# Capítulo 3: Clusters Heterogéneos

Dado que se ha hecho una descripción de los principales métodos para multiplicar matrices, ya es posible poner toda la atención sobre las redes de computadoras instaladas que se pueden utilizar para cómputo paralelo. En este sentido, y tal como se puntualiza en el capítulo anterior, es necesario conocer con bastante nivel de detalle la arquitectura de cómputo subyacente para, como mínimo, analizar los métodos que han sido propuestos.

Inicialmente se describirán las características principales de las redes de computadoras instaladas para ser utilizadas como máquinas paralelas. Dado que las redes de computadoras no han sido concebidas inicialmente para realizar cómputo paralelo (al menos en el contexto de las aplicaciones científico-numéricas), se deben identificar con la mayor claridad posible las capacidades en cuanto a: cómputo, interconexión de los procesadores, sincronización y escalabilidad.

Con las características de las redes de computadoras bien definidas, es posible analizar los métodos paralelos de multiplicación de matrices y de acuerdo a este análisis decidir si es posible que su implementación en clusters obtenga rendimiento aceptable. En este análisis se encontrará que los métodos de cálculo paralelo de la multiplicación de matrices que han sido propuestos no son adecuados para los clusters heterogéneos.

A partir del análisis anterior, se propone un nuevo algoritmo de multiplicación de matrices en paralelo, mostrando sus características principales en cuanto a balance de carga de los cálculos que debe realizar cada estación de trabajo, balance de carga de la red de interconexión y requerimientos de memoria que impone el algoritmo. Una vez definido este algoritmo *inicial*, se proponen cambios en la forma de llevar a cabo cómputo y comunicaciones a fin de aprovechar la capacidad (si la hay) de solapar (o hacer en *background*) cómputo local con comunicaciones. También se analizan las formas de aplicar los mismos conceptos de paralelización de la multiplicación de matrices en clusters heterogéneos a otros problemas, aunque es claro que es muy difícil hacerlo en general.

# 3.1 Características de los Clusters

Como se puntualiza en el primer capítulo, las redes de computadoras constituyen la plataforma de cómputo paralelo más ventajosa en cuanto a la relación costo/rendimiento. Y esta relación costo/rendimiento es aún mejor en el caso de utilizar las redes de computadoras que ya están instaladas. En este sentido, y para la descripción/discusión que sigue, se consideran redes de computadoras instaladas a todas las redes locales, independientemente de que las computadoras interconectadas sean PCs, estaciones de trabajo o computadoras con procesamiento simétrico (SMP: Symmetric MultiProcessing) [71]. De hecho, en todo este capítulo

- NOW (Network of Workstations),
- redes locales,
- redes de computadoras,
- redes de estaciones de trabajo,
- redes de PCs,
- clusters,

son utilizados como sinónimos.

Es importante identificar las características propias de las redes locales de computadoras que son utilizadas para cómputo paralelo ya que en general los algoritmos propuestos para resolver la mayoría (sino todas) las aplicaciones en paralelo, incluyendo la operación de multiplicación de matrices, deben ser evaluados en este contexto. Si bien es cierto que la mayoría (sino todos) los algoritmos paralelos propuestos pueden ser implementados sobre redes locales de computadoras con mayor o menor grado de dificultad, también es cierto que el rendimiento puede ser *muy* diferente. Dado que en general la paralelización de las operaciones que se han propuesto es *orientada* a un tipo de arquitectura, la implementación de estos algoritmos paralelos debe ser analizada teniendo en cuenta las características propias de las redes locales de computadoras a utilizar como plataforma de cálculo paralelo.

En las subsecciones que siguen, se identifican las características según se consideren como propias de las redes de computadoras homogéneas o propias de las redes de computadoras heterogéneas. Inicialmente, también se describe como una característica de la red local de computadoras utilizada para cómputo paralelo la propia red física de interconexión de máquinas, que de hecho es la que se utiliza para el pasaje de mensajes entre los distintos procesadores.

Toda la caracterización de las redes locales que se hace tiene como referencia las computadoras paralelas tradicionales, más específicamente las multicomputadoras. Este punto de referencia tiene importancia desde dos puntos de vista:

- Todo el conocimiento y la experiencia adquirida en cuanto a diseño de hardware de procesamiento y de interconexión. En este sentido, se tiene un punto de referencia sólido con respecto al cual es más sencillo describir características y es posible comparar e identificar similitudes y diferencias.
- Paralelización de aplicaciones y rendimiento obtenido. Como se puntualiza anteriormente, los algoritmos que tienen rendimiento razonable en una máquina paralela normalmente están orientados a la arquitectura de la misma máquina paralela. Por lo

tanto, al identificar similitudes y diferencias con respecto a las máquinas paralelas tradicionales se tiende a simplificar el análisis de los algoritmos paralelos propuestos en general y en particular las formas de procesar en paralelo el trabajo necesario para multiplicar matrices.

### 3.1.1 Características de la Red de Interconexión de Procesadores

Gran parte del esfuerzo de las máquinas paralelas tradicionales ha sido dedicado a las redes de interconexión de procesadores. De hecho, se considera que la red de interconexión de procesadores y en particular su rendimiento es elemental en las computadoras paralelas [69] [71] [87].

Para las computadoras paralelas con (o basadas en) memoria física distribuida, la red de interconexión se puede identificar claramente como la que provee comunicación entre los procesadores. Expresado de otra manera, si a una computadora paralela con memoria distribuida se le quita la red de interconexión de procesadores deja de ser una computadora paralela y se transforma en un conjunto de computadoras separadas o módulos de CPU-Memoria, sin la capacidad de cooperación para la resolución de un problema. En el caso de las computadoras paralelas con memoria físicamente compartida, quitar la red de interconexión de los procesadores con la (*única*) memoria elimina completamente la posibilidad de ejecutar aplicaciones sobre el hardware que queda. Dado que las redes de computadoras utilizadas para cómputo paralelo claramente son computadoras de memoria distribuida, se continuará considerando a la red de interconexión para la transmisión de datos entre procesadores (o computadoras directamente).

En relación con la flexibilidad de una red de interconexión de procesadores, se busca no solamente que haya una forma de transferir datos entre dos procesadores sino que también se tenga el máximo de comunicaciones al mismo tiempo. Los ejemplos clásicos en este sentido se enfocan en la capacidad o no de que todos los pares posibles de procesadores se puedan comunicar al mismo tiempo, o que sea posible que un solo paso (o en una cantidad de pasos independiente de la cantidad de procesadores que se tengan interconectados) se pueda transferir información desde un procesador hacia todos los demás.

La flexibilidad que tenga una red de interconexión definirá la facilidad (o dificultad) de las aplicaciones de usuario para resolver la comunicación entre sus procesos. La idea subyacente es que nunca se debería perder de vista que cada uno de los procesadores será el encargado de la ejecución de uno o más procesos que se comunicará con otros procesos asignados a otro/s procesador/es.

En términos de costo se tiene una relación invariante a través de las distintas posibilidades de redes de interconexión: a mayor flexibilidad y/o rendimiento de la red de interconexión el costo también aumenta. El crecimiento del costo varía según la red de interconexión que se utilice, pero en muchos de los casos aumentar la cantidad de procesadores de la computadora paralela implica un crecimiento más que lineal del costo de la red de procesadores. En el caso particular de las redes de computadoras instaladas, el costo es cero (en general, despreciable), dado que ya están interconectadas.

Como se mencionó antes, la red de interconexión de los procesadores de una computadora paralela construida a partir de una red local es la misma red. En el contexto particular de las redes de computadoras instaladas, redes locales o LAN (Local Area Networks), la red de interconexión más utilizada es la definida por el protocolo estándar IEEE 802.3 [73] [109] [108]. Este estándar es conocido inicialmente como red Ethernet de 10 Mb/s por su capacidad de transmisión de $10^6$ bits por segundo. Las características de esta red de interconexión son muy bien definidas y conocidas en términos de hardware y de lo que tiene relación directa con su flexibilidad y rendimiento.

Además, la mayoría de las características de la red Ethernet de 10 Mb/s son similares a la red Ethernet de 100 Mb/s, también llamada *Ethernet Rápida* (*Fast Ethernet*), donde se cambian solamente los parámetros/índices referidos a rendimiento de las comunicaciones. Esta similitud está ejemplificada en, y también aprovechada por, muchas de las empresas de hardware de comunicaciones que se encargan de diseñar y construir placas de interfase de comunicaciones (NIC: Network Interface Card) con ambas capacidades de transmisión de datos y denominadas placas de red de 10/100 Mb/s. Además, y siempre dentro de la norma 802.3 se ha definido también Gigabit Ethernet con capacidad de transmisión de $10^9$ bits por segundo [105] [76] y se está considerando también la definición del estándar para $10^{10}$ bits por segundo o 10-Gbps [110].

La Figura 3.1 muestra esquemáticamente la forma lógica básica en que se conectan las computadoras en una red local utilizando Ethernet. Se puede notar fácilmente que es de tipo bus, donde las características principales de cada transferencia de datos son:

- no se manejan prioridades ni es predecible el tiempo de acceso al medio,
- tiene un único emisor,
- ocupa el único canal de comunicaciones,
- puede tener múltiples receptores,
- el modo de acceso al medio es CSMA/CD (Carrier Sense, Multiple Access / Collision Detect).

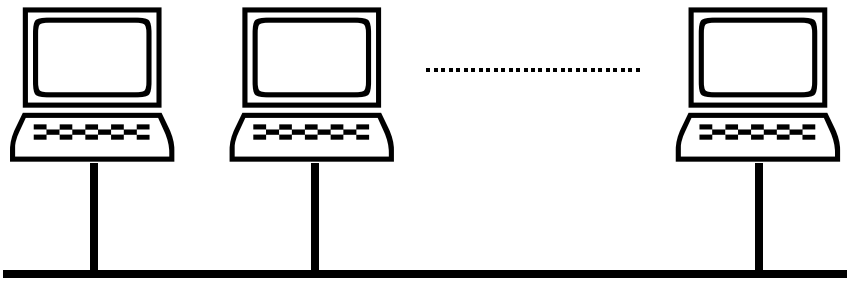

Figura 3.1: Red Ethernet.

Las dos primeras características implican claramente que no puede haber más de una transferencia de datos simultáneamente, porque de hecho hay un único canal de comunicaciones que es compartido por todas las computadoras. La anteúltima característica enunciada hace muy natural la implementación de las comunicaciones del tipo *broadcast* y/o *multicast*, donde desde una computadora se emite un mensaje que es recibido en todas las demás o en un subconjunto de las demás de la red respectivamente. El hardware de comunicaciones inicialmente adoptado en la mayoría de las instalaciones estuvo basado en cables coaxiales, con lo cual se logra que la topología física sea igual a la

topología lógica de la Figura 3.1.

Gradualmente, el cableado (*wiring rules*) utilizado en la mayoría de las instalaciones se ha cambiado hacia la utilización del cable de par trenzado con *hubs* que son básicamente concentradores y repetidores de comunicaciones. La Figura 3.2 muestra que desde el punto de vista del cableado la red tiene topología estrella, pero dado que los hubs distribuyen una misma señal en todos los cables la interconexión lógica sigue siendo la de un bus.

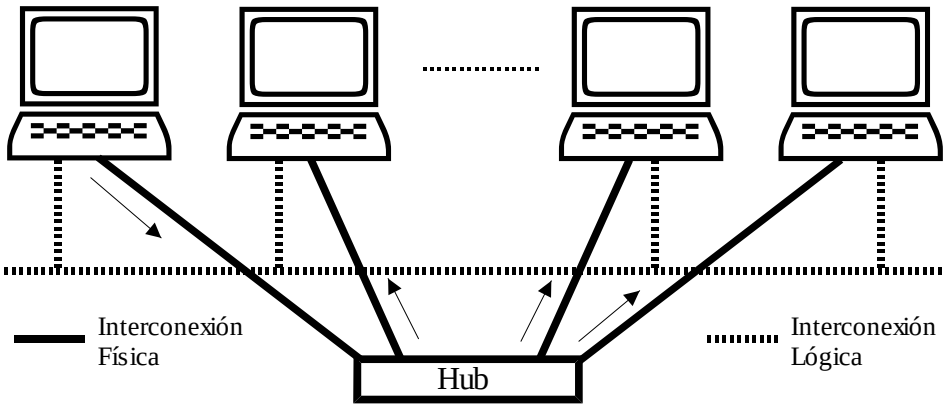


Figura 3.2: Red Ethernet con Hub.

En la Figura 3.2 también se muestra que la computadora ubicada a la izquierda envía un dato y todas las demás están en condiciones de recibirlo simultáneamente, a menos que se produzca una colisión. En este sentido, cada hub no solamente es un repetidor de las señales que llegan por cada cable sino que además es un *repetidor broadcast*, es decir que la señal que llega por un cable es repetida hacia *todos* los demás.

La utilización de *switches* en vez de hubs en las redes Ethernet se considera un gran avance, dado que además de tener todas las características de los hubs, los switches tienen la capacidad de aislar las comunicaciones que son punto a punto [109] [108] [138]. Esta aislación en los *switches* se produce cuando el hardware detecta que hay transmisión de datos punto a punto entre dos de las computadoras que están interconectadas y por lo tanto varias transmisiones de datos punto a punto se pueden realizar de manera simultánea. De todas maneras, por estar definido en el estándar Ethernet, se conservan todas las operaciones definidas por Ethernet, como el broadcast a nivel físico y en ese caso el switch se comporta como un hub.

