Capítulo 3

Multiprocesadores y multicomputadoras

3.1. Introducción

Las computadoras paralelas pueden clasificarse *grosso modo* en dos grupos:

1. *Multiprocesadores*, con arquitectura de memoria *compartida*. Se componen de un arreglo de procesadores y un banco de memoria común conectados a través de un *bus* rápido o uno de los varios tipos de red de interconexión *dinámica multietapas*. En esta arquitectura los procesadores se comunican mediante la lectura o escritura a la memoria común o compartida.

2. *Multicomputadoras*, con arquitectura de memoria *distribuida*, es decir, no compartida. En esta arquitectura se tiene un arreglo de microprocesadores, cada uno con su propia memoria local, interconectados en un esquema *estático*. Los procesadores se comunican mediante el envío y recepción de mensajes.

En este capítulo estudiaremos las características principales de ambas arquitecturas sin entrar en detalles de alguna máquina comercialmente disponible. Resulta importante destacar que en las implementaciones desarrolladas en el transcurso del presente trabajo, ha sido utilizada una *multicomputadora de componentes comunes*, cuya arquitectura también ha sido denominada de *tipo Beowulf* [11].

3.2. Multicomputadora de hipercubo

3.2.1. Estructura recursiva del hipercubo

Sean N y b dos enteros tales que $N \geq b \geq 2$. Sea k un entero tal que $b^{k-1} < N \leq b^k$. Sea $\langle m \rangle = \{0, 1, 2, \cdots, m-1\}$.

Un hipercubo de N nodos en base b, denotado como *hipercubo* (N, b, k), es un grafo $G = (V, E)$, donde

$$V = \{x \mid x = (x_k x_{k-1} \cdots x_1)_b \wedge x_i \in \langle b \rangle\}$$

es decir, los nodos están etiquetados con números en base b, y

$$E = \{(x, y) \mid x, y \in V \wedge \exists (1 \leq j \leq k)(x_j \neq y_j \wedge \forall (i \neq j) x_i = y_i)\}.$$

Esto es, dos nodos en G están conectados si y solo si sus etiquetas difieren en exactamente un dígito en base b.

Si $N = b^k$ entonces el grafo G es un hipercubo (N, b, k) *completo* de *dimensión* k o simplemente un cubo (N, b, k). En cualquier otro caso, G es denominado un hipercubo (N, b, k) *incompleto*.

El hipercubo (N, b, k) admite una caracterización recursiva simple. Si A es un conjunto de cadenas sobre $\langle b \rangle$, entonces definimos

$$B = xA = \{x\alpha \mid \alpha \in A \wedge x \in \langle b \rangle\} \tag{3.1}$$

donde $x\alpha$ es la cadena que resulta de concatenar x y α. Definamos una secuencia de conjuntos $V^{(i)}$, para $i = 0, 1, \cdots, k-1$, donde

$$V^{(i)} = \bigcup_{x \in \langle b \rangle} V_x^{(i)}, \tag{3.2}$$

y $V_x^{(i)}$ se define recursivamente como

$$V_x^{(0)} = \{x\}, \tag{3.3}$$

y

$$V_x^{(i)} = xV^{(i-1)} \tag{3.4}$$

para toda $x \in \langle b \rangle$. Se deduce que $V_x^{(i)}$ consiste en el conjunto de cadenas de longitud $i+1$, cuyo caracter más a la izquierda es x. Además, para toda $x \in \langle b \rangle$

$$|V_x^{(0)}| = 1 \quad \text{y} \quad |V^{(0)}| = b, \tag{3.5}$$

y, para toda $i \geq 1$ y $x \in \langle b \rangle$,

$$|V_x^{(i)}| = b^i \quad \text{y} \quad |V^{(i)}| = b^{i+1}. \tag{3.6}$$

Sea $G^{(0)}$ definido como

$$G^{(0)} = (V^{(0)}, E^{(0)}), \tag{3.7}$$

donde $V^{(0)}$ se define como en (3.2) y

$$E^{(0)} = \left\{ (s,t) \mid s,t \in V^{(0)}, s < t \right\}. \tag{3.8}$$

Se puede observar que $G^{(0)}$ es un grafo completo sobre $V^{(0)}$ y que

$$|E^{(0)}| = \frac{b(b-1)}{2}. \tag{3.9}$$

Definamos ahora

$$G_x^{(1)} = (V_x^{(1)}, E_x^{(1)}), \tag{3.10}$$

donde $V_x^{(1)}$ está definido de (3.2) a (3.4), y

$$E_x^{(1)} = \left\{ (x\alpha, x\beta) \mid (\alpha, \beta) \in E^{(0)} \right\}. \tag{3.11}$$

Esto es, $G_x^{(1)}$ se obtiene simplemente de añadir el prefijo x a cada uno de los nodos de $G^{(0)}$ y, como consecuencia, $G_x^{(1)}$ es isomorfo a $G^{(0)}$, para toda $x \in \langle b \rangle$.

Sea

$$G^{(1)} = (V^{(1)}, E^{(1)}), \tag{3.12}$$

donde $V^{(1)}$ está definido de (3.2) a (3.4), y

$$E^{(1)} = \bigcup_{x \in \langle b \rangle} E_x^{(1)} \cup \left\{ (s\alpha, t\beta) | \alpha \in V^{(0)}; s,t \in V^{(0)} \wedge s < t \right\}. \tag{3.13}$$

Es evidente que

$$|E^{(1)}| = b^2(b-1). \tag{3.14}$$

En otras palabras, primero tomamos b copias de $G^{(0)}$ y las renombramos como $G_x^{(1)}$ para $x \in \langle b \rangle$. Si consideramos cada uno de esos $G_x^{(1)}$ como un *supernodo*, se obtiene $G^{(1)}$ construyendo un cubo de base b sobre ellos. Un supernodo $G_x^{(1)}$ está conectado a un supernodo $G_y^{(1)}$ si y solamente si un

nodo $x\alpha$ en $G_x^{(1)}$ está conectado a su *correspondiente* nodo $y\alpha$ en $G_y^{(1)}$, para cada α en $V^{(0)}$.

En general, $G^{(i)}$ está definido por

$$G^{(i)} = (V^{(i)}, E^{(i)}), \tag{3.15}$$

donde $V^{(i)}$ está definido de (3.2) a (3.4), y

$$E^{(i)} = \bigcup_{x \in \langle b \rangle} E_x^{(i)} \cup \left\{ (s\alpha, t\beta) | \alpha \in V^{(i-1)}; s, t \in V^{(0)} \wedge s < t \right\}. \tag{3.16}$$

De (3.16) se tiene la relación de recurrencia

$$|E^{(i)}| = b|E^{(i-1)}| + \frac{b^{i+1}(b-1)}{2}, \tag{3.17}$$

cuya solución es

$$|E^{(i)}| = \frac{i+1}{2} b^{i+1}(b-1). \tag{3.18}$$

Teorema 3.1 *Para $j \geq 1$, el grafo $G^{(j-1)}$ definido en forma recursiva anteriormente, es un cubo (b^j, b, j).*

Demostración
(Por inducción) De las definiciones (3.7) y (3.8) se sigue que $G^{(0)}$ es un cubo $(b, b, 1)$. Supongamos que el enunciado del teorema se cumple para $j = i-1$. Para mostrar que se cumple en el caso de $j = i$, considérese

$$G^{(i)} = (V^{(i)}, E^{(i)}).$$

De acuerdo a (3.16), las aristas de $E^{(i)}$ se dividen en dos grupos: las que están en $G_x^{(i)}$ y las que interconectan diferentes $G_x^{(i)}$. Como las $G_x^{(i)}$ son isomorfos a $G^{(i-1)}$, por hipótesis de inducción todas las aristas en $E_x^{(i)}$ satisfacen la definición del hipercubo. Por otra parte, por la manera en que están definidas, las aristas del segundo grupo satisfacen la definición del hipercubo dada al inicio de esta sección. □

