Tarea 1: Estudio de la API OpenMP

1. Directivas OMP

Las directivas de OpenMP son instrucciones especiales que se agregan al código fuente para indicar a un compilador cómo debe paralelizar ciertas secciones del código. Estas directivas permiten a los desarrolladores aprovechar la capacidad de procesamiento paralelo de las arquitecturas de múltiples núcleos sin tener que realizar programación detallada de bajo nivel.

#pragma omp parallel: Esta directiva crea un equipo de hilos y se utiliza para indicar una sección de código que debe ejecutarse en paralelo. Cada hilo ejecutará la sección de código dentro del bloque paralelo.

```
#pragma omp parallel [clause[ [, ]clause] ...] new-line
        structured-block
```

#pragma omp for: Se utiliza para paralelizar un bucle for. Los hilos creados con #pragma omp parallel ejecutarán iteraciones diferentes del bucle en paralelo.

```
#pragma omp for [clause[[,] clause] ... ] new-line
        for-loops
```

#pragma omp parallel for: Combina las funcionalidades de **#pragma omp parallel** y **#pragma omp for**, paralelizando un bucle.

```
#pragma omp parallel for [clause[[,] clause] ...] new-line
        for-loop
```

#pragma omp sections: Divide el bloque de código en secciones, y cada sección puede ejecutarse en paralelo. Es común combinar esto con **#pragma omp section**.

```
#pragma omp sections [clause[[,] clause] ...] new-line
        {
        [#pragma omp section new-line]
                structured-block
        [#pragma omp section new-line
                structured-block ]
        ...
        }
```

#pragma omp parallel sections: Similar a **#pragma omp sections**, pero también crea un equipo de hilos para ejecutar las secciones en paralelo.

```
#pragma omp parallel sections [clause[ [, ]clause] ...]
        new-line
        {
        [#pragma omp section new-line]
                structured-block
        [#pragma omp section new-line
        structured-block ]
        ...
        }
```

#pragma omp single: Indica que el bloque de código dentro de esta directiva debe ejecutarse solo por un hilo. Se elige arbitrariamente el primer hilo que llega a la región **single**.

```
#pragma omp single [clause[[,] clause] ...] new-line
        structured-block
```

#pragma omp task: Crea una tarea independiente que puede ser ejecutada en paralelo. Las tareas son unidades de trabajo que pueden asignarse a hilos disponibles.

```
#pragma omp task [clause[ [, ]clause] ...] new-line
        structured-block
```

#pragma omp master: Indica que el bloque de código dentro de esta directiva debe ejecutarse solo por el hilo maestro (el hilo que ejecuta la región paralela).

```
#pragma omp master new-line
        structured-block
```

#pragma omp critical: Establece una región crítica, lo que significa que solo un hilo a la vez puede ejecutar el bloque de código dentro de esta directiva.

```
#pragma omp critical [(name)] new-line
        structured-block
```

#pragma omp barrier: Sincroniza los hilos en el equipo, asegurando que ningún hilo avance más allá de esta directiva hasta que todos los demás hayan alcanzado la misma directiva.

```
#pragma omp barrier new-line
```

#pragma omp taskwait: Espera a que todas las tareas en paralelo creadas por la región paralela actual se completen antes de continuar.

```
#pragma omp taskwait newline
```

#pragma omp atomic: Indica que una operación debe realizarse atómicamente, es decir, sin interferencia de otros hilos.

```
#pragma omp atomic new-line
        expression-stmt

expression-stmt: one of the following forms:
        x binop = expr
        x++
        ++x
        x--
        --x
```

#pragma omp flush: Garantiza la sincronización de las variables entre hilos. Se utiliza para forzar la actualización de las variables compartidas en la memoria.

```
#pragma omp flush [(list)] new-line
```

#pragma omp ordered: Indica que el bucle o la sección deben ejecutarse en orden secuencial, a pesar de la ejecución paralela.

```
#pragma omp ordered new-line
        structured-block
```

#pragma omp threadprivate: Declara variables que deben ser privadas para cada hilo en un equipo. Cada hilo tiene su propia copia de estas variables.

```
#pragma omp threadprivate(list) new-line
```

2. Cláusulas OMP

Las cláusulas de OpenMP se utilizan junto con las directivas de OpenMP para especificar cómo se deben manejar las variables en un entorno paralelo.