La Figura 3.3 muestra una red Ethernet donde se llevan a cabo dos comunicaciones punto a punto simultáneas que son posibles por la actividad del switch. Las ventajas que proporcionan los switches sobre los hubs son evidentes, pero también se debe tener en cuenta que el costo de los switches es sustancialmente mayor que el de los hubs a pesar de que se están extendiendo rápidamente en la instalación de las nuevas redes locales. Por otro lado, la utilización de switches se considera esencial en la instalación de redes de computadoras destinadas directamente a cómputo paralelo, del tipo Beowulf [19] [103] [111].

Las redes Ethernet con cableado orientado al uso de hubs son importantes no solamente por

la cantidad de instalaciones actualmente funcionando sino porque son claramente menos costosas que las demás alternativas que se comercializan. Son menos costosas en el hardware necesario (placas, conectores y cables) y en lo referente a instalación: desde mano de obra (técnicos) hasta reconocimiento y puesta en marcha del hardware. Todo esto necesariamente reduce los costos de instalación y de mantenimiento de las redes Ethernet.

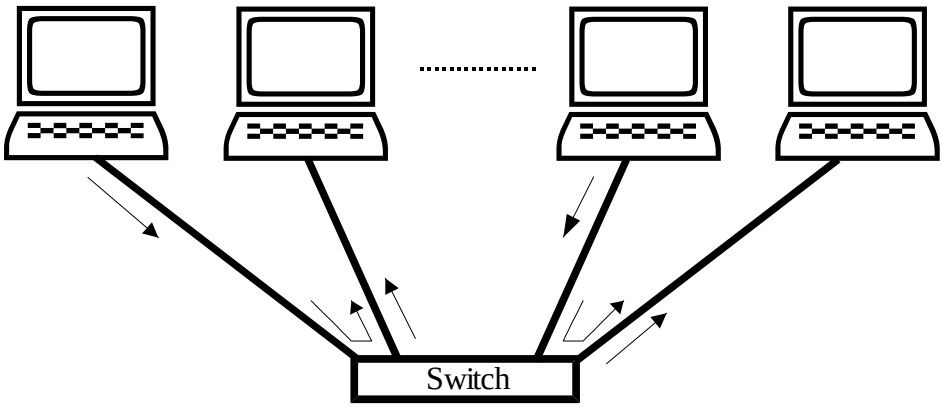


Figura 3.3: Red Ethernet con Switch.

La reducción de costo que representan las redes basadas en hubs con respecto a las demás alternativas de interconexión de computadoras implica una gran inercia en el mantenimiento de las redes Ethernet instaladas así como también en la instalación de nuevas redes con este hardware.

Desde el punto de vista de las aplicaciones que hacen uso de la red de interconexión, de forma tanto o más homogénea que la utilización de Ethernet a nivel físico es la utilización de IP (Internet Protocol) y los que lo utilizan, como TCP (Transmission Control Protocol) o UDP (User Datagram Protocol) [89] [97] [112] [32]. Es así que tanto a nivel físico como en los protocolos de utilización de la red de interconexión, la gran mayoría de las redes locales son homogéneas y con características comunes y bien conocidas.

### 3.1.2 Cluster Homogéneo como Máquina Paralela

En realidad es difícil encontrar una red local con todas las computadoras interconectadas iguales, a menos que la red local se haya instalado *recientemente*. En la mayoría de los casos recientemente significa no más de algunos meses, aunque depende de la organización en la que se utilice y la finalidad de la misma red.

Aún así es útil considerar el caso homogéneo a los efectos de identificar/describir las características de los clusters desde el punto de vista de cómputo paralelo ya que:

- Permite una separación más clara de las características, mejorando por lo tanto su análisis.
- Todas (o la *mayoría de*) las características que se identifiquen para el caso homogéneo serán compartidas por los clusters heterogéneos. Puesto de otra forma, las redes de computadoras tienen características que en sí mismas se deben tener en cuenta a la hora

de la paralelización de aplicaciones independientemente de que las máquinas sean homogéneas o no. Desde este punto de vista, las redes de computadoras heterogéneas agregan otras características que también deben ser tenidas en cuenta para la paralelización de aplicaciones.
- Sistemas como los denominados Beowulf, o simplemente las redes locales instaladas para cómputo paralelo son usualmente homogéneas. En este sentido, las características que se identifican como importantes en un ambiente homogéneo se deben tener en cuenta no solamente en las redes ya instaladas que son homogéneas sino también en todos estos sistemas de cómputo paralelo basados en redes locales de computadoras.

Resumiendo, las principales características técnicas de las redes de computadoras homogéneas que se deben tener en cuenta para el procesamiento paralelo son:
- Acoplamiento débil.
- Bajo rendimiento de la red de interconexión de procesadores.

**Acoplamiento**. Las redes locales de computadoras se construyen a partir de computadoras completas interconectadas con el fin de compartir algún tipo de recurso. Los recursos clásicos que se han compartido son espacio de almacenamiento, datos almacenados e impresora/s. Por lo tanto, el hardware de las distintas computadoras en general no tiene ninguna relación. De hecho, la única condición para que una computadora esté conectada en una red local es la instalación de una placa de interfase de comunicaciones (NIC) y no mucho más que las rutinas de software dedicadas a manejarla. Por lo tanto, desde el punto de vista de la arquitectura de cómputo de cada computadora, implica agregar un dispositivo más de entrada/salida de datos. Ahora volviendo a la visión de la red local como una máquina paralela esto implica que:
- La memoria física de las computadoras en la red local está totalmente distribuida y no hay ningún medio de hardware que facilite compartirla.
- El procesamiento en una red local es totalmente asincrónico, y no hay ningún medio de hardware que facilite la sincronización.

La única forma de hacer que un procesador lea o escriba una posición de memoria en otra máquina es utilizando apropiadamente la red de comunicaciones. De la misma manera, la única forma de sincronización entre los procesadores de cada computadora de la red local es utilizando apropiadamente la red de comunicaciones. Es claro que ambas cosas son posibles, pero también es claro que las redes locales no han sido diseñadas con ninguno de estos dos propósitos y por lo tanto la penalización en rendimiento para lograr la sincronización entre procesadores y/o memoria compartida puede ser muy grande en términos de rendimiento.

Descartar el uso de memoria compartida por la penalización en rendimiento implica necesariamente descartar los algoritmos diseñados para los multiprocesadores, que son computadoras paralelas con memoria compartida. En el caso de los algoritmos numéricos en general y de los algoritmos de multiplicación de matrices en particular, implica en muchos casos descartar algoritmos que han probado ser muy efectivos en cuanto a obtención de rendimiento satisfactorio en multiprocesadores.

El problema de sincronización de los procesadores no parece ser tan problemático como el de la memoria distribuida. En principio, la necesidad de sincronización no es tan fuerte en

la mayoría de los algoritmos, al menos no se proponen algoritmos en función de que se utilicen máquinas con procesadores sincronizados o no. Por otro lado, el hecho de que los procesadores no estén sincronizados por hardware no implica que sea imposible sincronizarlos y por lo tanto si la sincronización es poco frecuente el impacto en el rendimiento no será relevante. Normalmente la frecuencia de sincronización entre los procesadores que no implica pérdida de rendimiento está dada de manera directa por los índices de rendimiento de la red de interconexión, específicamente por el tiempo de inicialización (latencia o *startup*) de comunicaciones.

**Rendimiento de la red de interconexión de procesadores**. El rendimiento de una red de interconexión está directamente relacionado con el tiempo de transferencia de los datos entre los procesadores de una computadora paralela. Esta visión del rendimiento no necesariamente es disjunta de la flexibilidad. De hecho, a mayor cantidad de transferencias de datos simultáneas entre pares de procesadores será también mayor la capacidad o la cantidad de datos que una red de interconexión puede transferir por unidad de tiempo.

Si bien es importante la idea de *ancho de banda* (tasa de transferencia) o ancho de banda asintótico, dado por la cantidad de datos por unidad de tiempo que se pueden transferir, otro de los índices de rendimiento importantes es el tiempo mínimo de comunicación entre dos procesadores. Este tiempo mínimo es básicamente el tiempo de inicialización de las comunicaciones (*startup*), o también denominado por algunos autores [146] [25] como *latencia* de comunicación entre procesadores. La forma clásica de cálculo del tiempo comunicaciones de una red de interconexión tiene en cuenta tanto la latencia como el ancho de banda asintótico de acuerdo con la siguiente ecuación [69] [68]

$$t(n) = \alpha + \beta n \tag{3.1}$$

donde

- $n$ es la unidad de información que se transfiere (bit, byte, representación de un número en punto flotante con precisión simple, etc.).
- $\alpha$ es el tiempo de latencia (*startup*) de la comunicación.
- $\beta$ es el valor inverso del ancho de banda asintótico de la red de comunicaciones, es decir que $1/\beta$ es el ancho de banda asintótico.

El principal inconveniente de las redes de interconexión de computadoras en cuanto a rendimiento es que no fueron diseñadas para cómputo paralelo. En este sentido, se ubican varios órdenes de magnitud por debajo de las demás redes de interconexión de las computadoras paralelas *tradicionales*. Es por eso que se torna muy importante evaluar su rendimiento desde el punto de vista de los procesos de usuario que componen una aplicación paralela.

Es muy difícil cuantificar de manera exacta la relación de las redes locales con respecto a las redes de interconexión de las computadoras paralelas tradicionales en general, utilizando los índices de rendimiento mencionados. Lo que es generalmente aceptado es que la peor relación está dada con respecto al tiempo de inicialización de los mensajes a comunicar dos procesadores. Más aún, en el contexto de las redes locales heterogéneas, el tiempo de inicialización de las comunicaciones suele depender de las computadoras

utilizadas dado que están involucrados los tiempos de llamadas al sistema operativo y su consiguiente sobrecarga en cuanto al mantenimiento y manejo de los protocolos (o *pila de protocolos*) utilizados. En un nivel más cercano al hardware, también están involucrados los tiempos de

- acceso a memoria,
- inicialización-utilización de canales de DMA (Direct Memory Access) en caso de ser utilizados y
- manejo de interrupciones relacionadas y/o interfase con la placa de red de cada computadora a comunicar.

Por otro lado, el rendimiento de la red de interconexión tiene relación directa con el rendimiento y con la granularidad de las aplicaciones paralelas que se pueden ejecutar sobre la computadora. Todo tiempo de comunicación tiende a degradar el tiempo total de ejecución de una aplicación paralela, a menos que se disponga y se aproveche al máximo la capacidad de solapar en el tiempo cómputo con comunicación. Esta capacidad de solapamiento de cómputo con comunicación es otra característica que se ha intentado aprovechar en el diseño de las redes de interconexión de las máquinas paralelas tradicionales.

Desde el punto de vista del rendimiento, si el tiempo de comunicación para la obtención de un resultado en el procesador $P_1$ es igual o mayor que el tiempo de cómputo necesario para calcularlo, entonces lo más razonable es llevar a cabo el procesamiento de forma local (en $P_1$), ahorrando tiempo y/o complejidad de la aplicación.

Resumiendo, el rendimiento de las redes locales es inferior al de las redes de interconexión de procesadores de una máquina paralela tradicional desde varios puntos de vista:

- Latencia y ancho de banda.
- Capacidad de solapamiento.
- *Heterogeneidad* de latencia dependiendo de la heterogeneidad de las máquinas.

### 3.1.3 Cluster Heterogéneo como Máquina Paralela

Las diferencias de velocidad de cómputo relativa de las computadoras en los clusters heterogéneos son lo más importante que se agrega a lo que ya se ha identificado en cuanto a la red de interconexión y a las redes locales con computadoras homogéneas. Por otro lado, la posibilidad de red de interconexión heterogénea es realmente muy poco probable en las redes locales instaladas y por lo tanto se descarta.

En general, las redes locales instaladas pueden ser ampliamente heterogéneas en cuanto al hardware de procesamiento o las computadoras interconectadas en la red local. En las redes locales con un mínimo tiempo de existencia (y *evolución*), se pueden encontrar múltiples modelos de computadoras, quizás algunas computadoras paralelas y/o computadoras con multiprocesamiento simétrico (SMP: Symmetric MultiProcessing) y múltiples modelos de PCs. En el caso de las PCs, también es posible que máquinas con el mismo procesador y operando con la misma frecuencia de reloj tengan distinta capacidad de procesamiento dependiendo de las diferencias que haya, por ejemplo, en velocidad de acceso a memoria, capacidad de memoria cache externa y frecuencia de operación del bus del sistema. Las

diferencias entre las computadoras conectadas en una red local normalmente están relacionadas con:

- Tiempo de existencia de la red local, con su consiguiente impacto en la reposición y/o actualización de las computadoras. En el caso de la reposición, se tiende a mantener un mínimo de computadoras que cada usuario dispone y a medida que transcurre el tiempo el hardware deja de ser fabricado y es reemplazado por otro (normalmente más veloz). En el caso de la actualización, se tiende a incorporar el mejor hardware de procesamiento a un mismo costo, como suele ser el caso de los componentes básicos como procesadores, memoria principal y discos.
- Evolución en cuanto a requerimientos. Las razones y condiciones por las cuales se produjo la instalación de una red local no necesariamente se mantienen invariantes. El cambio de las tareas o la incorporación de nuevas tareas que se llevan a cabo en una red local generalmente implican la incorporación de nuevas computadoras, quizás específicas para las tareas a desarrollar en el ambiente en el que está instalada la red local.