Teorema 3.2 *Sean d, k y m enteros positivos. Entonces un cubo (d^{mk}, d^m, k) contiene un cubo $(d^{mk}, d^{m/p}, kp)$ para todo entero positivo p que divide a m.*

Demostración
Considérense dos nodos x y y en el cubo (N, d^s, t) con $s < m$. Sean $x = x_t x_{t-1} \cdots x_2 x_1$ y $y = y_t y_{t-1} \cdots y_2 y_1$ donde x_i y y_i pertenecen a $\langle d^s \rangle$. Considerando al mismo par de nodos en el cubo completo (N, d^m, k), sean $x = \bar{x}_k \bar{x}_{k-1} \cdots \bar{x}_2 \bar{x}_1$ y $y = \bar{y}_k \bar{y}_{k-1} \cdots \bar{y}_2 \bar{y}_1$ donde $\bar{x}_i$ y $\bar{y}_i$ pertenecen a $\langle d^m \rangle$. Dado que $s < m$ y $m = ps$, se tiene que $\langle d^s \rangle$ es un subconjunto de $\langle d^m \rangle$ y que $\bar{x}_1 = x_p x_{p-1} \cdots x_1$, $\bar{x}_2 = x_{2p} x_{2p-1} \cdots x_{p+1}$ y, en general, para $1 \leq i \leq k$,

$$\bar{x}_i = x_{ip} x_{ip-1} \cdots x_{(i-1)p+1}.$$

De forma similar se relacionan las $\bar{y}_i$ y las y_i.

Existe una arista que conecta a x y y en el cubo (N, d^s, t) si y solo si existe $1 \leq q \leq t$ tal que $x_q \neq y_q$ y $x_i = y_i$ para $i \neq q$. Por lo tanto

$$\bar{x}_{\lceil q/p \rceil} \neq \bar{y}_{\lceil q/p \rceil} \text{ y } \bar{x}_j = \bar{y}_j \text{ para toda } j \neq \left\lceil \frac{q}{p} \right\rceil,$$

esto es, x y y son nodos adyacentes en el cubo (N, d^m, k). □

3.2.2. Propiedades topológicas de un cubo (N, b, k)

A continuación enunciaremos algunas de las propiedades topológicas mas relevantes de los hipercubos (N, b, k) completos.[1]

Lema 3.3 *Si $b = N$, entonces el cubo $(N, N, 1)$ es un grafo completo sobre N nodos.*

Demostración
Se sigue inmediatamente de la definición del hipercubo. □

Teorema 3.4 *En un cubo (N, b, k) completo,*

1. *El grado de cada nodo es $(b-1)\log_b N$, y*
2. *Existen un total de $\frac{1}{2}(b-1)N \log_b N$ aristas.*

Demostración
Para cada enunciado del teorema tenemos:

1. Los nodos en $G^{(k-1)}$ están etiquetados con enteros de k dígitos en la base b. Por construcción, un nodo está conectado por una arista a cada uno de los nodos cuyas etiquetas difieren en exactamente un dígito. Dado que hay $b-1$ nodos que difieren en un dígito dado y hay un total de $k = \log_b N$ dígitos, el grado de cada nodo es $(b-1)\log_b N$.

[1] Para una exposición más detallada consultar [54], pp. 67-98.

2. De lo anterior, y dado que hay un total de N nodos, se sigue que el número total de aristas es $\frac{1}{2}(b-1)N\log_b N$.

□

Teorema 3.5 *El* diámetro *de un cubo* (N, b, k) *es* $\log_b N$.

Demostración
El diámetro es el máximo de la distancia más corta entre un par de nodos en $G^{(k-1)}$. Es evidente que este máximo se tiene cuando las etiquetas de un par de nodos difieren en cada uno de los k dígitos. Como hay $k = \log_b N$ dígitos, el teorema se cumple. □

Como el diámetro está relacionado al retraso de las comunicaciones entre nodos, los algoritmos implementados en cubos de base b, para $b > 2$, tendrán menor retraso en las comunicaciones que los cubos de base 2.

El siguiente teorema nos proporciona un algoritmo para construir un esquema de interconexión de N procesadores con un diámetro determinado de antemano.

Teorema 3.6 *Dados* N *y* d, *existen* k *y* b *tales que el diámetro del cubo* (N, b, k) *es menor o igual a* d.

Demostración
Definiendo $b = \lceil N^{1/d} \rceil$, se construye recursivamente el cubo (b^d, b, d). Si se remueven los nodos de $(N+1)$ a b^d con las correspondientes aristas, el grafo resultante tendrá la propiedad requerida. □

A continuación, estableceremos algunas propiedades de las trayectorias en un cubo (N, b, k). Sean x y y un par de nodos cualquiera, donde $x = x_k x_{k-1} \cdots x_1$ y $y = y_k y_{k-1} \cdots y_1$.

Sea $H_b(x, y)$ la distancia de Hamming entre x y y, definida como el número de dígitos base b en los cuales difieren x y y.

Por ejemplo, $H_2(1110, 1001) = 3$.

Definimos $x|y_j$ como el nodo cuya etiqueta se obtiene reemplazando al dígito x_j en x por y_j. De manera similar, definimos $x|x_i \leftarrow j$ como el nodo cuya etiqueta se obtiene al reemplazar el i-ésimo dígito x_i por $j \in \langle b \rangle$.

Teorema 3.7 *Sea* $H_b(x, y) = t$. *Entonces,*

1. *La distancia mínima entre el nodo* x *y el nodo* y *es* t.

2. *Existen un total de* $(b-1)k$ *trayectorias disjuntas entre el nodo* x *y el nodo* y, *de las cuales*

a) *t trayectorias son de longitud t,*

b) $t(b-2)$ *trayectorias son de longitud* $t+1$*, y*

c) $(k-t)(b-1)$ *trayectorias son de longitud* $t+2$*,*

donde $b^k = N$*.*

Demostración

Sean $i_1 < i_2 < \cdots < i_t$ las posiciones de los t dígitos en los cuales difieren x y y. La trayectoria más corta entre x y y se obtiene al ir reemplazando sucesivamente en x los t dígitos en los cuales difiere de y. Por lo tanto, una trayectoria de x a y es

$$x \to x|y_{i_1} \to (x|y_{i_1})|y_{i_2} \to \cdots \to (\cdots((x|y_{i_1})|y_{i_2})\cdots)|y_{i_t} = y.$$

Si en lugar de iniciar en el dígito i_1, iniciamos primero en el dígito i_j y convertimos hasta el dígito i_t, para después corregir del dígito i_1 hasta el i_{j-1}, podremos generar las t trayectoria de x a y. Es claro, por la manera en que se construyen las trayectoria, que son disjuntas. Con ésto, se prueban tanto el inciso (1) como el inciso $(2)(a)$. Ahora, para demostrar el inciso $(2)(b)$, tomemos la trayectoria

$$x \to (x|x_{i_j} \leftarrow s) \to \cdots \to (y|y_{i_j} \leftarrow s) \to y,$$

donde $x_{i_j} \neq s$ y $y_{i_j} \neq s$, y $s \in \langle b \rangle$.

