Teoria 6-open mp

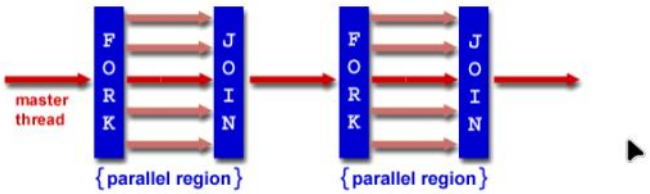
viernes, 11 de abril de 2025 08:39

Estándar de programación paralela basado en directivas.

- Usa directivas, librerías y variables de entorno.
- Las directivas dan soporte para concurrencia, sincronización y manejo de datos sin necesidad de locks, conducuines y demás
- Por abajo usa pthreads

Open MP la idea es tomar el código que a hiciste y sin mucho esfuerzo hacerlo concurrente.

- Comienza con un único hilo (hilo master).
- ***Fork***: Al encontrar un constructor paralelo (o directiva paralela), el hilo master crea un grupo de hilos
- El bloque encerrada por el constructor de la región paralela es ejecutada en paralelo entre todos los hilos.
- ***Join***: cuando el conjunto de hilos finaliza el bloque paralelo, se sincronizan y terminan, continuando únicamente el hilo master.



Constructor parallel:
El más importante, especifica qué bloque de código se hace en paralelo.
Se crea un equipo de hilos pero el programador es quien distribuye.
Al final de la región hay una barrera implícita, solo el master sigue la ejecución

Admite private y firstprivate que se usen ara determinar que dats son privados a cada hilo y cuáles se comparten.
Private crea una copia a cada variable.

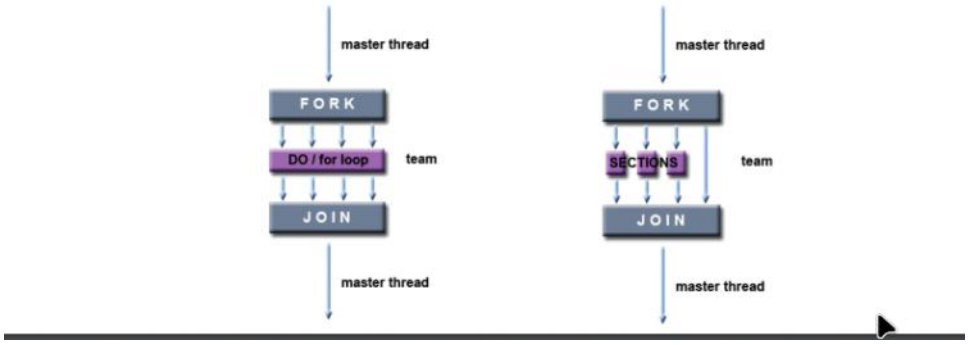
Shared se usa para decirle que las variables que le paso son compartidas.

Num_threads se usa paa indicar cuántos hilos a usar

Con un if podemos controlar si se generan hilos o no. Si da false, se ejecuta secuencia, sino paralelo

Parallel puede ser usado para especificar concrrencia entre iteraciones y tareas, no crea hreads nuevos.

- Diferentes tipos de constructores.
  - Directiva *for* → Divide las iteraciones de un bucle entre los hilos (paralelismo de datos).
  - Directiva *sections* → Divide el trabajo en secciones separadas (paralelismo funcional).



For: las iteraciones tienen que ser independientes pq sino el resultado pede no sel el correcto.
Tenés que saber de antemano cuantas iteraciones vas a hacer.
El índice es privado por defecto, no se lo puede editar en el bucle por los hilos.
No se puede usar break

- Cláusulas disponibles:
  - *shared*, *private*, *firstprivate*
  - *lastprivate* (lista de variables): funciona como *private*, sólo que la variable original queda con el valor de la última iteración del bucle
  - *reduction* (operador:variable, …): realiza una operación de reducción usando el operador indicado con las múltiples copias de la variable correspondiente

Cláusual reduction

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

#define N 1000

int main () {
   int v[N], i, sum=0;
   /* Inicializar v */
   #pragma omp parallel
   {
        #pragma omp for reduction(+:sum)
        for (i=0; i<N; i++)
                sum += v[i];
   }
   …
}
```

Ejemplo: acá se dividen las iteraciones entre la cantidad de hilos. Se hacen sumas por separado y después se van juntando.

Cláusula nowait: cuando un hilo no tiene más iteraciones que hacer no frena en una barrera sino que sigue ejecutando

```
#pragma omp parallel
{
      #pragma omp for nowait
      for (i=1; i<n; i++)
            b[i] = (a[i] + a[i-1]) / 2.0;

      #pragma omp for nowait
      for (i=0; i<m; i++)
            y[i] = sqrt(z[i]);
}
```

Cláusula schedule: especifica como se distribuyen las teraciones entre los hilos.

- Statuc: busca darle aprox misma cant a cada uno. Se le puede pasar varable chnk y le decis vos
- Dynamic: divide y les asigna bajo demanda. Se puede pasar en chunk cuanto asignar al principio
- Guided: parecido en dynamic, a medida que necesito menos hilos deja de usarlos??? Creo
- Auto: delega lo que prefiera el compilador o el sistema

Constructor sections: no lo vamo a usar probablemente 😊
Una sección por hilos, sino quedan ociosos

Paralelismo anidado tampoco se usa mucho.

Constructores para sincronzación:

- Single: permite que un bloqe de código sea ejeccutado por un solo hilo en la región paralela. El bloque lo ejecuta solo el primer hilo que llega.

```
#pragma omp single [lista de cláusulas]
{ /* bloque estructurado */ }
```

- Master es pareccido pero el que ejecuta es si o si el hilo máster
- Barrier: usar con cuidado pq muchas directivas las usan implícitamente
- Critical: para ejecuar una sección critica. Un solo hilo po vez

```
#pragma omp critical [nombre]
{ /* bloque estructurado */ }
```

Hay que ponerle nombre pq sino se llaman todas igual y se quedan trabadas aunque haya algun hilo en otra parte

Funcióne de openMP:lero lero, leer de las diapos si lo quiero

Tasking: se usa para resolver aquellos problemas que no se podian paralelizar. Ej paralelizar whiles, fors con cant indedfinida de iteraciones, algoritms recursivos.