Para obtener el máximo rendimiento posible en una red heterogénea necesariamente se debe llevar a cabo un balance de carga de procesamiento adecuado. Si se parte de asumir que todas las computadoras tendrán que realizar la misma cantidad de procesamiento, se llegará a que la utilización de un mayor número de computadoras implica peor rendimiento. De hecho, la máquina paralela completa funciona en términos del tiempo de procesamiento de la computadora más lenta que se utiliza porque será la última en completar los cálculos asignados y por lo tanto el procesamiento completo no se terminará sino hasta que la *peor* (en términos de rendimiento de procesamiento) termine.

La idea básica para balancear la carga de procesamiento de las computadoras en una red local es simple en cuanto a definición aunque no necesariamente simple en general para su implementación: si una computadora, $ws_1$, es *n* veces más veloz para llevar a cabo el mismo procesamiento que otra, $ws_2$, entonces $ws_1$ debe recibir una tarea *n* veces más compleja en términos de cómputo que $ws_2$. Esta idea básica entonces, significa *nada más* que tener en cuenta las velocidades relativas entre las computadoras para la asignación de tarea/s de cómputo.

Se podría agregar que otra de las características a mencionar en las redes heterogéneas es el de la diferencia de capacidad de almacenamiento en memoria principal. Sin embargo, se debe aclarar que en la mayoría de las redes locales instaladas esta diferencia no es muy amplia ni es proporcional a la diferencia entre velocidades relativas. Haciendo referencia al ejemplo anterior, es muy difícil encontrar que si una computadora $ws_1$ es *n* veces más rápida que otra $ws_2$ el tamaño de memoria de $ws_1$ es *n* veces mayor que $ws_2$. De hecho, es bastante frecuente encontrar tamaños de memoria bastante similares en computadoras con velocidades relativas muy diferentes. En el caso extremo de encontrar que la diferencia entre los tamaños de memoria es muy grande, se puede recurrir a calcular las velocidades relativas entre las computadoras en función de la utilización de memoria swap de forma tal que se tenga en cuenta que las computadoras con menor cantidad de memoria principal disponible llevan a cabo los cálculos haciendo uso de memoria swap y por lo tanto esto se verá directamente reflejado en su velocidad de cómputo.

En un contexto más general de procesamiento heterogéneo las variaciones son también

mayores. Reportes como [22] por un lado identifican los problemas que pueden causar a nivel de estabilidad numérica o convergencia de algoritmos la gran variación en cuanto a representación de los datos numéricos (básicamente de punto flotante). Otros, como [41] por ser más cercanos al aporte teórico llegan a niveles de complejidad mayores aún en cuanto a estimación de eficiencia. Desde el punto de vista de las redes locales de computadoras se puede considerar que estos inconvenientes no son muy probables. En el caso específico de la representación de datos, la gran mayoría de (sino *todas*) las computadoras interconectadas en redes locales utilizan procesadores estándares que a su vez se adhieren explícitamente a los estándares de representación definidos por IEEE, dado por el ANSI/IEEE 754-1984 [72] para números en punto flotante.

## *3.2 Cómputo Paralelo en Clusters Heterogéneos*

Las características enumeradas de las redes locales necesariamente deben ser tenidas en cuenta para la paralelización de las tareas de cómputo que se resuelvan sobre esta plataforma de hardware de procesamiento. Resumiendo de la sección anterior, las características de una red local vista como una máquina paralela son:

- Computadora de memoria distribuida, multicomputadora débilmente acoplada.
- Red de interconexión de procesadores Ethernet.
- Bajo rendimiento de la red de interconexión.
- Heterogeneidad de procesamiento, traducida en distintas velocidades relativas.

El impacto de cada una de estas características en la paralelización de tareas a resolver en redes locales de computadoras instaladas se verá en cada una de las subsecciones que siguen. Es conveniente recordar que la paralelización que se haga sobre las redes locales o, lo que es equivalente, los algoritmos paralelos que se ejecuten sobre las redes locales, tendrá/n impacto directo sobre el rendimiento obtenido. Por lo tanto, se deben favorecer (tal como se ha hecho tradicionalmente en el ámbito de cómputo de alto rendimiento) los algoritmos paralelos que tengan en cuenta la arquitectura de cómputo subyacente. Los algoritmos numéricos en general no escapan a este principio, ni los pertenecientes al álgebra lineal, ni la multiplicación de matrices en particular.

### 3.2.1 Multicomputadora Débilmente Acoplada

El modelo de programación que normalmente se impone en las multicomputadoras débilmente acopladas y aún en las multicomputadoras en general es el de pasaje de mensajes [3] [52] que a su vez se deriva de CSP (Communicating Sequential Processes) [66]. Este modelo de pasaje de mensajes implica que el programa paralelo será un conjunto de procesos secuenciales que se comunican y/o sincronizan, donde normalmente la sincronización en este contexto implica alguna forma directa o indirecta de comunicación entre procesos.

Puede considerarse que la característica de hardware mencionada anteriormente en lo referente a memoria físicamente distribuida tiene relación directa con el modelo de pasaje

de mensajes. Dado que la memoria está físicamente distribuida, es muy difícil compartir memoria entre procesos asignados a (ejecutándose en) diferentes procesadores sin una penalización relativamente alta en cuanto a rendimiento.

Tradicionalmente se ha asumido y en muchos casos experimentado que por un lado es cierto que en última instancia cualquier algoritmo puede ser implementado sobre una máquina paralela de memoria distribuida débilmente acoplada con suficiente esfuerzo (normalmente a nivel de capas intermedias de software). Pero por otro lado, también se ha experimentado que también es cierto que la probabilidad de obtener rendimiento aceptable u óptimo en cuanto a rendimiento es mayor cuando se adopta el modelo de programación de pasaje de mensajes en este ambiente de hardware. De hecho, el caso específico de la gran cantidad de algoritmos propuestos que coexisten para la multiplicación de matrices no hace más que corroborar esta realidad. Puesto de otra manera, dado un algoritmo paralelo basado en memoria compartida por ejemplo, es altamente probable que se pueda diseñar e implementar un algoritmo (que resuelve la misma tarea) basado en pasaje de mensajes que obtenga igual o mejor rendimiento. A priori, es aún más probable que dado un algoritmo basado en pasaje de mensajes no pueda ser superado en rendimiento por otro algoritmo basado en memoria compartida que resuelve la misma tarea.

El área específica de las operaciones matriciales en general, las provenientes del álgebra lineal en particular y específicamente la multiplicación de matrices no escapa a la *regla* mencionada anteriormente. Por lo tanto, en el ambiente específico de procesamiento paralelo que proporcionan las redes locales de computadoras los algoritmos a considerar (como *mejores*) serán los que puedan expresarse en términos de pasaje de mensajes sin ningún tipo de adaptación o traducción intermedia que implique sobrecarga de procesamiento sobre la básica de las mismas operaciones a resolver.

### 3.2.2 Red Ethernet para Interconexión de Procesadores

Como se explica previamente, las redes Ethernet tienen una gran variación en cuanto a rendimiento (10 o 100 Mb/s, por ejemplo) y cableado, o lo que podría considerarse como *topología* de hardware (utilización de hubs y/o switches, por ejemplo). Sin embargo, por la definición misma de estándar definido para la interconexión de máquinas [73] en todos los casos se mantiene la capacidad de broadcast físico de los mensajes. Es decir que, independientemente del rendimiento y del cableado de las redes locales Ethernet, siempre se podrá enviar desde una computadora y recibir el mismo mensaje en todas las demás computadoras de la misma red.

A nivel de los protocolos (o *pila* o *serie* de protocolos [112] [32]) que se utilizan sobre las redes Ethernet, la capacidad de comunicaciones del tipo broadcast se mantiene al menos en los protocolos más cercanos al hardware como IP, IGMP y UDP [112] [34] [96] [95]. La capacidad de llevar a cabo broadcast *físico* es muy relevante en las redes Ethernet, al menos por dos razones:

- Desde el punto de vista de escalabilidad: en relación con la utilización del medio físico de comunicación (cableado), el broadcast físico es *independiente* de la cantidad de receptores. Es decir que el tiempo de comunicaciones tiende a ser el mismo independientemente de la cantidad de computadoras que intervienen en una

transferencia de datos (básicamente *reciben* datos). No se puede asegurar que el tiempo total de un broadcast sea *exactamente* el mismo para cualquier cantidad de receptores porque el tiempo total depende de otros factores como:

- Método de sincronización de receptores, para que un mismo dato sea simultáneamente recibido por todos.
- Método de reconocimiento de recepción (*acknowledge*) de datos o método/s para asegurar que los datos enviados han sido recibidos en todas las computadoras.
- Tasa de pérdida en cuanto a recepción de datos de cada computadora, que depende a su vez de la calidad de la placa de red que cada computadora utiliza y del cableado mismo.
- Disponibilidad de *buffers* de recepción y envío de datos.

Sin embargo, se considera que cada uno de estos factores son relativamente menos importantes que la transmisión de los datos misma (implican la suma de tiempos uno o varios órdenes de magnitud inferiores respecto del tiempo de los datos transferidos).

- Desde el punto de vista del rendimiento: tal como es generalmente aceptado, la utilización de broadcast en los programas paralelos es muy extendida [146]. Si los mensajes broadcast entre procesos no se implementan utilizando el broadcast físico de las redes Ethernet, se transforman en múltiples transferencias punto a punto (que comunican *siempre* los *mismos* datos). Además, es claro que toda comunicación de datos entre computadoras implica la utilización del cableado. Si en el cableado no se utilizan switches, se tiene que cada comunicación implica la utilización de todo el medio de comunicaciones (Figura 3.1 y Figura 3.2) y por lo tanto este medio de comunicaciones debe ser multiplexado entre las distintas transferencias de datos. Expresado de otra manera, en todas las redes locales en las que no se utilizan switches de comunicación, todas las transferencias de datos se secuencializan y de hecho cada comunicación punto a punto entre procesos ocupa todo el medio de comunicaciones. De esta manera se llega a que el tiempo de comunicaciones de los mensajes broadcast se multiplica por la cantidad de receptores. Por otro lado, aún en el caso en que se utilizan switches en el cableado de las redes locales, la posibilidad de asignar más de un proceso a una computadora se debe tener en cuenta para no desaprovechar la capacidad de realizar todas las comunicaciones punto a punto de manera simultánea. Desde otro punto de vista, dado que Ethernet por definición tiene broadcast independientemente del cableado, siempre que la implementación de los mensajes broadcast lo utilice no se tendrá *ninguna* penalización en cuanto al tiempo de transferencia de datos entre computadoras.

Por otro lado, y también desde el punto de vista del rendimiento, la utilización de mensajes punto a punto entre procesos tiende a que se tenga en total mayor tiempo de comunicaciones. Dado que, como se ha mencionado, cuando se tiene cableado sin switches necesariamente se tendrá que el tiempo de transferencia de datos aumenta de manera proporcional a la cantidad de mensajes punto a punto que se llevan a cabo. Esta situación se mantiene en el caso en que se utilizan switches en el cablead pero no se maneja adecuadamente la situación en la que más de un proceso es asignado a una computadora y los mensajes se multiplexan en el canal de comunicaciones.

Por lo tanto, en el ambiente específico de procesamiento paralelo que proporcionan las redes locales de computadoras los algoritmos a considerar (como *mejores*) serán los que puedan expresarse en términos de mensajes del tipo broadcast. Estos mensajes pueden ser

implementados de manera directa en las redes Ethernet y de esa manera:
- Se evitan los costos de algoritmos de broadcast utilizando mensajes punto a punto.
- Se tiene mejor escalabilidad al menos en cuanto a comunicaciones entre procesos.
- Se evita la penalización por el uso secuencial del medio de comunicaciones en las redes locales que no tienen cableado con switches.

Tomando como referencia la construcción de máquinas paralelas de bajo costo a partir de computadoras interconectadas con Ethernet, en el caso extremo se podría reducir aún más el costo, dado que los switches que se utilizan actualmente (y que son considerados *esenciales*) podrían ser reemplazados por hubs sin penalización en cuanto a rendimiento. Sin embargo, este extremo implica que *todos* los programas a ejecutar están expresados en términos de mensajes broadcast. Sin embargo, esta puede no ser una buena decisión en el contexto de las redes locales instaladas, donde ya está/n instalado/s los switches en particular.

### 3.2.3 Bajo Rendimiento de la Red de Interconexión

El rendimiento de las redes de locales de computadoras estándares y específicamente de las redes Ethernet instaladas es muchas veces considerado inaceptable para llevar a cabo cómputo paralelo [12]. De hecho, existen muchas propuestas de interconexión de computadoras de bajo costo (PCs o las estaciones de trabajo menos costosas) estándares que:
- Se interconectan con interfases de red de mayor rendimiento y más costosas [12] [91].
- Se interconectan con más de una interfase de red de bajo costo [65] [48] [78].
- Se interconectan con interfases de red específicamente diseñadas para cómputo paralelo. Unp de los primeros proyectos reportados en este sentido es el TTL_PAPERS: Purdue's Adapter for Parallel Execution and Rapid Synchronization [39] [40] [67] [38] [PAPERS].

Todas estas posibilidades y varias combinaciones posibles se pueden encontrar en [37] en el contexto de procesamiento paralelo utilizando PCs con sistema operativo Linux, pero en la mayoría de los casos pueden ser aplicadas a redes de computadoras en general.

