

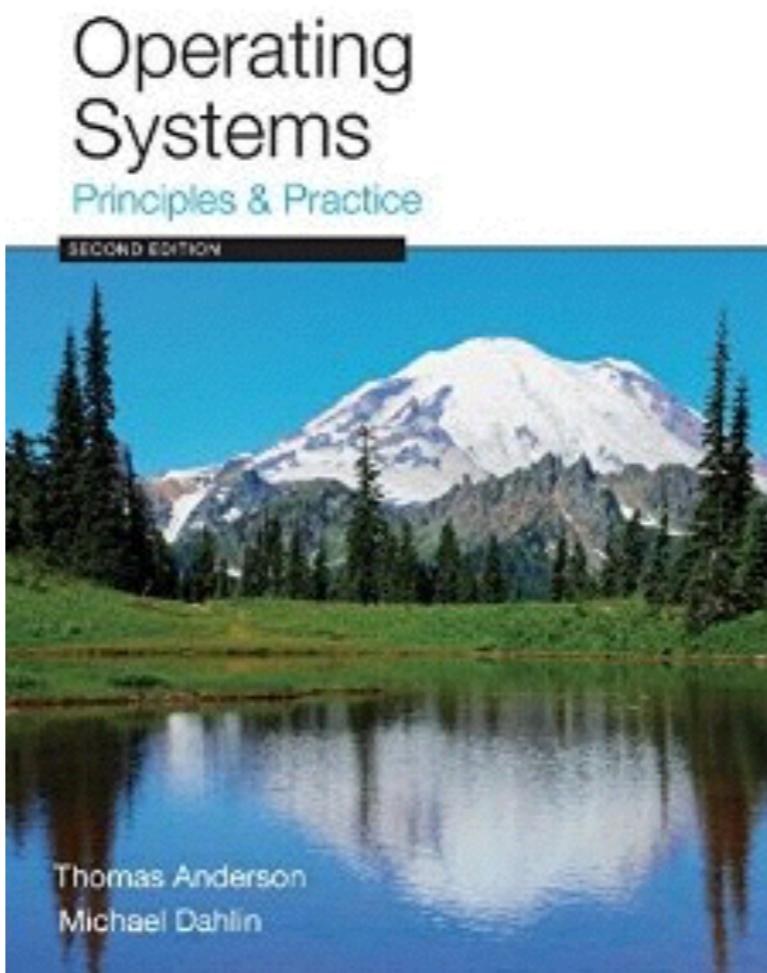


Sistemas Operativos

Capítulo 10 El Sistema de Archivos

Prof. Javier Cañas R.

Estos apuntes están tomados en parte del texto:
“Operating System: Principles and Practice” de T.
Anderson y M. Dahlin



Temario

1. Introducción
2. La Abstracción del Sistema de Archivos
3. Dispositivos de Almacenamiento
4. Archivos y Directorios
5. Disponibilidad del Almacenamiento

1 Introducción

- Los sistemas computacionales deben ser capaces de almacenar información en forma confiable.
- Los usuarios hacen fuertes demandas sobre los sistemas de almacenamiento:
 - **Confiabilidad:** Datos guardados en forma segura, aún ante caídas del SO.
 - **Alta capacidad y bajo costo:** 1 GB almacena 300 fotos aprox. ¿Qué significa imprimir 1 TB de texto?. Calcular!.
 - **Alto desempeño:** El acceso a datos por los programas debe ser rápido.
 - **Espacio de nombres:** Facilidad de identificar datos de interés.
 - **Compartición controlada:** Compartir datos controladamente.

Persistencia

- El contenido de la DRAM se puede perder ante una caída o falla de energía.
- Memoria no volátil tiene una mayor capacidad, menor costo por bit, pero mayor tiempo de acceso.
- El acceso en memorias no volátiles es por bloque, tanto en discos rotacionales como SSD. El tamaño de estos bloques varía entre 512B a 2KB.
- El tiempo de acceso a un sector de disco es alrededor de 10mseg y el tiempo de acceso de DRAM es de 100nseg

Impacto en Desempeño

- Dadas las limitaciones de los dispositivos de almacenamiento actuales, la ilusión de alta confiabilidad y desempeño de un sistema de archivo es imperfecta.
- ¿Qué pasa al editar un documento grande con imágenes que periódicamente se auto-salvan?.
 - Podrían ocurrir situaciones inesperadas.

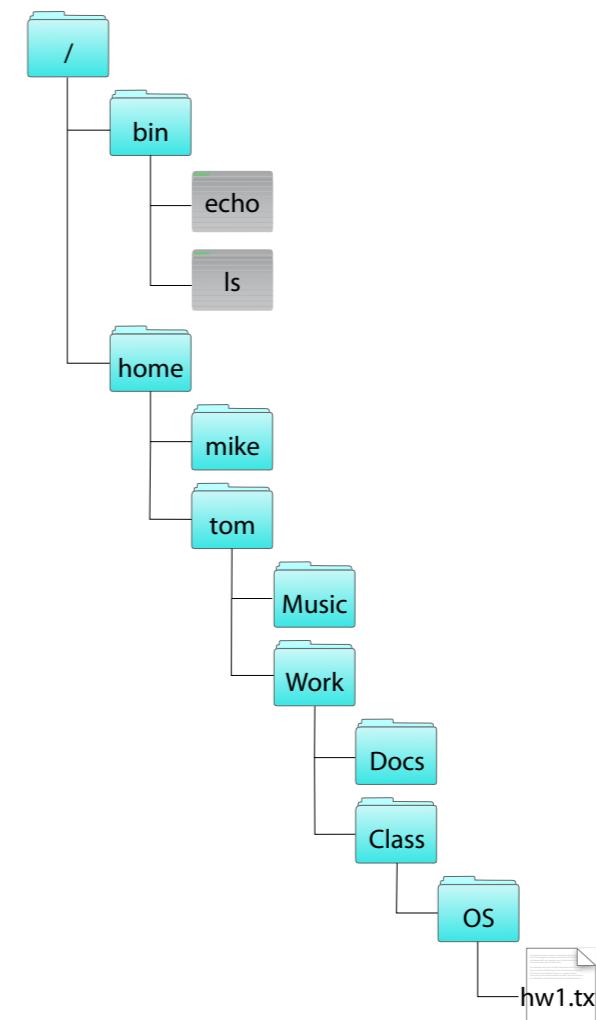
- Situaciones inesperadas:
 - **Desempeño bajo:** Aún pequeñas actualizaciones significan reescribir completamente el archivo. Cada “auto-save” puede tomar varios segundos.
 - **Corrupción:** Si la aplicación reescribe datos, un crash puede dejar inconsistente la versión actual del documento.
 - **Pérdida de archivo:** Si en vez de reescribir el documento, la aplicación la actualiza a un nuevo archivo, un crash puede dejar la aplicación sin copias.

2 La Abstracción del Sistema de Archivo

- Un sistema de archivo es una abstracción del SO que proporciona datos con nombre en forma persistente.
- Un sistema de archivo está definido por los conjuntos de datos (archivo) y los nombres (directorio).
- Un archivo puede ser: `/Users/javiercanas/Documents/MATLAB/Scripts/Chol.m`. Este archivo puede estar en el bloque `0x0BF53ED`.
- La metadata es la información que maneja el SO. Por ejemplo: tamaño, tiempos diversos, inf. de seguridad.
- Los datos son sólo un arreglo de bytes sin ningún tipo. Son las aplicaciones las que interpretan estos bytes.

Directarios

- Los archivos contienen datos y metadatos, los directorios proporcionan los nombres.
- Los directorios se organizan jerárquicamente.
- En Unix, . indica el directorio actual, .. el directorio padre y ~ el directorio *home* del usuario.

