Tema IV: Algoritmos y Computación MIMD

Básicamente existen tres tipos de algoritmos:

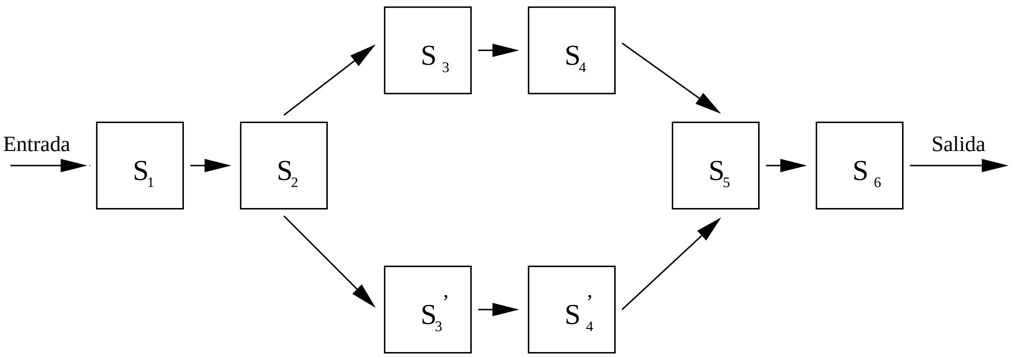
- De tubería (*pipelined*).
- De partición.
- De relajación (*asincronizados*).

Algoritmos MIMD de tubería.

Conjunto ordenado de segmentos tales que la salida de uno de ellos es entrada del siguiente.

La entrada se hace por el primer segmento y la salida por el último.

En principio, pueden ejecutarse simultáneamente dos segmentos tales que la salida de uno sea irrelevante para la entrada del otro.



Ejemplo 1: Sumar cuatro matrices cuadradas de orden n:

$$A = (a_{ij}),\ B = (b_{ij}),\ C = (c_{ij}),\ D = (d_{ij}).$$

(a) Solución secuencial.

para $i \leftarrow 1$ **hasta** n **hacer**

  para $j \leftarrow 1$ **hasta** n **hacer**

    $x_{ij} \leftarrow a_{ij} + b_{ij} + c_{ij} + d_{ij}$

devolver (x_{ij})

Número de operaciones: $3 \cdot n^2$.

(b) Solución **SIMD–PRAM** con n^2 procesadores.

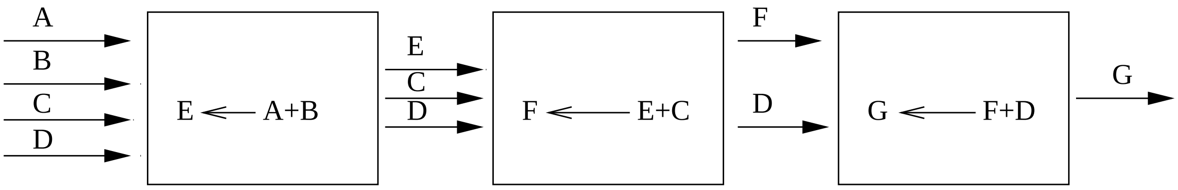
para cada P_{ij} tal que $1 \leq i, j \leq n$ **hacer**

  $x_{ij} \leftarrow a_{ij} + b_{ij} + c_{ij} + d_{ij}$

devolver (x_{ij})

Número de pasos: 3.

(c) Solución **MIMD–Pipelined** con 3 procesadores.



- Paso 1

$$P_1 \left\{ \begin{array}{ll} \text{recibe} & a_{11}, b_{11}, c_{11}, d_{11} \\ \text{ejecuta} & e_{11} \leftarrow a_{11} + b_{11} \\ \text{pasa} & e_{11}, c_{11}, d_{11} \text{ a } P_2 \end{array} \right.$$

- Paso 2

$$P_1 \left\{ \begin{array}{ll} \text{recibe} & a_{12}, b_{12}, c_{12}, d_{12} \\ \text{ejecuta} & e_{12} \leftarrow a_{12} + b_{12} \\ \text{pasa} & e_{12}, c_{12}, d_{12} \text{ a } P_2 \end{array} \right.$$

$$P_2 \left\{ \begin{array}{ll} \text{recibe} & e_{11}, c_{11}, d_{11} \\ \text{ejecuta} & f_{11} \leftarrow e_{11} + c_{11} \\ \text{pasa} & f_{11}, d_{11} \text{ a } P_3 \end{array} \right.$$

- Paso 3

$$P_1 \begin{cases} \text{recibe} & a_{13}, b_{13}, c_{13}, d_{13} \\ \text{ejecuta} & e_{13} \leftarrow a_{13} + b_{13} \\ \text{pasa} & e_{13}, c_{13}, d_{13} \text{ a } P_2 \end{cases}$$

$$P_2 \begin{cases} \text{recibe} & e_{12}, c_{12}, d_{12} \\ \text{ejecuta} & f_{12} \leftarrow e_{12} + c_{12} \\ \text{pasa} & f_{12}, d_{12} \text{ a } P_3 \end{cases}$$

$$P_3 \begin{cases} \text{recibe} & f_{11}, d_{11} \\ \text{ejecuta} & g_{11} \leftarrow f_{11} + d_{11} \\ \text{devuelve} & g_{11} \end{cases}$$

Número de pasos: $n^2 + 2$.

