

OpenMP.

César Pedraza Bonilla

Universidad Nacional de Colombia
Departamento de Ingeniería de Sistemas e Industrial

capedrazab@unal.edu.co

15 de septiembre de 2020

Overview

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

- 1 introducción.
- 2 Creación de hilos.
- 3 Sincronización.
- 4 Anidamiento paralelo.
- 5 Programación con OpenMP

Introducción.

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

- API para escribir aplicaciones multihilo.
- Conjunto de directivas y librerías para programación multihilo.
- Simplifica la programación de aplicaciones multihilo.



Introducción.

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

- La mayoría de las construcciones de OpenMP son directivas.
- Las construcciones de OpenMP se aplican a bloques de código, un punto de entrada y un punto de salida.
- Por ejemplo:

```
1 #pragma omp parallel num_threads(4)
2
3 #include <omp.h>
```

Introducción.

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

```
1 #include "omp.h"
2 void main()
3 {
4     #pragma omp parallel //inicio de region paralela
5     {
6         int ID = omp_get_thread_num();
7         printf(" hello( %d) ", ID);
8         printf(" world( %d) \n", ID);
9     } //fin de region paralela
10 }
```

Compilación: gcc -fopenmp hello.c -o hello

Introducción.

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

- OpenMP usa el modelo de hilos.
- Los hilos se comunican a través de variables compartidas.
- Condiciones de carrera: el programador realiza operaciones con resultados de hilos, que no necesariamente se han ejecutado.
- Se usan métodos de sincronismo, pero que son muy costosos computacionalmente. Se debe revisar la forma en que se acceden a las variables para evitar excesos en sincronismos.

Creación de hilos.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

- Paralelismo estilo *fork()-join()*
- Existe un hilo maestro.
- Ejemplo: crear 4 hilos:

```
1 omp_set_num_threads(4); #pragma omp parallel num_threads(4)
2
3 int omp_get_num_threads(); //numero de hilos presentes
4 int omp_get_thread_num(); //Thread ID
5 double omp_get_wtime(); //Tiempo en segundos desde un punto fijo en el pasado
```

Ejercicio: paralelizar el algoritmo de Leibniz mediante openMP.

Creación de hilos.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

False sharing.

Código para calcular pi.

```
1  #include <omp.h>
2  static long num_steps = 100000;
3  double step;
4  #define NUM_THREADS 2
5  void main ()
6  {
7      int i, nthreads; double pi, sum[NUM_THREADS];
8      step = 1.0/(double) num_steps;
9      omp_set_num_threads(NUM_THREADS);
10     #pragma omp parallel
11     {
12         int i, id, nthrds;
13         double x;
14         id = omp_get_thread_num();
15         nthrds = omp_get_num_threads();
16         if (id == 0) nthreads = nthrds;
17         for (i=id, sum[id]=0.0; i< num_steps; i=i+nthrds) {
18             x = (i+0.5)*step;
19             ssum[id] += 4.0/(1.0+x*x);
20         }
21     }
22     for(i=0, pi=0.0; i<nthreads; i++) pi += sum[i] * step;
23 }
```

Creación de hilos.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

False sharing.

- Se observa que se usa un arreglo para acumular los cálculos de cada hilo.
- Los elementos de *sum* son adyacentes en memoria. Puede causar al protocolo de caché que sean leídos más frecuentemente en memoria, dado que son llevados a *cores* distintos. A éste fenómeno se le llama *false sharing*.
- También se observa cuando se copian datos frecuentemente a caché, que sólo van a ser leídos, a causa de posiciones de memoria que van a ser escritas y que son adyacentes.
- Se puede atenuar separando las posiciones de memoria a las que cada hilo accede.

Creación de hilos.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

False sharing.

Código para calcular pi.

```
1  #include <omp.h>
2  static long num_steps = 100000;
3  #define PAD 8    // assume 64 byte L1 cache line size
4  double step;
5  #define NUM_THREADS 2
6  void main ()
7  {
8      int i, nthreads; double pi, sum[NUM_THREADS][PAD]; //se separan las posiciones para acumular
9      step = 1.0/(double) num_steps;
10     omp_set_num_threads(NUM_THREADS);
11     #pragma omp parallel
12     {
13         int i, id, nthrds;
14         double x;
15         id = omp_get_thread_num();
16         nthrds = omp_get_num_threads();
17         if (id == 0) nthreads = nthrds;
18         for (i=id, sum[id]=0.0; i< num_steps; i=i+nthrds) {
19             x = (i+0.5)*step;
20             ssum[id][0] += 4.0/(1.0+x*x);
21         }
22     }
23     for(i=0, pi=0.0; i<nthreads; i++) pi += sum[i] * step;
24 }
```

Sincronización.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Son herramientas o estrategias para imponer un orden y protección en el acceso a datos.

- De alto nivel. Crítica, atómica, barrera, ordenada.
- De bajo nivel. Flush, locks.

* Se explicarán más adelante.

Sincronización.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Sincronización crítica.

Hay exclusión mutua: sólo 1 hilo ejecuta al tiempo una sección crítica.

```
1 float res;  
2 #pragma omp parallel  
3 {  
4     float B; int i, id, nthrds;  
5     id = omp_get_thread_num();  
6     nthrds = omp_get_num_threads();  
7     for(i = id; i < niters; i + nthrds){  
8         B = big-job(i);  
9         #pragma omp critical  
10         consume (B, res); // just 1 thread executes cosume at a time  
11     }  
12 }
```

Sincronización.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Sincronización Atómica.

