Taller 2: OpenMP

Esteban Aguero Perez (estape)

Arquitectura de Computadores II

22 de febrero de 2019

Investigación

1. ¿Qué es OpenMP?

   Es una especificación para un set de directivas del compilador, rutinas de bibliotecas y variables del entorno que pueden ser usados para la especificación de alto nivel de paralelismo en programas Fortran y C/C++.[1]

2. ¿Cómo se define una región paralela en OpenMP utilizando pragmas?

   Los pragmas están diseñados para que incluso si no lo soporta, este se ejecute correctamente pero sin ningún paralelismo. [2]

   Se define con:

   ```
   #pragma args
   {
       // Region de código en paralelo
   }
   ```

3. ¿Cómo se define la cantidad de hilos a utilizar al paralelizar usando OpenMP?

   Con el la bandera [2]

   ```
   #pragma omp parallel num_threads(n)
   ```

   También con la variable de ambiente

   ```
   set OMP_NUM_THREADS=<number of threads to use>.
   export OMP_NUM_THREADS=<number of threads to use>.
   ```

4. ¿Cómo se compila un código fuente c para utilizar OpenMP y qué encabezado debe incluirse?

   ```
   #include <omp.h>
   ```

   Y al compilar se usa la bandera [2]

   ```
   -fopenmp
   ```

5. ¿Cómo maneja OpenMP la sincronización entre hilos y por qué esto es importante?

Ejemplo Pi Serial

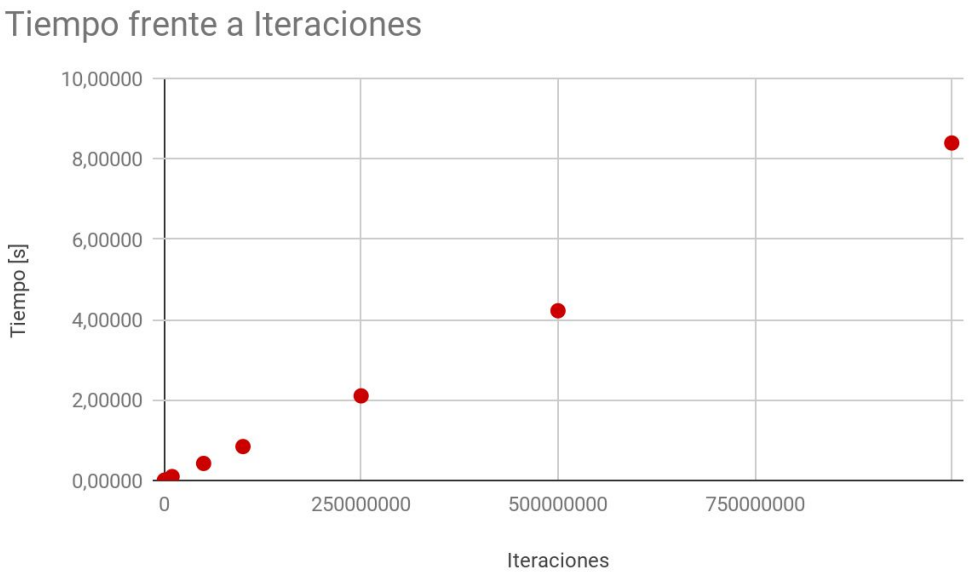
- Detecte qué sección del código podrı´a paralelizarse por medio de la técnica de multihilo.

  La parte que detecto puede ser paralelizable, es la parte del for.

- Con respecto a las variables de la aplicación (dentro del código paralelizable) ¿cuáles deberı´an ser privadas y cuáles deberı´an ser compartidas? ¿Por qué?

  Debería ser compartida el valor de suma ya que el valor de este debe ser acumulado y compartido en hilos, y privados x ya que este es independiente entre hilos.

- En la siguiente gráfica de Iteraciones vs Tiempo se muestran los resultados obtenidos.