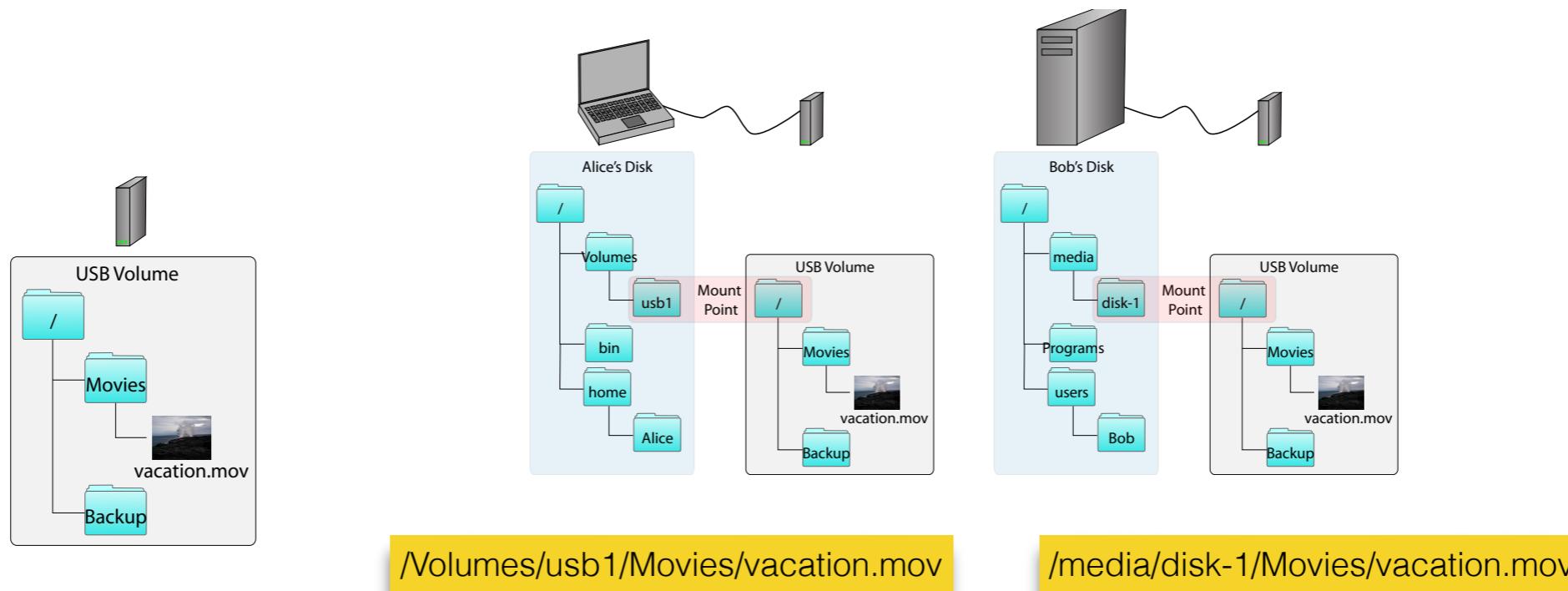


Volúmenes

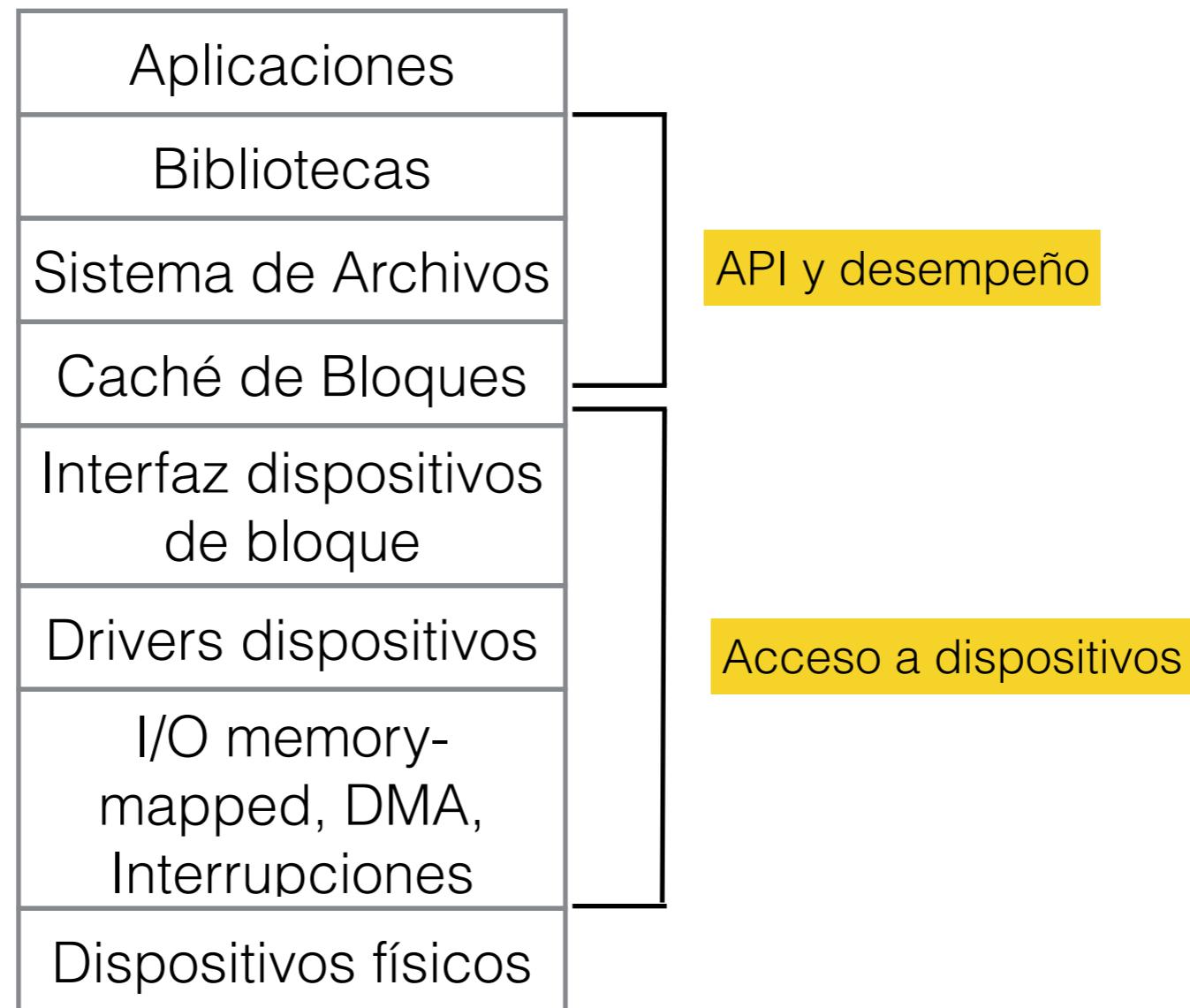
- Cada instancia de un sistema de archivos, administra los archivos y directorios de un **volumen**.
- Un **volumen** es un conjunto de recursos físicos de almacenamiento que forman un dispositivo de almacenamiento.
- Un **volumen** corresponde a un disco lógico. En un caso simple, corresponde a un drive de disco. También un drive puede estar particionado y almacenar múltiples volúmenes. También muchos discos pueden estar agrupados en un volumen simple.

... Volúmenes

- Múltiples sistemas de archivos almacenados en múltiples volúmenes se pueden montar en un sistema de directorio simple.