Permite hacer exclusión mutua pero sólo aplica para leer y actualizar una variable. La operación atómica debe ser del tipo:

$x \text{ binop} = \text{expr}$, $x++$, $++x$, $x--$, $--x$

Pej, se protege la actualización de la variable X.

```
1 #pragma omp parallel
2 {
3     double tmp, B;
4     B = DOIT();
5     tmp = big_ugly(B);
6     #pragma omp atomic
7     tmp = big_ugly(B);
8     X += tmp;
9 }
```

Sincronización.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Sincronización por barrera.

Cada hilo espera hasta que los todos hayan llegado hasta un punto determinado del programa.

```
1 #pragma omp parallel
2 {
3   int id=omp_get_thread_num();
4   A[id] = big_calc1(id);
5   #pragma omp barrier
6   B[id] = big_calc2(id, A);
7 }
8 }
```

Anidamiento paralelo.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

- Una construcción paralela es SPMD (Single Program, Multiple Data).
- La forma en que se separan las cargas de un programa en hilos:
 - Loop construct
 - Sections/section constructs
 - Single construct
 - Task construct

* Se explicarán más adelante.

Anidamiento paralelo.

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Loop construct

```
1 #pragma omp parallel
2 {
3     #pragma omp for
4     for (l=0; l<N; l++){
5         NEAT_STUFF(l);
6     }
7 }
```

La variable *l* se crea de forma privada para cada hilo.

Anidamiento paralelo.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

```
1 //secuencial
2 for(i=0;i<N;i++) { a[i] = a[i] + b[i];}
3
4 //OpenMP parallel region
5 #pragma omp parallel
6 {
7     int id, i, Nthrds, irstart, iend;
8     id = omp_get_thread_num();
9     Nthrds = omp_get_num_threads();
10    irstart = id * N / Nthrds;
11    iend = (id+1) * N / Nthrds;
12    if (id == Nthrds-1)iend = N;
13    for(i=irstart;i<iend;i++) { a[i] = a[i] + b[i];}
14 }
15
16 //OpenMP parallel region and a worksharing for construct
17 #pragma omp parallel
18     #pragma omp for
19     for(i=0;i<N;i++) { a[i] = a[i] + b[i];}
```

Anidamiento paralelo.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

La cláusula *schedule*.

- *schedule(static [,chunk])*. Se asignan bloques de tamaño *chunk* a cada hilo.
- *schedule(dynamic[,chunk])*. Cada hilo ejecuta *chunk* iteraciones de una cola
- *schedule(guided[,chunk])*. Cada hilo ejecuta bloques de hilos de forma dinámica. El tamaño del bloque es grande al inicio y luego disminuye.
- *schedule(runtime)*. El tamaño del bloque es tomado de la variable *OMP_SCHEDULE*
- *schedule(auto)*. Escoje la máquina.

`omp_schedule.c`

Anidamiento paralelo.

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Iteraciones anidadas

```
1 #pragma omp parallel for collapse(2)
2 for (int i=0; i<N; i++) {
3     for (int j=0; j<M; j++) {
4         .....
5     }
6 }
```

Se especifica el número de iteraciones a ser paralelizadas (2)

Anidamiento paralelo.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Reducción.

- La reducción es la combinación de valores en una variable de acumulación.
- La cláusula para hacer reducción es `reduction (op : list)`.
- Dentro de la iteración:
 - Se crea una copia local de cada variable y es inicializada.
 - La variable local es la que se actualiza.
 - Las copias locales se reducen a un solo valor y almacenado en la variable original.

Anidamiento paralelo.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Reducción.

Ejemplo secuencial:

```
1 double ave=0.0, A[MAX]; int i;  
2 for (i=0; i< MAX; i++) {  
3   ave += A[i];  
4 }  
5 ave = ave/MAX;
```

Con OpenMP:

```
1 double ave=0.0, A[MAX]; int i;  
2 #pragma omp parallel for reduction (+:ave)  
3 for (i=0; i< MAX; i++) {  
4   ave += A[i];  
5 }  
6 ave = ave/MAX;
```

Programación con OpenMP.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Sincronización con barrera.

Cada hilo espera hasta que todos hayan llegado hasta ese punto.

```
1
2 #pragma omp parallel shared (A, B, C) private(id)
3 {
4     id=omp_get_thread_num();
5     A[id] = big_calc1(id);
6     #pragma omp barrier
7     #pragma omp for
8     for(i=0;i<N;i++){ C[i]=big_calc3(i,A); }
9     #pragma omp for nowait
10    for(i=0;i<N;i++){ B[i]=big_calc2(C, i); }
11    A[id] = big_calc4(id);
12 }
```

Programación con OpenMP.

OpenMP

César Pedraza
Bonilla

Introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Sentencia master

Especifica un bloque de código que sólo va a ser ejecutado por el hilo maestro.

```
1
2 #pragma omp parallel
3 {
4   do_many_things();
5   #pragma omp master
6   { exchange_boundaries(); }
7   #pragma omp barrier
8   do_many_other_things();
9 }
```


Programación con OpenMP.

OpenMP

César Pedraza
Bonilla

introducción.

Creación de
hilos.

Sincronización.

Anidamiento
paralelo.

Programación
con OpenMP

Sentencia single

Especifica un bloque de código que se desea ejecutar por un solo hilo. No necesariamente el maestro. Implícitamente lleva una barrera al final. Se puede omitir con *nowait*

```
1 #pragma omp parallel
2 {
3   do_many_things();
4   #pragma omp single
5     { exchange_boundaries(); }
6   do_many_other_things();
7 }
```