Ejemplo Pi Paralelo

- Analice el código dado. ¿Cómo se define la cantidad de hilos a ejecutar? ¿Qué funcionalidad tiene el #pragma omp single? ¿Qué función realiza la lı´nea: `#pragma omp for reduction(+:sum) private(x)`

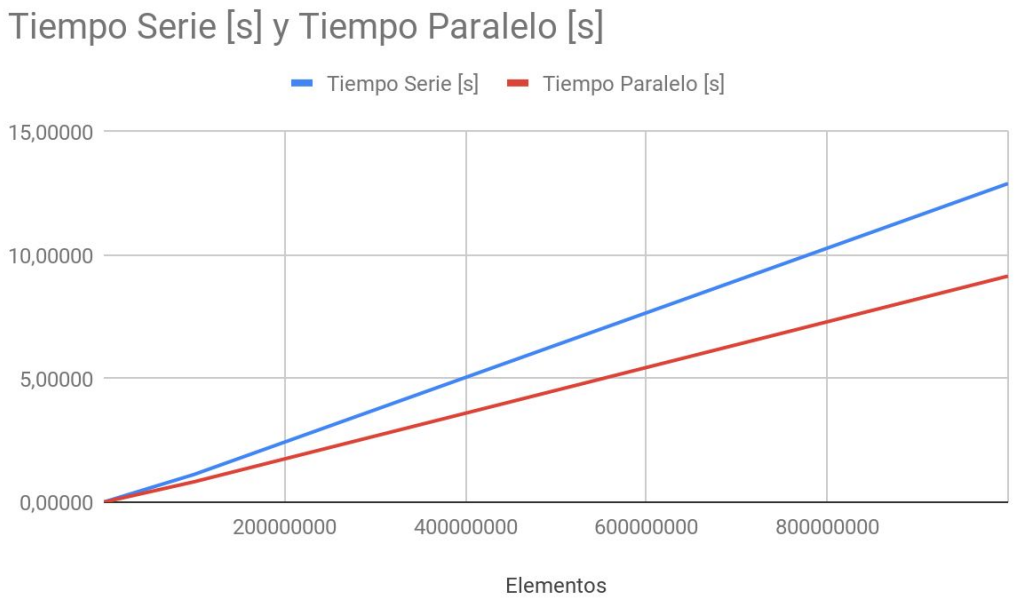
  La cláusula reduction es una directiva especial que le dice al compilador que tiene que generar un código que tiene acumular valores de diferentes loops de cierta forma u orden. Y la definición private define que es privada entre hilos, por defecto son compartidas excepto por las definidas en el cuerpo del bloque paralelo.

- La gráfica a continuación muestra la cantidad de hilos vs el tiempo.

Para el resultado anterior se empleó un procesador de dos núcleos, por lo que el cambio más dramático se da cuando pasa de un hilo a dos. Pero no es si no hasta el valor de cinco hilos que se encuentra el tiempo mínimo, a partir de ahí en vez de mejorar más, empieza a empeorar. Con ocho hilos se encuentra uno de los valores más altos luego del valor mínimo y es que en este caso con tantos hilos, en lugar de ser más rápido, resulta ser más lento ya que ahora se tienen más cantidad de hilos luchando por tener tiempo de CPU, ocasionando que se estabilice la mejora o hasta empeora al aumentar el número de hilos.

Saxpy

- Para realizar la comparación entre SAXPY serie y paralelo, se tomaron dos hilos para la ejecución, debido a que la máquina que emplee para realizar las pruebas es de doble núcleo.



Ejercicios Prácticos

- Para el ejercicio se implementó un calculador de integrales, se encuentra en el siguiente repositorio, además del código empleado para este taller y con las instrucciones para compilar (readme).

  https://github.com/estape11/ArquitecturaComputadoresII/tree/master/Taller2

Referencias

[1] Make - GNU Project . [Online] Recuperado de : https://www.openmp.org/about/openmp-faq/#WhatIs el 22 de febrero de 2019.

[2] Guide into OpenMP: Easy multithreading programming for C++ . [Online] Recuperado de : https://bisqwit.iki.fi/story/howto/openmp/ el 22 de febrero de 2019.