3 Arquitectura del Sistema de Archivos

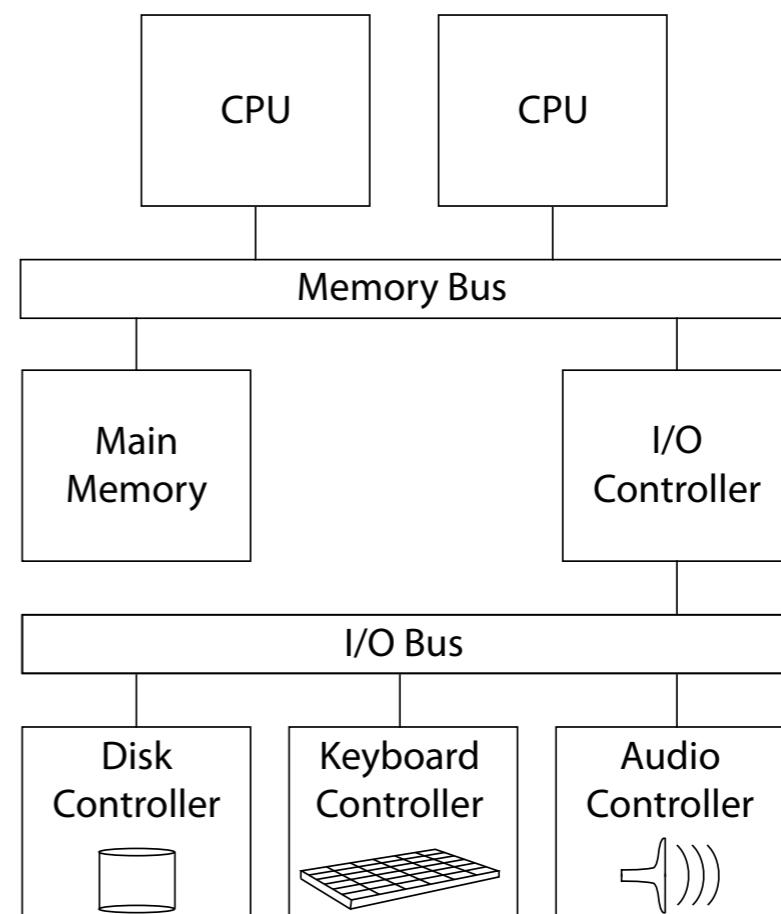


Caché de Bloques

- La Caché de bloques mantiene en memoria principal bloques de disco recientemente leídos, actúa como buffer cuando son escritos y posteriormente son escritos en disco.
- Actúan también como puntos de sincronización frente a escritura y lecturas.
- Para optimizar el desempeño se utiliza el prefetching: si un proceso lee los primeros dos bloques, el SO lee los siguientes 10.

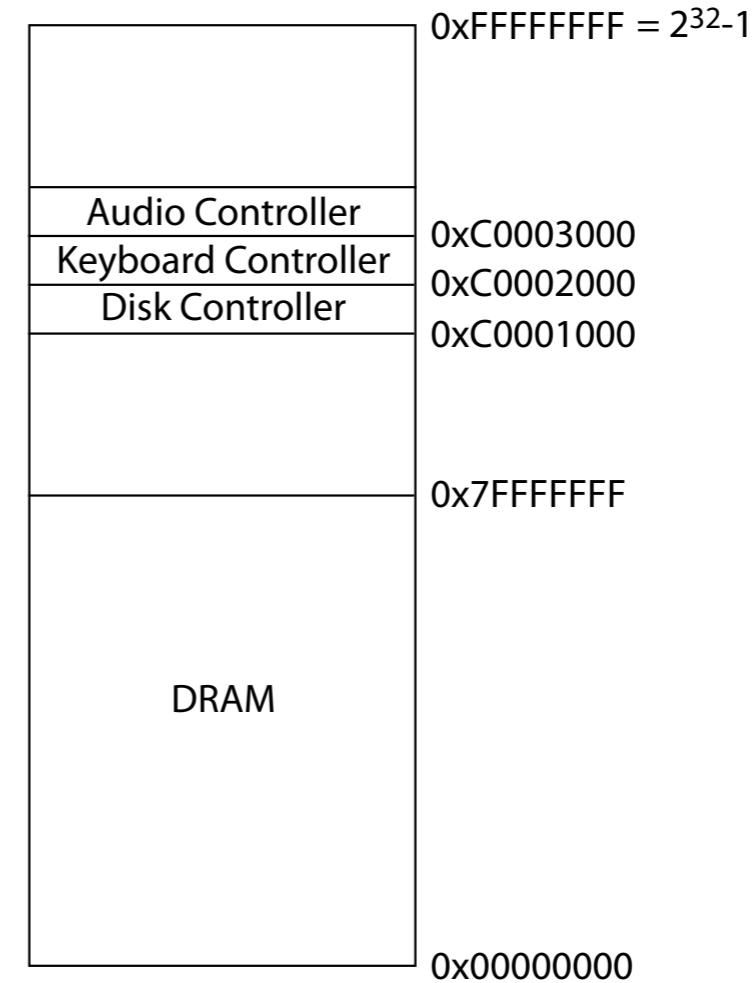
E/S mapeada en memoria

- Dispositivos de E/S se conectan al bus de memoria a través de un bus de E/S.
- Para permitir la escritura de registros de control, se implementa un E/S mapeada en memoria: cada dispositivo tiene sus registros de control en un rango de direcciones de la RAM



Ejemplo de mapa de memoria

- Lecturas y escrituras de la CPU a estos rangos de direcciones, no van a la memoria principal sino a los controladores de E/S.
- Por ejemplo el SO lee el último carácter del teclado leyendo la dirección física 0xC00002000.
- Este método es el dominante actualmente para accesar dispositivos.



DMA: Acceso Directo a la Memoria

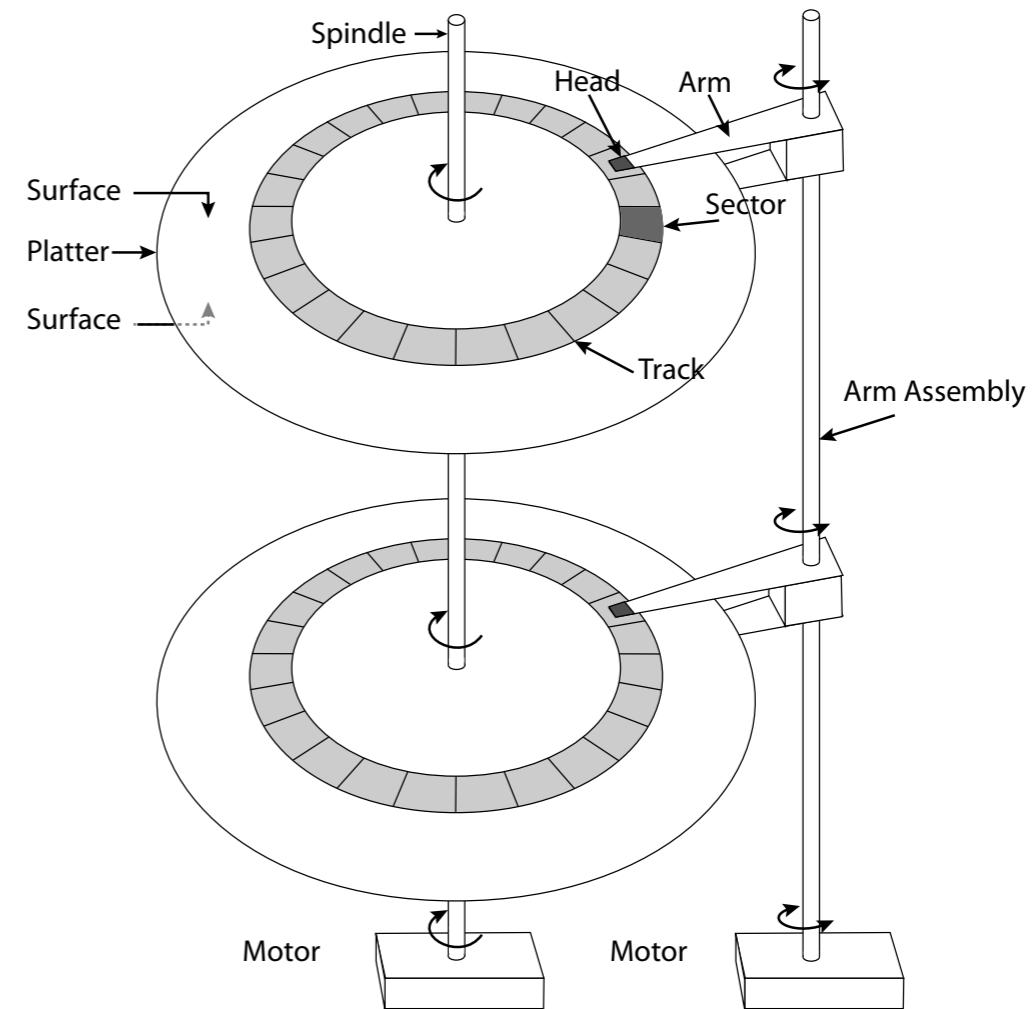
- Muchos dispositivos de E/S transfieren datos en bloques de bytes. Por ejemplo leer una palabra de disco significa transferir 1 o 2 KB.
- Al utilizar DMA, el dispositivo de E/S copia un bloque de datos directamente entre el buffer del controlador y la memoria principal.
- Una vez iniciada la transferencia DMA, el SO no puede utilizar la página física para cualquier otro propósito hasta que la transferencia se complete. Se dice que el SO pincha (pin) la página hasta que esté completa.