Sin embargo, en todos estos casos se tiene como contrapartida el considerable aumento de costo dado que:
- Se utiliza hardware de interconexión más costoso o más hardware (más interfases de red). Siempre el hardware de interconexión con mayor rendimiento será más costoso y en muchos casos este costo se multiplica cuando el hardware no es de uso masivo como las redes Ethernet. En el caso de utilizar más placas de red, es claro que el costo de hardware de interconexión se multiplica por la cantidad placas que se agregan en cada computadora.
- Tanto el software de base como las herramientas de utilización se vuelven más complejas. En muchos casos el mismo kernel del sistema operativo debe ser modificado y en la mayoría de los casos se deben agregar capas intermedias de software para que las aplicaciones paralelas no sean afectadas o sean afectadas lo menos posible en cuanto a la interfase de utilización de las capacidades de comunicación. Todos estos agregados implican costo de diseño e implementación (que suele reducirse mucho por la utilización de bibliotecas de uso libre), y de mantenimiento, dado que cualquier cambio

en el hardware de red o en el kernel del sistema operativo o en las bibliotecas puede tener impacto directo en las aplicaciones paralelas que lo utilizan.
- Se descartan automáticamente todas las redes locales ya instaladas, que en general son de muy bajo costo y rendimiento relativo: Ethernet de 10 Mb/s.

Teniendo en cuenta estas referencias, es muy probable que la decisión de utilizar las redes de computadoras instaladas implica la revisión de los patrones y frecuencia de las comunicaciones de los programas paralelos a ejecutar. No hacerlo implica asumir que los programas paralelos son lo suficientemente "buenos" como para que la reducción de varios órdenes de magnitud en cuanto al rendimiento de las comunicaciones no implique *ningún* cambio o cambios menores en el rendimiento total.

El impacto del bajo rendimiento de las comunicaciones de las redes locales instaladas sobre el rendimiento total de los programas paralelos a ejecutar siempre depende del patrón de comunicaciones y especialmente de la frecuencia de las comunicaciones entre los procesos del programa paralelo. Es aquí donde se torna muy importante la granularidad de las aplicaciones y del mismo programa paralelo a ejecutar. Teniendo en cuenta el modelo de tiempo de comunicación de datos de cualquier red de interconexión, Ecuación (3.1), los dos aspectos a considerar son: latencia y ancho de banda asintótico.

La latencia de las redes locales de comunicaciones es quizás el aspecto que debe ser tomado con mayor atención desde el punto de vista del rendimiento por dos razones:
- Como se dijo previamente, este tiempo es varios órdenes de magnitud mayor que en las redes de interconexión de las computadoras paralelas *tradicionales*. Por lo tanto se debe tener a priori la tendencia de que la cantidad de datos de las transferencias entre los procesos de un programa paralelo debería ser varios órdenes de magnitud mayores. Esto evidentemente impone una restricción muy fuerte (tan *fuerte* como órdenes de magnitud mayores) sobre la paralelización que se realice o los programas paralelos que se ejecuten sobre las redes de computadoras. Este impacto es o debe ser directamente visible en la granularidad, ya que asumiendo que la cantidad de datos que se transfiere es la misma, aumentar el tamaño de los mensajes tiende a producir menor cantidad de mensajes con menor frecuencia y esto de hecho implica programas con granularidad mayor. Puesto de otra manera, la granularidad mínima de los programas se restringe dado que la latencia es muy grande y por lo tanto reducir la granularidad implica reducir el rendimiento de manera significativa.
- También como se puntualizó antes, la latencia de las comunicaciones suele depender o puede depender de las computadoras que intervienen en la transferencia de datos. Esto significa que no necesariamente se cumple que el tiempo de latencia para transferir datos de una computadora $ws_i$ a otra $ws_j$ es igual al tiempo de latencia de la transferencia de datos de $ws_i$ a otra $ws_k$, $\forall$ *i, j, k*. Puesto de otra forma, es bastante probable que el tiempo de latencia a tener en cuenta en las redes de computadoras esté relacionado con el *mayor* tiempo de latencia entre todos los pares de computadoras interconectadas. Esta restricción no es independiente de la anterior sino que es complementaria porque ahora la latencia a tener en cuenta es la mayor de toda la red, que a su vez será igual o mayor de la definida por el estándar Ethernet.

En lo referente específicamente al bajo rendimiento del ancho de banda asintótico de la red, también afecta el rendimiento de los programas paralelos de manera significativa.

Nuevamente el impacto más fuerte en rendimiento lo tendrán los programas paralelos con granularidad más fina.

Suponiendo, para ejemplificar, que se utiliza una red Ethernet de 10 Mb/s y asumiendo que:
- el tiempo de latencia es cero,
- se transfieren los datos a la máxima capacidad de la red sin que haya ninguna sobrecarga (overhead) de empaquetado, protocolos, capas de software, etc.,

se pueden transferir datos a una tasa de 1.25 MB/s. Siguiendo con el ejemplo, y considerando el contexto mismo de la operación de multiplicación de matrices, son necesarias 1990000 operaciones de punto flotante (utilizando la cantidad de operaciones dada en la tercera ecuación del capítulo anterior), para resolver una multiplicación de matrices de 100×100 elementos. Dado que es hoy bastante común encontrar computadoras en una red local con capacidad de procesamiento de 300 Mflop/s o más, el tiempo aproximado para resolver la multiplicación de matrices de 100×100 elementos es poco menos de 0.0067 segundos (6.7 ms). ¡Y el tiempo para transferir una matriz de 100×100 elementos representados en punto flotante de precisión simple sobre la red es de 0.032 segundos o 32 ms! Por lo tanto, calcular el resultado localmente es 32/6.7 ≅ 4.7 veces mejor que recibir la respuesta desde otra computadora conectada por una red Ethernet de 10 Mb/s.

El ejemplo anterior tiene varias simplificaciones y varias alternativas de *solución* (como usar una red de 100 Mb/s), pero también da una idea de los órdenes de magnitud de tiempos en los cuales se opera y los peligros de asumir que los programas paralelos diseñados para máquinas paralelas no tendrán penalización en rendimiento. Muchas veces, este tipo de análisis ha llevado a asumir (prematuramente) que no es útil resolver en paralelo los problemas numéricos en redes de locales computadoras. Una vez más, para evitar el impacto del ancho de banda de la red de comunicaciones sobre el rendimiento se tiende a aumentar la granularidad. Pero muchas veces no es posible (o no alcanza con) disminuir la frecuencia y aumentar el tamaño de los mensajes como en el ejemplo comentado para resolver el problema del tiempo de latencia de los mensajes. De hecho, en el ejemplo se considera que el tiempo de latencia es cero.

El aumento de granularidad que tiene relación con mejorar el rendimiento de las aplicaciones paralelas utiliza una de las características que es bastante frecuente en los problemas numéricos. Normalmente los problemas numéricos se resuelven con una o más operaciones sobre datos estructurados en arreglos (multidimensionales, en general). También es común que el procesamiento involucrado es uno o más órdenes de magnitud mayor que la cantidad de datos a procesar. En el caso de la multiplicación de matrices ya se ha explicado que la cantidad de datos sobre la que se opera es $O(n^2)$ y la cantidad de operaciones necesarias es $O(n^3)$. Por lo tanto, al aumentar la cantidad de datos es relativamente sencillo aumentar la granularidad, dado que se deben realizar mayor cantidad de operaciones. Esto es equivalente a considerar en el ejemplo anterior matrices de 1000×1000 elementos con todo lo demás invariante:
- El tiempo de cálculo ahora es de aproximadamente 6.7 s.
- El tiempo de comunicaciones para recibir la matriz resultado es de 3.32 s.

De todas maneras, se debe recordar que aumentar el tamaño de los mensajes

necesariamente implica procesar mayor cantidad de datos o, lo que es lo mismo, aumentar el tamaño mínimo de los problemas u operaciones a resolver.

Resumiendo, los programas paralelos que tengan mejor rendimiento en las redes locales de computadoras serán los de mayor granularidad. En este sentido, las redes de computadoras no difieren de la mayoría (o *todas*) las demás plataformas de cómputo paralelo. Pero las redes de interconexión Ethernet implican un esfuerzo mucho mayor en el aumento de granularidad mínima de los programas paralelos y este esfuerzo es proporcional a la diferencia de latencia y ancho de banda que tienen las redes Ethernet con respecto a las redes de interconexión de procesadores en las computadoras paralelas tradicionales.

### 3.2.4 Heterogeneidad de procesamiento

Tal como se ha explicado previamente, la heterogeneidad en capacidad de procesamiento en una red local se traduce directamente en las distintas velocidades relativas de las computadoras interconectadas en redes locales. Avanzando en el nivel de detalle del problema de balance de carga o balance de carga proporcional a las velocidades relativas se deben resolver dos aspectos:

- Identificar con la mayor precisión posible las velocidades relativas de las computadoras a utilizar.
- Distribución de la carga de procesamiento.

**Velocidad Relativa**. Identificar con precisión las velocidades relativas de las computadoras no es sencillo en general. De hecho, es uno de los grandes problemas que intentan resolver los programas de *benchmark* y sobre los cuales sigue habiendo discusión. Afortunadamente, a medida que se avanza en la especificación del área de los problemas numéricos a resolver la situación suele encontrarse más *estable* o más posible de predecir en cuanto a velocidad de cálculo relativa. De hecho, en las aplicaciones numéricas y las del álgebra lineal en particular el procesamiento es sumamente regular y por lo tanto el cálculo de velocidades relativas entre diferentes computadoras también es más sencillo. En este ámbito, dos de los caminos a tomar son:

- Utilizar un tamaño reducido del problema a resolver para ejecutarlo en todas las computadoras y calcular las diferencias de velocidades en base al tiempo de procesamiento en cada computadora. Suponiendo que, por ejemplo, se debe calcular una FFT (Fast Fourier Transform), por ejemplo de $10^7$, $10^8$ o más datos, se puede hacer el cálculo de FFT con $10^5$ o $10^6$ datos en todas las computadoras y tomar las diferencias relativas directamente proporcionales al tiempo de cómputo de esta FFT “reducida”. Se debe ser bastante cuidadoso en cuanto al tamaño del problema dado que se pueden producir resultados muy erróneos [11] si los datos del problema “reducido” se pueden almacenar totalmente en memoria cache en algunas computadoras y en otras no. Por otro lado, si se toman conjuntos de datos demasiado grandes el tiempo de ejecución de este “mini-benchmark” sería muy alto.
- Utilizar la velocidad de cada computadora para la multiplicación de matrices como referencia en cuanto a velocidad de cálculo. Sería el caso anterior en particular para la multiplicación de matrices. Aunque como se aclaró en el capítulo anterior la multiplicación de matrices es particularmente apropiada para aprovechar todas las optimizaciones de código, es de esperar que las diferencias en el tipo de procesamiento

que cada operación o problema numérico impone tenga consecuencias similares de rendimiento en los procesadores de las computadoras que se utilizan. En el caso del problema anterior, implicaría tomar como referencia en cuanto a velocidad de procesamiento el tiempo de ejecución de una multiplicación de matrices de, por ejemplo, 1000×1000 elementos para hacer la distribución del procesamiento necesario para calcular la FFT de $10^7$, $10^8$ ó más datos. De esta manera se asume que si una computadora $ws_i$ es dos veces más rápida que otra $ws_j$ para hacer una multiplicación de matrices también esta diferencia de capacidad de procesamiento se mantiene para realizar el cálculo necesario para una FFT. De esta manera se ahorra el costo de ejecutar un problema "reducido" para cada aplicación aunque se agrega algo de imprecisión derivada de la diferencia de procesamiento que puede haber entre una multiplicación de matrices y cualquier otro problema numérico que se resuelve en una red de computadoras.

En ambos casos, es claro que existe un costo asociado en lo que se refiere al cálculo mismo de las velocidades relativas que no existe en las máquinas paralelas homogéneas en cuanto a la capacidad de cálculo de los elementos de procesamiento. Para el caso específico de las multiplicaciones de matrices no hay diferencias entre los dos caminos anteriores. Ampliando el espectro de aplicaciones a L3 BLAS, la segunda opción es sumamente apropiada dado que, de hecho, la operación básica en todas las definidas en L3 BLAS es la misma multiplicación de matrices. En el caso de ampliar aún más el espectro a las operaciones provenientes del álgebra lineal (a las operaciones definidas en LAPACK, por ejemplo), el tipo de cálculo de la multiplicación de matrices sigue siendo muy similar y, además, las operaciones que son importantes desde el punto de vista del rendimiento (y tiempo de cómputo necesario), siguen siendo las de L3 BLAS.

En el ámbito de las computadoras con elementos de procesamiento heterogéneos, la capacidad de cálculo de cada computadora o procesador suele utilizarse para el cálculo de la heterogeneidad de la máquina paralela [147]. De hecho, en general se recurre a alguna forma de "potencia de cálculo relativa de cada computadora $ws_i$" o $pw(ws_i)$. En este contexto, suele ser muy útil recurrir a la capacidad de cómputo de $ws_i$ expresada en millones de operaciones de punto flotante por segundo o Mflop/s($ws_i$) y utilizarla para el cálculo de $pw(ws_i)$

$$pw(ws_i) = \frac{Mflop/s(ws_i)}{\underset{j=0..P-1}{máx}\ (Mflop/s(ws_j))} \tag{3.2}$$

donde se tienen en cuenta $P$ computadoras identificadas como $ws_0$, ..., $ws_{P-1}$ y $pw(ws_i)$ se calcula en función de la más veloz.

**Distribución de Carga**. Una vez establecido el método con el cual calcular las velocidades relativas y las velocidades relativas específicas entre las computadoras de la red local a utilizar, se debe distribuir la carga de procesamiento de cada computadora.

