Control #8 OpenMP

Viernes 15 de Julio de 2022

Nombre:_____

La siguiente función implementa la multiplicación de una matriz A de dimensión NxN y una matriz B de dimensión Nx1.

```
void mvmult(const double A[N][N], const double B[N], double C[N])
{
    double sum;
    for (int i=0; i<N; i++) {
          sum = 0;
          for (int j=0; j<N; j++){
          sum += A[i][j] * B[j];

          C[i] = sum;
    }
}
```

a) Paralelice la función anterior con OpenMP.

Se asume que la cantidad de threads es p. Cada threads debe tener acceso a las matrices de entrada A y B y a la matriz de salida C. Debido a esto, estas variables deben ser shared. La variable sum debe ser privada a cada thread.

```
void mvmult(const double A[N][N], const double B[N], double C[N])
{
    double sum;
    #pragma omp parallel for num_threads(p) shared(A,B,C) private(sum)
    for (int i=0; i<N; i++) {
          sum = 0;
          for (int j=0; j<N; j++){
          sum += A[i][j] * B[j];

          C[i] = sum;
    }
}
```

b) Determine el speedup y eficiencia del código paralelo. Suponga que el número de threads p es un divisor de N.

Para determinar el speedup y eficiencia, es necesario calcular primero la complejidad algorítmica T_s del algoritmo:

$$T_s = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} = N^2$$

La comlejidad paralela, se obtiene observando que cada thread realiza N/p iteraciones del ciclo en forma paralela. No se considera overhead y se considera complejidad asintótica:

$$T_p = \sum_{i=0}^{N/p-1} \sum_{j=0}^{N-1} \approx \frac{N}{p} \cdot N = \frac{N^2}{p}$$

Luego, el speedup es:

$$S = \frac{T_s}{T_p} = p$$

La eficiencia es:

$$E = \frac{S}{p} = 1$$