3 Dispositivos de Almacenamiento

- Para optimizar el desempeño de un sistema de archivos, es necesario tanto para el diseñador del sistema de archivos como el diseñador de aplicaciones conocer las características físicas de los dispositivos de almacenamiento.
- Comercialmente hay dos tipos de almacenamiento persistente: *discos magnéticos* (HDD) y *memorias flash* (SSD).

Disco magnético

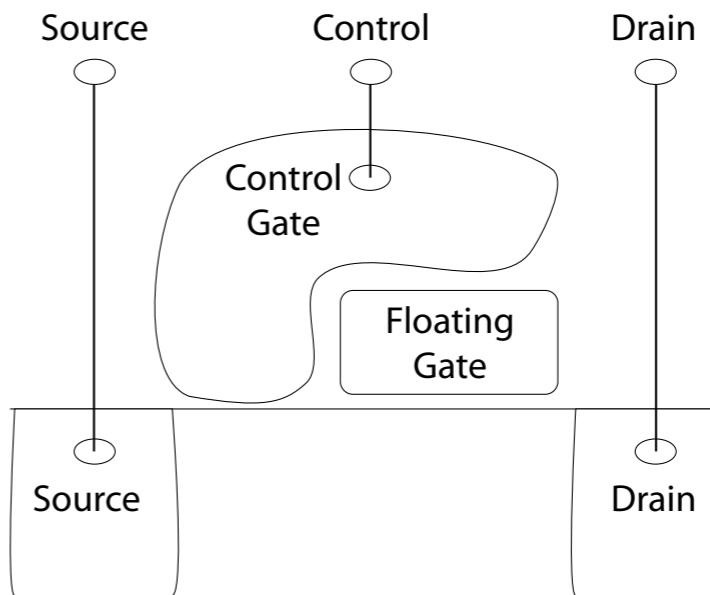
- El tamaño de un sector es típicamente **512B**.
- Buffer controlador= 16MB app
- $T_{\text{acceso}} = T_{\text{seek}} + T_{\text{rotacion}} + T_{\text{transferencia}}$
- Rotación=7200 RPM (120Hz)
- $T_{\text{seek}} = 10\text{mseg}$ a 12mseg



Almacenamiento Flash

- Antes de escribir, el bloque debe borrarse escribiendo un “1”.
- Sólo se puede borrar por **bloque de borrado**: 128KB a 512KB.
- Flash NAND: páginas entre 2KB a 4KB (bloques de lectura y escritura)
- Tiempo de escritura y lectura: Decenas de microsegundos

Transistor de gate flotante



El gate flotante almacena carga.

... Flash

- Almacenamiento Flash se puede organizar como NOR y como NAND.
- NOR: Se pueden leer y escribir palabras individuales. Se utiliza para firmware.
- NAND: Se pueden escribir y leer una página a la vez. Tiene mayor densidad que la NOR.

Comparación



Metric	Spinning Disk	Flash
Capacity/Cost	Excellent	Good
Sequential BW/Cost	Good	Good
Random I/O per Second/Cost	Poor	Good
Power Consumption	Fair	Good
Physical Size	Good	Excellent

4 Archivos y Directorios

- El sistema de archivos mapea nombres y offsets con el almacenamiento físico de bloques de una manera eficiente.
- Las ideas que sustentan esta implementación son 4:
 1. Directorios
 2. Estructuras de índices
 3. Mapas de espacio libre
 4. Heurísticas de localidad

Directorios

- El sistema de archivos mapea nombres de archivo y offset a bloques específicos en dos pasos:

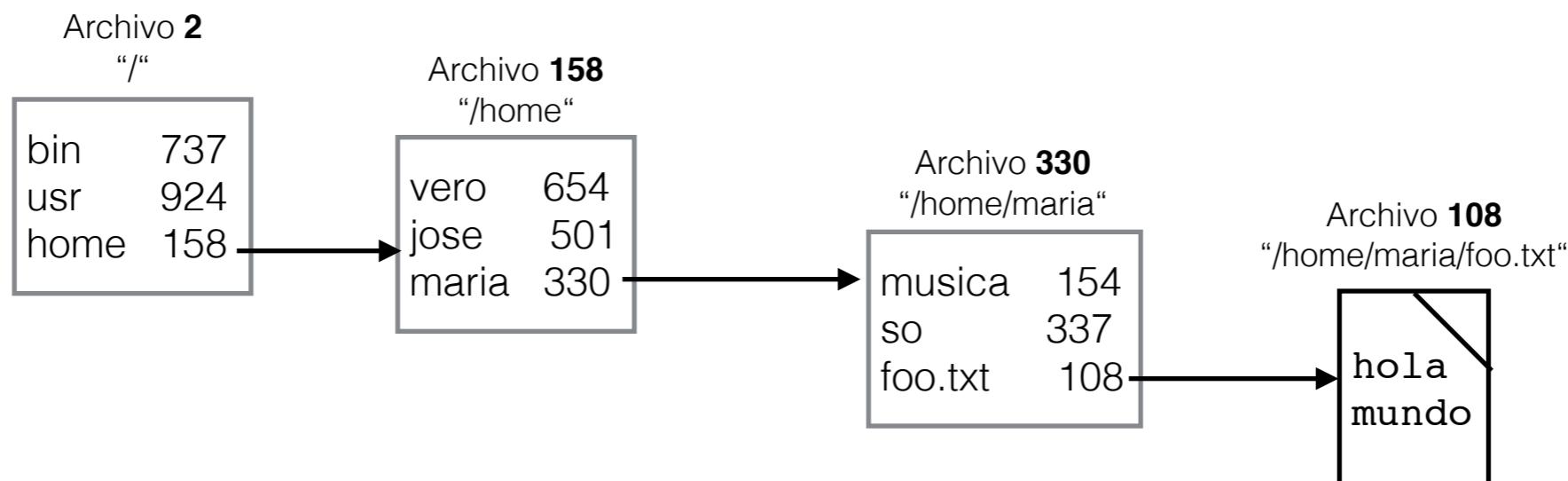


- La estructura de índice es una estructura de datos persistente que por flexibilidad es normalmente un árbol.
- Un directorio es un archivo que contiene una lista de nombre → número

musica 340
tareas 219
test.txt 986

... Directorios

- Un directorio es un archivo. En UNIX y Linux, se usa el número 2 como número de archivo para el directorio root.



Búsqueda de Bloques

- Una vez convertido un nombre en un número de archivo utilizando un directorio, es necesario encontrar los bloques de datos que lo componen.
- Adicionalmente el sistema de archivos debe proveer:
 - Soporte para acceso secuencial
 - Acceso random a cualquier bloque
 - Overhead limitado para archivos pequeños
 - Escalabilidad para soportar archivos grandes
 - Almacenamiento de metadata

