Computación de Altas Prestaciones
HPC

Problemas Tema 2: Vectorización y paralelización de bucles

2.1.- El siguiente bucle tiene varios tipos de dependencias. Encuentra todas las dependencias verdaderas, antidependencias y dependencias de salida, y elimine las antidependencias y dependencias de salida cambiando el nombre.

```
for (i=0; i<100; i=i+1) {
        Y[i] = X[i] / c;        /* S1 */
        X[i] = X[i] + c;        /* S2 */
        Z[i] = Y[i] + c;        /* S3 */
        Y[i] = c - Y[i];        /* S4 */
}
```

SOLUCION:

The following dependences exist among the four statements:
1. There are true dependences from S1 to S3 and from S1 to S4 because of Y[i]. These are not loop carried, so they do not prevent the loop from being considered parallel. These dependences will force S3 and S4 to wait for S1 to complete.
2. There is an antidependence from S1 to S2, based on X[i].
3. There is an antidependence from S3 to S4 for Y[i].
4. There is an output dependence from S1 to S4, based on Y[i].

The following version of the loop eliminates these false (or pseudo) dependences.

```
for (i=0; i<100; i=i+1 {
        T[i] = X[i] / c; /* Y renamed to T to remove output dependence */
        X1[i] = X[i] + c;/* X renamed to X1 to remove antidependence */
        Z[i] = T[i] + c;/* Y renamed to T to remove antidependence */
        Y[i] = c - T[i];
}
```

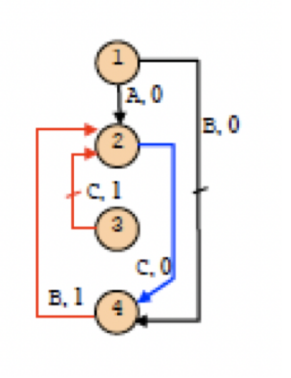
After the loop, the variable X has been renamed X1. In code that follows the loop, the compiler can simply replace the name X by X1. In this case, renaming does not require an actual copy operation, as it can be done by substituting names or by register allocation. In other

2.2.- El siguiente bucle se va a ejecutar en una máquina multithread:

```
do i = 1, N-2
        (1) A(i) =B(i)
        (2) C(i) = A(i) + B(i-1)
        (3) D(i) = C(i+1)
        (4) B(i) = C(i) * 2
enddo
```

Genere el grafo de dependencias
Escribe una versión paralela de dicho bucle y suponiendo que el tiempo de ejecución de una instrucción es T, ¿Cuántas veces más rápido es la ejecución al paralelizar en P procesadores?.¿Cuál será la máxima aceleración con infinitos procesadores y despreciando el tiempo de sincronización.

SOL




```
doall i = 1, N-2              ; 1 y 3 se pueden ejecutar en paralelo
(1)  A(i) = B(i)
(3)  D(i) = C(i+1)
enddoall

[barrera]

do i = 1, N-2                 ; este bloque hay que ejecutarlo en serie
(2)  C(i) = A(i) + B(i-1)
(4)  B(i) = C(i) * 2
enddo
```

Tiempo de ejecución en serie Ts= Nx4xT

Tiempo de ejecución con mejoras Tp= (Nx2xT)/p + Nx2xT

Si p = infinito S= 2.

2.3.- Examine los siguientes bucles y analice su posible paralelización:

a) ¿ Tiene dependencias las iteraciones del siguiente bucle?

```
for (i=0;i<100;i++) {
        A[i] = B[2*i+4];
        B[4*i+5] = A[i];
}
```

I

SOL:

5 -4 /GCD(2,4) = ½ =0,5 no es entero => No hay dependencias

b) Encuentra todas las dependencias verdaderas, antidependencias y dependencias de salida, y elimine las antidependencias y dependencias de salida cambiando el nombre.

```
for (i=0;i<100;i++) {
    A[i] = A[i] * B[i]; /* S1 */
    B[i] = A[i] + c; /* S2 */
    A[i] = C[i] * c; /* S3 */
    C[i] = D[i] * A[i]; /* S4 */
}
```

SOL

Output dependencies

S1 and S3 cause through A[i]

Anti-dependencies

S4 and S3 cause an anti-dependency through C[i]

Re-written code

```
for (i=0;i<100;i++) {
T[i] = A[i] * B[i]; /* S1 */
B[i] = T[i] + c; /* S2 */
A1[i] = C[i] * c; /* S3 */
C1[i] = D[i] * A1[i]; /* S4 */}
```

True dependencies

S4 and S3 through A[i]

S2 and S1 through T[i]

c) Considere el siguiente bucle:

```
for (i=0;i < 100;i++) {
        A[i] = A[i] + B[i]; /* S1 */
        B[i+1] = C[i] + D[i]; /* S2 */
}
```

¿Hay dependencias entre S1 y S2? ¿ Es paralelizable? En caso negative, muestre como hacerlo parallelizable.

