Lección 13

Entrada y Salida

- **5.4 DISCOS**
- **5.4.1 Hardware de disco**
- Los discos son de varios tipos. Los más comunes son los discos magnéticos (discos duros y flexibles). Se caracterizan por el hecho de que las operaciones de lectura y escritura son igual de rápidas, lo que los hace ideales como memoria secundaria (como paginación o sistemas de archivos, por ejemplo).
- Algunas veces se utilizan arreglos de estos discos para ofrecer un almacenamiento altamente confiable. Para la distribución de programas, datos y películas, son también importantes varios tipos de discos ópticos (CD-ROMs, CD-grabable y DVD).
- **Discos magnéticos**
- Los discos magnéticos se organizan en cilindros, cada uno de los cuales contiene tantas pistas como cabezas apiladas en forma vertical. Las pistas se dividen en sectores. El número de sectores alrededor de la circunferencia es por lo general de 8 a 32 en los discos flexibles, y hasta varios cientos en los discos duros. El número de cabezas varía entre 1 y 16.
- Los discos antiguos tienen pocos componentes electrónicos y sólo producen un flujo de bits serial simple. En estos discos el controlador realiza la mayor parte del trabajo. En otros discos, en especial los discos **IDE** (**Electrónica de Unidad Integrada**) y **SATA** (**ATA Serial**), la unidad de disco contiene un microcontrolador que realiza un trabajo considerable y permite al controlador real emitir un conjunto de comandos de nivel superior. A menudo el controlador coloca las pistas en caché, reasigna los bloques defectuosos y mucho más.
- Una característica de dispositivo que tiene implicaciones importantes para el software controlador del disco es la posibilidad de que un controlador realice búsquedas en dos o más unidades al mismo tiempo. Éstas se conocen como **búsquedas traslapadas**.

- **RAID**
- Como hemos visto, el procesamiento en paralelo se utiliza cada vez más para agilizar el rendimiento de la CPU. Con el paso de los años, a varias personas se les ha ocurrido que la E/S en paralelo podría ser una buena idea también. En su artículo de 1988, Patterson y colaboradores sugirieron seis organizaciones de discos específicas que se podrían utilizar para mejorar el rendimiento del disco, su confiabilidad o ambas características (Patterson y colaboradores, 1988).
- Estas ideas fueron adoptadas de inmediato por la industria y han conllevado a una nueva clase de dispositivo de E/S conocido como **RAID**. Patterson y sus colaboradores definieron RAID como **Arreglo Redundante de Discos Independientes**.
- La idea básica detrás de un RAID es instalar una caja llena de discos a un lado de la computadora (que por lo general es un servidor grande), reemplazar la tarjeta controladora de discos con un controlador RAID, copiar los datos al RAID y después continuar la operación normal. Como los discos SCSI tienen un buen rendimiento, bajo costo y la capacidad de tener hasta siete unidades en un solo controlador (15 para SCSI amplio), es natural que la mayoría de los RAIDs consistan en un controlador RAID SCSI más una caja de discos SCSI que el sistema operativo considere como un solo disco grande. De esta forma no se requieren cambios en el software para utilizar el RAID.
- Además de aparecer como un solo disco para el software, todos los RAIDs tienen la propiedad de que los datos se distribuyen entre las unidades, para permitir la operación en paralelo. Patterson y sus colaboradores definieron varios esquemas distintos para hacer esto, y ahora se conocen como RAID nivel 0 hasta RAID nivel 5. Además hay unos cuantos niveles menores que no analizaremos.
- El término “nivel” es algo equivocado, debido a que no hay una jerarquía involucrada; simplemente son seis organizaciones distintas posibles.

