Pipeline

Maximiliano A. Eschoyez
Programación Eficiente — UBP

2010

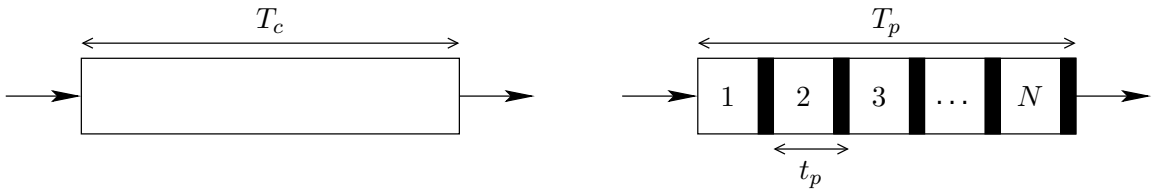
Dentro de una computadora cada evento está gobernado por el reloj de la CPU. Nada se puede mover a mayor o menor velocidad. El ciclo de reloj determina la velocidad de trabajo.

Dado que no se puede aumentar indefinidamente la velocidad de trabajo, se pueden realizar algunos trucos para mejorar la performance general sin incrementar el ciclo de reloj.

Una de la técnicas utilizadas se denomina “Pipeline”. Esta técnica consiste en dividir un bloque de procesamiento en subbloques menores y hacer trabajar en paralelo a todos ellos. En la versión original del bloque, cuando ingresa un dato lo bloquea hasta que se termine el procesamiento. En la versión pipeline cada dato que ingresa ocupa un único subbloque, permitiendo procesar parcialmente hasta n datos al mismo tiempo (siendo n la cantidad de subbloques).

El nombre “Pipeline” deja a las claras la forma de trabajo: los datos egresan en el orden que ingresaron, como si los datos fueran bolitas que introducimos en una manguera.

La ventaja principal de esta forma de procesamiento es que se consigue una mejor tasa de generación de resultados sin necesidad de recaer en técnicas de procesamiento más costosas y complejas.



(a) Bloque monolítico (b) Bloque con Pipeline

Figura 1: Bloque monolítico versus Bloque con Pipeline

T_c = Latencia de la lógica monolítica
T_p = Latencia del Pipeline completo
t_p = Latencia para cada subbloque del Pipeline
N = Cantidad de subbloques que conforman el Pipeline

En el caso ideal, todos los subbloques poseen el mismo retardo. En la realidad, algunos subbloques son más veloces que otros, sin embargo, el tiempo de respuesta está condicionado por el subbloque más lento.

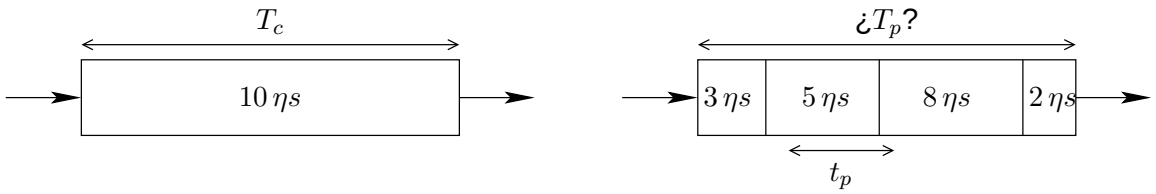
$$t_p < T_c$$
$$T_p = N \times t_p$$
$$T_c < T_p$$

Por lo tanto, la performance (throughput) es:

$$\text{Perf}_c = \frac{1}{T_c} \tag{1}$$

$$\text{Perf}_p = \frac{1}{t_p} \tag{2}$$

Ejemplo Pipeline Ideal de Bloques con Igual Retardo



(a) Bloque monolítico (b) Bloque con Pipeline

Figura 2: Diagramas del ejemplo 1.

Cuando ingresa un único dato (lo mismo para la espera del primer resultado), el tiempo de espera es:

$$\begin{aligned} T_c &= 10\,\eta s \\ T_p &= N \times t_p \\ &= 4 \times 5\,\eta s = 20\,\eta s \end{aligned}$$

Por lo tanto, la performance (throughput) es:

$$\begin{aligned} \text{Perf}_c &= \frac{1}{T_c} = \frac{1}{10\,\eta s} = 100\,Mhz \\ \text{Perf}_p &= \frac{1}{t_p} = \frac{1}{5\,\eta s} = 200\,Mhz \end{aligned}$$

En el caso de tener que procesar 1000 datos tendríamos:

$$\begin{aligned} T_c &= 1000 \times 10\,\eta s = 10\,ms \\ T_p &= (1000 + 3) \times 5\,\eta s = 5,015\,ms \approx 5\,ms \end{aligned}$$

Donde el valor 3 está representando el retardo de espera para que se genere el primer resultado.