Como

$$H_b(x|x_{i_j} \leftarrow s, y|y_{i_j} \leftarrow s) = t - 1,$$

la trayectoria anterior es de longitud $t+1$. Además, existen $t-1$ trayectorias disjuntas entre $(x|x_{i_j} \leftarrow s)$ y $(y|y_{i_j} \leftarrow s)$. Dado que existen $b-2$ posibles valores para s, variando el valor de j de 1 hasta t obtenemos las $t(b-2)$ trayectorias de longitud $t+1$.

Para demostrar $(2)(c)$, sean

$$m_1 < m_2 < \cdots < m_{k-t}$$

tales que

$$x_{m_j} = y_{m_j} \text{ para } j = 1, \cdots, k-t.$$

Tomando la trayectoria

$$x \to (x|x_{m_j} \leftarrow s) \to \cdots \to (y|y_{m_j} \leftarrow s) \to y,$$

donde $x_{m_j} \neq s$ y $y_{m_j} \neq s$, y $s \in \langle b \rangle$.

Es evidente que

$$H_b(x|x_{m_j} \leftarrow s, y|y_{m_j} \leftarrow s) = t,$$

y en consecuencia la trayectoria anterior es de longitud $t+2$. Dado que hay $b-1$ valores para s, cambiando j de 1 hasta $k-t$, obtenemos $(k-t)(b-1)$ trayectorias disjuntas de longitud $t+2$. □

Obsérvese que como corolario del teorema anterior tenemos que si $b = 2$ entonces no existen trayectorias de longitud $t+1$. Más aun, en un cubo de base 2, si $t < \log_2 N$, entonces existen t trayectorias de longitud t y $\log_2 N - t$ trayectorias de longitud $t+2$.

Teorema 3.8 *Sean x e y dos nodos en un cubo (N, b, k), donde $H_b(x, y) = t$. Entonces, existen*

1. $\binom{t}{2}$ *ciclos pares de longitud* $2t$.
2. $\binom{t(b-2)}{2} + t(k-t)(b-1)$ *ciclos pares de longitud* $2(t+1)$.
3. $\binom{(k-t)(b-1)}{2}$ *ciclos pares de longitud* $2(t+2)$.
4. $t^2(b-2)$ *ciclos impares de longitud* $2t+1$.
5. $t(k-b)(b-1)(b-2)$ *ciclos impares de longitud* $2t+3$.

Demostración
Puesto que dos trayectorias de longitud t pueden combinarse para obtener un ciclo de longitud $2t$, se cumple el inciso (*1*). Cualquier trayectoria de longitud t de x a y puede combinarse con una trayectoria de longitud $t+1$ para obtener un ciclo impar de longitud $2t+1$. Debido a que hay t trayectorias del primer tipo y $t(b-2)$ del segundo, se cumple el inciso (*4*). De manera similar pueden mostrarse (*2*), (*3*) y (*5*) □

Como corolario del teorema anterior tenemos que en un cubo de base 2 no existen ciclos impares.

3.2.3. Incrustamiento de topologías simples

Debido a que en fechas recientes se tienen disponibles varias computadoras paralelas cuyas arquitecturas tienen un esquema de interconexión de hipercubo, resulta de particular interés el portar a este tipo de arquitectura, algoritmos basados en otras topologías, tales como arreglos lineales, anillos, mallas bidimensionales o árboles binarios completos.

El problema del mapeo de algoritmos paralelos en arquitecturas paralelas está relacionada al problema de incrustamiento de grafos. Se conoce desde hace bastante tiempo que el problema general de incrustamiento de grafos es NP-completo. Más aun, el incrustamiento de árboles arbitrarios sobre hipercubos binarios es también NP-completo. Sin embargo, pueden deducirse algoritmos de incrustamiento para grafos altamente estructurados, tales como los arreglos, anillos y árboles binarios completos.

Sea $G_g = (V_g, E_g)$ el grafo a ser incrustado en el grafo $G_h = (V_h, E_h)$. Al primer grafo le llamaremos *huésped* y al segundo grafo *anfitrión*. Se supone que el anfitrión es precisamente el cubo (N, b, k) y que $|V_g| \leq |V_h|$. Sea $d(x, y)$ la trayectoria más corta, medida en términos del número de aristas, entre los nodos x y y del grafo G_g. Sea $f : V_g \to V_h$ una función de incrustamiento tal que $f(x) \neq f(y)$ si $x \neq y$, es decir, f es inyectiva. El factor de *dilatación* $D_f(x, y)$ para $x \neq y$ está definido por

$$D_f(x, y) = \frac{H_b(f(x), f(y))}{d(x, y)}.$$

Sea

$$D_f = \max_{x \neq y}(D_f(x, y)). \tag{3.19}$$

Es obvio que si $D_f = 1$ entonces las aristas en G_g son mapeadas a aristas en G_h y que la función de incrustamiento preserva la propiedad de vecindad del grafo que es incrustado. El cociente

$$E_f = \frac{|V_h|}{|V_g|} \geq 1 \tag{3.20}$$

es denominado el *factor de expansión*.

Los códigos binarios de Gray reflejados han sido utilizados como una herramienta para el incrustamiento de arreglos, anillos y mallas en hipercubos binarios. Para extender estos incrustamientos a un cubo (N, b, k), necesitaremos una clase de códigos generalizados, denominados códigos de Gray de base b.

Códigos de Gray de base b

Supongamos que $b = 2^m$, $m \geq 1$, $N = 2^{km}$ y $n = km$. Se define

$$G^{(2)}(1) = \{0, 1\}.$$

Sea

$$G^{(2)}(n) = \{G_0^{(2)}(n), G_1^{(2)}(n), \cdots, G_{2^n-1}^{(2)}(n)\},$$

donde $G_i^{(2)}(n)$ son las codificaciones binarias (en n bits) del entero i, para $i = 0, \cdots, 2^n - 1$.

Entonces definimos $G^{(2)}(n+1)$ recursivamente como se muestra enseguida:

$$G^{(2)}(n+1) = \{0G_0^{(2)}(n), \cdots, 0G_{2^n-1}^{(2)}(n), 1G_{2^n-1}^{(2)}(n), \cdots, 1G_0^{(2)}(n)\}.$$

Puede verse que

$$|G^{(2)}(n)| = 2^n.$$

Sea $i \in \langle N \rangle$, donde $i = i_n i_{n-1} \cdots i_1 i_0$ en binario. Debido a que $0 \leq i < 2^n$, se tiene que $i_n = 0$.

Sea

$$G_i^{(2)}(n) = g_n g_{n-1} \cdots g_2 g_1$$

y sean

$$E^{(2)} : \langle N \rangle \to G^{(2)}(n)$$

la función de codificación y

$$D^{(2)} : G^{(2)}(n) \to \langle N \rangle$$

la función decodificadora. Si $\oplus$ denota la operación binaria de OR exclusivo, puede demostrarse que

$$E^{(2)} = G_i^{(2)}(n),$$

donde $g_j = i_j \oplus i_{j-1},$ para toda $j = 1, 2, \cdots, n$, y

$$D^{(2)}(G_i^{(2)}(n)) = i,$$

donde $i_j = g_{j+1} \oplus \cdots \oplus g_n$ para toda $j = 0, \cdots, n-1$.

Una propiedad de los códigos de Gray arriba expuestos es que dos palabras sucesivas del código difieren en exactamente un bit y por consiguiente la distancia de Hamming entre palabras sucesivas es la unidad. Este resultado se extiende en el siguiente lema.

