TEST-BP3.pdf

celssdfgh

Arquitectura de Computadores

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

PREGUNTAS BP3

¿Qué tipo de reparto se realiza en el siguiente codigo?

```
#pragma omp parallel for schedule(static)
for(i=0; i<10000; i++)
      if(i==0)
            omp_get_schedule(&kind, &chunk);
      v3[i] = v1[i] + v2[i];
}
```

a. El que indique la variable de control interno run-sched-var
b. El que indique la variable de control interno de def-sched-var
c. Ninguna otra respuesta es correcta
d. Static

(ya que hacemos un get_schedule y no un set)

Si le piden que realice un estudio de escabilidad de un codigo que calcula el producto de dos matrices

a. Representaría en una grafica el tiempo de ejecucion en funcion del tamaño de las matrices
b. Representaría en una grafica el tiempo de ejecucion en funcion del numero de nucleos
c. Representaría en una grafica la ganancia en velocidad(o ganancia en prestaciones) en funcion del numero de nucleos
d. No haría nada de lo indicado en el resto de respuestas

Indica que reparto de iteraciones a hebras es correcto suponiendo 2 hebras y la clausula schedule(guided,3)

a.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	0	0	0	0	0	0	1	1	0	0	0

b.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	0	0	0	0	0	1	1	1	0	0	0

c.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	0	0	1	1	1	0	0	0	1	1	1

d.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	0	0	0	0	0	1	1	1	0	0	1

(guided hace: (NUM_ITERACIONES - NUM_ITERACIONES_DADAS)/NUM_HEBRAS)
(en este caso seria:
Hebra 0: (12-0)/2 → 6 iteraciones
Hebra 1: (12-6)/2 → 3 iteraciones
Hebra 2: (12-6-3)/2 → menos que 3 (Chunk). Se comporta como Dynamic dándole o intentando darle a chunk iteraciones a cada hebra

¿Qué tipo de reparto realiza el siguiente codigo?

```
#pragma omp parallel or schedule (runtime)
for(i=0; i<1000;i++)
      v1[1] = v3[i] + v2[i];
```

a. El que indique la variable de control interno def-sched-var
b. El que indique la variable de control interno run-sched-var
c. El que devuelve la funcion de entorno omp_set_dynamic()
d. Las otras respuestas no son correctas

(lo podemos comprobar con la tabla de variables de entorno)

¿Cuál es la funcion de la clausula if en el siguiente codigo?

```
#pragma omp parallel if(n>20)
```

a. Las otras respuestas no son correctas
b. No ejecutar el codigo del bloque estructurado si n<=20
c. Ejecutar las ramas if y else del bloque estructurado en paralelo
d. Evitar problema de sobrecarga introducida al paralelizar el codigo para tamaños del problema pequeños

(si n es muy pequeña, no merecería la pena paralelizar)

¿Con cuantas hebras se ejecuta este codigo si previamente se ha fijado la variable OMP_NUM_THREADS = 8?

```
omp_set_num_threads(4);
#pragma omp parallel num_threads(2)
printf("hello\n")
```

a. 2
b. 1
c. 8
d. 4

(tiene mas prioridad la clausula num_threads(2))

¿Que codigo cree mejor para conseguir multiplicar una matriz triangular por un vector?

```
int m[N][N], v[N], r[N] = {0};
```

a.
```
for(int j=0; j<N; j++)
      for(int i=0; i<N; i++)
            r[i] += m[i][j]*v[j];
```
b.
```
for(int i=0; i<N; j++)
      for(int j=0; j<=i; i++)
            r[i] += m[i][j]*v[j];
```
c.
```
for(int i=0; i<N; j++)
      for(int j=0; j<N; i++)
            r[i] += m[i][j]*v[j];
```
d.
```
for(int i=0; i<N; j++)
      for(int i=0; i<=i i++)
            r[i] += m[i][j]*v[j];
```

(es la unica que calcula unicamente la parte que nos interesa de la matriz (la parte triangular, que en este caso es la que estaría en la parte izq de la matriz triangular inferior)

¿Cuál de las siguientes formas es la correcta para fijar el numero de hebras para un programa OPENMP?

a. En un programa OpenMP, usando la función omp_max_threads(4) al principio de la funcion main
b. En un programa OpenMP, usando la funcino omp_num_threads(4) al principio de la funcion main
c. En la consola del sistema, usando la variable de entorno export OMP_THREAD_LIMIT = 4
d. En un programa OpenMP, usando la funcion omp_set_num_threads(4) al principio de la funcion main

(es la única forma de modificar el numero de hebras)

El siguiente codigo se ejecuta en paralelo sobre 2 hebras para N=1024 repartiendo las iteraciones del bucle mas externo usando la directiva for con la clausula schedule(static, chunk), se puede asegurar que todos los threads realizaran el mismo trabajo (es decir, el mismo numero de operaciones)

```
int m[N][N], v[N], r[N] = {0}

for(int i=0; i<N; ++i)
      for(int j=0; j<=I; ++j)
            r[i] += m[i][j] * v[j];
```

a. En algunas ejecuciones sera correcta y en otras no
b. Correcta siempre en todas las ejecuciones
c. Sera correcta para algunos valores de chunk y para otros no
d. Incorrecta en todas las ejecuciones