SOL

There is an anti-dependence between iteration i and i+1 for array B. This can be avoided by renaming the B array in S2

2.4. Represente el grafo de dependencias de los siguientes bucles:

a)

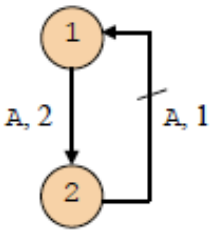
```
do i = 1, N-2
        (1) A(i) =B(i) +2
        (2) C(i) = A(i-2) +A(i+1)
enddo
```

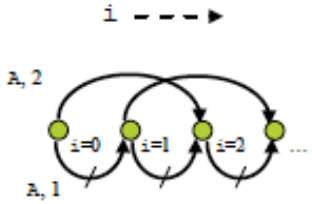
b)

```
do i = 1, N-2
do j = 1, N-2
        (1) A(i,j) =A(i,j-1) *2
        (2) C(i,j) = A(i-2.j+1) +1
Enddo
enddo
```

SOL a.-

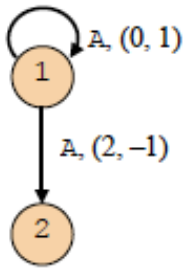
```
do i = 2, N-2
1:  A(i) = B(i) + 2
2:  C(i) = A(i-2) + A(i+1)
enddo
```



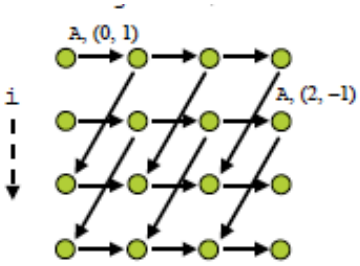


SOL b.-

```
do i = 2, N-1
do j = 1, N-2
1:  A(i,j) = A(i,j-1) * 2
2:  C(i,j) = A(i-2,j+1) + 1
enddo
enddo
```



grafo de dependencias



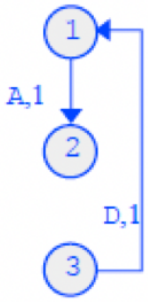
espacio de iteraciones

2.5.- El siguiente bucle se va a ejecutar en una máquina SMP de 4 procesadores:

```
do i = 1, 994
      (1) A(i) = D(i) - 10
      (2) B(i+1) = B(i+1) * A(i-1)
      (3) D(i+1) = D(i+1) + D(i+1)
Enddo
```

a) Genere el grafo de dependencias y el espacio de iteraciones correspondientes al bucle. Escribe una versión paralela de dicho bucle que pueda ejecutarse en P procesadores sin tener que utilizar funciones de sincronización.

SOL




```
A(1) = D(3) - 10
B(2) = B(2) * A(0)
B(3) = B(3) * A(1)
doall i = 1, 992
    D(i+1) = D(i+1) + D(i+1)
    A(i+1) = D(i+1) - 10
    B(i+3) = B(i+3) + A(i+1)
enddoall
D(994) = D(994) + D(994)
A(994) = D(994) - 10
D(995) = D(995) + D(995)
```

b) Para el caso de esta máquina de 4 procesadores, escribe el código que ejecutará cada procesador (el bucle paralelo) si el reparto de las iteraciones entre los procesadores es (1) estático consecutivo; y (2) dinámico por chunks (trozos), con trozos de 50 iteraciones. ¿Cuántas iteraciones ejecutará el procesador P1 en el primer caso? ¿y en el segundo?

El bucle doall de 992 iteraciones se puede repartir como se quiera.

1.- estático consecutivo: cada procesador ejecuta 992/4 = 128

P0: i=1,248
P1: i 249,496
P2: i = 497,744
P3: i= 745,942

1.- dinámico cada procesador ejecuta 50 iteraciones

P1 no se puede saber 250 si es el que mas trabaja y 200 o 242 en otros casos.