Lema 3.9 *Sea $i, j \in \langle N \rangle$. Si $j = (i + 2^d) \bmod N$, para alguna $0 < d < n$, entonces*

$$H_2(G_i^{(2)}(n), G_j^{(2)}(n)) = 2,$$

donde $H_2(x, y)$ es la distancia de Hamming entre las cadenas binarias x e y.

Demostración
Sea $s \geq d$ la posición del bit en la cual el acarreo, al sumar 2^d a i, deja de propagarse. Entonces se tiene que si

$$i = i_n i_{n-1} \cdots i_{s+1} i_s \cdots i_{d+1} i_d i_{d-1} \cdots i_1 i_0,$$

$$j = j_n j_{n-1} \cdots j_{s+1} j_s \cdots j_{d+1} j_d j_{d-1} \cdots j_1 j_0,$$

entonces $i_r \neq j_r$, para $r = d, d+1, \cdots, s$, y los restantes bits correspondientes en i y j son iguales. Si

$$E^{(2)}(i) = a_n a_{n-1} \cdots a_2 a_1$$

y

$$E^{(2)}(j) = b_n b_{n-1} \cdots b_2 b_1,$$

de la definición de $E^{(2)}$ y de las i e j, se tiene que $a_d \neq b_d$ y $a_{s+1} \neq b_{s+1}$, y el resto de los bits son iguales, por lo tanto se cumple el lema. □

A continuación extenderemos la definición de códigos de Gray al caso en base b, cuando $b = 2^m$ y $m > 1$. Con este propósito, introducimos una función biyectiva

$$f : \{0,1\}^{km} \to \{0, 1, \cdots, b-1\}^k.$$

Si

$$x = x_{km} x_{km-1} \cdots x_{(k-1)m+1} x_{(k-1)m} \cdots x_m \cdots x_2 x_1,$$

y

$$\xi = \xi_k \xi_{k-1} \cdots \xi_2 \xi_1,$$

entonces $f(x) = \xi$ si y solo si, para toda $r = 1, \cdots k$,

$$(\xi_r)_b = (x_{rm} x_{rm-1} \cdots x_{(r-1)m+2} x_{(r-1)m+1})_2.$$

Es decir, la función f cambia un número de su representación binaria a la representación correspondiente en base b. Por ejemplo, si $k = 3$ y $m = 2$,

$$f(011011) = 123.$$

A partir de f podemos definir la función de codificación para la base b,

$$E^{(b)} = f^{-1} \circ E^{(2)} \circ f \tag{3.21}$$

donde $\circ$ denota a la composición funcional. Por lo tanto,

$$E^{(b)}(i) = (f^{-1} \circ E^{(2)} \circ f)(i) = f(E^{(2)}(f^{-1}(i))).$$

En consecuencia, la función decodificadora en base b, $D^{(b)}$, está dada por

$$D^{(b)} = f^{-1} \circ D^{(2)} \circ f. \tag{3.22}$$

Lema 3.10 *Sean $i, j \in \langle N \rangle$ tales que $j = (i + 2^d) \bmod N$, para $0 < d < n$. Entonces,*

$$H_b(G_i^{(b)}(k), G_j^{(b)}(k)) = \begin{cases} 2 & \text{para } s > d \text{ si } 0 \equiv d(\bmod\ m) \\ 1 & \text{para } d \leq s \leq d + m - 1 - t \text{ si } t \equiv d(\bmod\ m) \\ 2 & \text{para } s > (d + m - 1 - t) \text{ si } t \equiv d(\bmod\ m) \end{cases}$$

donde s se define como en la demostración del lema (3.9), y $1 \leq t \leq m - 1$.

Demostración
Sean

$$f(E^{(2)}(i)) = A_k A_{k-1} \cdots A_2 A_1$$

y

$$f(E^{(2)}(j)) = B_k B_{k-1} \cdots B_2 B_1.$$

donde

$$\begin{aligned} A_r &= a_{rm} a_{rm-1} \cdots a_{(r-1)m+1}, \\ B_r &= b_{tm} b_{tm-1} \cdots b_{(t-1)m+1} \end{aligned}$$

Debido a que $d > 0$, si $d \equiv 0(\bmod\ m)$ entonces se tiene que $d = rm$ para alguna $1 \leq r \leq k$. Por lo anterior y el lema (3.9) se cumple que $A_r \neq B_r$. Más aún, como $s + 1 > d$ y $a_{s+1} \neq b_{s+1}$, se tiene que existe $p > r$ tal que $A_p \neq B_p$. Con lo anterior probamos el caso cuando $0 \equiv d(\bmod\ m)$. El resto de los incisos se demuestra de manera similar. □

Incrustamiento de arreglos lineales, anillos y mallas

Si tenemos un *arreglo lineal* dado, con p procesadores $P_0, P_1, \cdots, P_{p-1}$, donde $p \leq N$, a incrustar en un cubo (N, b, k), entonces el incrustamiento debe hacerse de tal forma que el procesador P_i esté directamente conectado a los procesadores P_{i-1} y P_{i+1}, excepto en el caso en que $i = 0$ o $i = p-1$. En este último caso, P_0 está conectado directamente con P_1 y P_{p-1} lo está con P_{p-2}. Debido a que las palabras en al código $G^{(b)}(k)$ poseen esta propiedad de adyacencia se tiene un incrustamiento sencillo mapeando cada P_i al nodo etiquetado $G_i^{(b)}(k)$, para $i = 0, \cdots, p - 1$.

Por otra parte, la estructura de adyacencia de un *anillo* de N procesadores $P_0, P_1, \cdots, P_{N-1}$ puede incrustarse en un cubo (N, b, k), con b par, mediante el mapeo del procesador P_i al nodo con la etiqueta $G_i^{(b)}(k)$ para $i = 0, \cdots, N - 1$. Puede observarse que tal mapeo corresponde a un *ciclo Hamiltoniano* sobre el grafo $G^{(k-1)}$.

Si r es par, $r < N$ y $m = (r-2)/2$, entonces un anillo de r nodos puede incrustarse en un cubo (N, b, k) mapeando los r procesadores a los nodos del cubo cuyas etiquetas pertenezcan a los siguientes subconjuntos ordenados de $G^{(b)}(k)$,

$$\left\{G^{(b)}_{0:m}(k), G^{(b)}_{N-m-1:N-1}(k)\right\},$$

donde $G^{(b)}_{i:j}(k)$ se refiere al conjunto $\{G^{(b)}_i(k), G^{(b)}_{i+1}(k), \cdots, G^{(b)}_j(k)\}$.

Podemos resumir lo anterior en el siguiente teorema.

Teorema 3.11 *Un cubo (N, b, k) permite la incrustación de arreglos lineales de longitud $p \leq N$ así como de anillos de longitudes impares, para $b > 2$, y pares, para $b \geq 2$. En todos estos incrustamientos se preserva la adyacencia, esto es $D_f = 1$, mediante la asociación del nodo i-ésimo del arreglo o anillo con la palabra i-ésima del código $G^{(b)}(k)$.*

Incrustamiento de árboles binarios

Debido a que el grafo $G^{(n)}$ de un cubo (N, b, n) es un grafo *conexo*, tiene un árbol generador. A continuación detallaremos un algoritmo para encontrar el árbol generador de profundidad mínima de un cubo $(N, 2, n)$.