(depende del chunk porque sip or ejemplo

¿Cual de las siguientes opciones permitiria comprobar que tipo de planificacion obtiene mejores resultados para un programa paralelo con ayuda de un script?

a. Schedule(Dynamic)
b. Schedule(guided)
c. Schedule(static)
d. Schedule(runtime)

(usando el script para cambiar el tipo de planificación y comprobar cual obtiene mejor resultado)

¿Qué codigo cree mejor para conseguir multiplicar una matriz triangular superior por un vector?

```
int m[N][N], v[N], r[N] = {0}
```

a.
```
for(int j=0; j<N; j++)
      for(int i=0; i<N; i++)
            r[i] += m[i][j]*v[j];
```

```
b. for(int i=0; i<N; j++)
        for(int j=0; j<=i; i++)
              r[i] += m[i][j]*v[j];
c. for(int i=0; i<N; j++)
        for(int j=i; j<N; i++)
              r[i] += m[i][j]*v[j];
d. for(int i=0; i<N; j++)
        for(int j=0; j<=N i++)
              r[i] += m[i][j]*v[j];
```

(es la unica que hace la matriz triangular superior)

Dado el codigo que se tiene a continuacion, ¿Qué tipo de reparto de iteraciones a hebras seria el mas optimo en tiempo de ejecucion?

```
#pragma omp parallel for
For(int i=0; i<100; i++)
      For(int j=0; j<I; j++)
            A[i][j] = 0;
```

a. Dynamic
b. El que indique la variable de control interno def-sched-var
c. Static
d. Runtime

(en cada iteración la carga es diferente j<i)

¿Cómo se puede modificar el reparto de iteraciones del bucle de una directiva #pragma omp for entre las hebras si usamos la clausula schedule(runtime)?

a. Usando la variable de entorno OMP_SCHEDULE o la funcion omp_set_schedule()
b. Usando solo la funcion omp_set_schedule()
c. Usando solo la variable de entorno OMP_SCHEDULE
d. Usando la variable de entorno OMP_SCHEDULE y la funcion omp_set_schedule()

(podemos usar ambas, o alguna de las dos como hemos hecho en el cuadernillo de practicas)

El tiempo de ejecucion de un programa paralelo…

a. Aumenta conforme el tamaño del problema disminuye
b. Se reduce conforme el tamaño del problema aumenta
c. Siempre será menor que el de su versión secuencial para cualquier tamaño del problema
d. Puede ser mayor que el tiempo de la versión secuencial para tamaños de problema pequeños, debido a la sobrecarga introducida al crear y destruir las hebras

Cuando se usa una planificación Dynamic de un bucle for en OpenMP, el tamaño del chunk…

a. Se adapta dinámicamente en funcion de la velocidad de cada hebra
b. Es siempre constante
c. Va decreciendo conforme se van ejecutando las iteraciones del bucle
d. Siempre debe ser mayor que 1

(aunque lo podemos cambiar durante la ejecucion, este valor puede valer >=1)

El parámetro chunk en el siguiente codigo determina:

```
#pragma omp parallel for schedule(guided,chunk)
```

a. El tamaño minimo del bloque iteraciones que OpenMP asignara a una hebra
b. El tamaño del bloque iteraciones optimo que OpenMP debe usar para minimizar el tiempo de ejecucion
c. El tamaño máximo del bloque iteraciones que OpenMP asignara a una hebra
d. El tamaño del bloque iteraciones que OpenMP asignara siempre a cada hebra

En una maquina con 8 cores y tras ejecutar export OMP_NUM_THREADS = 4
¿Cuántas iteraciones ejecuta la hebra master en la región parallel?

```
int N = omp_get_max_threads();
omp_set_num_threads(2);
#pragma omp parallel for num_threads(6) if(N>=4) schedule(static)
for(int i=0; i<12; i++)
      printf(“ thread: \%d iteracion: \%d \n”, omp_get_thread_num(),
i);
```

a. 12
b. 2
c. 4
d. 6

(temenos N=4 y 6 hebras por la clausula, y al ser static y el chunk de 1, se darán 2 iteraciones a cada hebra)

¿Cómo se podría establecer el numero de hebras a ejecutar en una región paralela desde programa?

a. Mediante el uso de la variable de entorno OMP_NUM_THREADS antes de la ejecucion del programa
b. Mediante el uso de la clausula num_threads en una directiva que abra la región paralela
c. Mediante el uso de la funcino omp_set_num_threads después de que una región paralela comience
d. Ninguna de las respuestas es correcta

(estamos en programa y se debe poner el num de hebras antes de la región paralela)

ESTE TEMA LO TIENES MASTICADÍÍÍÍÍSIMO

¡~~UNA HORA~~ UN TRIDENT MÁS Y YA LO TIENES!