2.1 Cláusulas de compartición de datos

default: Esta cláusula establece el comportamiento predeterminado para las variables no especificadas de manera explícita en otras cláusulas.

shared: Se utiliza para indicar que una variable es compartida entre todos los hilos en un equipo. Todos los hilos ven y pueden modificar la misma instancia de la variable en la memoria.

private: Declara que cada hilo debe tener su propia copia privada de la variable. Cada hilo trabaja con su propia instancia de la variable y no comparte datos con otros hilos.

firstprivate: Similar a **private**, pero también inicializa la variable privada de cada hilo con el valor de la variable original antes de entrar en la región paralela.

lastprivate: Se utiliza con bucles **for** paralelos y garantiza que la variable sea igual a la versión privada de la variable en el hilo que ejecuta la última iteración del bucle.

reduction: Se utiliza para realizar operaciones de reducción en variables. Crea una variable privada para cada hilo y, al final de la región paralela, combina todas las instancias privadas en una única instancia mediante una operación de reducción (como suma o producto).

nowait: Indica al compilador que no debe esperar a que todos los hilos terminen antes de continuar con el código fuera de la región paralela. Puede mejorar el rendimiento al reducir la sincronización.

2.2 Cláusula de copia de datos

copyin: Especifica que el valor de una variable privada en un hilo debe ser copiado en la variable compartida fuera de la región paralela.

copyprivate: Similar a **copyin**, pero opera en el sentido opuesto: copia el valor de una variable compartida en la variable privada de cada hilo al salir de la región paralela.

3. Planificadores OMP

Mecanismos que determinan como se distribuyen las iteraciones de un bucle entre los hilos de un equipo.

static (estático): En este planificador, las iteraciones del bucle se dividen estáticamente entre los hilos antes de la ejecución. Cada hilo recibe un bloque de iteraciones contiguas. Este enfoque es simple y puede ser eficiente cuando las iteraciones tienen una duración similar.

dynamic (dinámico): Las iteraciones se asignan dinámicamente a los hilos en tiempo de ejecución. Cada hilo recibe un conjunto de iteraciones y, cuando termina, obtiene un nuevo conjunto. Este enfoque es útil cuando las duraciones de las iteraciones son impredecibles o varían significativamente.

guided (guiado): Similar a **dynamic**, pero las iteraciones se asignan en bloques de tamaño decreciente. Inicialmente, se asigna un bloque grande, y a medida que los hilos terminan, reciben bloques más pequeños. Puede ayudar a equilibrar la carga cuando hay variabilidad en las duraciones de las iteraciones.

auto (automático): Deja que el sistema decida el planificador más adecuado según heurísticas internas. Esta opción permite a OpenMP elegir el planificador basándose en el contexto y la arquitectura del sistema.

runtime (tiempo de ejecución): Permite al planificador ser especificado en tiempo de ejecución mediante variables de entorno o funciones de OpenMP en el código. Proporciona flexibilidad para cambiar el planificador sin recompilar el código.

4. Variables de entorno

Las variables de entorno permiten ajustar el comportamiento de OpenMP sin necesidad de modificar el código fuente del programa, proporcionando flexibilidad y control sobre la ejecución paralela.

OMP_SCHEDULE: Esta variable de entorno se utiliza para especificar el planificador de bucles por defecto y el tamaño de bloque en OpenMP. Puedes configurar valores como static, dynamic, guided, etc., junto con un tamaño de bloque (por ejemplo, static,4).

OMP_NUM_THREADS: Determina el número máximo de hilos que se deben utilizar para la ejecución paralela. Puedes establecer este valor en el número de hilos que deseas que se utilicen en tu aplicación. Por ejemplo, export OMP_NUM_THREADS=4.

OMP_DYNAMIC: Esta variable controla si OpenMP permite la variabilidad dinámica del número de hilos. Un valor de true indica que OpenMP puede ajustar el número de hilos en tiempo de ejecución según la carga de trabajo.

OMP_NESTED: Determina si se permiten regiones anidadas en OpenMP. Un valor de true permite la anidación de regiones paralelas, mientras que un valor de false no lo permite.

OMP_STACKSIZE: Establece el tamaño de la pila para cada hilo de OpenMP. Puedes ajustar este valor según tus necesidades, especialmente si experimentas problemas de agotamiento de la pila en hilos.

OMP_WAIT_POLICY: Controla la política de espera para la sincronización de hilos. Puedes establecer este valor en active para que los hilos en espera realicen un bucle activo, o passive para usar una espera más eficiente en términos de energía.

OMP_MAX_ACTIVE_LEVELS: Indica el número máximo de niveles de anidación de regiones paralelas permitidos.

OMP_THREAD_LIMIT: Define el número máximo de hilos que pueden ser utilizados en el programa.