Algoritmo de árbol generador para cubos de base 2

1. Sea $j = 1$. Iniciando en el nodo 0^n, incluir en el árbol a la arista que conecta 0^n con $0^{n-1}1$.

2. Para $j = 2$ hasta n, colocar las aristas que conectan al nodo $0^{n-j+1}x_j$ con $0^{n-j}1x_j$, donde x_j es una cadena binaria de $(j-1)$ bits.

El árbol binario generado de este modo puede utilizarse para difundir datos de uno de los procesadores al resto de los nodos en el cubo. Si en lugar de construir el árbol a partir del nodo 0^n, deseamos hacerlo desde algún otro nodo Z, podemos obtener este árbol del anterior reemplazando cada nodo por el nodo etiquetado con el OR exclusivo de la etiqueta del nodo con la etiqueta de Z. A este procedimiento se le denomina *traducción*.

Por otra parte, debido a que varios algoritmos paralelos, como por ejemplo el del abanico asociativo, utilizan árboles binarios, consideraremos enseguida al incrustamiento de árboles binarios completos. El primer resultado que presentamos enseguida es en un sentido negativo.

Teorema 3.12 *No es posible incrustar un árbol binario completo con $2^n - 1$ nodos (por lo tanto, de n niveles) en un cubo $(2^n, 2, n)$, para $n \geq 3$.*

Demostración
Cada hoja en el árbol tiene una arista hacia el padre. Por lo tanto, para cada hoja, deben encontrarse $(n-1)$ nodos vecinos. El total de aristas restantes al quitar las hojas, es $(n-1)2^{n-1}$. Ahora podemos contar el número de aristas que pueden absorber los nodos restantes. Se deben de considerar dos casos.

Caso 1. n es impar. En este caso, el número de nodos que pueden recibir aristas de las hojas está dado por

$$\sum_{i=1}^{(n-1)/2} 2^{2i-1}.$$

Cada uno de estos nodos ya tiene tres aristas ocupadas. El nodo restante puede absorber n aristas. Entonces, el total de aristas que pueden ser absorbidas está dado por

$$(n-3)\left(\sum_{i=1}^{(n-1)/2} 2^{2i-1}\right) + n = \frac{n-3}{3}(2^n - 2) + n.$$

Debido a que para $n \geq 3$,

$$\frac{n-3}{3}(2^n - 2) + n < (n-1)2^{n-1},$$

entonces no hay suficientes vértices, incluyendo al vértice extra, que puedan absorber a las aristas que conectan a las hojas. Por lo tanto, en este caso el árbol binario completo no puede ser incrustado en el cubo $(2^n, 2, n)$.

Caso 2. n es par. En este caso, el número de nodos que pueden recibir aristas de las hojas es

$$\sum_{i=0}^{(n-2)/2} 2^{2i}.$$

Cada uno de esos nodos ya tiene ocupadas tres aristas, excepto el nodo raíz que tiene grado 2. El nodo restante, que está fuera del árbol, puede absorber n aristas. Entonces el número total de aristas que pueden ser absorbidas es

$$(n-3)\sum_{i=0}^{(n-2)/2} 2^{2i} + n + 1 = (n-3)\left(\frac{2^n - 1}{3}\right) + n + 1.$$

Nuevamente, para $n \geq 3$, debido a que

$$(n-3)\left(\frac{2^n - 1}{3}\right) + n + 1 < (n-1)2^{n-1},$$

no pueden absorberse todas las aristan que provienen de las hojas. Por lo tanto, en este caso tampoco es posible el incrustamiento. □

Si no es posible incrustar un árbol binario completo de tamaño $2^n - 1$ en un cubo $(2^n, 2, n)$, ¿cuál es el tamaño máximo de un árbol binario completo que puede incrustarse en este cubo? El siguiente teorema establece que sólo un árbol que ocupe aproximadamente la mitad de los nodos en el cubo puede ser incrustado en él.

Teorema 3.13 *Un árbol binario completo con* $2^{n-1} - 1$ *nodos puede incrustarse en un cubo* $(N, 2, n)$ *con* $D_f = 1$.

Demostración
Demostraremos el teorema, probando que el grafo $G_n = (V_n, E_n)$ puede ser incrustado en un cubo $(N, 2, n)$, donde

$$V_n = V'_n \cup V''_n$$

y

$$E_n = E'_n \cup E''_n,$$

donde (V'_n, E'_n) es un árbol binario completo con $2^{n-1} - 1$ nodos. Además

$$V''_n = \{l_1, l_2, \cdots, l_{n+1}\}$$

y

$$E''_n = \{(l_i, l_{i+1}) |\ i = 1, 2, \cdots, n\} \cup \{(r, l_1), (l_{n+1}, l)\}$$

donde r es la raíz y l la hoja más a la derecha del árbol binario (V'_n, E'_n).

La demostración se realizará por inducción sobre n. Cuando $n = 2$ el cubo $(4, 2, 2)$ es un ciclo de cuatro nodos. Cualquiera de esos nodos puede tomarse como el árbol binario de $2^{n-1} - 1 = 1$ nodo, y este nodo es tanto r como l, a los que se hace referencia anteriormente. Los nodos restantes forman el conjunto V''_2.

Supongamos que el incrustamiento del grafo es posible para toda $n < m$, donde $m \geq 3$, y consideremos el caso cuando $n = m$. Observemos que el cubo $(N, 2, n)$ puede descomponerse en dos subcubos $(N/2, 2, (n-1))$, lo cuales son precisamente la proyección en alguna dimensión, que puede ser sin perdida de generalidad, el bit más significativo. Primero hacemos una incrustación de $G^{(0)}_{n-1}$ en cualquiera de los subcubos. Dado que esto es independiente de las etiquetas de los nodos, escogemos una incrustación de $G^{(1)}_{n-1}$ en el otro subcubo, donde los nodos $r^{(1)}$ y $l^{(1)}$ sean vecinos de $l^{(0)}_1$

y $l_2^{(0)}$ en $G_{n-1}^{(0)}$, respectivamente. El incrustamiento requerido de G_n se obtiene haciendo $l_1^{(0)}$de $G_{n-1}^{(0)}$ el nodo raíz, introduciendo las aristas $(l_1^{(0)}, r^{(1)})$ y $(l_2^{(0)}, l_1^{(1)})$ y borrando los nodos $(l_3^{(0)}, l_4^{(0)}, \cdots, l_n^{(0)})$ y sus aristas correspondientes. Se puede comprobar que G_n contiene al árbol binario requerido. □

Del teorema anterior, es evidente que el árbol binario completo más grande que puede ser incrustado en un cubo binario utiliza menos de la mitad de los nodos del cubo. Sin embargo, un *árbol binario completo de doble raíz* puede incrustarse en el cubo binario utilizando todos los nodos del cubo. Un árbol binario completo de doble raíz de nivel 1 tiene dos nodos conectados por una arista. Un árbol binario completo de doble raíz de nivel n consiste de dos nodos de nivel 1 conectados como se mencionó anteriormente, y cada uno de esos nodos a su vez está conectado con un nodo que es la raíz de un árbol binario completo de $n-1$ niveles.

Existen 2^{l-1} nodos en el nivel l, para $l > 1$, de un árbol binario completo de doble raíz. Como el nivel 1 contiene 2 nodos, es evidente que un árbol binario completo de doble raíz contiene 2^n nodos.

Teorema 3.14 *Un árbol binario completo de doble raíz con* 2^n *nodos puede ser incrutado en un cubo* $(2^n, 2, n)$ *con* $D_f = 1$.

Para la demostración de este teorema, y otros relacionados remitiremos al lector a [54].

