Examen BP3 Resuelto.

Recuerda que me puedes seguir en wuolah :P

Pd: para cuando subo el examen, no se la nota ni los resultados, así que puede ser que alguna respuesta no sea correcta, aunque están consensuadas post-examen. Y otra cosa, no te fijes en las respuestas marcadas, fíjate en las respuestas.

1) ¿Qué tipo de reparto se realiza en el siguiente código?

```
#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 `def-sched-var`.
- ○ c) Ninguna otra respuesta es correcta.
- ○ d) `static`

Respuesta: d), ya que hacemos un get_schedule y no un set.

2) Si le piden que realice un estudio de escalabilidad de un código que calcula el producto de dos matrices.

- ○ a) Representaría en una gráfica el tiempo de ejecución en función del tamaño de las matrices.
- ○ b) Representaría en una gráfica el tiempo de ejecución en función del número de núcleos.
- ○ c) Representaría en una gráfica la ganancia en velocidad (o ganancia en prestaciones) en función del número de núcleos.
- ○ d) No haría nada de lo indicado en el resto de respuestas.

Respuesta: c) – Fuente: Miami me lo confirmó xD pero es la que más sentido tiene.

3) Indica qué reparto de iteraciones a hebras es correcto suponiendo hebras y la cláusula schedule(guided,3)

a)

iteración	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)

iteración	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)

iteración	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)

iteración	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

Respuesta: b)

IMPORTANTE: GUIDED HACE:
(NUM_ITERACCIONES-NUM_ITERACCIONES_DADAS)/NUM_HEBRAS

Donde en este caso sería:
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 intentadolo, darle chunk iteraciones a cada hebra.

Sigue bajando que no me entraba la foto xd

4) ¿Qué tipo de reparto realiza el siguiente código?

```
#pragma omp parallel for schedule (runtime)
for (i=0 ; i<1000 ; i++)
  v1[i] = 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 función de entorno `omp_set_dynamic()`.
- ○ d) Las otras respuestas no son correctas.

Respuesta: b), run-sched-var como podemos ver en la tabla de variables de entorno.

5) ¿Cuál es la función de la cláusula if en el siguiente código?
#pragma omp parallel if(n>20)

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

Respuesta: d), Si n es muy pequeña, no merecería la pena paralelizar.

6) ¿Con cuántas hebras se ejecuta este código si previamente se ha fijado la variable de entorno 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

Respuesta: a), ya que tiene más prioridad la cláusula num_threads(2).

7) ¿Qué código 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 ; i++)
  for (int j=0 ; j<=i ; j++)
    r[i] += m[i][j] * v[j];
```

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

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

Respuesta: b), ya que es la única que calcula únicamente 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)

8) ¿Cuál de las siguientes formas es la correcta para fijar a el número de hebras para un programa OPENMP ?

a) En un progarma OpenMP, usando la función `omp_max_threads(4)` al principio de la función main.

b) En un programa OpenMP, usando la función `omp_num_threads(4)` al principio de la función main.

c) En la consola del sistema, usando la variable de entorno `export OMP_THREAD_LIMIT=4`

d) En un programa OpenMP, usando la función `omp_set_num_threads(4)` al principio de la función main.

Respuesta: d), ya que es la única forma de modificar el número de hebras (leer bien)

9)

El siguiente código se ejecuta en paralelo sobre 2 hebras para $N=1024$ repartiendo las iteraciones del bucle más externo usando la directiva *for* con la cláusula *schedule(static,chunk)*, se puede asegurar que todos los threads realizarán el mismo trabajo (es decir, el mismo número 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];
```

Esta afirmación es:

- ○ a) En algunas ejecuciones será correcta y en otras no.
- ○ b) Correcta siempre en todas las ejecuciones.
- ○ c) Será correcta para algunos valores de chunk y para otros no.
- ○ d) Incorrecta en todas las ejecuciones.

Respuesta: c), depende del chunk, ya que si por ejemplo, el tamaño fuese 3, la última hebra que cogiese iteraciones cogería solo 1 en vez de 3.

10) ¿Cuál de las siguientes opciones permitiría comprobar qué tipo de planificación obtiene mejores resultados para un programa paralelo con ayuda de un script?

¿Cuál de las siguientes opciones permitiría comprobar qué tipo de planificación obtiene mejores resultados para un programa paralelo con ayuda de un *script*?

- ○ a) `schedule(dynamic)`
- ○ b) `schedule(guided)`
- ○ c) `schedule(static)`
- ○ d) `schedule(runtime)`

Respuesta d), Usando el script para cambiar el tipo de planificación y comprobar cual obtiene mejor resultado.

11)

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

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

Usuario Profesores

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

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

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

Respuesta: b), ya que es la única que hace la matriz triangular superior (c) hace inferior).

