Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores

(Licentiate Degree Program in Computer Engineering)

Curso: CE-4302 Arquitectura de Computadores II

(Course: CE-4302 Computer Architecture II)

Evaluación Taller 2: OpenMP

(Workshop 2 evaluation: OpenMP)

Profesor:

(Professor)

Ing. M.Sc. Jeferson González Gómez

Fecha: 20 de febrero de 2019

(Date)

Evaluación Taller 2. OpenMP, parte I

Micro-investigación

Previo a la realización de los ejercicios prácticos. Realice una pequeña búsqueda en internet, que responda las siguientes interrogantes:

1. ¿Qué es OpenMP?
2. ¿Cómo se define una región paralela en OpenMP utilizando pragmas?
3. ¿Cómo se define la cantidad de hilos a utilizar al paralelizar usando OpenMP?
4. ¿Cómo se compila un código fuente c para utilizar OpenMP y qué encabezado debe incluirse?
5. ¿Cómo maneja OpenMP la sincronización entre hilos y por qué esto es importante?

OpenMP es formalmente una extensión de lenguaje, y está presente en versiones modernas de los compiladores (gcc, por ejemplo) por lo que su instalación no es necesaria.

Primer ejemplo: cálculo de pi

Como primer ejemplo, se muestra una aplicación de paralelismo para el cálculo de pi, por medio de una aproximación discreta de:

$$\int_0^1 \frac{4}{1+x^2} dx$$

Pi serial

La implementación serial para el cálculo de pi por la aproximación, se muestra en el código pi.c. Este código utiliza métodos de OpenMP, exclusivamente para la medición de tiempo, pero requiere que su compilación se realice con esta extensión. Para este código

1. Analice la implementación del código y detecte qué sección del código podría paralelizarse por medio de la técnica de multihilo.
2. Con respecto a las variables de la aplicación (dentro del código paralelizable) ¿cuáles deberían ser privadas y cuáles deberían ser compartidas? ¿Por qué?
3. Realice la compilación del código, utilizando el método requerido para aplicaciones de OpenMP.
4. Ejecute la aplicación. Realice un gráfico de tiempo para al menos 4 números de pasos distintos (iteraciones para cálculo del valor de pi).

Pi paralelo

Una posible implementación paralela de la aproximación se muestra en el archivo pi_par.c. La aplicación calculará *pi* para diferente cantidad de hilos de ejecución. A partir del código:

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

2. Realice la compilación del código.

3. Ejecute la aplicación. Realice un gráfico con tiempo de ejecución para las diferentes cantidades de hilos mostradas en la aplicación. Compare el mejor resultado con la cantidad de procesadores de su sistema. Aumente aún más la cantidad de hilos. Explique por qué, a partir de cierta cantidad de hilos, el tiempo aumenta.

Ejercicios prácticos

1. Realice un programa que aplique la operación SAXPY tanto serial (normal) como paralelo (OpenMP), para al menos tres tamaños diferentes de vectores. Mida y compare el tiempo de ejecución entre ambos.

2. Realice una aplicación paralelizable que requiera gran cantidad de procesamiento. Su aplicación podría ser una aproximación a una integral, una serie, un conjunto de operaciones, etc.

Entregables

- En TecDigital: Archivo de texto con nombre Taller1_Nombre_completo.docx (odt, doc, etc) que contenga las respuestas a la microinvestigación y preguntas teóricas, así una captura de pantalla del contenido de los códigos fuente. Adicionalmente, el documento puede incluir un link a un repositorio con los códigos fuente en lugar de las capturas de pantalla.