Trident®

ESTIIIIIRA TUS MOMENTOS

Indica cual de las siguientes opciones obtendrá mejores prestaciones para multiplicar una matriz triangular por un vector

a.
```
#pragma omp for schedule(guided)
for(i=0; i<N; i++)
      v2[i] = 0;
      for(j=0; j<=i; j++)
            v2[i] += m[i][j] * v1[j];
}
```

b.
```
#pragma omp for private(j) schedule(guided)
for(i=0; i<N; i++)
      v2[i] = 0;
      for(j=0; j<=i; j++)
            #pragma omp critical
            v2[i] += m[i][j] * v1[j];
      }
}
```

c.
```
#pragma omp for private(j) schedule(guided)
for(i=0; i<N; i++)
      v2[i] = 0;
      for(j=0; j<=i; j++)
            v2[i] += m[i][j] * v1[j];
}
```

d.
```
#pragma omp for private(j) schedule(guided)
for(i=0; i<N; i++)
      v2[i] = 0;
      for(j=0; j<N; j++)
            v2[i] += m[i][j] * v1[j];
}
```

Analiza el codigo mostrado a continuacion e indica que habria que cambiar para que se imprima la siguiente salida. Cuando OMP_NUM_THREADS = 4

```
int i, n=3;
#pragma omp parallel for private(n)
for(i=0; i< omp_get_max_threads(); i++)
      printf("Thread %d imprime: %d", omp_get_thread_num(), i+n)
Salida por pantalla:
      Thread 0 imprime: 3
      Thread 1 imprime: 4
      Thread 2 imprime: 5
      Thread 3 imprime: 6
```

a. Cambiar private por firstprivate
b. Cambiar private por copyprivate
c. No hay que cambiar nada
d. Cambiar private por lastprivate

¿Cual de los siguientes metodos para determiner el numero de hebras que ejecutaran la siguiente región paralela es el mas prioritario?

a. La clausula num_threads
b. La funcion omp_set_num_threads
c. La clausula if
d. La variable de entorno OMP_NUM_THREADS

Dado el codigo que se tiene a continuación ¿Qué tipo de reparto de iteraciones a hebras seria el mas optimo en tiempo de ejecucion?

```
#pragma omp parallel for
For(int i=0; i<100; i++)
      A[i] += b[i]*c[i];
```

a. Dynamic
b. Static
c. Runtime
d. Guided

(ya que cada hebra en cada iteración, tiene la misma carga)

Indica que reparto de iteraciones a hebras es correcto suponiendo 3 hebras y la clausula schedule(static)

a.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	0	1	1	2	2	0	0	1	1	1	0

b.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	1	2	2	1	2	0	1	2	0	1	2

c.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	0	0	0	1	1	1	1	2	2	2	2

d.

Iteracion	0	1	2	3	4	5	6	7	8	9	10	11
hebra	0	0	0	1	1	1	2	2	2	2	2	0

Indica que reparto de iteraciones a hebras es correcto suponiendo 4 hebras y la clausula schedule(static,3)

a.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	0	0	1	1	1	2	2	2	3

b.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	0	0	0	1	1	1	1	2	2

c.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	1	2	3	0	1	2	3	0	1

d.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	3	3	0	0

¿En el siguiente fragmento de codigo, cuantas hebras están ejecutando la región paralela?

```
long sum = 0, N=10, a[10], b[10], c[10];
#pragma omp parallel
{
      int ithread = omp_get_thread_num();
      int nthread = omp_get_num_threads();
      #pragma omp sections
      {
            #pragma omp section
            for(long i=0; i<N; i+=nthread)
                  c[i] = a[i] + b[i];

            #pragma omp section
            for(long i=ithread; i<N; i+=nthread)
                  c[i] = a[i] + b[i];
      }
```

a. El numero de hebras posible sera siempre igual al numero de procesadores lógico que tenga la maquina donde se ejecuta el codigo
b. 2
c. N
d. Las que indique la funcion omp_get_thread_num()

Las variables de control internas de OpenMP

a. Solo se pueden modificar mediante el uso de variables de entorno o en la consola del sistema
b. Las otras respuestas son incorrectas
c. Solo se pueden modificar mediante el uso de las funciones que proporciona el API de OpenMP
d. Pueden ser accedidas directamente por el programador

¿Qué muestra la ejecucion del siguiente programa por pantalla suponiendo que se ejecuta en un nodo de ATCgrid?

```
int n = (int)(omp_get_max_threads()/4);
#pragma omp parallel for num_threads (6) if(n>6)
for (int i=0; i<n; i++)
      printf("h: \%d ", omp_get_thread_num());
```

a. h:0
b. Las otras respuestas no son correctas
c. h:0 h:0 h:0 h:0 h:0 h:0
d. h:0 h:1 h:2 h:3 h:4 h:5

Indica que reparto de iteraciones a hebras es correcto suponiendo 3 hebras y la clausula schedule(dynamic,2)

a.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	0	0	1	1	1	2	2	2	0

b.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	1	1	0	0

c.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	2	2	0	1

d.

Iteracion	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	0	0	0	2