Ejercicios Tema 3 – Parte 2

Programación Paralela

Baldomero Imbernón Tudela

Grado en Informática

Ejercicio 1. El uso de `#pragma omp sections` es comúnmente utilizado a la hora de ejecutar dos funciones o más funciones de forma paralela. Se pide:

a) Realizar el código paralelo para ejecutar dos funciones en dos secciones distintas, mostrando para cada hilo, el tiempo total consumido en cada sección. El código secuencial se aporta en la (Fig. 1). En el ejemplo figura `omp_get_wtime()`, que se utiliza para medir tiempos, al igual que `wtime()` vista en clase.

b) (Opcional) Describir lo que sucede cuando aumentamos el número de hilos.

```
int main (){
     double timeIni, timeFin;
     timeIni = omp_get_wtime();
     printf("Ejecutando tarea 1\n");
     tarea_uno();
     printf("Ejecutando tarea 2\n");
     tarea_dos();
     timeFin = omp_get_wtime();
     printf("Tiempo total = %f segundos\n",timeFin – timeIni);
}
```

Fig. 1. Función main() de ejecución de dos tareas en secuencial.

Ejercicio 2. Realizar un código paralelo, o con alguno ya realizado en ejercicios anteriores de este u otro seminario, que demuestre las diferentes maneras de asignar iteraciones a los distintos `threads` (`static, dynamic o guide`). Este proceso se denomina, planificación de tareas o `Schedule`.

Ejercicio 3. Podemos definir el producto escalar de dos vectores, como el producto de sus componentes y la suma total de todos ellos.

```
total += a[i] * b[i];
```

a) Se pide realizar la paralelización con OpenMP del producto escalar de dos vectores con tamaños (3000, 4000 y 5000), variando el número de hilos hasta un máximo de 8. Se debe estudiar la evolución del tiempo de ejecución con respecto a la versión secuencial.

b) Realizar la paralelización con el uso de la directiva `#pragma omp critical` en vez de reducir con la directiva `#pragma omp....reduction.`

c) Realizar una gráfica o tabla comparativa de los tiempos de la versión secuencial, paralela (apartado a) y paralela (apartado b).

Ejercicio 4. Realizar un código paralelo en OpenMP que obtenga máximo o el mínimo del vector.

José M. Cecilia - Tlf: (+34) 968 278587 - jmcecilia@ucam.edu

Ejercicio 5. Realizar un código paralelo de la multiplicación de un vector por una matriz. Se pide:

a) Estudiar el rendimiento de la paralelización con tamaños de matriz (1000x1000, 2000x2000 y 3000x3000) y vectores de (1000, 2000 y 3000) elementos, respectivamente. El núcleo de cálculo se muestra a continuación, con un ejemplo práctico (Fig. 2), junto al núcleo secuencial del cálculo (Fig. 3).

b) Estudiar el impacto en el rendimiento de aplicar paralelización anidada.

$$\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix} \text{ X } \begin{pmatrix}2\\ 1\end{pmatrix} = \begin{pmatrix}1\,x\,2+2\,x\,2\\ 3\,x\,1+4\,x\,1\end{pmatrix}$$

Fig 2. Ejemplo practico.

```
for (int i = 0; i < n; i++) {
      sumas[i] = 0;
      for (int j = 0; j < n; j++) {
          sumas[i] += A[i][j] * x[j];
      }
}
```

Fig 3. Núcleo secuencial.

Ejercicio 6. Ejecutar el código de (Fig 4) en Hertz y comentar lo que sucede si dejamos activada la permisividad de paralelismo anidado con `omp_set_nested(1)` o por el contrario la desactivamos.

```
#include <omp.h>
#include <stdio.h>
void threads_por_nivel(int nivel)
{
    printf("Nivel %d: Número de hilos en el nivel %d -
%d\n",nivel,nivel,omp_get_num_threads());

}
int main()
{

    omp set nested(1)
    omp set num threads(2);
    #pragma omp parallel
    {
        threads por nivel(1);
         omp set num threads(2);
        #pragma omp parallel
        {
            threads por nivel(2);
             omp set num threads(2);
            #pragma omp parallel
            {
                threads por nivel(3);
            }
        }
    }
    return(0);
}
```

Fig. 4. Código ejemplo de paralelismo anidado.

A continuación, escoger alguna de las implementaciones vistas en clase u otra que el alumno quiera plantear, y aplicar esta técnica.

José M. Cecilia - Tlf: (+34) 968 278587 - jmcecilia@ucam.edu