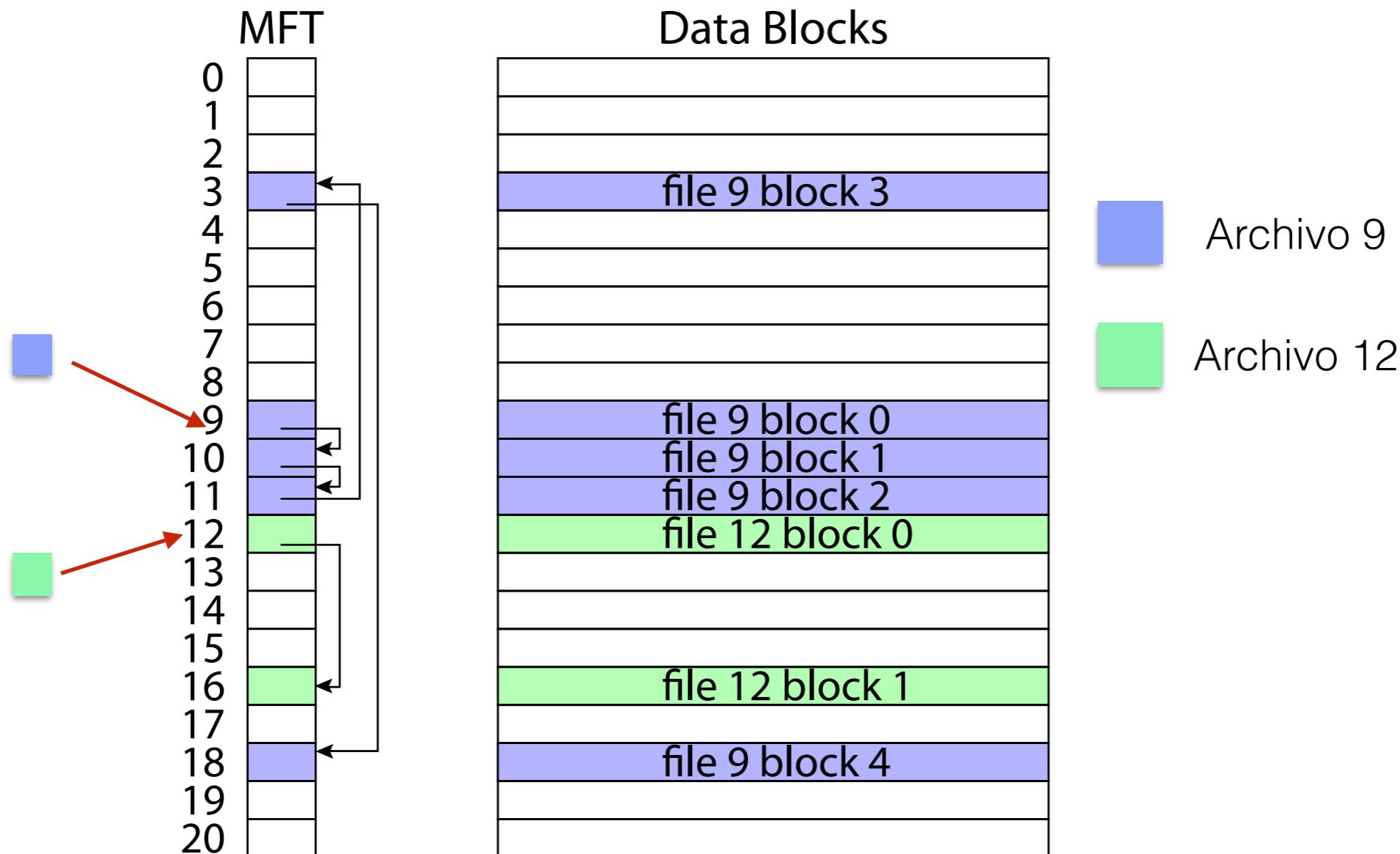
Opciones comunes de organización

	FAT	FFS	NTFS
Index structure	Linked list	Tree (fixed, assym)	Tree (dynamic)
granularity	block	block	extent
free space allocation	FAT array	Bitmap (fixed location)	Bitmap (file)
Locality	defragmentation	Block groups + reserve space	Extents Best fit defrag

Microsoft File Allocation Table (FAT)

- Su estructura es de una lista enlazada de índices.
 - Estructura simple, fácil de implementar.
 - Se utiliza en memorias USB
- La versión FAT-32 soporta hasta 2^{28} bloques y 4GB
- Cada archivo es una lista enlazada de bloques.

FAT con 2 archivos



FAT: Observaciones

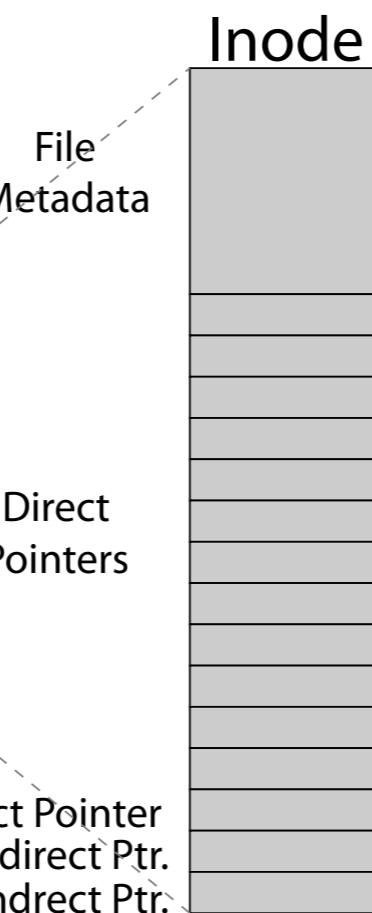
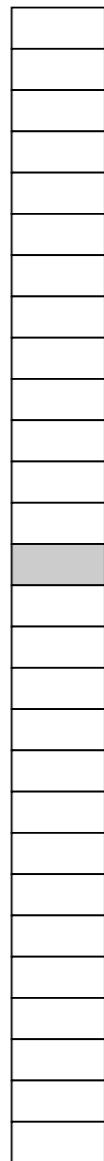
- Pro
 - ▶ Es fácil encontrar un bloque libre
 - ▶ Es fácil agregar a un archivo
 - ▶ Es fácil borrar un archivo
- Contra
 - ▶ Acceso lento a archivos pequeños
 - ▶ Acceso random lento
 - ▶ Fragmentación
 - Los bloques pueden estar muy dispersos
 - Archivos en el mismo directorio pueden estar muy dispersos
 - Se hace más complejo si el disco está lleno

Berkeley UNIX FFS (Fast File System)

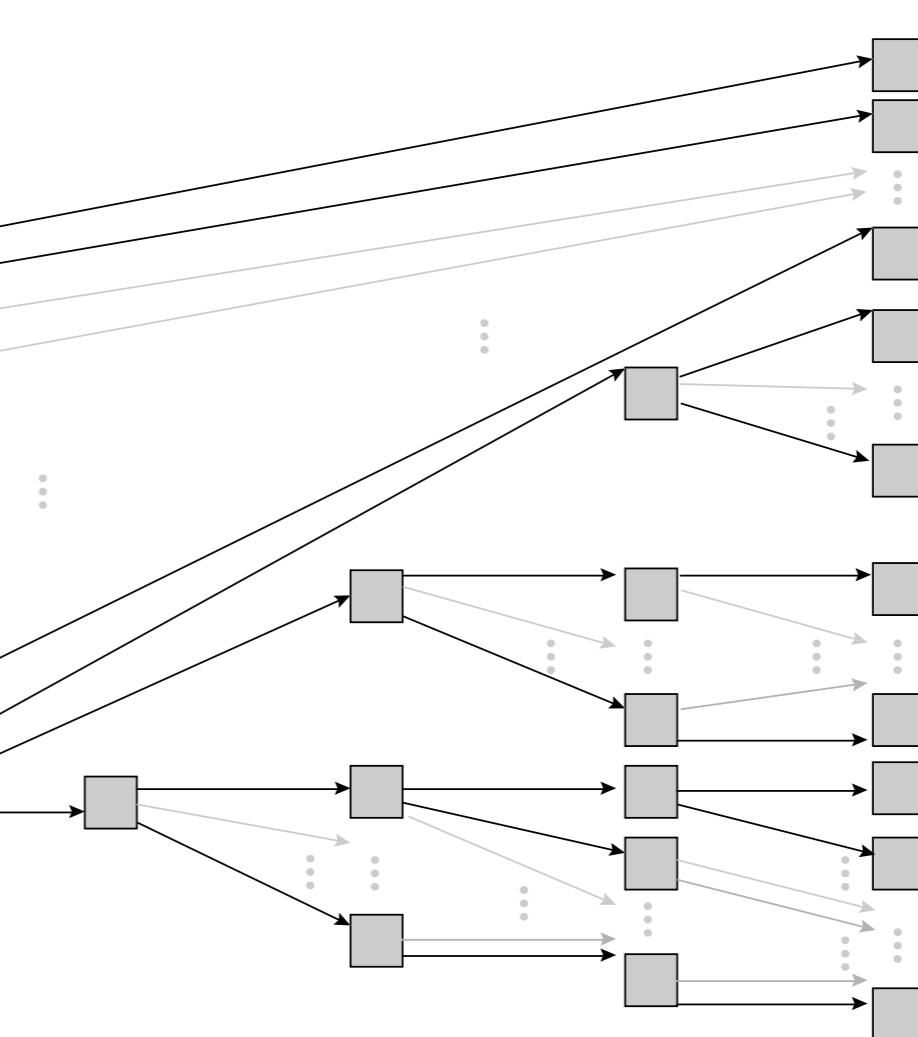
- Utiliza tabla de inodos, análoga a la FAT
- Inodo:
 - Metadata: permisos, dueño, tiempos de acceso,...
 - Tiene 12 punteros directos a bloques. Con bloques de 4KB, el tamaño máximo es 48KB
 - Puntero a bloque indirecto: con bloques de 4KB, 1K de bloques de datos lo que da 4MB.
 - Puntero indirecto doble
 - Puntero indirecto triple