El nivel RAID 0 se ilustra en la figura 5-20(a). Consiste en ver el disco virtual simulado por el RAID como si estuviera dividido en bandas de *k* sectores cada una, en donde los sectores 0 a *k* 1 son la banda 0, los sectores *k* a *2k* 1 son la banda 1, y así en lo sucesivo. Para *k* 1, cada banda es un sector; para *k* 2, una banda es de dos sectores, etc. La organización RAID de nivel 0 escribe bandas consecutivas sobre las unidades utilizando el método por turno rotatorio (*round-robin*), como se muestra en la figura 5-20(a) para un RAID con cuatro unidades de disco. Al proceso de distribuir datos sobre varias unidades de esta forma se le conoce como **reparto de bloques** (*striping*). Para cada ejemplo, si el software emite un comando para leer un bloque de datos de cuatro bloques consecutivos que empieza en un límite de bloque, el controlador RAID descompondrá este comando en cuatro comandos separados, uno para cada uno de los cuatro discos, y hará que operen en paralelo. Así, tenemos E/S en paralelo sin que el software lo sepa.

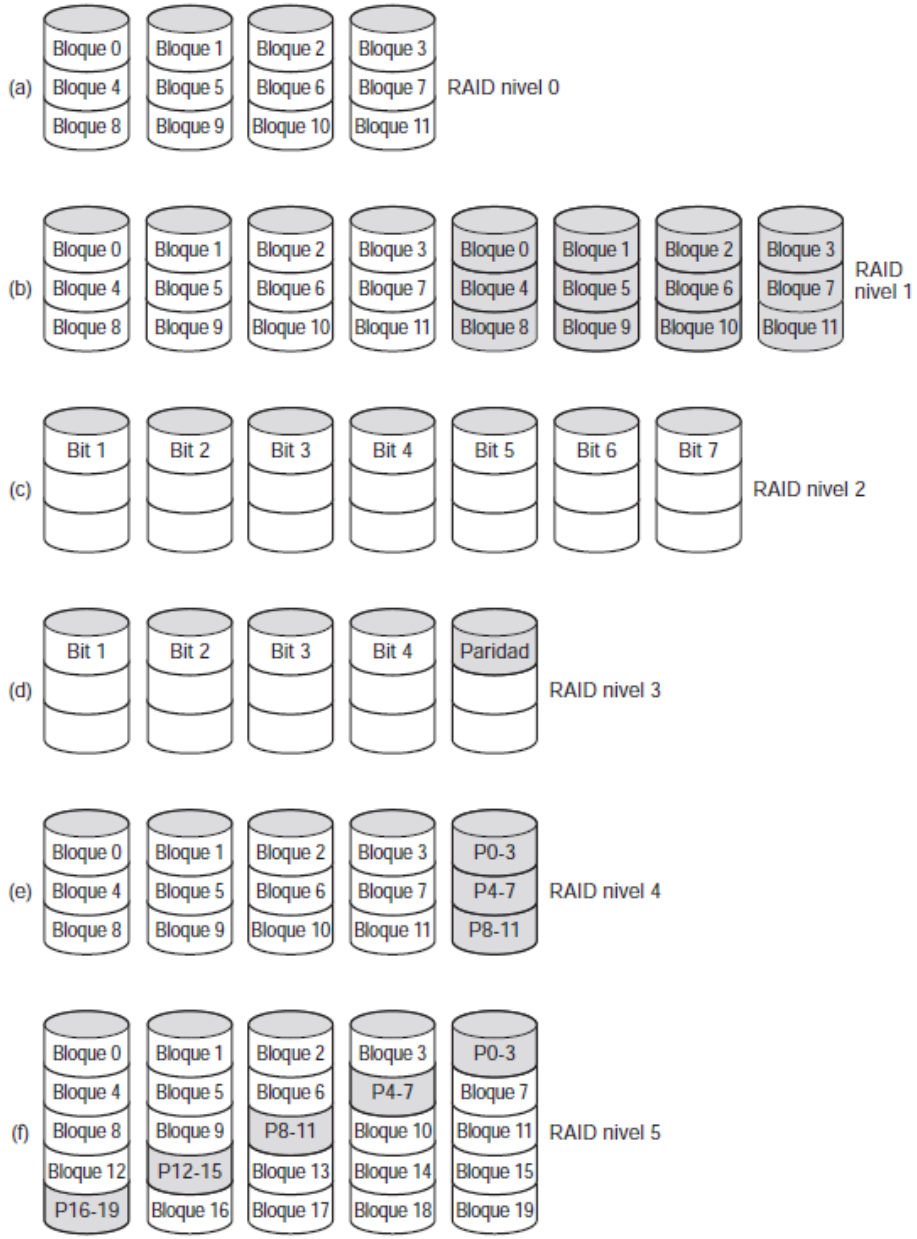

Figura 5-20. Niveles 0 a 5 de RAID. Las unidades de respaldo y de paridad se muestran sombreadas.