La línea de tiempo de ambos bloques será la siguiente:

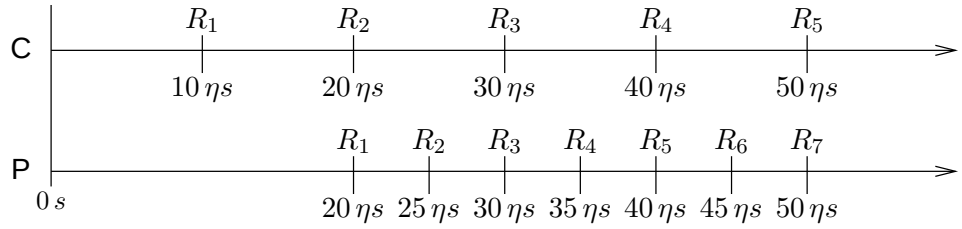
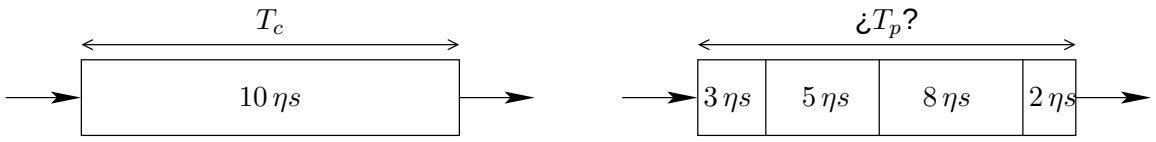


Figura 3: Línea de tiempo del ejercicio 1.

Ejemplo Pipeline Ideal de Bloques con Distinto Retardo



(a) Bloque monolítico (b) Bloque con Pipeline

Figura 4: Diagramas del ejemplo 2.

En el caso que los subbloques sean de distinto retardo, el bloque más lento es el que determina la frecuencia de trabajo del Pipeline.

Cuando ingresa un único dato (lo mismo para la espera del primer resultado), el tiempo de espera es:

$$\begin{aligned} T_c &= 10\,\eta s \\ T_p &= N \times t_p \\ &= 4 \times 8\,\eta s = 32\,\eta s \end{aligned}$$

Por lo tanto, la performance (throughput) es:

$$\text{Perf}_c = \frac{1}{T_c} = \frac{1}{10\,\eta s} = 100\,Mhz$$
$$\text{Perf}_p = \frac{1}{t_p} = \frac{1}{5\,\eta s} = 125\,Mhz$$

En el caso de tener que procesar 1000 datos tendríamos:

$$\begin{aligned} T_c &= 1000 \times 10\,\eta s = 10\,ms \\ T_p &= (1000 + 3) \times 8\,\eta s = 8,024\,ms \approx 8\,ms \end{aligned}$$

Donde el valor 3 está representando el retardo de espera para que se genere el primer resultado.

La línea de tiempo de ambos bloques será la siguiente:

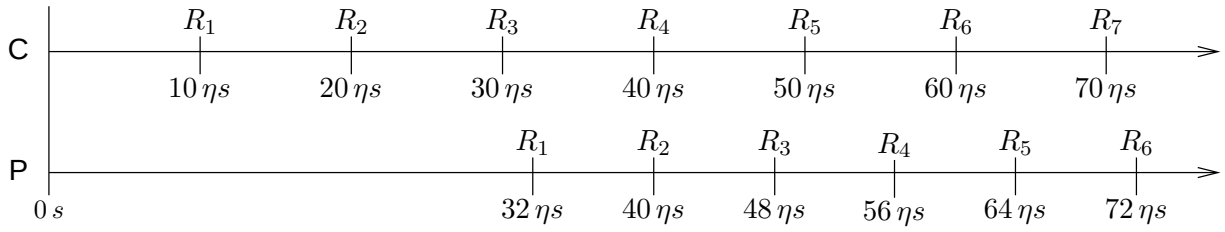


Figura 5: Línea de tiempo del ejercicio 2.