de forma paralela, la cual será capaz de recibir desde terminal cuantos hilos creará el programa ya que deberás usar el identificador del hilo en la implementación de la suma, además deberás realizar un experimento práctico que consiste en ejecutar la solución paralela en varias ocasiones y con diferente número de hilos.

Ejecutarás en una terminal ***A*** la actividad anterior y obtendrás los tiempos de cada ejecución para presentar los resultados en una tabla con el siguiente formato:

No. de Hilos	Tiempo de ejecución T(p)	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5
1						
2						
4						
6						
20						
50						
100						

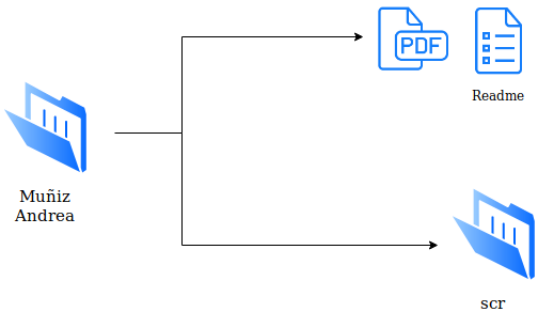
Cuadro 1: Tabla de tiempo de ejecución

La celda **Tiempo de Ejecución T(p)** es el promedio de las cinco pruebas para el número de hilos correspondiente para el renglón.

Después abrirás una terminal ***B*** donde compilarás y pondrás en ejecución el código que se te proporcionó para está práctica, llamado **cod01.c** y volverás a obtener los valores de *Tiempo de ejecución T(p)* y los reportaras de la misma manera que la tabla anterior, especificando que las pruebas se realizaron con el programa **cod01.c** en la Terminal **B** y el programa que suma los primeros 1 000 000 000 números en la Terminal ***A***.

4.1. Archivos a entregar

- Para la entrega de esta práctica deberás crear una carpeta con tu nombre y apellido, en ella pondrás un archivo *readme*, donde están las especificaciones sobre el programa y el ***PDF*** correspondiente, una sub-carpeta llamada *src* en la cual estarán los códigos fuente.



- Como parte de la práctica 1, deberás entregar un reporte con los siguientes requisitos y contestando a las preguntas:
  - Anexar las dos tablas de tiempos obtenidas.
  - Añadir en el reporte la información sobre tu equipo de cómputo que obtuviste en la Actividad 1, es decir, el número de núcleos y procesadores.
  - ¿Cómo afecta el número de procesadores y núcleos en el Cuadro 1 que presentaste?
  - Explicar cómo afecta el programa **cod01.c** al desempeño de tu algoritmo y cuál es el objetivo del código **cod01.c**.

4.2. Apoyo

Si usas latex para tus reportes aquí te proporciono el código de la tabla para el reporte, basta con que llenes la tabla con los valores obtenidos en tu experimento práctico.

```
\begin{table}[h]
    \begin{tabular}{ |c|c|c|c|c|c|c|}
     \hline
    No. de Hilos & Tiempo de ejecuci n T(p) & Prueba 1 & Prueba 2 & Prueba 3 & Prueba 4 & Prueba 5\\
    \hline
    1 & &&&& \\
    \hline
    2 & &&&&\\
    \hline
    6 & &&&&\\
    \hline
    8 & &&&&\\
    \hline
    20 & &&&&\\
    \hline
    50 & &&&&\\
    \hline
    100 & &&&&\\
    \hline
    \end{tabular}
        \caption{Tabla de tiempo de ejecuci n}
\label{table:1}
\end{table}
```

Otro recurso disponible es: https://www.tablesgenerator.com/