Sobre la idea de cálculo de velocidad relativa de cada computadora dada en le Ecuación (3.2) se avanza para llegar a la "potencia de cálculo relativa normalizada de $ws_i$" o directamente $pw_i$

$$pw_i = \frac{Mflop/s(ws_i)}{\sum_{j=0}^{P-1}(Mflop/s(ws_j))} \tag{3.3}$$

de forma tal que se tiene

$$\sum_{j=0}^{P-1}(pw_j) = 1 \tag{3.4}$$

Y con esta métrica o índice se debe cumplir que cada computadora $ws_i$ debe llevar a cabo $pw_i$ (o, en porcentaje, $pw_i \times 100$ %), del total del trabajo a realizar.

En el caso específico de la multiplicación de matrices, esto es equivalente a establecer que la computadora $ws_i$ realice todos los cálculos necesarios para $pw_i \times n^2$ de los elementos de la matriz resultado C, lo cual es equivalente a establecer que $ws_i$ ejecute $pw_i \times (2n^3 - n^2)$ operaciones. En el caso más general de las aplicaciones de álgebra lineal y de los problemas numéricos, la identificación de la tarea a resolver en cada computadora $ws_i$ quizás no sea tan claramente definible. Sin embargo, como mínimo en todos los problemas para los cuales se han diseñado e implementado métodos paralelos evidentemente ya se ha hecho algún tipo de división del trabajo total en partes (aunque normalmente estas partes han sido *iguales*).

## 3.3 Principios de Paralelización de Aplicaciones en Clusters

Habiendo hecho la descripción de los clusters en general, y de los clusters heterogéneos en particular desde el punto de vista de procesamiento paralelo, los dos principios de paralelización que se deberían seguir para la obtención de rendimiento paralelo optimizado son:

- Balance de carga dado por la distribución de datos, que a su vez se hace de acuerdo con la capacidad de cálculo relativa de cada computadora.
- Comunicaciones de tipo broadcast únicamente, de tal manera que se aprovecha al máximo la capacidad de las redes Ethernet.
- Distribución de los datos de manera unidimensional, siguiendo casi de manera unívoca el propio hardware de interconexión física definido por el estándar Ethernet. Esta distribución de los datos, además, facilita la utilización de mensajes broadcast dado que, por ejemplo, no hay definidas *filas* y *columnas* de procesadores como en las redes estáticas bidimensionales de interconexión de procesadores.

Se podría afirmar que estos principios son específicamente asociados a los clusters heterogéneos. En el caso específico de la multiplicación de matrices, ninguno de los algoritmos que se describieron en el capítulo anterior *sigue* o *respeta* estos tres principios de paralelización. De hecho, el algoritmo de Fox específicamente se ha modificado para

que los mensajes broadcast se puedan llevar a cabo de manera optimizada en redes bidimensionales de interconexión de procesadores.

Sin embargo, también se pueden seguir otros principios de paralelización derivados del área de aplicaciones de álgebra lineal y procesamiento numérico en general, tales como

- La adopción del modelo de procesamiento SPMD (Single Program - Muliple Data), que simplifica notablemente la programación, el depurado y la optimización de aplicaciones de cómputo paralelo.
- El aprovechamiento de la capacidad (si existe) de solapamiento de las comunicaciones con respecto al cómputo local de cada máquina. Aunque en el caso específicos de las redes locales no está asegurada esta capacidad, es posible intentar, al menos, reducir el tiempo de latencia de las comunicaciones llevando a cabo las comunicaciones en *background* con respecto al cómputo local en cada máquina.

# *3.4 Multiplicación de Matrices en Paralelo*

Aunque ya se han dado algunas ideas en cuanto a la multiplicación de matrices sobre redes de computadoras heterogéneas, es necesario avanzar en cuanto al nivel de detalle y precisión del método por el cual se calcula en paralelo C = A×B. Este método ya ha sido explicado de forma resumida en [136] [131] [137].

De manera similar a la presentación tradicional de los algoritmos numéricos orientados a computadoras paralelas de memoria distribuida, se explica por un lado la distribución de los datos y por otro la secuencia de pasos a ejecutar en cada computadora (procesador) para llegar al resultado esperado. Como en todos los algoritmos orientados a computadoras paralelas de memoria distribuida, se descarta casi desde el inicio la posibilidad de replicación de los datos en todos los procesadores (en este caso, computadoras de la red local) dado que esto impone un requerimiento muy elevado de memoria en cada una de ellas.

## 3.4.1 Distribución de Datos

Asumiendo que cada una de las $P$ computadoras a utilizar $ws_i$ ($0 \leq i \leq P$-1), tiene calculada su "potencia de cálculo relativa normalizada", $pw_i$, que cumple con la Ecuación (3.3) y con la Ecuación (3.4), una de las formas de balancear la carga de procesamiento es establecer que cada estación de trabajo $ws_i$ calcule $pw_i$ del total de datos de la matriz C. En el contexto de la paralelización de cómputo sobre matrices, definir que una determinada parte de una matriz X, o submatriz $X^{(i)}$, sea calculada en una computadora $ws_i$ es equivalente a asignar $X^{(i)}$ a $ws_i$. Es decir que los datos de la submatriz que se calcula en una computadora residen localmente en esa computadora.

Las formas más simples en que se determina la cantidad de datos a calcular de la matriz C en una computadora se muestran de manera esquemática en la Figura 3.4, donde:

- $C^{(i)}$ es la submatriz de C asignada a la computadora $ws_i$, o lo que es igual la parte de C

que reside y se calcula en $ws_i$.

- La Figura 3.4-a) muestra la asignación por filas ($pw_i \times n$ filas), es decir que $ws_i$ calcula una parte de C, $C^{(i)}$, de $fC_i = pw_i \times n$ filas por $n$ columnas. Esta forma de asignación de los datos de una matriz es denominada particionamiento por bloques de filas (*block-striped rowwise partitioning*) en [79].
- La Figura 3.4-b) muestra la asignación por columnas ($pw_i \times n$ columnas), es decir que $ws_i$ calcula una parte de C, $C^{(i)}$, de $n$ filas por $cC_i = pw_i \times n$ columnas. Esta forma de asignación de los datos de una matriz es denominada particionamiento por bloques de columnas (*block-striped columnwise partitioning*) en [79].
- La Figura 3.4-c) muestra la asignación por "bloques en general", es decir que $ws_i$ calcula una parte de C, $C^{(i)}$, de $fC_i = q$ filas por $cC_i = r$ columnas tal que $qr = pw_i n^2$. Esta forma de asignación de los datos de una matriz puede considerarse directamente relacionada con la que se denomina particionamiento en tablero por bloques (*block-checkerboard partitioning*) en [79].

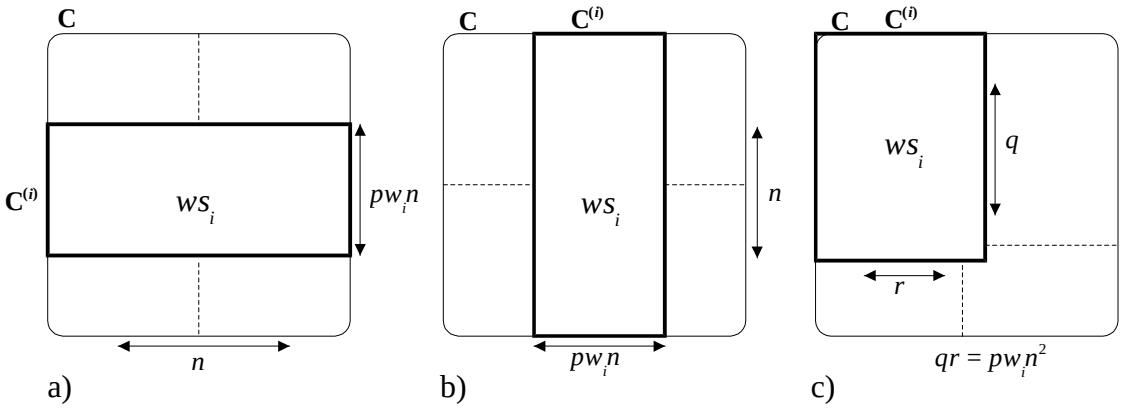


Figura 3.4: Formas Simples de Asignación de Datos.

De las formas de asignación de datos que se muestran en la Figura 3.4, se puede notar que:

- La asignación por filas es totalmente equivalente a la asignación por columnas.
- La asignación por "bloques en general" es bastante más compleja que las anteriores y no parece aportar ningún beneficio sobre las otras. Por otro lado, tiende a ser una distribución bidimensional y por lo tanto no sigue uno de los principio de paralelización enumerados en la sección anterior, por lo tanto se puede, en principio, descartar.

A pesar de que existen formas más complejas y consideradas más apropiadas para la distribución de los datos para cómputo paralelo, se elige distribuir la matriz resultado C por filas para llevar a cabo la multiplicación de matrices en paralelo sobre clusters. La relación y las implicaciones de llevar a cabo este particionamiento comparándolo con los más complejos se explicará después de completada la presentación del algoritmo, para mejorar la claridad en la presentación del algoritmo propuesto. En resumen, la computadora $ws_i$ "tiene a su cargo" (contiene los datos de y calcula) la submatriz $C^{(i)}$ que es de $fC_i = pw_i \times n$ filas y $n$ columnas, es decir $C^{(i)}_{fC_i \times n}$.

Una vez definida la distribución de la matriz resultado C en las computadoras de la red, la forma de distribuir los datos de las matrices A y B se define en función de los cálculos

locales a realizar en cada computadora. Dado que se ha definido que la computadora $ws_i$ tiene que computar $C^{(i)}_{fC_i \times n}$, debe realizar la multiplicación

$$C^{(i)}_{fC_i \times n} = A^{(i)}_{fA_i \times n} \times B \quad (3.5)$$

En la Figura 3.5 se muestran en cada matriz (sombreados) los datos involucrados en el cálculo de $C^{(i)}$ de acuerdo con la Ecuación (3.5).

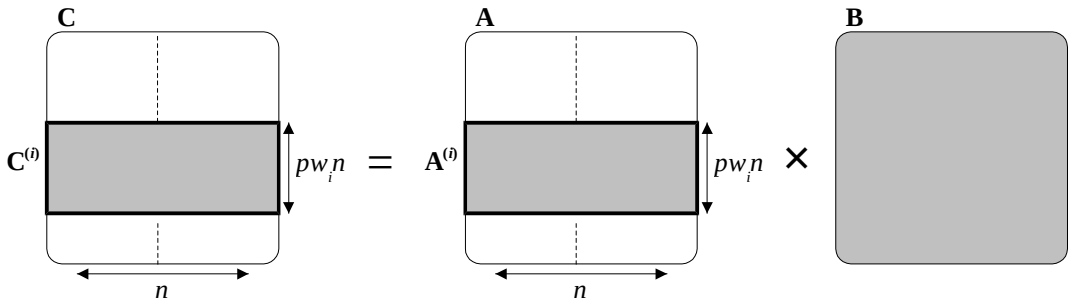


Figura 3.5: Cálculo de una Submatriz.

Por lo tanto, queda bastante claro que la computadora $ws_i$ debería tener localmente los datos de la submatriz de A, $A^{(i)}$, de la Figura 3.5, y esto significa que la distribución de la matriz A entre las *P* computadoras $ws_0$, ..., $ws_{P-1}$ es igual a la de la matriz C. En resumen, la computadora $ws_i$ "tiene a su cargo" (o contiene los datos de) la submatriz de A, $A^{(i)}$, que es de $fA_i = fC_i = pw_i \times n$ filas y *n* columnas, es decir $A^{(i)}_{fA_i \times n}$.

De acuerdo con la Ecuación (3.5) y la Figura 3.5, lo óptimo en cuanto a disponibilidad de datos en $ws_i$ para el cálculo de $C^{(i)}$ sería tener disponible (en memoria local) toda la matriz B. Con esto, se podría llevar a cabo el cálculo simultáneo de todos los $C^{(i)}$ porque todas las computadoras tendrían todo lo necesario para hacerlo. Como se aclaró anteriormente, todos los datos de las matrices se deberían distribuir entre las distintas computadoras para no establecer requerimientos de memoria demasiado grandes y por lo tanto la matriz B también debería ser distribuida entre las computadoras. De acuerdo con la Ecuación (3.5) y la Figura 3.5, todas las computadoras requieren toda la matriz B, por lo tanto esta matriz se distribuye en partes iguales entre todas las computadoras $ws_0$, ..., $ws_{P-1}$. Durante la ejecución del programa paralelo, las computadoras se deben comunicar para intercambiar-recibir las partes de la matriz B que necesitan para el cálculo de la porción de la matriz resultado. Una vez más, esta distribución puede hacerse de forma equivalente por filas o por columnas. Teniendo en cuenta que todas las computadoras harán el cálculo de $C^{(i)}$ en función de esta distribución, debería analizarse (al menos de manera aproximada), si hay alguna de ellas que tenga alguna ventaja sobre la otra.

**Distribución de B por filas**. Si la matriz B se distribuye en partes iguales por filas entre las computadoras, cada submatriz $B^{(i)}$ sería de *n*/*P* filas por *n* columnas, y se tendrían todos los datos de las matrices distribuidos tal como lo muestra la Figura 3.6 para la computadora $ws_i$. De esta manera, el cálculo parcial de la submatriz $C^{(i)}$ (y que se denotará $C^{(i_i)}$), que cada computadora puede llevar a cabo con lo datos que dispone localmente está dado por la multiplicación de matrices de la forma que lo muestra la Figura 3.7.