FFS

Inode Array



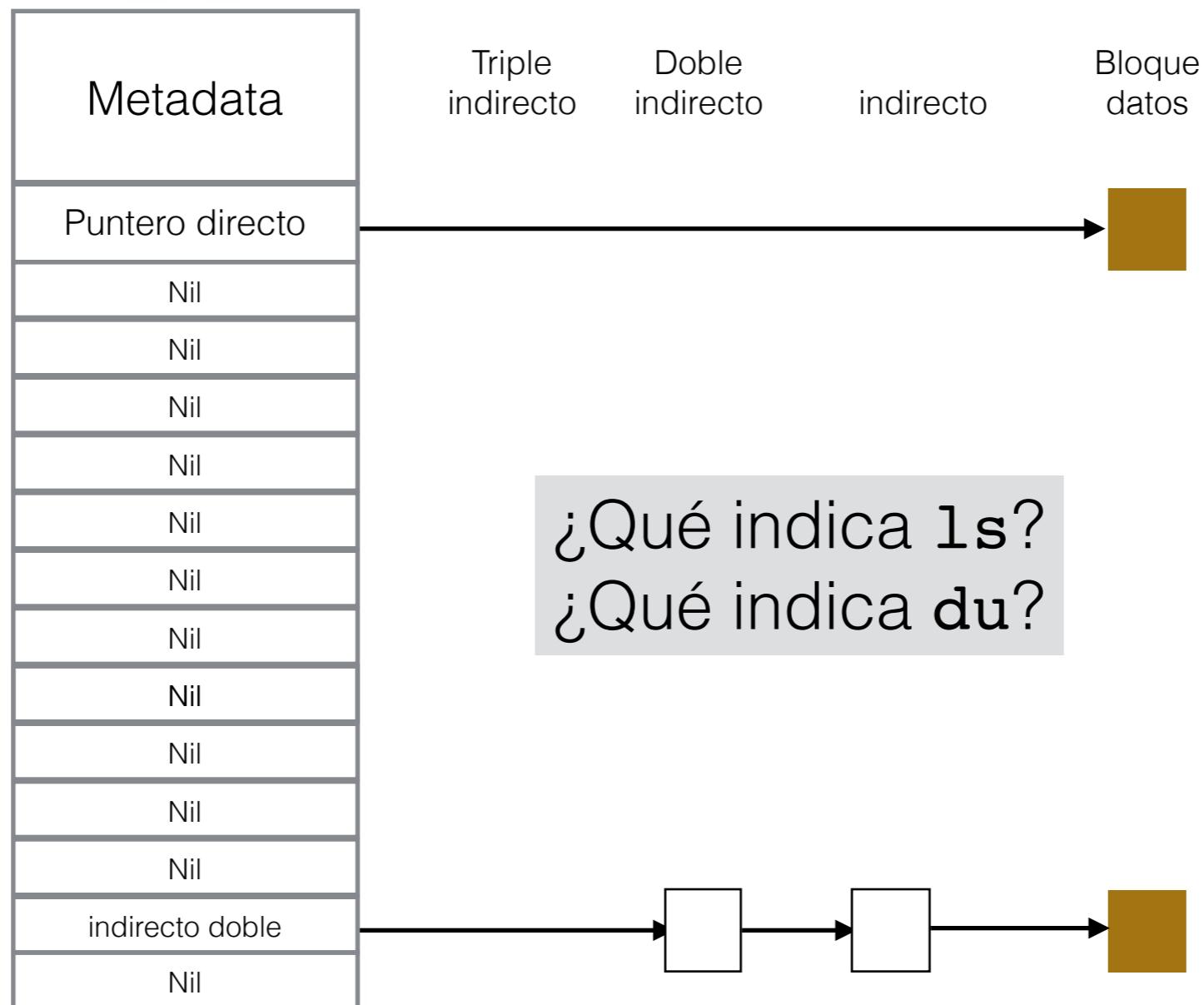
Triple Double
Indirect Indirect Indirect Data
Blocks Blocks Blocks



Observaciones

- Cada archivo se representa por un árbol lo que permite eficiencia en la búsqueda de bloques.
- Nodos con alto grado: minimiza los seek, un bloque indirecto apunta a 1024 bloques!.
- Estructura fija: Los primeros **d** punteros siempre apuntan a los primeros **d** bloques.
- Soporta archivos poco densos, es decir, contienen espacios sin datos. Por ejemplo un bloque directo y una indirecto triple permite eficientemente almacenar un archivo grande sin ocupar bloques.

Archivo poco denso



FFS: Manejo de espacio libre

- FFS asigna un bitmap con 1 bit por bloque. El bit **i** indica si el bloque **i** está libre o en uso. Al formatear el disco el bitmap queda fijo.
- **Heurísticas de localidad.** FFS utiliza dos heurísticas: *colocación de bloques* y *espacio reservado*.
- **Colocación de bloques:** FFS ubica los datos para optimizar los accesos comunes a bloques, datos y metadatos y diferentes archivos del mismo directorio.

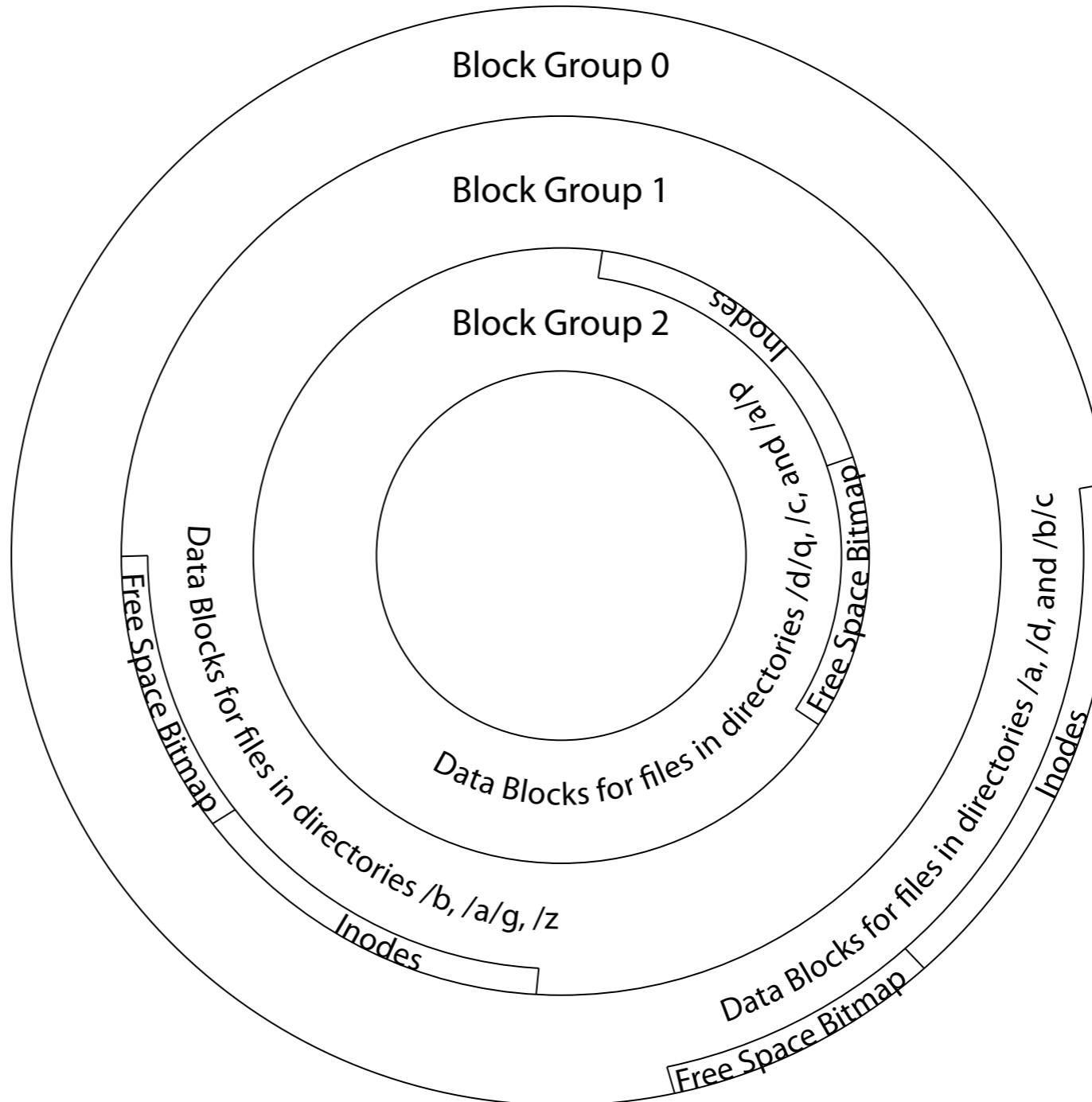
FFS: heurística de colocación de bloques