La difusión de mensajes en un cubo de base b puede efectuarse utilizando el árbol generador en el cubo (N, b, k). El algoritmo siguiente es una generalización del que se presentó al inicio de esta sección para cubos de base 2.

Algoritmo de árbol generador para cubo de base b

1. Iniciando en el nodo 0^k, se incluyen en el árbol las aristas que conectan con los nodos $x0^{k-1}$, para $x = 1, 2, \cdots, b-1$.

2. Para $j = 2$ hasta k, colocar la arista que conecta $x0^{k-j+1}$ a $xy0^{k-j}$ para toda $y = 1, 2, \cdots, b-1$, y x es un entero de $(j-1)$ dígitos de base b.

3.3. Multiprocesadores de memoria compartida

3.3.1. Conceptos básicos

La característica principal de los multiprocesadores de memoria compartida consiste en que poseen un banco de módulos de memoria los cuales están conectados a un conjunto de procesadores a través de una red de interconexión, y que cada procesador es una máquina autónoma con su propia memoria local.

El principal factor que influye en el desempeño de este tipo de organización de computadora paralela es el de los denominados *conflictos de acceso*. Si k de los N procesadores generan direcciones que hacen referencia a una localidad en un solo módulo de memoria, entonces tomará k ciclos accesar los datos y, en principio, el algoritmo se *degradaría* por un factor de k. Entonces, con el objeto de tener eficiencia global, se necesita que

(a) Los procesadores generen direcciones de modo tal que ningún par de procesadores requieran datos de un mismo módulo, esto es, que el patrón de acceso sea una *permutación*.

(b) Los datos estén distribuidos a través de los módulos de memoria de tal forma que permitan *acceso paralelo*.

(c) La red de interconexión debe tener la capacidad necesaria para *alinear* y entregar los datos a los respectivos procesadores.

La más simple de las redes de interconexión es el *bus de alta velocidad*. Si bien en esta red los procesadores no pueden accesar simultaneamente la memoria, fijando de antemano el número de procesadores y módulos de memoria, se puede determinar el ancho de banda requerido en el bus para atender todas las peticiones de acceso dentro de un retraso específico. Un ejemplo de computadora comercial con este tipo de arquitectura es la Encore Multimax de 20 procesadores. Este esquema no es muy adecuado para un número grande de procesadores debido a problemas de saturación del bus.

En el otro extremo están los *conmutadores de barras cruzadas* $N \times N$, los cuales son redes de interconexión de N entradas y N salidas. Los procesadores están conectados a las entradas y los módulos de memoria a las salidas. En la interconexión de la línea de entrada i-ésima y la línea de salida j-ésima hay un switch de *dos estados (on/off)*. El encendido de este switch conecta al procesador P_i con el módulo de memoria M_j. En total, existen N^2 switches en un conmutador de barras cruzadas $N \times N$.

Sea $\pi : \langle N \rangle \to \langle N \rangle$ una permutación. Se dice que una red *efectua* una permutación π si existe una configuración de los switches en la red de modo que la entrada i esté conectada a la salida $\pi(i)$ para $0 \leq i < N$. Puede

verificarse que el conmutador de barras cruzadas puede efectuar todas las $N!$ permutaciones. Por lo tanto, el *poder combinatorio (PC)* que se define como el cociente del número de permutaciones efectuadas entre el número total de permutaciones, para el conmutador de barras cruzadas es la unidad. El multiprocesador experimental C.mmp de la Universidad de Carnegie-Mellon, fue construido usando un conmutador de barras cruzadas de 16×16.

Al proceso de calcular los estados de los switches individuales para efectuar una permutación, se le denomina *ruteo*. El conmutador de barras cruzadas permite una estrategia de ruteo simple que puede ser implementada en *paralelo*. Sin embargo una de las principales desventajas de los conmutadores de barras cruzadas es su costo, además de las limitantes técnicas tales como la *tolerancia a fallas*, el *mantenimiento*, etc. Por lo tanto, resulta más conveniente tener conmutadores de barras cruzadas de menor tamaño y mayor confiabilidad, tales como 2×2 o 4×4. Debido a esto, el problema principal en la interconexión es el de diseño de redes de conmutadores pequeños, digamos 2×2, que sean funcionalmente equivalentes a los conmutadores $N \times N$. En la discusión subsecuente utilizaremos a los conmutadores 2×2 como bloques básicos para la construcción de redes de interconexión.

Un conmutador de barras cruzadas de 2×2 contiene obviamente 4 switches, s_{00}, s_{01}, s_{10} y s_{11}. Poniendo a s_{00} y s_{11} en *encendido* y a los otros dos en *apagado*, efectuamos la permutación identidad

$$\pi = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Este es denominado el estado *directo* del conmutador. De modo similar, poniendo s_{01} y s_{10} en encendido y a los otros en apagado, efectuamos la permutación

$$\pi = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Este es denominado estado *cruzado* del conmutador. Por tanto, el estado del conmutador pueden ser controlados por un bit. El valor en 0 de este *bit de control* corresponderia al *estado directo* mientras que el valor de 1 al *estado cruzado*. El *estado de red* de una red compuesta de conmutadores 2×2 corresponderia a la especificación de los estados individuales de cada conmutador. Entonces, una red de interconexión formada por L conmutadores 2×2 tiene un total de 2^L estados diferentes.

Una red en la cual existe *al menos una* trayectoria de cada una de las terminales de entrada a cada una de las terminales de salida es denominada red de *acceso completo*. Si existe una única trayectoria entre cada par de

entrada y salida, entonces la red es denominada *red de trayectoria única* o red de Banyano. Una red es denominada *red bloqueante* si existe al menos una permutación no efectuada por la red, es decir, si $CP < 1$, en el otro caso, cuando $CP = 1$, es denominada *red no bloqueante*.

Tanto el número de trayectorias como el poder combinatorio de una red dependen críticamente del número de etapas y del esquema de interconexión entre etapas.

Podemos observar que si $N = 2^n$ y considerando una red de k etapas, formada por conmutadores 2×2 se tiene lo siguiente.

1. En una red de k etapas, existen a lo más $kN/2$ conmutadores 2×2.
2. En una red de k etapas, si $k \leq n$ entonces existe *a lo más una* trayectoria entre cada par de terminales de entrada y salida.
3. Para esta red son necesarias n etapas para tener acceso completo.
4. El número de permutaciones realizadas por una red de k etapas es a lo más $2^{kN/2}$ para $1 \leq k \leq n$.
5. En una red de trayectoria única, el número de permutaciones efectuadas por la red es igual al número de estados diferentes de la red.
6. Si π es una permutación efectuada por una red de acceso completo, entonces más de un estado de la red efectua la misma permutación π.

Sea $N = 2^n$ y supongamos que tenemos una red $N \times N$ construida de L conmutadores 2×2. Entonces existen 2^L estados de la red. Para tener poder combinatorio completo, el número de estados debe ser al menos tan grande como el número total de permutaciones, y por tanto

$$2^L \geq N!$$

esto es,

$$L \geq \log N!.$$

Mediante la aproximación de Stirling

$$N! \approx (2\pi)^{-1/2} N^N e^{-(N+1/2)}$$

obtenemos que, para algunas constantes c_1 y c_2,

$$L \geq N \log N - c_1 N + c_2.$$

Por lo anterior, se cumple el siguiente teorema.

Teorema 3.15 *El número mínimo de conmutadores* 2×2 *necesarios para efectuar las* $N!$ *permutaciones, es* $N \log N$*, esto es* $L = \Omega(N \log N)$*.*

Del resultado anterior podemos deducir que un conmutador de barras cruzadas $N \times N$ está lejos de ser óptimo.