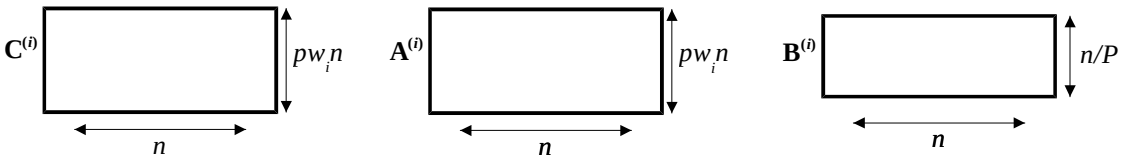


Figura 3.6: Submatrices en $ws_i$ con B Distribuida por Filas.

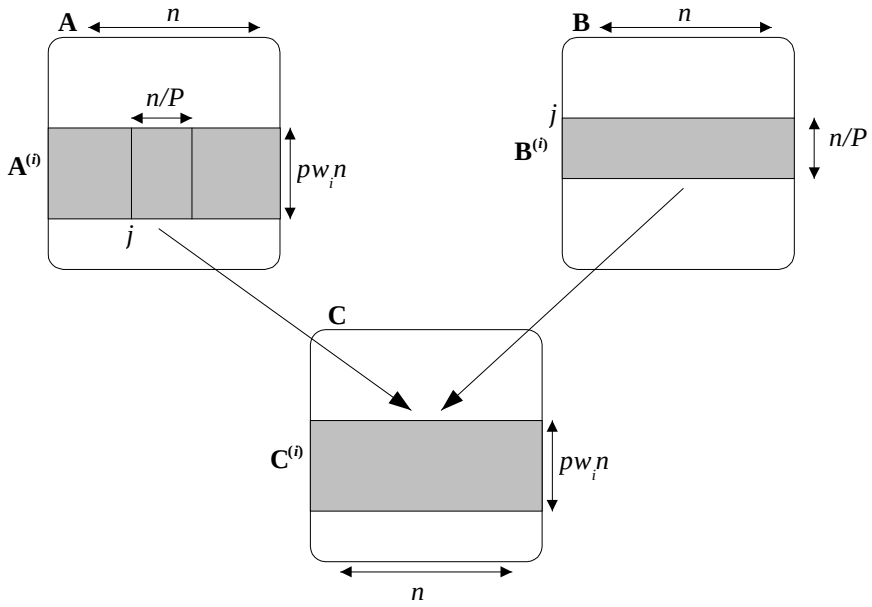


Figura 3.7: Cálculo Parcial de $C^{(i)}$ con B distribuida por Filas.

De manera análoga, con los datos de $B^{(k)}$ de cada computadora $ws_k$ se podrá agregar el cálculo parcial de una parte de $A^{(i)}$ (de $pw_in$ filas y $n/P$ columnas) con $B^{(k)}$ (de $n/P$ filas y $n$ columnas), a $C^{(i_j)}$, tal como el cálculo parcial que se muestra en la Figura 3.7. De esta manera, con sucesivos cálculos parciales de $C^{(i)}$ se llega al resultado final. Esto significa que cada cálculo parcial de $C^{(i)}$ involucra una parte de $A^{(i)}$, todo un bloque $B^{(k)}$, y todo $C^{(i)}$, o sea la multiplicación de dos matrices: una de $pw_in \times n/P$ y otra de $n/P \times n$.

**Distribución de B por columnas**. Si la matriz B se distribuye en partes iguales por columnas entre las computadoras, cada submatriz $B^{(i)}$ sería de $n$ filas por $n/P$ columnas, y se tendrían todos los datos de las matrices distribuidos tal como lo muestra la Figura 3.8 para la computadora $ws_i$.

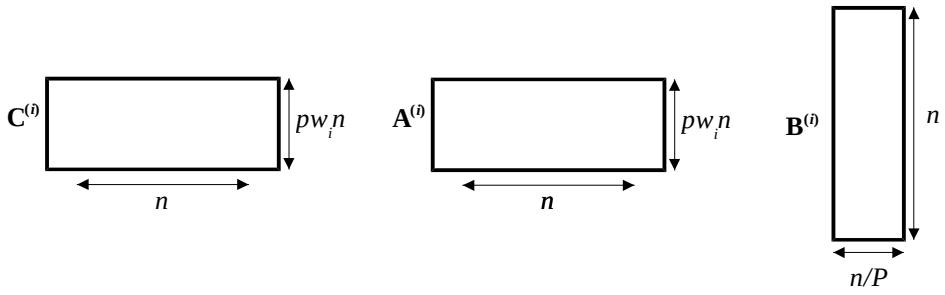


Figura 3.8: Submatrices en $ws_i$ con B Distribuida por Columnas.

De esta manera, el cálculo parcial de la submatriz $C^{(i)}$ (y que se denotará también $C^{(i_i)}$), que cada computadora puede llevar a cabo con lo datos que dispone localmente está dado por la multiplicación de matrices de la forma que lo muestra la Figura 3.9. En ese caso, el cálculo parcial de $C^{(i)}$ que se puede llevar a cabo con los datos $B^{(k)}$ de cada computadora $ws_k$ es el de un bloque de columnas de $C^{(i)}$, y que se denotará $C^{(i_k)}$. En este caso, la multiplicación a realizar es la que corresponde a toda la matriz A, que es de $pw_i n \times n$, con todo un bloque de B, que es de $n \times n/P$, y se calcula una parte de $C^{(i)}$ de $pw_i n \times n/P$.

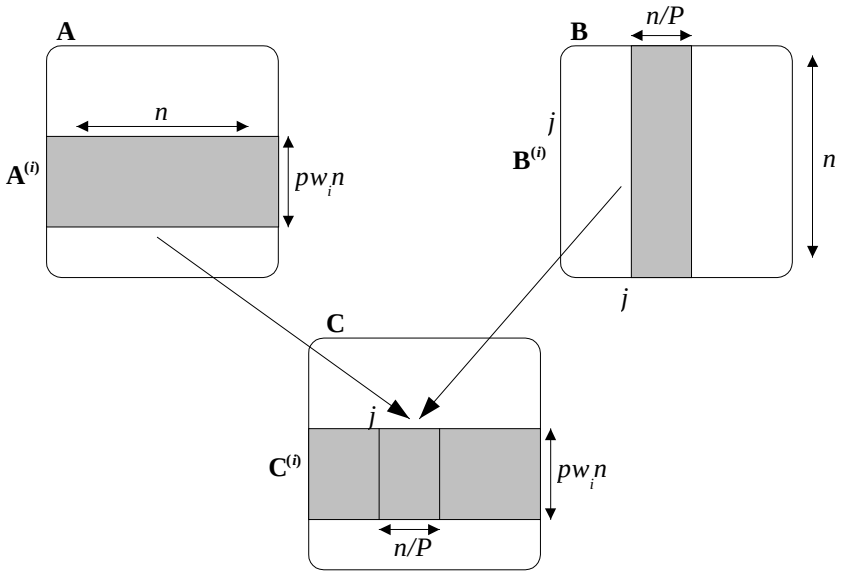


Figura 3.9: Cálculo Parcial de $C^{(i)}$ con B distribuida por Columnas.

Es claro que ambas alternativas de distribución de los datos de B no altera la cantidad de operaciones que se deben realizar, aunque en principio parece más sencilla (o un poco más intuitiva respecto de la definición de la multiplicación de matrices misma), la distribución por columnas, y por lo tanto será la elegida para el algoritmo.

**Resumen y Comentarios Acerca de la Distribución de Datos**. Las principales características de la distribución de datos entre las computadoras son:

- La matriz A y la matriz C se distribuyen por filas, con la cantidad de filas dada por la potencia de cálculo relativa de cada máquina. Es decir que la computadora $ws_i$ con potencia de cálculo relativa normalizada $pw_i$ tendrá una submatriz de A, $A^{(i)}$, y una submatriz de C, $C^{(i)}$, de $fA_i = pw_i n$ filas por $n$ columnas.
- La matriz B se distribuye de manera uniforme por columnas, lo cual implica que todas las computadoras tendrán la *misma* cantidad de columnas (y por lo tanto datos) de B. Es decir que la computadora $ws_i$ tendrá una submatriz de B, $B^{(i)}$, de $n$ filas por $cB_i = n/P$ columnas, donde $P$ es la cantidad total de computadoras.

Aunque se cumple que

$$\sum_{j=0}^{P-1}(pw_j) = 1 \quad \text{y} \quad \sum_{j=0}^{P-1}(n/P) = n$$

no necesariamente se tendrá que $fA_i = pw_i n$ sea un número entero, dado que $0 < pw_i < 1$, y por lo tanto, operando con números enteros, lo más probable es que

$$df = \lfloor fA_1 \rfloor + ... + \lfloor fA_P \rfloor < n$$

donde $\lfloor fA_i \rfloor$ es el mayor número entero tal que $\lfloor fA_i \rfloor < fA_i$.

Las filas que "quedan" sin repartir, *n-df*, se reparten uniformemente entre las computadoras $ws_0$, ..., $ws_{n-df-1}$. Dado que normalmente $P << n$ esta fila "agregada" puede considerarse irrelevante respecto a la posibilidad de generar desbalance de carga de cómputo. El principio a seguir con respecto a la cantidad de columnas de B ($n/P$ no necesariamente es un número entero), es similar.

## 3.4.2 Cómputo

Como es de esperar para la mayoría (o todos) los algoritmos numéricos que se resuelven en paralelo, se diseña bajo el modelo SPMD (Single Program - Multiple Data) y está dado directamente en pseudocódigo en la Figura 3.10 para la computadora $ws_i$.

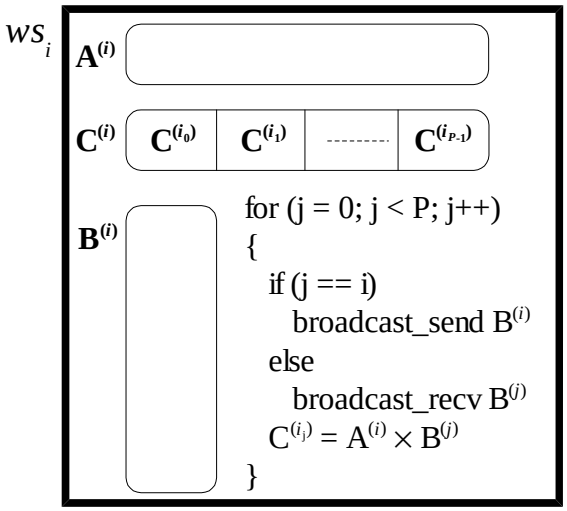


Figura 3.10: Pseudocódigo de la Multiplicación en $ws_i$.

donde

- P es la cantidad total de computadoras.
- Las matrices A y C están distribuidas por filas, y $A^{(i)}$ y $C^{(i)}$ son las submatrices que $ws_i$ contiene localmente.
- La matriz B está distribuidas por columnas, y $B^{(i)}$ es la submatriz que $ws_i$ contiene localmente.

La Figura 3.11 muestra esquemáticamente la secuencia de pasos de ejecución del algoritmo para cuatro computadoras. Tanto del pseudocódigo de la Figura 3.10 como de la secuencia de pasos que se muestra en la Figura 3.11, se puede ver que las características más importantes del algoritmo son:

- Tiene $P$ pasos de comunicación broadcast (mensajes broadcast), dado que cada una de las computadoras envía un mensaje broadcast y recibe $P$-1 mensajes broadcast.
  - Tiene $P$ pasos de cómputo local, dado que todas las computadoras realizan $P$ cálculos parciales ($C^{(i_k)}$, $\forall\ k = 0, 1, ..., P\text{-}1$) de la parte de la matriz C que almacenan localmente.
- La cantidad total de pasos Secuenciales de ejecución es $2P$, y el tiempo de cómputo total se calcula aproximadamente como lo muestra la Ecuación (3.6),

$$t_{par} = P\ (t_{bcast} + t_{cómp}) \qquad (3.6)$$

donde

- $t_{bcast}$ es el tiempo necesario para llevar a cabo una operación de broadcast de una submatriz $B^{(i)}$, recordando que $|B^{(i)}| \cong |B^{(j)}|$ $0 \le i, j \le P\text{-}1$, donde $|X|$ indica la cantidad de datos (elementos) de la matriz X.
- $t_{cómp}$ es el tiempo necesario para ejecutar el cómputo local en todas las computadoras, que es el mismo dado que la carga está balanceada en función de los datos de A y C que cada computadora almacena localmente.

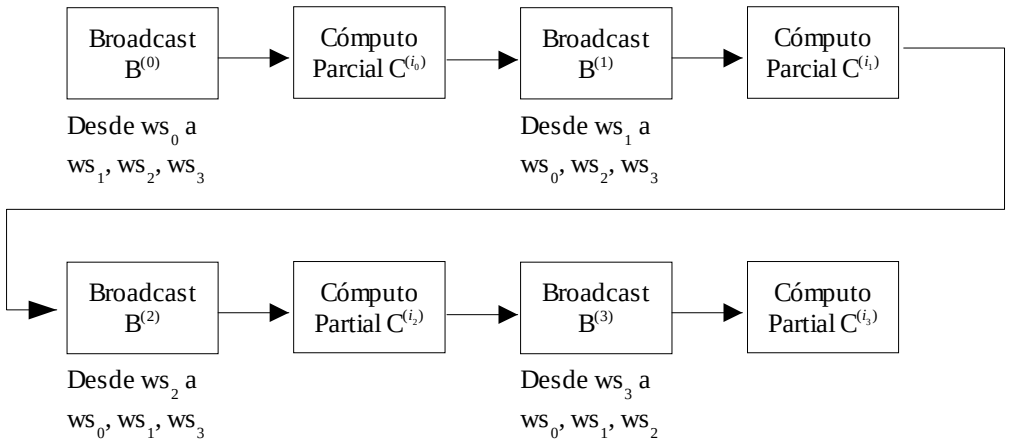


Figura 3.11: Secuencia de Pasos de Ejecución para Cuatro Computadoras.