- La heurística tiene 4 partes:
 - **Dividir disco en grupos de bloques.** El tiempo de seek entre cualquier bloque es pequeño.
 - **Distribuir metadata.** La idea es evitar muchos seek entre datos y metadatos. Cada porción de inodos y bitmap se partitiona y distribuye en el disco.
 - **Ubicar archivo en un grupo de bloques.** Si FFS no encuentra bloques los distribuye en diferentes grupos (perdiendo eficiencia)
 - **Ubicación de bloques de datos.** FFS utiliza la heurística del primer bloque libre. Cuando se escribe un nuevo bloque, FFS lo escribe en el primer bloque libre del grupo.

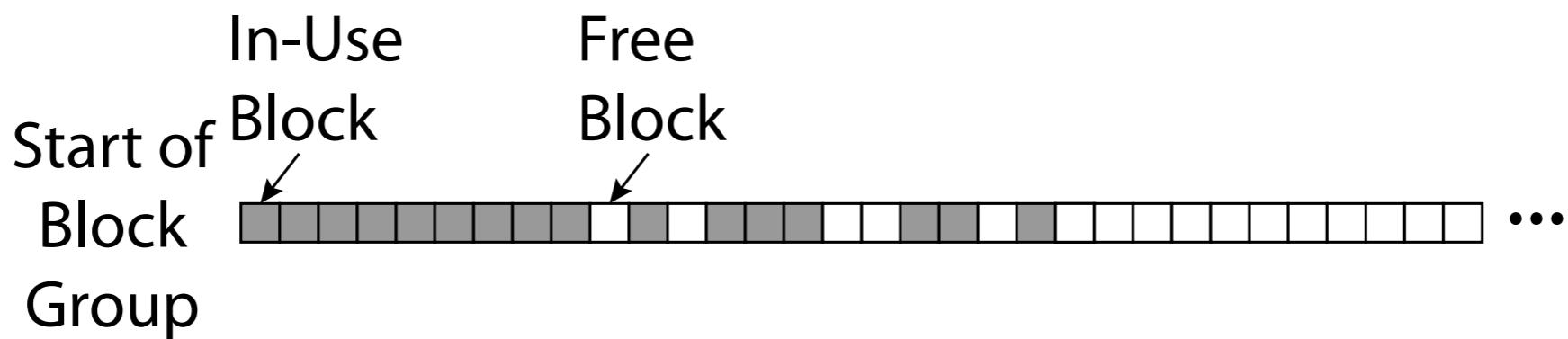
División del Disco en FFS

FFS divide el disco en conjuntos de bloques cercanos llamados **block groups** (alrededor de 100).

Cada grupo de bloques contiene un 1% de bitmaps y de inodos (por ejemplo)

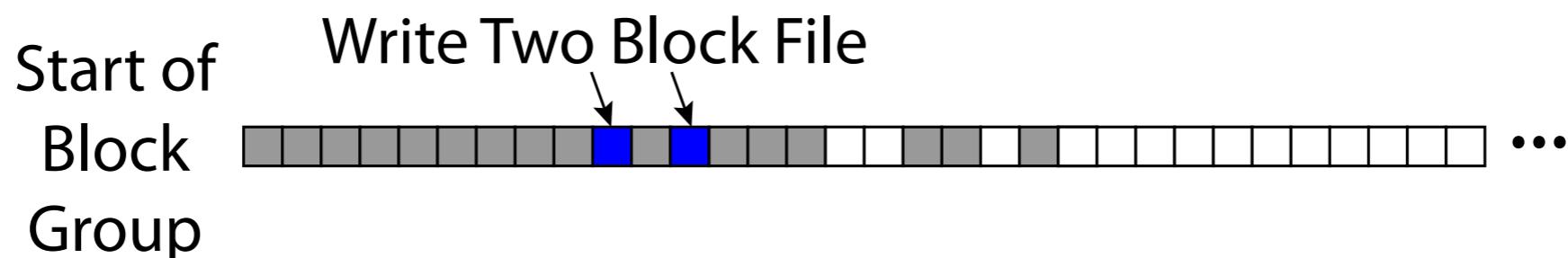


Heurística First Fit para asignación de bloques



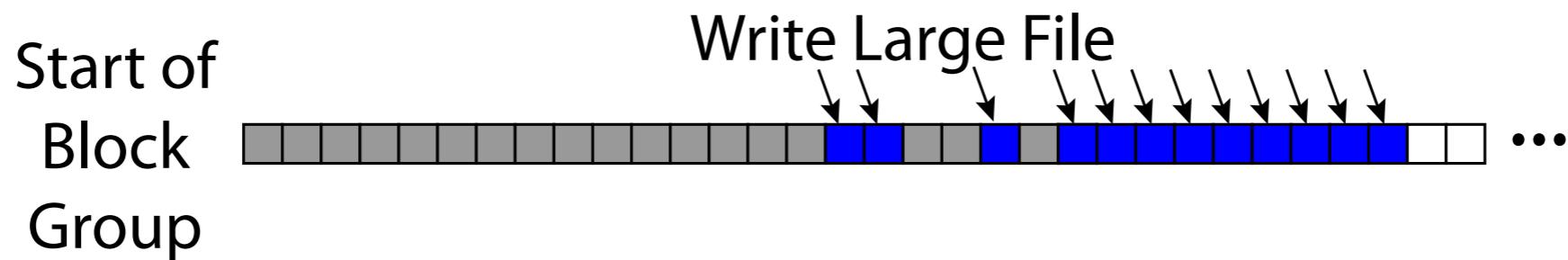
... Heurística First Fit para asignación de bloques

Archivos pequeños llenan los espacios libres cerca del comienzo de un grupo de bloques



... Heurística First Fit para asignación de bloques

Archivos grandes llenan los espacios libres cerca del comienzo de un grupo de bloques y después escriben datos en un rango secuencial de bloques



FFS: Observaciones

- Pro
 - Almacenamiento eficiente para archivos grandes y pequeños.
 - Localidad para archivos grandes y pequeños.
 - Localidad para datos y metadatos.
- Contra
 - Ineficiente para archivos diminutos. (Un archivo de 1B requiere un inodo y un bloque de datos).
 - Necesita reservar un 10% a 20% de espacio libre para prevenir fragmentación.