2.6- Para cada uno de los siguientes bucles indique en qué casos existen dependencia entre las dos instrucciones dadas. Justifique la respuesta.

a)

```
do i = 1, 100
      A(2*i) = ...
      ... = A(2*i+1)
enddo
```

SOL:
(1-0)/GCD(2,2) =1/2=0,5 no es entero => No hay dependencias

b)

```
do i = 5, 100
      A(i-5) = ...
      ... = A(2*i+90)
enddo
```

SOL:
(90-(-5))/GCD(2,1) =95/1 = 95 es entero => Puede haber dependencias, pero no la hay porque los intervalos de escritura son de A(0) a A(95) y el de lectura es de A(100) a A(290)

c)

```
do i = 1, 100
      A(3*i+100) = ...
      ... = A(2*i-1)
enddo
```

SOL:
(-1-100)/GCD(3,2) =-101/1 = -101 es entero => Puede haber dependencias.
Interseccion de los intervalos de escritura son de A(103) a A(400) y el de lectura es de A(1) a A(199)
Para escritura con i= 1 en A(103) depende de lectura con i =52 en A(103)

d)

```
do i = 1, 100
      A(6*i+3) = ...
      ... = A(3*i+81)
enddo
```

SOL:
(81-3)/GCD(6,3) =78/3 = 26 es entero => Puede haber dependencias.
Interseccion de los intervalos de escritura son de A(9) a A(603) y el de lectura es de A(84) a A(181)
Para escritura con i= 27 en A(165) depende de lectura con i=28 en A(165)
En i=2 se lee A(87) que se va escribir en i= 14 A(87) antidependencia

2.7.- Indique cómo paralelizar de la manera más efectiva el bucle siguiente para una máquina con 40 procesadores. Escriba el pseudocódigo para el caso de planificación estática consecutiva y realice una estimación de la aceleración que se obtendrá.

```
do i = 0, 19
        do j = 0, 199
                A(i, j) = A(i, j) + 1
                B(2i+1, j) = B(2i, j) * 2
        enddo
enddo
```

Sol= La aceleración será de 40 si no se tiene en cuenta la creación de los threads.

El Bucle es totalmente paralelizable

2.8.- Considere el siguiente bucle a ejecutar en una máquina paralela:

```
do i = 1, 24
     do j = 1, i
          (1) A(i,j) = B(i,j) + 1
          (2) C(i,j) = A(i-1,j) * 2
     enddo
enddo
```

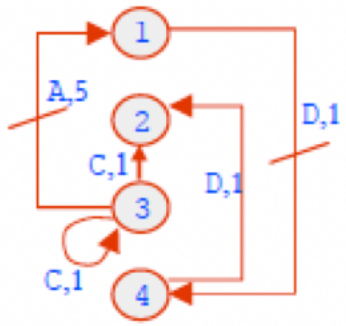
a) Genere el grafo de dependencias y decide cómo paralelizar el código de la manera más eficiente posible. Escribe el código correspondiente al caso de un sistema de 4 procesadores, en el que la planificación de las tareas es dinámica por trozos de tamaño fijo, 4 iteraciones en concreto.

b) Indique el número de tareas de planificación necesarias y haz una estimación razonada del speed-up que se puede conseguir al ejecutar el programa de esa manera. Supón que el coste de ejecución de una instrucción como las del bucle es 1, y que una operación de sincronización tiene un coste de 10.

SOL: b → 6 tareas de planificación $S = 3$, eficiencia = 0,75

2.9.- Dado el siguiente bucle, indique cómo ejecutarlo en paralelo de la manera más eficiente posible. Escriba el código resultante. Si el tiempo de ejecución de cada instrucción es T, realice una estimación de la aceleración y de la eficiencia que se conseguirá al utilizar 10 procesadores. Justifique los resultados.

```
do i = 2, 93
     (1) A(i) = D(i) - 10
     (2) B(i+1) = C(i-1) * D(i-2)
     (3) C(i) = C(i-1) + A(i+5)
     (4) D(i-1) = E(i+1) * 5
enddo
```




```
do i = 2, 93                          ; en serie (un procesador)
    C(i) = C(i-1) + A(i+5)
enddo

D(1) = E(3) * 5                       ; prólogo
B(3) = C(1) * D(0)
B(4) = C(2) * D(1)

doall i = 2, 91                       ; dos iteraciones menos
    A(i) = D(i) - 10
    D(i) = E(i+2) * 5
    B(i+3) = C(i+1) * D(i)
enddoall

A(92) = D(92) - 10                    ; epilogo
A(93) = D(93) - 10
D(92) = E(94) * 5
```

S = 368 T / 125 T = 2,94

2.10.- Dado el siguiente bucle, genera el grafo de dependencias e indica cómo paralelizar el código sin necesidad de sincronización (efectuando peeling.)