3.3.2. Análisis de redes bloqueantes

A continuación describiremos una clase de redes denominadas *redes de línea base*, BL_N, con N entradas y salidas.

Sea $N = 2^n$, BL_2 es una red de una sola etapa consistente en un conmutador 2×2. Para $N \geq 4$, suponiendo que se tienen copias de $BL_{N/2}$, la red BL_N se construye de la siguiente forma: La *primera etapa* está formada por una *columna* de $N/2$ conmutadores 2×2 numerados del 0 al $N/2 - 1$. Esta etapa es seguida por dos copias de $BL_{N/2}$ denominadas las subredes superior e inferior. Las entradas de cada subred están numeradas de 0 a $N/2 - 1$. La interconexión entre la primera etapa y las dos subredes se hace de tal forma que las dos salidas de un conmutador 2×2 se distribuyan de igual forma entre ambas subredes. En concreto, la salida superior del conmutador i de la primera etapa se conecta a la entrada i de la subred superior y la salida inferior del mismo conmutador i va a la entrada i de la subred inferior. Las terminales de salida de la subred superior se numeran del 0 al $N/2 - 1$ y las de la red inferior del $N/2$ al $N - 1$.

Lema 3.16 *La red* BL_N *es una red bloqueante.*

Demostración
De la definición anterior es fácil observar que la red tiene $\log N$ *etapas* con $N/2$ conmutadores 2×2 en cada una de ellas. La red BL_N tiene entonces un total de $N/2 \log N$ conmutadores 2×2. Por lo tanto, el número total de estados de la red BL_N es

$$2^{(N/2) \log N} = \sqrt{N^N},$$

los cuales son menos de $N!$. □

Lema 3.17 *La red* BL_N *es una red de acceso completo de trayectoria única.*

Demostración
Este lema es fácil de probar por inducción sobre la estructura recursiva de BL_N. □

Lema 3.18 *El número de permutaciones efectuadas por BL_N es $\sqrt{N^N}$ y además*

$$\lim_{N\to\infty} CP = 0.$$

Demostración

De las propiedades de acceso completo y trayectoria única de BL_N se tiene que cada uno de los diferentes estados de la red corresponden a diferentes permutaciones efectuadas por la red. En consecuencia BL_N efectua $\sqrt{N^N}$ permutaciones. Y como

$$CP = \frac{\sqrt{N^N}}{N!},$$

utilizando la aproximación de Stirling, se demuestra la segunda afirmación. □

Sea $x \in \langle N \rangle$ donde $N = 2^n$ y

$$x = (x_n x_{n-1} \cdots x_2 x_1)_2.$$

La *mezcla perfecta* σ es una permutación $\sigma : \langle N \rangle \to \langle N \rangle$ tal que

$$\sigma(x) = (x_{n-1} x_{n-2} \cdots x_2 x_1 x_n)_2,$$

es decir, $\sigma(x)$ es una *rotación izquierda* sobre los bits de la representación en binario de x. En notación decimal se expresa como

$$\sigma(x) = (2x + \lfloor 2x/N \rfloor) \bmod N.$$

Para $1 \leq k \leq n$, se define la k-ésima *submezcla* $\sigma_{(k)}(x)$ como

$$\sigma_{(k)}(x) = (x_n x_{n-1} \cdots x_{k+1} x_{k-1} x_{k-2} \cdots x_2 x_1 x_k)_2,$$

esto es, como una mezcla perfecta de los k bits menos significativos de x. De forma similar, se define $\sigma^{(k)}(x)$ como la k-ésima *supermezcla*:

$$\sigma^{(k)}(x) = (x_{n-1} x_{n-2} \cdots x_{n-k+1} x_n x_{n-k} \cdots x_2 x_1)_2,$$

es decir, como la mezcla perfecta de los k bits más significativos de x. La *inversa*, σ^{-1}, de la mezcla perfecta es denominada la *separación*. Si σ^k denota la composición de σ realizada k veces, esto es,

$$\sigma^k = \underbrace{\sigma \cdot \sigma \cdots \sigma}_{k \text{ veces}}$$

entonces se tiene que

$$\sigma^{-1} = \sigma^{n-1}.$$

La permutación de *inversión de bits* es $\rho : \langle N \rangle \to \langle N \rangle$, donde

$$\rho(x) = (x_1 x_2 \cdots x_{n-1} x_n)_2.$$

La permutación de *mariposa* es $\beta : \langle N \rangle \to \langle N \rangle$ tal que

$$\beta(x) = (x_1 x_{n-1} x_{n-2} \cdots x_3 x_2 x_n)_2.$$

Para $1 \leq k \leq n$, se definen las k-ésimas sub y superinversión de bits, $\rho_{(k)}(x)$ y $\rho^{(k)}(x)$, como las operaciones de iversión de bits sobre los k bits menos y más significativos, respectivamente, de x. De forma similar se definen $\beta_{(k)}(x)$ y $\beta^{(k)}(x)$.

Una permutación de *intercambio* $e_i : \langle N \rangle \to \langle N \rangle$ es tal que

$$e_i(x) = (x_n x_{n-1} \cdots x_{i+1} \bar{x}_i x_{i-1} \cdots x_2 x_1)_2,$$

donde $\bar{y}$ es el complemento de y.

Denotaremos por E al conjunto de $2^{N/2}$ permutaciones efectuadas por una columna de $N/2$ conmutadores de barras cruzadas de 2×2.

Una red *omega* $N \times N$, Ω_N, está formada por $\log N$ etapas de conmutadores 2×2 donde la interconexión entre etapas es una mezcla perfecta. Además las terminales de entrada y las entradas de la primera etapa están también conectadas por una mezcla perfecta. Por lo tanto, Ω_N puede ser descrita brevemente como

$$\Omega_N = (\sigma E)^n.$$

La red de *cubo n binario indirecto*, C_N, está definida como

$$C_N = E\beta_{(2)} E\beta_{(3)} E \cdots E\beta_{(n)} E\sigma^{-1}.$$

Esta red recibe su nombre debido a la relación que guarda con el hipercubo binario. Para establecer esta relación supóngase que las terminales i de entrada y salida están ambas conectadas al procesador P_i, donde $0 \leq i \leq N$, esto es, que la red sirve como interconexión entre procesadores. Estableciendo los conmutadores de la etapa i en el estado cruzado y todos los demás en el estado directo, se puede verificar que C_N simula a las conexiones en la dimensión i del cubo binario de N nodos.

Si tenemos una red de interconexión e intercambiamos las terminales de salida con las de entrada, obtenemos una nueva red a la cual se denomina la *inversa*.

Dos redes son *topológicamente equivalentes* si una red puede ser obtenida a partir de la otra mediante un reetiquetado de las terminales y de los

conmutadores. Se dice que dos redes son *funcionalmente equivalentes* si efectuan el mismo número de permutaciones. Se puede comprobar que la red de cubo n binario indirecto C_N, la red omega Ω_N, la inversa de la red de linea base BL_N^{-1}, todas son topológicamente equivalentes a la red BL_N. Esta equivalencia topológica implica claramente equivalencia funcional.

3.3.3. Análisis de redes no bloqueantes

Sea $N = n^2$ para alguna $n \geq 2$. La *red de Clos simétrica* de tres etapas consiste de n copias de conmutadores de barras cruzadas $n \times m$ y $m \times n$ en la primera y tercera etapas, donde $m = 2n - 1$, mientras que la etapa media está formada por m copias de conmutadores de barras cruzadas $n \times n$. Las interconexiones entre etapas siguen una *regla distributiva*: la salida i del conmutador j de una etapa está conectada a la entrada j del conmutador i de la siguiente etapa.

