Programación Concurrente

Preguntas de final

Ramiro Martínez D'Elía

2021

Implemente una solución, paralela, al problema de multiplicación de matrices de $n * n$ con $P < n$

Si el número de procesos es menor que n, no podremos asignar una fila a cada proceso. En su lugar, debemos hacer que cada proceso trabaje con una porción (*strpe*) del arreglo.

```
1  Process[w = 1..P]
2
3  # El proceso w, procesara las filas first a last
4  int first = (w-1) * (n/p) + 1;
5  int last = first + n/p - 1;
6
7  for(a_row = first to last)
8
9      for [b_col = 1 to n]
10         # Inicializamos la celda acumuladora de C.
11         c[a_row, b_col] = 0;
12
13         # Iteramos entre las columnas (A) y filas (B) de interes.
14         for [k = 1 to n]
15             c[a_row, b_col] = c[a_row, b_col] + (a[a_row, k] * b[k, b_row]);
16 end;
17 End;
```

(a) Suponga n=128 y que cada procesador es capaz de ejecutar un proceso. ¿Cuántas asignaciones, sumas y productos se hacen secuencialmente (caso en el que P=1)?

- En este caso $strip \rightarrow n$
- Línea 11: Hace tantas pasadas como columnas tenga B (n) por el tamaño del strip $\rightarrow n^2$
- Línea 15: Hace n pasadas por tantas pasadas como columnas tenga B por el tamaño del strip $\rightarrow n^3$

$Asignaciones = n^3 + n^2 = 128^3 + 128^2 = 2097152 + 16384 = 2113536$

$Sumas = n^3 = 128^3 = 2097152$

$Productos = n^3 = 128^3 = 2097152$

(b) Manteniendo n=128. Si los procesadores P1 a P7 son iguales, y sus tiempos de asignación son 1, de suma 2 y de producto 3, y si P8 es 4 veces más lento, ¿Cuánto tarda el proceso total concurrente? ¿Cuál es el valor del speedup (Tiempo secuencial/Tiempo paralelo)?. Modifique el código para lograr un mejor speedup.

- En este caso $strip = n/p = 128/8 = 16$

$Asignaciones = n^2 \times 16 + n \times 16 = 128^2 \times 16 + 128 \times 16 = 262144 + 2048 = 264192$

$Sumas = n^2 \times 16 = 128^2 \times 16 = 262144$

$Productos = n^2 \times 16 = 128^2 \times 16 = 262144$

Los procesos 1 a 7, tardaran lo mismo:

$264192 \times 1ut + 262144 \times 2ut + 262144 \times 3ut = 1574912ut$

El proceso 8, es 4 veces más lento que el resto. Por lo cual, tardará 4 veces más:

$1574912ut \times 4 = 6299648ut$

Por consiguiente, el proceso concurrente tardará $6299648ut$ en finalizar. Ya que, el proceso 8 será el último, en terminar su trabajo.

Con las unidades de tiempo, de los procesadores más eficientes, el proceso secuencial tardará:

$2113536 \times 1ut + 2097152 \times 2ut + 2097152 \times 3 = 12599296ut$

Por consiguiente el Speedup obtenido será de 2.

Para mejorar el Speedup podríamos balancear la carga de trabajo, de los procesadores, de manera distinta. Por ejemplo; haciendo que el procesador 8, el más lento, trabaje sobre un strip más pequeño.

- Múltiplo de 7 más cercano a 128 $\rightarrow$ 126
- Tamaño del stripe, para el procesador 8 $\rightarrow 128 - 126 = 2$
- Tamaño del stripe, para el resto de los procesadores $\rightarrow 126/7 = 18$

Asignaciones $P_8 = 128^2 \times 2 + 128 \times 2 = 33024$

Sumas $P_8 = 128^2 \times 2 = 32768$

Productos $P_8 = 128^2 \times 2 = 32768$

Tiempo $P_8 = 33024 \times 1ut + 32768 \times 2ut + 32768 \times 3 = 196864ut$

Asignaciones $P_{resto} = 128^2 \times 18 + 128 \times 18 = 297216$

Sumas $P_{resto} = 128^2 \times 18 = 294912$

Productos $P_{resto} = 128^2 \times 2 = 294912$

Tiempo $P_{resto} = 297216 \times 1ut + 294912 \times 2ut + 294912 \times 3 = 1771776ut$

Con estos nuevos tiempos el speedup será de 7,1.

Dado el siguiente programa, indice si es posible que finalice.

```
1 bool continue = true;
2 bool try = false;
3 co  while (continue) { try = true; try = false; } #(P)
4 /   <await(try) continue = false> #(Q)
5 oc
```

Con una política débilmente fair, el programa podría no terminar. Ya que, *try* no se mantiene verdadera hasta ser vista por (Q).

Con una política fuertemente fair, el programa podría terminar. Ya que, *try* se convierte en verdadera con infinita frecuencia.