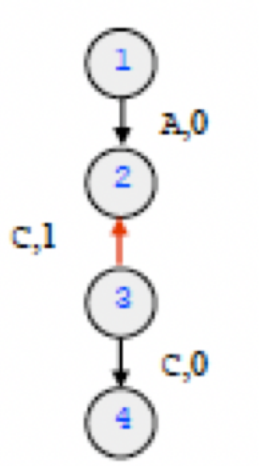
```
do i= 1, 198
        A(i) = B(i-1) * 3
        B(i+1) = C(i-1) + 1
        C(i) = C(i) * 0,5
        enddo
```

SOL:

```
B(2) = C(0) + 1
A(1) = B(0) * 3
A(2) = B(1) * 3
A(3) = B(2) * 3
doall i = 1, 195
   C(i) = C(i) * 0,5
   B(i+2) = C(i) + 1
   A(i+3) = B(i+2) * 3
enddo
C(196) = C(196) * 0,5
C(197) = C(197) * 0,5
B(198) = C(196) + 1
C(198) = C(198) * 0,5
B(199) = C(197) + 1
```

2.11.- Dado el siguiente bucle, genera el grafo de dependencias e indica cómo paralelizar el código sin necesidad de sincronizar los procesos.

```
do i= 1, 1000
        A(i) = A(i) +1
        B(i+1) = C(i)* A(i)
        C(i+1) = C(i+1) +D(i)
        F(i) =C(i+1)/F(i)
enddo
```




```
A(1) = A(1) + 1
B(2) = C(1) * A(1)

doall i = 1, 999
    A(i+1) = A(i+1) + 1
    C(i+1) = C(i+1) + D(i)
    B(i+2) = C(i+1) * A(i+1)
    F(i) = C(i+1) / F(i)
enddoall

C(1001) = C(1001) + D(1000)
F(1000) = C(1001) / F(1000)
```

2.12.- El siguiente bucle se ejecuta en una máquina de 10 procesadores. Analiza cómo paralelizar el bucle de la manera más efectiva y escribe el código que ejecutará cada procesador para el caso de planificación estática consecutiva
Nota:cada procesador se identifica por la variable privada pid = 0..9.

```
do i = 2, 101
        do j = 0, 199
                A(i,j) = A(i,j+2) * 2
                B(i,j) = B(i-2,j) * 3
        enddo
enddo
```

SOL

```
doall i = 2, 101
   do j = 0, 199
      A(i,j) = A(i,j+2) * 2
   enddo
enddoall

doall j = 0, 199
   do i = 2, 101
      B(i,j) = B(i-2,j) * 3
   enddo
enddoall
```

2.13.- Se Se ejecuta en un sistema paralelo de 4 procesadores el siguiente bucle:

```
do i = 1, 40
     <código>
enddo
```

Todas las iteraciones son independientes entre sí y se sabe que el tiempo de ejecución de las mismas es T (i)) = i (es decir, la primera iteración 1, la segunda 2, etc.).
Calcula el tiempo de ejecución en paralelo del programa, y el factor de aceleración respecto al caso serie, si la panificación de tareas es:

a) estática: 1. consecutiva; y 2. entrelazada

b) dinámica: 1. Autoplanificación con: Z = 1 y Z = 5
2. autoplanificación guiada, Z_s = ceil [(N – i) / P]
3. trapezoidal, $Z_1 = 9$, k = 1

Considere que el tiempo de cada operación de planificación dinámica es 1.

Sol:
a1 → *S*= 2,3 *eficiencia* = 0,58
a2 → *S*= 3,8 *eficiencia* = 0,94
b1 → *S*= 3,6 *eficiencia* = 0,9 y S = 2,9 *eficiencia* = 0,73
b2 → *S* = 3,6 eficiencia = 0,9
b3 → *S* = 3,2 eficiencia = 0,79

2.14.- Se ejecutan en paralelo, entre 4 procesadores, 100 iteraciones de un determinado bucle, independientes entre sí.
El tiempo de ejecución de cada iteración es de 1 s, salvo una de las iteraciones, que tiene un tiempo de ejecución de 20 s.
Calcula, de manera aproximada, el speed-up y la eficiencia que se consigue en los siguientes casos:

(a) el reparto de iteraciones es estático consecutivo (con trozos del mismo tamaño).
(b) el reparto de iteraciones es estático entrelazado (con trozos del mismo tamaño).
(c) el reparto de iteraciones es dinámico, 1 a 1 (calcula el peor y el mejor caso); coste de una operación de asignación de tareas, 50 ms.

SOL:

a) S= 119/44 = 2,7 eficiencia 0,68

b) S= 119/44 = 2,7 eficiencia 0,68

c) Caso mejor S= 119/31 = 3,8 eficiencia 0,96
Caso mejor S= 119/45,25 = 2,6 eficiencia 0,66