La primera y tercera etapa de la red de Clos requieren n^2m switches cada una y la etapa intermedia requiere n^2m switches también. Por lo tanto el costo de total de una red Clos de tres etapas $N \times N$ es

$$C(n^2, 3) = 3n^2m = 6N^{3/2} - 3N. \tag{3.23}$$

Puede verse que para $N \geq 36$, la red de Clos de tres etapas tiene menor costo que la red de barras cruzadas de una sola etapa.

Sea $N = n^{r+1}$ y $m = (2n-1)$. Consideremos una red de Clos de tres etapas con la primera y tercera etapas formadas por n' copias de conmutadores de barras cruzadas $n \times m$ y $m \times n$, respectivamente. La etapa intermedia consistiría de m copias de conmutadores de barras cruzadas $n' \times n'$. Podemos reemplazar recursivamente cada uno de los conmutadores de la etapa intermedia nuevamente por una rede de Clos de tres etapas. Se detiene la recursión cuando los conmutadores de la etapa intermedia sean 2×2. Puede observarse que

$$C(n^{r+1}, 2r + 1) = mC(n^r, 2r - 1) + 2mn^{r+1}, \tag{3.24}$$

con la condición inicial

$$C(n^2, 3) = 3n^2(2n - 1). \tag{3.25}$$

Al resolver la recurrencia anterior se tiene

$$C(n^{r+1}, 2r + 1) = \frac{n^2(2n - 1)}{n - 1} \left[(5n - 3)(2n - 1)^{r-1} - 2n^r\right]. \tag{3.26}$$

Para n grande se tiene que

$$\begin{aligned} C(n^{r+1}, 2r+1) &\approx 2n^2[5n \cdot (2n)^{r-1} - 2n^r] \\ &\leq 2[(5/2)2^r - 2]n^{r+2}. \end{aligned}$$

Por lo tanto

$$C(N, 2r+1) = O(N^{1+\varepsilon}) \tag{3.27}$$

donde $\varepsilon = 1/(r+1)$.

Por lo anterior, si $r \geq 1$, el costo de la red de Clos $C(N, 2r+1)$ es menor que N^2, el costo del conmutador de barras cruzadas $N \times N$. Sin embargo, al compararse con la cota inferior $\Omega(N \log N)$ resulta que la red de Clos no es *óptima*.

A continuación especificaremos una red, ideada por Benes[2], que al mismo tiempo que es óptima satisface la condición $CP = 1$.

Sea N un número que no es primo tal que $N = n \times r$. Una red de Benes de tres etapas B_N, está formada de r copias de conmutadores de barras cruzadas $n \times n$ en la primera y tercera etapas con la etapa intermedia formada por n copias de conmutadores de barras cruzadas $r \times r$. La interconexión entre etapas satisface la propiedad distributiva mencionada en la red de Clos.

Sea $N = 2^k$, $n = 2$ y $r = N/2$, para alguna $k \geq 1$. Cuando $N = 2$, la red de Benes B_2 es un solo conmutador de barras cruzadas 2×2. Ahora, suponiendo que disponemos de las suficientes copias de redes de Benes $B_{N/2}$, construimos la red de Benes B_N como sigue: la primera y tercera etapas consisten de $N/2$ copias de conmutadores de barras cruzadas 2×2, y la etapa intermedia contiene dos copias de $B_{N/2}$, donde las interconexiones entre etapas satisfacen la propiedad distributiva estándar.

Sea $C(N)$ el número de conmutadores 2×2 y $d(N)$ el número de etapas en la red de Benes B_N, donde $N = 2^k$. Puede verse que

$$C(N) = 2C(N/2) + N$$

con la condición inicial $C(2) = 1$. Y

$$d(N) = d(N/2) + 2$$

con la condición de frontera $d(2) = 1$. Resolviendo las anteriores ecuaciones obtenemos

$$C(N) = N \log N - \frac{N}{2} \tag{3.28}$$

y

$$d(N) = 2 \log N - 1. \tag{3.29}$$

de donde podemos deducir que la red de Benes es óptima.

[2]Ver [54], pp.121-127.

3.4. Multicomputadora de componentes comunes

En esta sección describiremos un tipo de multicomputadora que se construye a partir de componentes de uso común en las computadoras personales. Este tipo de multicomputadora, denominada más comúnmente *clusters Beowulf* [11], resulta interesante como una opción de bajo costo para el cómputo paralelo.

De acuerdo a la ley de Moore, el desempeño de un microprocesador común[3] se duplica cada 18 meses. Cualquier persona relacionada con la tecnología de la computación está conciente de lo que esto significa. El enorme volúmen de ventas de hardware a nivel de usuario, y la competencia que esto genera, ha llevado al desarrollo de hardware cada vez más eficiente simultaneamente con una reducción de costos. Lo anterior ha hecho posible la construcción de computadoras de alto desempeño a partir enteramente de componentes comúnes y de software libre, obteniendo así una significativa ventaja costo/desempeño sobre máquinas que requieren hardware y software especializado. El desarrollo de sistemas operativos libres (Linux, FreeBSD), de los estándares de paso de mensajes (MPI), y la existencia de dispositivos de interconexión de redes de cómputo de alta velocidad (Fast Ethernet), han creado las condiciones para la aparición de las multicomputadoras de componentes comunes.

La primer multicomputadora de este tipo fue construida en el CESDIS de la NASA en 1995 [11]. Utilizando 16 procesadores Intel Pentium Pro y un switch fast ethernet como medio de interconexión, se obtuvieron desempeños sostenidos de más de un Gigaflop en diferentes aplicaciones científicas. A partir de entonces son cada vez más los centros de investigación y las universidades que han construido sus propias multicomputadoras de componentes comúnes para sus proyectos y desarrollos relacionados al supercómputo y cómputo paralelo.

3.4.1. Arquitectura

Usualmente, la arquitectura de una multicomputadora de componentes comúnes es de interconexión en bus. Este bus es usualmente un conmutador de ethernet o fast ethernet, si bien se han utilizado otro tipo de hardware, como canal de fibra, myrinet, etc.

Generalmente la multicomputadora consta de un *nodo maestro* y varios *nodos esclavos*. El nodo maestro es una computadora completa, mientras que

[3] En adelante, *común* se refiere a que es de uso común en las computadoras personales o PCs

los nodos esclavos usualmente no poseen monitores, ni teclados, y pueden ser nodos sin disco duro.

En la mayoria de este tipo de multicomputadoras el sistema operativo utilizado es Linux, con algunas modificaciones, y el software utilizado para programarlas es alguna implementación libre de PVM [38] o MPI [67].

3.4.2. Interfaz de paso de mensajes MPI

El MPI es un estándar para la interfaz de paso de mensajes requerida en la programación de multicomputadoras [67]. La existencia de MPI ha hecho posible el desarrollo de un grado aceptable de portabilidad del software paralelo. MPI proporciona una interfaz de programación basada en envio y recepción de mensajes que es independiente de cualquier arquitectura paralela particular, además de que puede enlazarse con C++ o Fortran [98]. Existen varias implementaciones disponibles públicamente de la libreria MPI. En el desarrollo del presente trabajo se ha utilizado la implementación LAM/MPI (*Local Area Multicomputer/Message Passing Interface*)[4].

[4]Disponible en http://www.lam-mpi.org/