En el contexto de las redes locales con interconectadas por redes Ethernet, el tiempo de comunicaciones a utilizar en la Ecuación (3.6), $t_{bcast}$, se puede calcular directamente utilizando la Ecuación (3.1), para lo cual es necesario conocer el tiempo de latencia y de ancho de banda asintótico de la red de comunicaciones. Sin embargo, esto no es posible en general para cualquier computadora paralela y en muchos casos aún en las redes locales interconectadas con Ethernet el tiempo de broadcast puede depender de la forma en que las bobliotecas de comunicaciones utilizadas para el pasaje de mensajes han sido implementadas. De hecho, es bastante usual que el tiempo de un mensaje broadcast con $k$ receptores sea $k$ veces mayor al de una comunicación punto a punto, dado que los mensajes broadcast se implementan como múltiples ($k$) mensajes punto a punto. En cualquier caso, se debe recalcar que tanto el tiempo de latencia como el ancho de banda asintótico de la red

de comunicaciones debería ser calculado desde los procesos de usuario que componen un programa paralelo [115] [136].

Por lo tanto, considerando que
- Se implementan los mensajes del tipo broadcast aprovechando la capacidad de broadcast de las redes Ethernet.
- Se conoce el tiempo de latencia de los mensajes entre procesos de un programa paralelo y se denota como $\alpha$.
- Se conoce el ancho de banda asintótico de los mensajes entre procesos de un programa paralelo y se denota como $1/\beta$ (expresado en términos del tipo de elementos que se procesan, tal como números en punto flotante de precisión doble: 8 bytes).
- La cantidad de datos de B que cada computadora tiene, $|B^{(i)}|$, está dada por $n$ filas por $cB_i = n/P$ columnas, es decir $n^2/P$ elementos.

$$t_{bcast} = \alpha + \beta\, n^2/P \tag{3.7}$$

Para calcular el tiempo de cómputo local a utilizar en la Ecuación (3.6), $t_{cómp}$, se pueden reutilizar los valores de capacidad de procesamiento de cada computadora utilizados para calcular las velocidades relativas normalizadas. De acuerdo con la Ecuación (3.3), para el cálculo de $pw_i$ ya se debería tener la capacidad de cómputo de $ws_i$ en Mflop/s, es decir Mflop/s($ws_i$). Estos valores de Mflop/s se pueden utilizar de dos maneras para el cálculo de tiempo local:
1. Teniendo en cuenta la capacidad de cómputo de una sola computadora: *Mflop/s*($ws_i$).
2. Teniendo en cuenta la capacidad de cómputo de la máquina paralela completa (con la suma de todas las computadoras): $\Sigma_{i=0}^{P-1}(Mflop/s(ws_i))$.

En el primer caso, teniendo en cuenta la capacidad de cómputo de una sola computadora, se utiliza el hecho de que todas las computadoras tienen que ejecutar la misma cantidad de operaciones de punto flotante y por lo tanto el tiempo de ejecución en todas ellas es el mismo. En el segundo caso, teniendo en cuenta la capacidad de cómputo de la máquina paralela completa, se utiliza que:
- Se conoce la capacidad de cálculo de la máquina paralela completa.
- Se conoce la cantidad total de operaciones que se deben realizar ($2n^3$-$n^2$).
- La cantidad total de operaciones se divide en $P$ pasos, por lo tanto en cada paso se llevan a cabo $1/P$ del total.

Por lo tanto, el tiempo de ejecución $t_{cómp}$ de cada uno de los pasos de procesamiento se puede estimar independientemente de la heterogeneidad de las computadoras con

$$t_{cómp} = \frac{2n^3 - n^2}{P\ pw} \tag{3.8}$$

donde $pw$ es la potencia de cálculo de la máquina paralela

$$pw = \sum_{i=0}^{P-1} Mflop/s(ws_i) \tag{3.9}$$

De esta manera, utilizando la Ecuación (3.7) y la Ecuación (3.8) en la Ecuación (3.6) se llega a que el tiempo de cómputo total del algoritmo está dado por

$$t_{par} = P\alpha + \beta n^2 + \frac{2n^3 - n^2}{pw} \qquad (3.10)$$

Es decir que, en el tiempo total se contabilizan *P* tiempos de latencia correspondientes a los *P* mensajes broadcast, más todo el tiempo de comunicaciones de los datos de la matriz B a la máxima velocidad de transferencia más el tiempo de cómputo de la multiplicación de matrices considerando la potencia de cálculo de la máquina paralela que es la suma de las potencias de cálculo de las computadoras utilizadas.

## 3.4.3 Cómputo Solapado con Comunicación

El algoritmo presentado previamente no tiene en cuenta la posibilidad de solapar cómputo con comunicaciones, dado que en la especificación misma (pseudocódigo de la Figura 3.10), no está contemplado. Como se ha mencionado antes, la posibilidad de solapar cómputo con comunicaciones en las computadoras paralelas tradicionales ha sido una constante y muchas veces asociada al diseño mismo de las redes de interconexión. En el caso de las computadoras estándares encontradas en las redes locales, esta capacidad no necesariamente se puede asegurar *a priori*. Sin embargo, adaptar el algoritmo anterior para que tenga la capacidad de solapar las comunicaciones con el cómputo local en cada computadora tiene varias ventajas:

- Aunque depende de la adaptación que se haga (sobrecarga agregada), en general no se pierde rendimiento con respecto al algoritmo presentado, ya que lo *peor* que puede suceder es que las comunicaciones se hagan secuenciales con respecto al cómputo.
- Se tiende a aprovechar la posibilidad de hacer cómputo solapado con comunicaciones en caso de que ésta exista en una o más computadoras. En general, se puede mejorar el rendimiento incluso aunque algunas computadoras puedan solamente recibir o enviar datos de forma solapada con el cómputo local.
- En el mejor de los casos, si la granularidad de la aplicación es suficientemente grande y además todas las computadoras tienen la capacidad de solapar las comunicaciones con cómputo local se puede llegar a un valor de rendimiento cercano al óptimo.
- El mismo algoritmo de cómputo solapado con comunicaciones puede identificar la capacidad o no que tiene cada máquina de hacer el solapamiento y de esa manera evaluar cada máquina por separado y la máquina completa (como una clase de *benchmark* para la evaluación de la capacidad de solapamiento de cómputo con comunicaciones).

La adaptación del algoritmo dado en la Figura 3.10 para aprovechar cómputo solapado con comunicaciones está dado en la Figura 3.12, donde

- P es la cantidad total de computadoras.
- Se mantiene la distribución de los datos de las matrices invariante con respecto al algoritmo anterior.
- Las funciones “broadcast_recv_b” y “broadcast_send_b” reciben y envían mensajes

broadcast de forma solapada si es posible (o en *background* desde el punto de vista del sistema operativo), mientras se computan los resultados parciales $C^{(i_j)}$.

- Se puede notar que el primer mensaje broadcast no se lleva a cabo de forma solapada dado que inicialmente solamente $ws_0$ tiene los datos de $B^{(0)}$ y para el primer paso de cómputo todas las computadoras necesitan $B^{(0)}$.
- El pseudocódigo de cada computadora se vuelve un poco más complejo pero no demasiado con respecto al de la Figura 3.10.

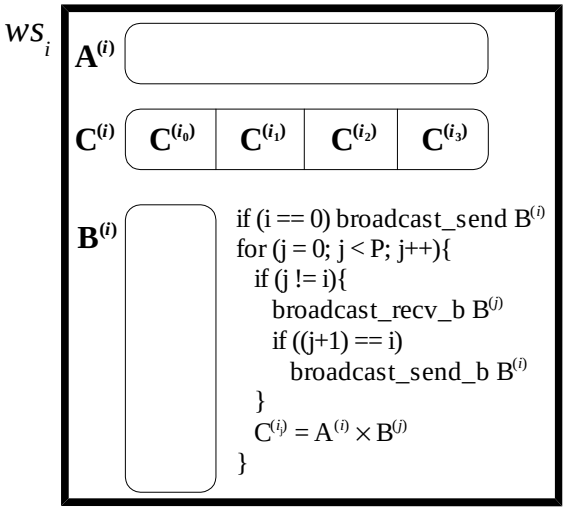


Figura 3.12: Pseudocódigo de Cómputo Solapado con Multiplicación en $ws_i$.

La Figura 3.13 muestra esquemáticamente la secuencia de pasos de ejecución del algoritmo para cuatro computadoras.

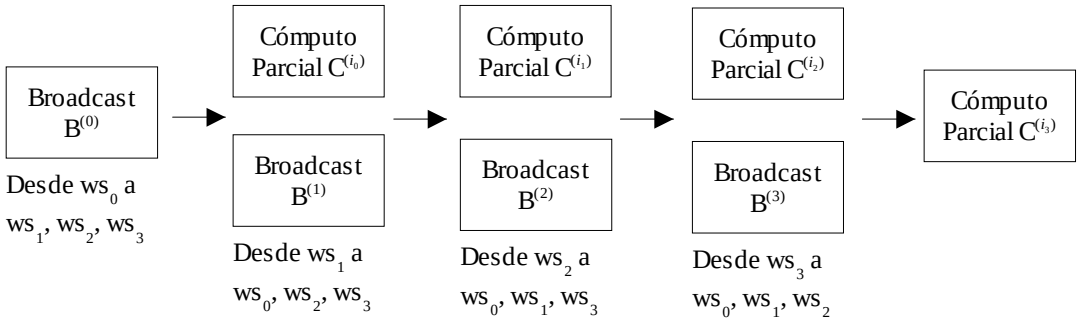


Figura 3.13: Cómputo Solapado con Comunicación en Cuatro Computadoras.

Tanto del pseudocódigo de la Figura 3.12 como de la secuencia de pasos que se muestra en la Figura 3.13, se puede notar que las características más importantes del algoritmo son:

- Tiene *P* pasos de comunicación broadcast (mensajes broadcast), dado que cada una de las computadoras envía un mensaje broadcast y recibe *P*-1 mensajes broadcast. Excepto el primer broadcast, los *P*-1 restantes se pueden llevar a cabo (dependiendo de las capacidades de las computadoras) de forma solapada con el cómputo local.
- Tiene *P* pasos de cómputo local, dado que todas las computadoras realizan *P* cálculos parciales ($C^{(i_k)}$, $\forall\ k = 0, 1, ..., P\text{-}1$) de la parte de la matriz C que almacenan localmente.
- La cantidad total de pasos secuenciales de ejecución es *P*+1.

El tiempo de cómputo total depende ahora de varios factores, tales como:

- La capacidad de solapar cómputo con comunicaciones de cada una de las computadoras. Esto determinará si las comunicaciones se llevan a cabo de forma "simultánea" con el cómputo como lo muestra la Figura 3.13 o no. Si una computadora no es capaz de solapar cómputo con comunicaciones, la tarea de cómputo y de comunicaciones se deberán hacer secuenciales.
- La influencia de la tarea de comunicación "en background" sobre el cómputo. Dado que se están transfiriendo datos, habrá uno o más procesos que controlen esta transferencia y por lo tanto utilizarán CPU y recursos como la jerarquía de memoria, por lo tanto el o los procesos de cómputo ahora competirán por estos recursos.
- La granularidad del problema, que determinará qué porcentaje de las comunicaciones se puede solapar con el cómputo. Expresado de otra manera, aunque una computadora pueda solapar cómputo con comunicaciones, si el tiempo de comunicar $B^{(i)}$ es mayor que el de cómputo local, necesariamente el tiempo de comunicaciones determinará el tiempo total de ejecución.
- Asumiendo el mejor de los casos, es decir que:
    - todas las computadoras son capaces de solapar cómputo con comunicaciones
    - y no hay penalización en la capacidad de cómputo por llevar a cabo transferencias de datos solapadas,

  el tiempo total de ejecución del algoritmo Figura 3.12 se puede estimar con la Ecuación (3.10),

$$t_{par} = t_{bcast} + (P\text{-}1)\ máx(t_{bcast}, t_{cómp}) + t_{cómp} \qquad (3.11)$$

donde $t_{bcast}$ y $t_{cómp}$ se calculan de acuerdo con la Ecuación (3.7) y con la Ecuación (3.8) respectivamente.

## 3.4.4 Reducción de Requerimientos de Memoria para Mensajes

Tanto el algoritmo presentado inicialmente en la Figura 3.10 con los períodos de cómputo y comunicación ejecutados de manera secuencial como el de la Figura 3.12, organizado para aprovechar la posibilidad de solapamiento de cómputo local con comunicaciones, tienen una característica común en cuanto a memoria para mensajes: ambos comunican toda una submatriz de B. Esto significa que, eventualmente, se necesitan buffers o memoria en cada computadora para $ws_i$, al menos, recibir los datos de la matriz B, $B^{(j)}$, de la computadora $ws_j$ sin destruir o sobreescribir los datos locales.

En el caso específico del algoritmo con cómputo y comunicaciones solapados esto no solamente genera requerimientos de memoria mayores sino que, además, hace cada mayor el tiempo inicial del envío de los primeros datos a todas las computadoras (Figura 3.12, primer paso de comunicación: envío de $B^{(0)}$) a medida que la matriz B es mayor y por lo tanto es mayor la submatriz involucrada en las comunicaciones.