Ejemplo 2: Resolver el sistema de ecuaciones lineales

$$
\begin{array}{lcl}
a_{11}x_1 & = & b_1 \\
a_{21}x_1 + a_{22}x_2 & = & b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & = & b_3 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & & \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n & = & b_n
\end{array}
$$

Admitiendo que $a_{ii} \neq 0$, para cada i,

$$x_i = \frac{b_i - \displaystyle\sum_{k=1}^{i-1} a_{ik}x_k}{a_{ii}}$$

(a) Solución secuencial.

para $i \leftarrow 1$ **hasta** n **hacer**

$suma \leftarrow 0$

para $j \leftarrow 1$ **hasta** $i-1$ **hacer**

$suma \leftarrow suma + a_{ij} \cdot x_j$

$x_i \leftarrow \frac{(b_i - suma)}{a_{ii}}$

(b) Solución **MIMD–pipelined** con n procesadores.

- A cada procesador P_i $(1 \leq i \leq n)$ le asociamos el proceso i: hallar x_i en la ecuación

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ii}x_i = b_i$$

  Necesita la salida de $P_1, \ldots, P_{i-1}$.

Proceso $pipelined(i)$

$suma \leftarrow 0$

para $j \leftarrow 1$ **hasta** $i - 1$ **hacer**

leer el valor x_j de la izquierda

enviar el valor x_j a la derecha

$suma \leftarrow suma + a_{ij} \cdot x_j$

$x_i \leftarrow \frac{(b_i - suma)}{a_{ii}}$

enviar el valor x_i a la derecha

Tiempo de ejecución: $\theta(n)$.

- Pasos que necesita P_n para recibir x_1: n.
- Operaciones que realiza $P(n)$: $\theta(n)$.

Algoritmos sistólicos

- Consta de un gran número de procesos (EP) similares conectados para intercambiar datos.

- Los datos se mueven a velocidad constante.

- Se trata de un modelo sincronizado: reloj global. En cada *tick* del reloj
  - ⋆ Cada EP realiza la computación sobre los datos recibidos en el tick anterior y/o los datos que tuviere.
  - ⋆ El dato producido lo transmitirá en el siguiente tick.

- El flujo de datos es discontinuo.

- Movimiento parecido al que realiza la sangre en los distintos latidos del corazón.

Ejemplo: *Algoritmo sistólico hexagonal para multiplicar matrices.*

Sean $A = (a_{ij}), B = (a_{ij})$ y $C^0 = (c^0_{ij})$ matrices cuadradas del mismo orden. Se trata de hallar $C^0 + A \cdot B$.

Consideramos una serie de EP que funcionan así:

- Las componentes a_{ij}, b_{ij}, c^0_{ij} son suministradas a los EP.
- Los datos a y b salen inalterados.
- El dato c sale como $c' \leftarrow c + a \cdot b$.

Descripción del algoritmo para el caso de matrices cuadradas de orden tres.

Algoritmos de partición

En el diseño de estos algoritmos:

- El ejemplar del problema se *descompone* en subejemplares.
- Se *resuelven* paralela e independientemente los subejemplares.
- Se *combinan* las soluciones parciales para obtener una solución global del ejemplar.

Paralelización de la técnica *divide y vencerás*.

Ejemplo: hallar la suma de $n = k \cdot r$ números enteros $a_1, a_2, \ldots, a_{k \cdot r}$, con k procesadores P_i.

- Inicializamos en 0 una variable global, S.
- Creamos k procesos, uno por cada P_i, que consisten en *sumar r números*.
- Se ejecutan en paralelo los k procesos.
- Al terminar un procesador su proceso, hay que añadir su valor a S (entra en la *sección crítica*).
- La variable S proporciona el dato de salida.

Descripción del pseudocódigo:

procedimiento $suma_{particion}(A[1,\ldots,k\cdot r])$

$S \leftarrow 0$

para cada P_i tal que $1 \leq i \leq k$ **hacer**

$S_i \leftarrow suma^*(A[(i-1)\cdot r+1,\ldots,i\cdot r])$

lock(S)

$S' \leftarrow 0$

para $i \leftarrow 1$ **hasta** k **hacer**

$S' \Longleftarrow S_i$

$S \leftarrow S + S'$

unlock(S)

devolver S

En donde $suma^*$ es un algoritmo secuencial para sumar números enteros y S' es una nueva variable global que implementa la sección crítica.

Tiempo de ejecución: $\theta(\frac{n}{k}+k)$

Número óptimo de procesadores: $\theta(\sqrt{n})$

Algoritmos de relajación

- Asincronizados.
- Capacidad de los procesadores de trabajar con los datos disponibles en ese instante.
- En estos algoritmos los procesadores pueden
  - ⋆ Trabajar juntos en una tarea común (como los de partición).
  - ⋆ Especializarse en alguna tarea específica (como los de tubería).

Nota: Existen algoritmos de tubería en donde algunos segmentos pueden ser paralelizados vía partición, y otros vía relajación.