Taller 1 - OpenMP

Jonathan Guzmán Araya
Ingeniería en Computadores, Tecnológico de Costa Rica
jonathan11@estudiantec.cr

1 Investigación

1. ¿Cuáles son los principales APIs en la industria basados en el paradigma de programación multi-proceso con memoria compartida? ¿Cuáles son las principales diferencias entre ellos? Entre los principales APIs podemos mencionar:

   (a) OpenMP
   (b) Pthreads (POSIX Threads)
   (c) Cilk
   (d) TBB (Intel Threading Building Blocks)

   Algunas diferencias entre ellos se pueden encontrar en la Tabla 1.

2. Respecto openMP que permiten las variables de entorno y funciones: OMP_NUM_THREADS y omp_get_max_threads().

   En el caso de openMP las variables de entorno y las funciones se relacionan entre sí para gestionar los hilos utilizados en una región paralela.

   - OMP_NUM_THREADS es una variable de entorno que se puede utilizar para controlar el número de hilos que OpenMP utiliza en las regiones paralelas de un programa.
   - omp_get_max_threads() es una función proporcionada por OpenMP que permite a un programa consultar el número máximo de hilos que se pueden utilizar en las regiones paralelas.

3. Si se utiliza gcc como compilador cómo es posible usar openMP? Cuáles son los pasos para poder utilizar el API?

   (a) Instalar GCC
   (b) Desarollar el código e importar la biblioteca
   (c) Compilar el codigo con gcc y OpenMP
   (d) Ejeuctar el archivo creado

4. ¿Qué es un pragma, específicamente en openMP qué hacen: omp parallel, omp single, omp for reduction? Bajo este contexto un pragma es una instrucción que proporciona indicaciones especiales sobre como deben gestionarse ciertas secciones de código.

   - omp parallel: esta se encarga de crear una región paralela en la que multiples hilos se ejecutan concurrentemente.

API	Lenguajes	Modelo de programación	Sincronización	Portabilidad	Escalabilidad
openMP	C, C++, Fortran	Directivas del compilador y funciones de biblioteca	Directivas pragmáticas para paralelizar bucles y secciones de código. Proporciona mecanismos de sincronización implícitos y explícitos	Diseñado para ser portátil entre diferentes plataformas y compiladores	Adecuado para aplicaciones con bucles paralelizables y tareas que se pueden dividir en subprocesos
Pthreads	C, C++	API de programación en C para la creación y gestión de hilos (threads)	Ofrece un conjunto de funciones para crear, gestionar y sincronizar hilos. Permite un alto grado de control y flexibilidad	Ampliamente compatible con sistemas POSIX, lo que lo hace portátil entre sistemas UNIX y Linux	Requiere un manejo más detallado de la concurrencia y sincronización por parte del programador
Cilk	C, C++	Extiende el lenguaje C/C++ con palabras clave específicas para la programación paralela	Utiliza la programación basada en tareas y divide automáticamente el trabajo en tareas más pequeñas	Menos común que OpenMP, pero todavía se encuentra en algunos entornos	Ofrece una abstracción más alta que OpenMP para la programación paralela basada en tareas
TBB	C++	Biblioteca de plantillas C++ que proporciona abstracciones para la programación paralela y concurrente	Ofrece abstracciones como flujos de flujo (flow graphs) y paralelismo de tareas, que simplifican la programación concurrente	Puede ser utilizado en sistemas compatibles con C++	Proporciona una mayor abstracción y simplificación para la programación paralela en C++

Table 1: Principales APIs de Programación Multiproceso con Memoria Compartida

- omp single: esta se encarga de que el bloque de código al que se le aplica debe ejecutarse por un solo hilo.
- omp for reduction: con esta al especificar que es con reduction se suele aplicar a variables compartidas al final del bucle para combinar los resultados parciales de cada hilo.

5. ¿Cómo se puede permitir la sincronización en openMP? OpenMP cuenta con varias formas para controlar la sincronización de hilos, entre ellas tenemos:

   - barrier: se utiliza para crear un punto de sincronización en el que todos los hilos participantes en una región paralela deben esperar hasta que todos los demás hilos alcancen ese punto antes de continuar.
   - critial: se utiliza para crear una sección crítica en la que solo un hilo puede ejecutar el código a la vez.
   - atomic: se utiliza para realizar operaciones de lectura, modificación y escritura de manera atómica en variables compartidas.
   - sections: se utiliza para dividir una región paralela en secciones, y los hilos trabajan en secciones específicas.

6. ¿Cómo se puede medir el tiempo de ejecución en openMP? Se puede utilizar de dos maneras diferentes:

   - omp_get_wtime: se utiliza para medir el tiempo de ejecución en segundos.
   - omp_get_wtick: se utiliza para para obtener la resolución del reloj, que es la diferencia mínima que se puede medir.