La reducción de requerimientos de memoria para los mensajes es relativamente sencilla y también basada en la idea de procesamiento por bloques: en vez de enviar (y recibir) toda

la submatriz de B local de cada computadora se envía (recibe) por partes. Si, por ejemplo, cada submatriz de B, $B^{(i)}$, se envía en diez partes, cada una de estas partes será de $|B^{(i)}/10|$ elementos, y por lo tanto se reducen a 1/10 los requerimientos de memoria para las comunicaciones de manera *inmediata*. De hecho, el "bloque de comunicaciones" básico definido para los algoritmos presentados es, justamente $n/10$, donde $n$ es el orden de las matrices cuadradas a multiplicar.

Se debe recordar que la matriz $B^{(n \times n)}$ se distribuye por bloques de columnas entre las computadoras, es decir que habiendo P computadoras:
- Cada una de ellas tendrá una submatriz de B con $n$ filas y $n/P$ columnas.
- Las submatrices se envían-reciben en "bloques" de $n/10$ filas (para reducir a 1/10 los requerimientos de memoria para comunicaciones) y $n/P$ columnas (todas las columnas).

En el caso específico del algoritmo con cómputo y comunicaciones solapados esto implica, además, reducir el tiempo inicial del envío de los primeros datos a todas las computadoras (Figura 3.12, primer paso de comunicación: envío de $B^{(0)}$) porque ahora solamente hay que comunicar 1/10 del total de los datos de la submatriz para comenzar a hacer los cálculos parciales. Más específicamente, la propia trasnmisión de $B^{(0)}$ se solapa con los cálculos parciales en los que está involucrada la submatriz $B^{(0)}$.

### 3.4.5 Características Generales

Más allá de las diferencias en cuanto a la posibilidad de solapar cómputo con comunicaciones, las características más importantes de ambas formas presentadas de calcular en paralelo una multiplicación de matrices $C = A \times B$ en $P$ computadoras son:
- Siguen el modelo SPMD (Single Program - Multiple Data).
- No hay replicación de datos, ya que todos los datos se distribuyen y un dato reside localmente en una única computadora.
- Las comunicaciones entre computadoras son únicamente de tipo broadcast, y se llevan a cabo de forma tal que no generan interferencias en la red de comunicaciones de manera independiente del cableado que se utilice (hubs, switches, etc.).
- Se tiende a la máxima granularidad, todos los datos que se tienen que comunicar de una computadora a otra se transfieren de una sola vez. Esto tiende a reducir la penalización dada por el tiempo de latencia (startup) de los mensajes.
- El balance de carga en cuanto a cómputo está dado por la cantidad de datos de la matriz resultado que debe calcular cada computadora, la cual a su vez está definida proporcionalmente a su capacidad de cálculo relativa con respecto a las demás computadoras que calculan en paralelo. Por lo tanto, con la distribución de los datos de la matriz resultado, C, todas las computadoras realizan el procesamiento en aproximadamente el mismo tiempo.
- El balance de comunicaciones está dado por la asignación de los datos de la matriz B, que son los únicos que se transfieren entre las computadoras. Dado que todas tienen aproximadamente la misma cantidad de datos de la matriz B y que todas las computadoras envían un mensaje broadcast y reciben $P$-1 mensajes broadcast, todas las computadoras envían y reciben aproximadamente la misma cantidad de datos.

## 3.4.6 Otras Formas de Distribuir Datos

Tal como se adelantó en la explicación de la distribución de los datos para el algoritmo, existen formas más complejas que la presentada y utilizada en esta capítulo para distribuir los datos de las matrices entre múltiples procesadores de una máquina paralela. De hecho, han surgido algunas discusiones al respecto a nivel de reportes técnicos tales como [47] [106]. Una de las formas consideradas más apropiadas es la que se denomina como particionamiento cíclico en tablero por bloques en [79], aunque quizás sea más conocida como descomposición cíclica de bloques bidimensional (*two-dimensional block cyclic decomposition*), tal como se la menciona en [26] [21] [45] y en el contexto de la biblioteca denominada ScaLAPACK (Scalable LAPACK) [27] [28] [ScaLAPACK]. La Figura 3.14 muestra esta forma de distribuir los datos de una matriz A de 7×8 elementos, considerando bloques de 1×1 elementos, en una malla de 3×2 procesadores.

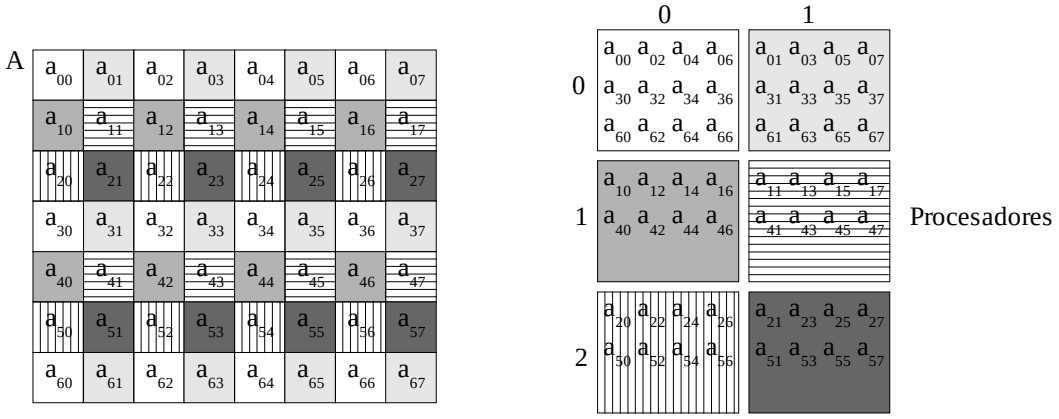


Figura 3.14: Descomposición Cíclica de Bloques Bidimensional.

Como lo muestra la Figura 3.14, la descomposición cíclica de bloques bidimensional está claramente orientada a las redes de interconexión de procesadores con topología de grilla o toro bidimensional. Además, es aplicable para cualquier tamaño de bloques de matrices, cualquier tamaño de matrices y cualquier malla o toro bidimensional de procesadores.

Toda la biblioteca ScaLAPACK asume que los datos (matrices y vectores) están distribuidos según esta descomposición. De hecho, dado que existen muchas alternativas para esta distribución, la representan con un arreglo descriptor de la distribución en el cual se identifica, entre otras cosas:

- Cantidad de filas de la matriz.
- Cantidad de columnas de la matriz.
- Cantidad de filas del bloque.
- Cantidad de columnas del bloque.

y este arreglo descriptor es utilizado (como parámetro) para resolver las rutinas básicas de cómputo tal como la multiplicación de matrices. La principal ventaja que se identifica explícitamente con esta distribución es la de balance de carga.

En el contexto de cómputo paralelo en redes de computadoras se deben hacer al menos tres consideraciones con respecto a la comparación de la Descomposición Cíclica de Bloques

Bidimensional con la presentada en este capítulo:
- Tamaño de los bloques a distribuir.
- Distribución de datos y su relación con la interconexión de los procesadores.
- Balance de carga del algoritmo.

**Tamaño de los Bloques a Distribuir**. Este es un serio inconveniente para las bibliotecas como ScaLAPACK y también PLAPACK [4] [141] [PLAPACK], que realizan las tareas de cómputo paralelo con un mismo tamaño de bloque (dado en ScaLAPACK por el arreglo descriptor de la distribución de datos). El tamaño de los bloques determina de manera directa dos aspectos en el procesamiento paralelo:
- Cómputo local en cada procesador.
- Comunicación entre procesadores.

Respecto del cómputo local, el tamaño de bloque está definido para obtener el óptimo rendimiento de los procesadores involucrados. Esto es relativamente *sencillo* en el caso de las máquinas paralelas con procesadores homogéneos, pero es inevitablemente un problema para las computadoras con procesadores heterogéneos. En este contexto, hasta es posible que dos procesadores puedan tener la misma capacidad de procesamiento (velocidad relativa o Mflop/s), pero con distintos valores de bloques, dado que esto depende fuertemente de las jerarquías de memoria (niveles y tamaños de memoria cache) que pueden ser muy diferentes.

Por lo tanto, tener un mismo tamaño de bloque es un problema más a resolver, dado que habría que determinar un valor que sea el *mejor*, lo cual no es sencillo en general dada la gran variedad de computadoras en las redes locales. Cualquiera sea el tamaño de bloque elegido, siempre se tendrá que algunas computadoras se aprovecharán al máximo en cuanto a rendimiento, mientras que en otras se procesará a una fracción del máximo rendimiento posible.

Por otro lado, en la distribución elegida no hay ningún tamaño de bloque predefinido, y esto tiene al menos dos consecuencias ventajosas:
- Se elimina un parámetro en las rutinas de procesamiento paralelo. La distribución de los datos es invariante, no hay más de una posibilidad para distribuir los datos de las matrices.
- El tamaño de bloques que optimiza el rendimiento de cada procesador se puede elegir localmente y no tiene efectos colaterales, no afecta el rendimiento ni el procesamiento en las demás computadoras.

Desde el punto de vista de ScaLAPACK esto puede ser considerado como una optimización que se debe hacer en el contexto de la heterogeneidad de las redes locales instaladas, pero que ScaLAPACK no resuelve (aún).

**Distribución de Datos e Interconexión de Procesadores**. Tanto en los los algoritmos orientados a computadoras paralelas de memoria distribuida presentados en el capítulo anterior como en los presentados en este capítulo, se puede notar que la distribución de los datos está íntimamente relacionada con el algoritmo de procesamiento y con la interconexión de los procesadores mismos. De hecho, esta es una característica distintiva de la mayoría (o *todos*) los algoritmos numéricos orientados a computadoras paralelas de memoria distribuida. Por lo tanto, una distribución bidimensional de datos está

directamente relacionada con una arquitectura de cómputo paralelo subyacente que tiene al menos dos características:
- Memoria distribuida.
- Procesadores interconectados con topología de malla o toro bidimensional.

Como ya se ha explicado, la interconexión de las redes locales de computadoras basadas en Ethernet (la *gran* mayoría) puede variar dependiendo principalmente del cableado, y más específicamente de la utilización de hubs y/o switches. Las redes en las que se utilizan switches se puede considerar la interconexión física como una malla o toro bidimensional (y de hecho, de varias maneras más, dada la flexibilidad de las redes con switches), pero es claro que esto no es válido para *todas* las redes, en donde se pueden encontrar hubs. Lo que sí es válido en general es que en todas las redes locales basadas en Ethernet es inmediato ("por hardware", dada la definición de Ethernet) considerar que la interconexión de las computadoras está proporcionada de manera lógica por un bus de comunicaciones, tal como el de la Figura 3.1 o el de la Figura 3.2.

Por lo tanto:
- No siempre se puede tener de manera inmediata una interconexión de las computadoras de las redes locales como una malla o toro bidimensional. Esto implica que todas las distribuciones de datos bidimensionales con sus algoritmos de cómputo orientados a redes de interconexión bidimensionales tendrán la tendencia de ser penalizados. Esta penalización será producto de la *serialización* de los mensajes sobre el bus de comunicaciones definido por Ethernet, sobre el cual se producirán las colisiones en las transferencias de datos.
- Al considerar una distribución de datos como la presentada en este capítulo se está considerando la interconexión de los procesadores en "topologías unidimensionales", tales como la del bus definido por Ethernet o los anillos con conexiones punto a punto. Más específicamente, la distribución de los datos presentada en este capítulo está orientada al aprovechamiento directo de las redes locales que en su mayoría están basadas en Ethernet y además de manera independiente de la variedad posible en el cableado de las mismas.

**Balance de Carga de los Algoritmos**. Como se afirma en general, el balance de carga es trivial en el caso de la Descomposición Cíclica de Bloques Bidimensional. Esta característica no necesariamente se mantiene en el contexto de los procesadores heterogéneos de las redes locales instaladas.

Pero desde el punto de vista de la distribución presentada en este capítulo, en términos generales no se pierde balance de carga ni generalidad de utilización en diferentes circunstancias, ya que:
- Se conserva el balance de carga de procesamiento en el caso de computadoras paralelas con procesadores homogéneos.
- Se conserva el balance de carga de procesamiento en el caso de computadoras paralelas con procesadores heterogéneos.
- El balance de carga es trivial, aunque en el caso de procesadores heterogéneos se debe conocer con anterioridad la velocidad relativa de los procesadores.

## 3.5 Resumen del Capítulo

A lo largo de este capítulo se han presentado:

- Las características de los clusters interconectados por redes Ethernet, tanto homogéneos como heterogéneos cuando se utilizan como plataformas de cómputo paralelo.
- Los principios de paralelización de aplicaciones a resolverse específicamente sobre clusters para obtener rendimiento optimizado. Estos principios se orientan a la utilización optimizada de todas las características de cómputo y procesamiento local en clusters.
- Dos algoritmos paralelos para multiplicación de matrices específicamente orientados a la obtención de rendimiento optimizado en clusters.
- Comentarios necesarios para la caracterización de cada algoritmo y también las comparaciones mínimas de estos algoritmos con los que se utilizan en ScaLAPACK, por ejemplo, básicamente en lo referente a la distribución de los datos.

No solamente quedan claras las diferencias a nivel de distribución de datos de los algoritmos propuestos en esta tesis, sino también en cuanto a la utilización (y en cierta forma *dependencia*) de los mensajes broadcast como la única forma de comunicar datos entre los procesos de una aplicación paralela.