- La siguiente opción, el nivel 1 de RAID que se muestra en la figura 5-20(b), es un RAID verdadero.
- Duplica todos los discos, por lo que hay cuatro discos primarios y cuatro discos de respaldo. En una operación de escritura cada bloque se escribe dos veces. En una lectura se puede utilizar cualquiera de las copias, con lo que se distribuye la carga entre más unidades. En consecuencia, el rendimiento de escritura no es mejor que para una sola unidad, pero el rendimiento de lectura puede ser de hasta el doble. La tolerancia a fallas es excelente: si una unidad falla, simplemente se utiliza la copia. La recuperación consiste tan sólo en instalar una nueva unidad y copiar toda la unidad de respaldo en ella.
- A diferencia de los niveles 0 y 1, que funcionan con bandas de sectores, el nivel 2 de RAID funciona por palabra, tal vez hasta por byte. Imagine dividir cada byte del disco virtual en un par de medios bits (*nibbles* de 4 bits), y después agregar un código de Hamming a cada uno para formar una palabra de 7 bits, de la cual los bits 1, 2 y 4 son bits de paridad. Imagine que las siete unidades de la figura 5-20(c) se sincronizan en términos de la posición del brazo y de la posición rotacional. Entonces sería posible escribir la palabra con código Hamming de 7 bits sobre las siete unidades, un byte por unidad.
- El nivel 3 de RAID es una versión simplificada del nivel 2 de RAID. Se ilustra en la figura 5-20(d). Aquí se calcula un solo bit de paridad para cada palabra de datos y se escribe en una unidad de paridad. Al igual que en el nivel 2 de RAID, las unidades deben tener una sincronización exacta, ya que las palabras de datos individuales están distribuidas a través de varias unidades.

- Los niveles 4 y 5 de RAID funcionan con bloques otra vez, no con palabras individuales con paridad, y no requieren unidades sincronizadas. El nivel 4 de RAID [véase la figura 5-20(e)] es como el nivel 0, en donde se escribe una paridad banda por banda en una unidad adicional. Por ejemplo, si cada banda es de *k* bytes, se aplica un OR EXCLUSIVO a todos los bloques y se obtiene como resultado un bloque de paridad de *k* bytes. Si una unidad falla, los bytes perdidos se pueden recalcular a partir del bit de paridad, leyendo el conjunto completo de unidades.
- Este diseño protege contra la pérdida de una unidad, pero tiene un desempeño pobre para las actualizaciones pequeñas. Si se cambia un sector, es necesario leer todas las unidades para poder recalcular la paridad, que entonces se debe volver a escribir. De manera alternativa, puede leer los datos antiguos del usuario y los datos antiguos de paridad, y recalcular la nueva paridad a partir de ellos. Incluso con esta optimización, una pequeña actualización requiere dos lecturas y dos escrituras Como consecuencia de la pesada carga en la unidad de paridad, puede convertirse en un cuello de botella. Éste se elimina en el nivel 5 de RAID al distribuir los bits de paridad de manera uniforme sobre todas las unidades, por turno rotatorio (*round-robin*), como se muestra en la figura 5-20(f).
- Sin embargo, en caso de una falla de unidad, la reconstrucción del contenido de la unidad fallida es un proceso complejo.

- **5.4.2 Formato de disco**
- Un disco duro consiste en una pila de platos de aluminio, aleación de acero o vidrio, de 5.25 o 3.5 pulgadas de diámetro (o incluso más pequeños en las computadoras notebook). En cada plato se deposita un óxido de metal delgado magnetizable. Después de su fabricación, no hay información de ninguna clase en el disco.
- Antes de poder utilizar el disco, cada plato debe recibir un **formato de bajo nivel** mediante software. El formato consiste en una serie de pistas concéntricas, cada una de las cuales contiene cierto número de sectores, con huecos cortos entre los sectores. El formato de un sector se muestra en la figura 5-25.

