Práctica

Introducción a Open Multiprocessing (Open MP)
Parte 2

Objetivo:

El estudiante conocerá el concepto de variables compartidas y privadas en OpenMP
El estudiante comprenderá uso y utilidad de variables compartidas entre los hilos y el uso de variables privadas para evitar condiciones de carrera.

Al final de la práctica el estudiante podrá identificar y establecer las variables privadas y compartidas dentro de una región paralela.

Previo

Investigar lo siguiente:
¿Qué es un *Data Race (condiciones de carrera)*?
¿Qué es lo que comparten y no compartes entre los hilos?
En el etapa de particionamiento de la fase de diseño de programas paralelos. ¿A qué se refiere la descomposición de dominio y funcional,

Antecedentes
Haber realizado la práctica Introducción a Open Multiprocessing (Open MP) parte 1

Introducción

OpenMP utiliza un modelo de programación de memoria compartida, esto es, hay un hilo al inicio y fin del programa, pero este número puede cambiar de forma dinámica durante su ejecución,es decir, cuando se crean regiones paralelas. En esas regiones paralelas la mayoría de las variables son compartidas entre los hilos.

Todo hilo tiene un contexto de ejecución, que consiste del espacio de direcciones al que el hilo puede tener acceso. El contexto de ejecución incluye variables estáticas, estructuras asignadas de forma dinámica y variables en el *stack*. Con lo anterior se puede decir que una variable compartida tiene la misma dirección en el contexto de ejecución de cada hilo y una variable privada tiene una dirección distinta en el contexto de ejecución de cada hilo.

Es importante entender que cuando en una región paralela se definen variables privadas, un hilo no puede tener acceso a las variables privadas de otro hilo. Cundo una variable se define como compartida todos los hilos pueden tener acceso al espacio en memoria asignado a esta. Figura 1

M.I. Elba Karen Sáenz García

Proyecto PAPIME PE104911
Pertinencia de la enseñanza del cómputo paralelo en el currículo de las ingenierías
Práctica - **Introducción a Open Multiprocessing (Open MP) Parte 2**
Facultad de Ingeniería – UNAM

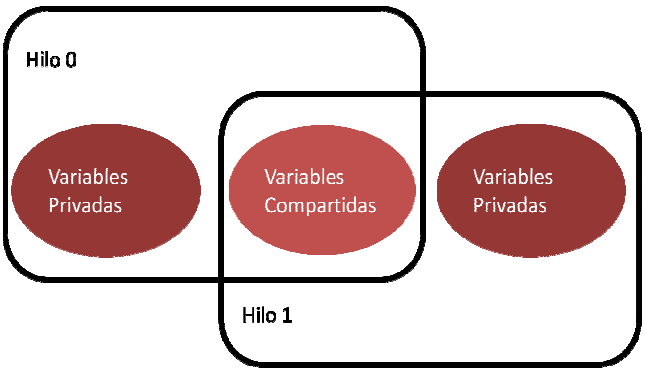

Figura 1

Dado que el objetivo de usar OpenMP es que parte de las actividades que realizará el programa se dividirá entre los hilos de la o las región o regiones paralelas definidas, en ocasiones, se producirán condiciones de carrera, debido a que varios hilos querrán tener acceso a las mismas localidades de memoria. En tal caso es conveniente hacer uso de la clausula ***prívate()*** que permite hacer una copia de las variables y tratarlas como distintas en cada hilo (se dice que las variables son privadas).El problema de utilizar variables privadas es que al final de una región paralela tienen un valor indefinido.

Como no todo es compartido entre los hilos, por ejemplo, las variables que se encuentran en el stack , también existe la clausula *shared*(), que permite indicar que la variable será compartida en la región paralela que se haya definido.

Clausulas para indicar variables compartidas y privadas

La sintaxis correspondiente para indicar que en una región paralela las variables son compartida o privadas es:

shared(var1,var2,…varn): Las variables son compartidas por todos los procesos ligeros
prívate(var1,var2,…varn) : Cada proceso ligero tiene una copia de la variable

Desarrollo

Realizar las siguientes actividades

Actividad 1: Del siguiente segmento de código.

Código Secuencial:
int a[1000], i;
for (i = 0; i < 1000; i++) a[i]=foo(i);

M.I. Elba Karen Sáenz García

¿Cómo se puede dividir el trabajo en dos hilos?

¿Qué tipo de descomposición se utiliza (De dominio o funcional)?

¿Cuáles deben ser variables compartidas y cuales privadas para que no existan *data races*?

Actividad 2
Para el segmento de código

```
int e;
main () {
  int x[10], j, k, m;   j = f(x, k);  m = g(x, k);
}
```

¿Cómo se divide el trabajo en dos hilos?

¿Qué tipo de descomposición se utiliza?

Clausula private

La siguiente función realiza la suma de dos arreglos bidimensionales a y b en paralelo, el resultado es almacenado en un tercer arreglo c.

```
void* f(float* c, int N) {
 float x, y; int i;
#pragma omp parallel for
   for(i=0; i<N; i++) {
         x = a[i]; y = b[i];
         c[i] = x + y;
   }
}
```

Si se analiza con detenimiento el código y se asume que hay dos hilos en la región paralela, se pueden plantear las siguientes preguntas:
¿Dónde se encontraría un *data race?*
¿Qué variables tendrían que ser privadas para evitar las condiciones de carrera?

En este caso la variables x,y tienen que ser privadas para que los hilos no modifiquen el resultados de los otros y se realizar una condición de carrera, y el código queda:

```
void* f(float* c, int N) {
  float x, y; int i;
 #pragma omp parallel for private(x,y)
    for(i=0; i<N; i++) {
          x = a[i]; y = b[i];
          c[i] = x + y;
    }
}
```

Actividad 3

Dado el siguente código, donde se agrega la biblioteca omp.h para el uso de las funciones *omp_get_thread_num() y omp_get_num_threads()* para obtener el identificador de cada hilo y el total de hilos en la región paralela, indicar cuales variables tienen que ser privadas y compartidas. Sdemás justificar la respuesta.

```
#include <omp.h>

main ()  {

int nhilos, tid;

#pragma omp parallel private(   ) shared(  )
  {

  /* Se obtiene y se imprime el identificador del hilo */
  tid = omp_get_thread_num();
  printf("Hola Mundo desde el hilo  = %d\n", tid);

  /* Solo el hilo maestro  o principal lo realiza */
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Numero de hilos= %d\n", nhilos);
    }
  }
}
```

Proyecto PAPIME PE104911
Pertinencia de la enseñanza del cómputo paralelo en el currículo de las ingenierías
Práctica - **Introducción a Open Multiprocessing (Open MP) Parte 2**
Facultad de Ingeniería – UNAM

Referencias

- **Introduction to parallel Programming ,Student book , Intel Software College , 2007**
- **Multicore Programming for Academia, Intel Software College, 2007**
- **https://computing.llnl.gov/tutorials/openMP/**
- **Michael J. Quinn, *Parallel Programming in C with MPI and OpenMP*, McGraw-Hill (2004).**

M.I. Elba Karen Sáenz García