PROGRAMACIÓN AVANZADA

Ciclos en paralelo en OpenMP

EJERCICIO:

Usando algún elemento de sincronización, o ambos, modifique el código para calcular AUC de manera que se elimine el arreglo con las sumas, y todas las sumas se vayan acumulando en una sola variable, de tipo **float**.

TRABAJO COMPARTIDO DENTRO DE LOS CICLOS

```
int i;
#pragma omp parallel
{
#pragma omp for
   for (i = 0; i < numSteps;
   i++)
   {
      someFunction(i);
   }
}
```

```
int i;
#pragma omp parallel for
  for (int i = 0;   i <
  numSteps; i++)
  {
     someFunction(i);
  }
```

ENFOQUE BÁSICO PARA PARALELIZAR CICLOS

Encuentra los ciclos que consumen más tiempo.

Estudio individual, por ejemplo: Búsqueda de herramientas “profiling”.

Modifique los bucles para que las iteraciones se ejecuten de forma independiente.

Coloque la directiva OpenMP.

OPERADORES DE REDUCCIÓN

```
reduction (op : list)
```

Las variables en “list” deben ser compartidas dentro de la región paralela.

Adentro de parallel o el bloque de construcción de trabajo en paralelo:

- Se crea una copia PRIVADA de cada variable de la lista y se inicializa de acuerdo al “op”.
- Estas copias son actualizadas localmente por los hilos.
- Al final del bloque de construcción, las copias locales se combinan de acuerdo al “op” a un solo valor y se almacena en la variable COMPARTIDA original.

EJEMPLO:

```
double avg=0.0;
int i;
#pragma omp parallel for
reduction(+ : avg)
   for (i = 0; i < 100; i++)
   {
       avg += someArray[i];
   }
avg = avg / 100;
```

Con este código, el compilador genera:

Una copia “local” de **sum** para cada hilo

Todas las copias locales de **sum** se suman y se almacenan en una variable “global”

EJERCICIO 1:

Modifique su programa para calcular el AUC de la función $\sqrt{x}$ utilizando un ciclo en paralelo y un operador de reducción.

Operador	Valor Inicial
&	~0
|	0
&&	1
||	0

Operador	Valor Inicial
+	0
*	1
-	0
^	0

OTRAS OPERACIONES DE REDUCCIÓN

EJERCICIO 2:
SUMAS DE FIBONACCI

Realice un programa en C que dado un número entero, calcule la suma de los números de Fibonacci en paralelo hasta ese término. Debe realizar aparte una función que, dado un índice, devuelva el número de Fibonacci en ese índice.

Recuerde, la sucesión de Fibonacci es una sucesión infinita de números naturales que comienza con los números 1 y 1, y a partir de ellos, cada término se obtiene sumando los dos anteriores. Por ejemplo, hasta el décimo término sería:

1	**1**	**2**	**3**	**5**	**8**	**13**	**21**	**34**	**55**

¿CÓMO MEDIR EL TIEMPO DE EJECUCIÓN EN C?

Librería <time.h>

- define cuatro tipos de variables,
- dos macros,
- y varias funciones para manipular la fecha y la hora.

EJERCICIO 3:
Para el ejercicio anterior, compare los tiempos de calcular las sumas hasta el término 30, en paralelo, y sin paralelizar.

```
#include <stdio.h>
#include <time.h>
int main() {
    clock_t t_inicio, t_final;
    double segundos;
    t_inicio = clock();

    /*código al que medir el tiempo*/

    t_final = clock();
    segundos = (double)(t_final -
t_inicio) / CLOCKS_PER_SEC;
    printf("%f", segundos); }
```

MUCHAS GRACIAS