12)

Dado el código que se tiene a continuación ¿Qué tipo de reparto de iteraciones a hebras sería el más óptimo en tiempo de ejecución?

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

Usuaria Profesores

- a) dynamic
- b) El que indique la variable de control interno `def-sched-var`.
- c) static
- d) runtime

Respuesta: a), Dynamic ya que, en cada iteración, la carga es diferente. (j<i)

13)

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

Usuaria Profesores

- a) Usando la variable de entorno `OMP_SCHEDULE` **O** la función `omp_set_schedule()`
- b) Usando sólo la función `omp_set_schedule()`
- c) Usando sólo la variable de entorno `OMP_SCHEDULE`
- d) Usando la variable de entorno `OMP_SCHEDULE` **Y** la función `omp_set_schedule()`

Respuesta: a), ya que podemos usar ambas, o alguna de las dos, como hemos hecho en el cuadernillo de la práctica.

14)

El tiempo de ejecución de un programa paralelo ...

Usuario Profesores

- 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

Respuesta: d), como hemos explicado anteriormente

15)

Cuando se usa una planificación `dynamic` de un bucle `for` en OpenMP, el tamaño del *chunk* ...

Usuario Profesores

- a) Se adapta dinámicamente en función 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

Respuesta: b), ya que, aunque lo podamos cambiar durante la ejecución (d)), este valor puede valer >=1

16)

El parámetro `chunk` en el siguiente código determina:

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

Usuario Profesores

a) El tamaño mínimo del bloque iteraciones que OpenMP asignará a una hebra
b) El tamaño del bloque iteraciones óptimo que OpenMP debe usar para minimizar el tiempo de ejecución
c) El tamaño máximo del bloque iteraciones que OpenMP asignará a una hebra
d) El tamaño del bloque iteraciones que OpenMP asignará siempre a cada hebra

Respuesta: a), como hemos explicado antes en una de las primeras preguntas

17)

En una máquina con 8 cores y tras ejecutar `export OMP_NUM_THREADS=4`, ¿cuántas iteraciones ejecuta la hebra máster 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);
```

Usuario Profesores

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

Respuesta: b), ya que N=4 y tenemos 6 hebras por la clausula y al ser static y el chunk de 1, se darán 2 iteraciones a cada hebra.

18)

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

Usuario Profesores

a) Mediante el uso de la variable de entorno `OMP_NUM_THREADS` antes de la ejecución del programa

b) Mediante el uso de la cláusula `num_threads` en una directiva que abra la región paralela

c) Mediante el uso de la función `omp_set_num_threads` después de que una región paralela comience

d) Ninguna de las respuestas es correcta

Respuesta: b) como estamos en programa y se debe poner el num de hebras antes de la región paralela.

19)

Dado el código que se tiene a continuación ¿Qué tipo de reparto de iteraciones a hebras sería el más óptimo en tiempo de ejecución?

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

Usuario Profesores

- a) dynamic
- b) static
- c) runtime
- d) El que indique la variable de control interno `def-sched-var`.

Respuesta: a), ya que iteraciones tienen cargas diferentes

20)

Indica cuál de las siguiente opciones obtendrá mejores prestaciones para multiplicar una matriz triangular por un vector

Usuaria Profesores

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(static)
for (i=0 ; i<N ; i++) {
  v2[i] = 0;
  for (j=0 ; j<N ; j++)
    v2[i] += M[i][j] * v1[j];
```

Respuesta: c), ya ha sido explicado antes

21)

Analiza el código mostrado a continuación e indica qué habría 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
```

Usuaria Profesores

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

Respuesta: a), ya que como sabemos de la BP2

22)

¿Cuál de los siguientes métodos para determinar el número de hebras que ejecutarán la siguiente región paralela es el más prioritario?

Usuaria Profesores

a) La cláusula `num_threads`
b) La función `omp_set_num_threads`
c) La cláusula `if`
d) La variable de entorno `OMP_NUM_THREADS`

Respuesta: c) (aparece en una página del seminario tal cual).

23)

Dado el código que se tiene a continuación ¿Qué tipo de reparto de iteraciones a hebras sería el más óptimo en tiempo de ejecución?

```
#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

Respuesta: b), ya que cada hebra en cada iteración, tiene la misma carga.

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

a)

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

b)

iteración	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)

iteración	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)

iteración	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

Respuesta: b), aunque supongo que el iteración 3 debería ser 0 y se ha puesto erróneamente.

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

hebra	0	0	0	1	1	1	2	2	2	3

b)

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

c)

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

d)

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

Respuesta: a)

26) ¿En el siguiente fragmento de código, 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 número de hebras posible será siempre igual al número de procesadores lógico que tenga la maquina donde se ejecuta el código

B) 2

C) N

D) Las que indiquen la función omp_get_thread_num()

Respuesta: b), pero no es seguro.