Figura 5-25. Un sector de disco.

- El preámbulo empieza con cierto patrón de bits que permite al hardware reconocer el inicio del sector. También contiene los números de cilindro y sector, junto con cierta información adicional. El tamaño de la porción de datos se determina con base en el programa de formato de bajo nivel. La mayoría de los discos utilizan sectores de 512 bytes. El campo ECC contiene información redundante que se puede utilizar para recuperarse de los errores de lectura.

- La posición del sector 0 en cada pista está desfasada de la pista anterior cuando se aplica el formato de bajo nivel. Este desplazamiento, conocido como **desajuste de cilindros**, se realiza para mejorar el rendimiento. La idea es permitir que el disco lea varias pistas en una operación continua sin perder datos.
- Suponga que una petición necesita 18 sectores, empezando en el sector 0 de la pista más interna. Para leer los primeros 16 sectores se requiere una rotación de disco, pero se necesita una búsqueda para desplazarse una pista hacia fuera para llegar al sector 17. Para cuando la cabeza se ha desplazado una pista, el sector 0 ha girado más allá de la cabeza, de manera que se necesita una rotación para que vuelva a pasar por debajo de la cabeza. Este problema se elimina al desviar los sectores como se muestra en la figura 5-26.

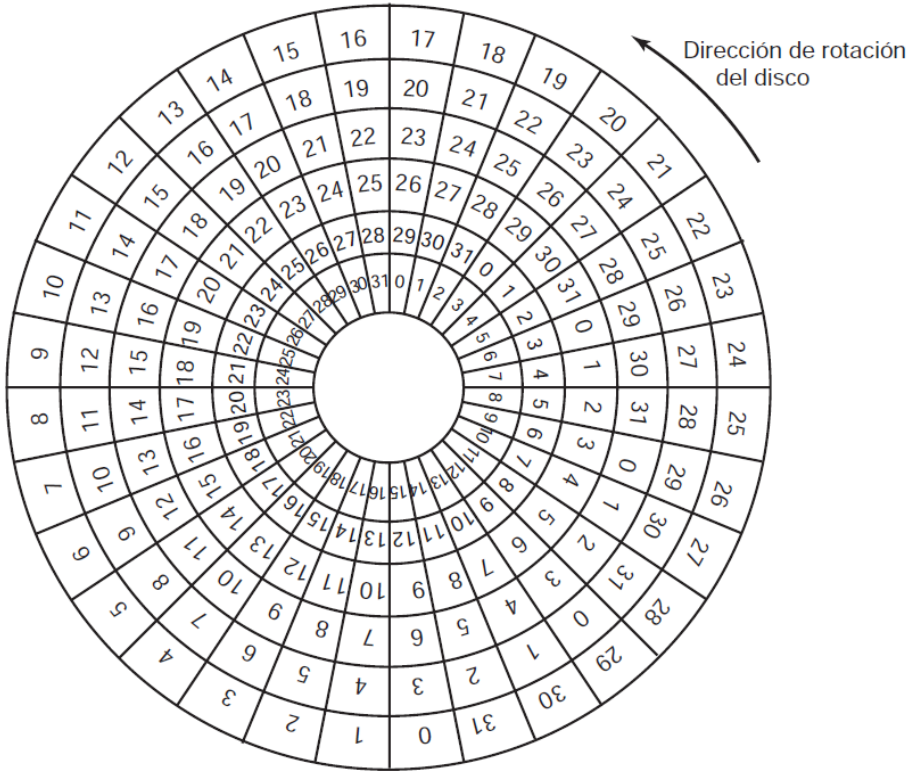

Figura 5-26. Una ilustración de la desviación de cilindros.

- Una vez que se completa el formato de bajo nivel, el disco se particiona. En sentido lógico, cada partición es como un disco separado. En el Pentium y la mayoría de las otras computadoras, el sector 0 contiene el **registro de inicio maestro** (MBR, por sus siglas en inglés), el cual contiene cierto código de inicio además de la tabla de particiones al final. La tabla de particiones proporciona el sector inicial y el tamaño de cada partición.
- El paso final de preparación de un disco para utilizarlo es realizar un **formato de alto nivel** de cada partición (por separado). Esta operación establece un bloque de inicio, la administración del espacio de almacenamiento libre (lista de bloques libres o mapa de bits), el directorio raíz y un sistema de archivos vacío. También coloca un código en la entrada de la tabla de particiones para indicarle qué sistema de archivos se va a utilizar en la partición, debido a que muchos sistemas operativos soportan varios sistemas de archivos incompatibles.
- Cuando se enciende la máquina, el BIOS se ejecuta al inicio, después lee el registro de inicio maestro y salta al mismo. Este programa de inicio comprueba a su vez cuál partición está activa. Después lee el sector de inicio de esa partición y lo ejecuta. El sector de inicio contiene un pequeño programa que por lo general carga un programa de inicio más grande, el cual busca en el sistema de archivos el kernel del sistema operativo. Ese programa se carga en memoria y se ejecuta.

- **5.4.3 Algoritmos de programación del brazo del disco**
- En primer lugar hay que considerar cuánto tiempo se requiere para leer o escribir en un bloque de disco. El tiempo requerido se determina en base a tres factores:
  - 1. Tiempo de búsqueda (el tiempo para desplazar el brazo al cilindro apropiado).
  - 2. Retraso rotacional (el tiempo para que el sector apropiado se coloque debajo de la cabeza).
  - 3. Tiempo de transferencia de datos actual.
- Para la mayoría de los discos, el tiempo de búsqueda domina los otros dos tiempos, por lo que al reducir el tiempo de búsqueda promedio se puede mejorar el rendimiento del sistema de manera considerable.
- Si el software controlador de disco acepta peticiones una a la vez, y las lleva a cabo en ese orden, es decir, **Primero en llegar, primero en ser atendido**, no se puede hacer mucho para optimizar el tiempo de búsqueda. Sin embargo, hay otra estrategia posible cuando el disco está muy cargado. Es probable que mientras el brazo esté realizando una búsqueda a favor de una petición, otros procesos puedan generar otras peticiones de disco. Muchos controladores de disco mantienen una tabla, indexada por número de cilindro, con todas las peticiones pendientes para cada cilindro encadenadas en una lista enlazada, encabezada por las entradas en la tabla.
- Dado este tipo de estructura de datos, podemos realizar mejorías con base en el algoritmo de programación del primero en llegar, primero en ser atendido. Para ver cómo, considere un disco imaginario con 40 cilindros. Llega una petición para leer un bloque en el cilindro 11; mientras la búsqueda en el cilindro 11 está en progreso, llegan nuevas peticiones para los cilindros 1, 36, 16, 34, 9 y 12, en ese orden, y se introducen en la tabla de peticiones pendientes, con una lista enlazada separada para cada cilindro. Las peticiones se muestran en la figura 5-28.

- Cuando termina la petición actual (para el cilindro 11), el software controlador de disco puede elegir qué petición manejar a continuación. Utilizando FCFS, el siguiente cilindro sería el 1, luego el 36, y así en lo sucesivo. Este algoritmo requiere movimientos del brazo de 10, 35, 20, 18, 25 y 3, respectivamente, para un total de 111 cilindros.
- De manera alternativa, siempre podría manejar la petición más cercana primero, para minimizar el tiempo de búsqueda. Dadas las peticiones de la figura 5-28, la secuencia es 12, 9, 16, 1, 34 y 36, y se muestra mediante la línea dentada en la parte inferior de la figura 5-28. Con esta secuencia, los movimientos del brazo son 1, 3, 7, 15, 33 y 2 para un total de 61 cilindros. Este algoritmo de **La búsqueda del trabajo más corto primero** (*Shortest Seek First,* SSF) recorta el movimiento total del brazo casi a la mitad, en comparación con FCFS.

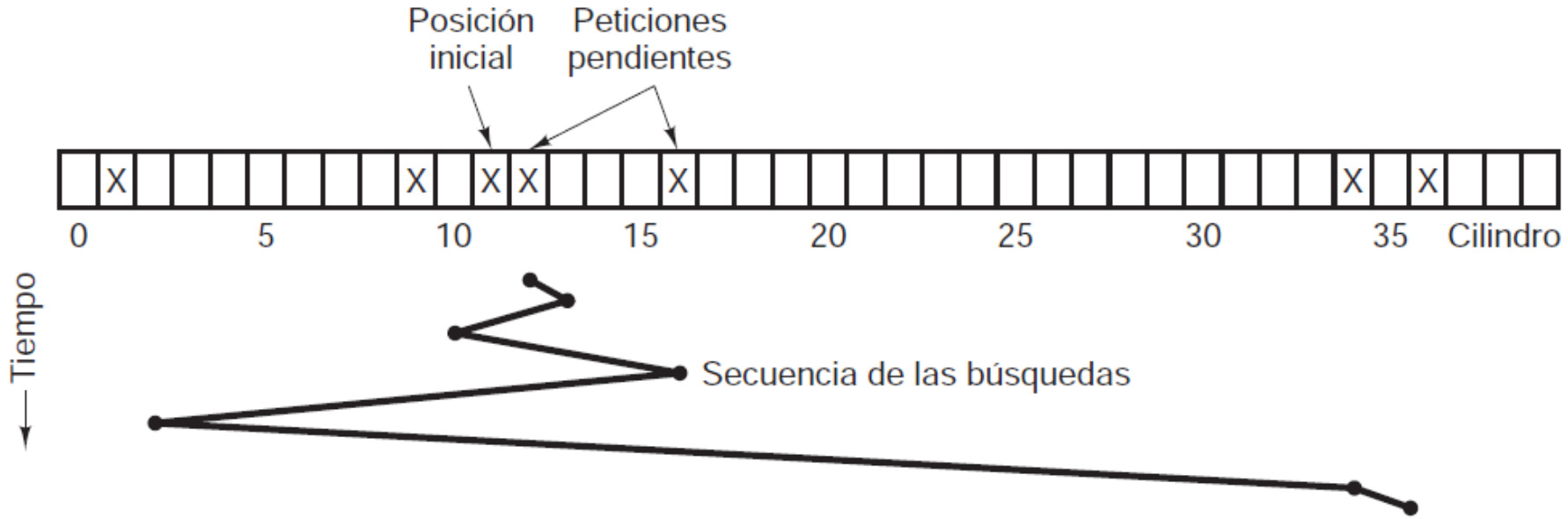

Figura 5-28. Algoritmo de planificación de disco de La búsqueda del trabajo más corto primero (SSF).

- Por desgracia, SSF tiene un problema. Suponga que siguen llegando más peticiones mientras se están procesando las peticiones de la figura 5-28. Por ejemplo, si después de ir al cilindro 16 hay una nueva petición para el cilindro 8, esa petición tendrá prioridad sobre el cilindro 1. Si después llega una petición para el cilindro 13, el brazo irá a continuación a 13, en vez de ir a 1. Con un disco que contenga mucha carga, el brazo tenderá a permanecer en la parte media del disco la mayor parte del tiempo, por lo que las peticiones en cualquier extremo tendrán que esperar.
- Los edificios altos también tienen que lidiar con esta concesión. El problema de planificar un elevador en un edificio alto es similar al de planificar un brazo de disco. Las peticiones llegan en forma continua, llamando al elevador a los pisos (cilindros) al azar. La computadora que opera el elevador podría llevar fácilmente la secuencia en la que los clientes oprimieron el botón de llamada, y atenderlos utilizando FCFS o SSF.
- Este algoritmo, conocido como **algoritmo del elevador** tanto en el mundo de los discos como en el mundo de los elevadores, requiere que el software mantenga 1 bit: el bit de dirección actual, *ARRIBA* o *ABAJO*. Cuando termina una petición, el software controlador del disco o del elevador comprueba el bit. Si es *ARRIBA,* el brazo o cabina se desplaza a la siguiente petición pendiente de mayor prioridad. Si no hay peticiones pendientes en posiciones mayores, el bit de dirección se invierte.
- Cuando el bit se establece en *ABAJO,* el movimiento es a la siguiente posición de petición con menor prioridad, si la hay.