5 Disponibilidad del Almacenamiento

- Como el HW de almacenamiento es imperfecto, los sistemas de almacenamiento se diseñan para detectar y corregir errores.
- El Hw detecta errores a través de Checksum de datos y a nivel de controladores de dispositivos. Se utiliza redundancias a través de arquitecturas RAID para reconstruir datos dañados.
- Tanto discos rotacionales como discos flash presentan dos tipos de fallas: sectores o páginas flash pueden perder datos o degradarse, un disco completo puede fallar.

Disponibilidad y Confiabilidad

- **Confiabilidad en almacenamiento:** Los datos recuperados son los mismos que se almacenaron.
- **Disponibilidad en almacenamiento:** Los datos están cuando uno los requiere.

Técnicas para mitigar fallas

- Se utilizan dos técnicas: Códigos correctores de error y remapeo.
- **Corrección de error:** Útil cuando algunos bits de un sector están corruptos. Al almacenar se agrega redundancia adicional. Estos bits permiten corregir en forma transparente, o en caso grave informar el error. Normalmente los errores varían de un sector por cada 10^{14} a 10^{16} bits.
- **Remapeo:** Discos rotacionales y flash se construyen con sectores de repuesto. Al detectar fallas, el controlador remapea sectores dañados.

Fallas de Dispositivos

- Un dispositivo falla cuando no es capaz de servir lecturas o escrituras en todos sus sectores. Cuando esto sucede, retorna no un bloque erróneo, sino un código de error.
- Las tasas de falla de un dispositivo se miden por el parámetro los parámetros:
 - MTTF: Mean Time to Failure o tiempo medio para falla. Tiempo promedio para que el dispositivo falle.
 - CAFR: Constant Anual Failure Rate. Tasa anual de fallas
- MTTF es el inverso del CAFR

MTTF y CAFR

- En 2011, el MTTF para discos rotacionales y discos flash es: 1.0×10^6 hrs. \leq MTTF $\leq 1.7 \times 10^6$ hrs.
- El CAFR se calcula a partir del MTTF. Por ejemplo si MTTF= 1.7×10^6 hrs.:

$$CAF R = \frac{1}{1.7 \times 10^6} \frac{\text{falla}}{\text{horas}} \times 24 \frac{\text{hrs.}}{\text{dia}} \times 365 \frac{\text{dias}}{\text{anos}} = 0.0051 = 0.5\%$$

- Si MTTF= 1.0×10^6 hrs.:

$$CAF R = \frac{1}{1.0 \times 10^6} \frac{\text{falla}}{\text{horas}} \times 24 \frac{\text{hrs.}}{\text{dia}} \times 365 \frac{\text{dias}}{\text{anos}} = 0.008760 = 0.9\%$$

MTTF y tiempo de vida

- Si un disco tiene un MTTF de un millón o más horas, no significa que el disco dure 100 años o más.
- Los discos se diseñan para operar con un tiempo de vida finito, alrededor de 5 años.
- El CAFR ($\sim 1/\text{MTTF}$) sólo se aplica durante su tiempo de servicio.

Ejemplo

Un datacenter tiene un file server con 100 discos, cada uno con un MTTF estimado de 1.5×10^6 hrs. ¿Cuál es el tiempo estimado que uno de estos discos falle?, o lo que es equivalente, ¿cuál es la tasa anual de falla para estos 100 discos?

...Ejemplo

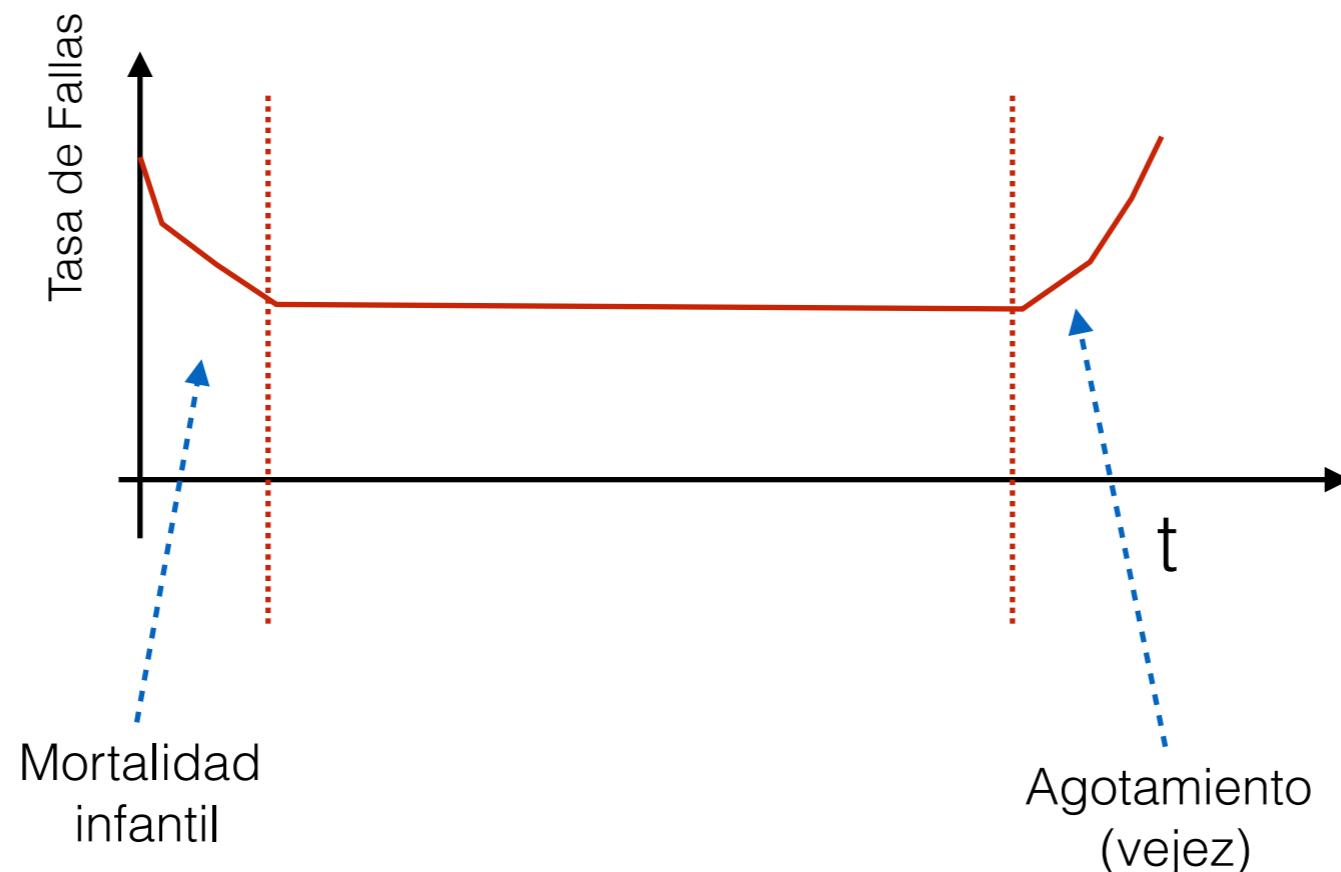
Los discos fallan en forma independiente y cada disco tiene una tasa de falla constante.

- Para un disco, MTTF= 1.5×10^6 hrs., el CAFR es

$$CAF R = \frac{1}{1.5 \times 10^6} \frac{\text{falla}}{\text{horas}} \times 24 \frac{\text{hrs.}}{\text{dia}} \times 365 \frac{\text{dias}}{\text{anos}} = 0.00584 = 0.584\%$$

- Como son 100 discos, el flujo de falla se multiplica por 100: CAFR= 0.584, es decir 58.5%. ***La tasa anual de falla para los 100 discos es 58.4%.***
- Como 100 discos fallan a una razón 100 veces mayor, el MTTF para los 100 discos es: MTTF= $24 \times 365 / \text{CAFR} = 1.5 \times 10^4$ hrs.

La Tina de Baño



Algunos dispositivos implementan una interfaz llamada **SMART** (Self-Monitoring, Analysis and Reporting Technology) que permite al SO monitorear eventos que pueden ser útiles en predicción de fallas.

RAID

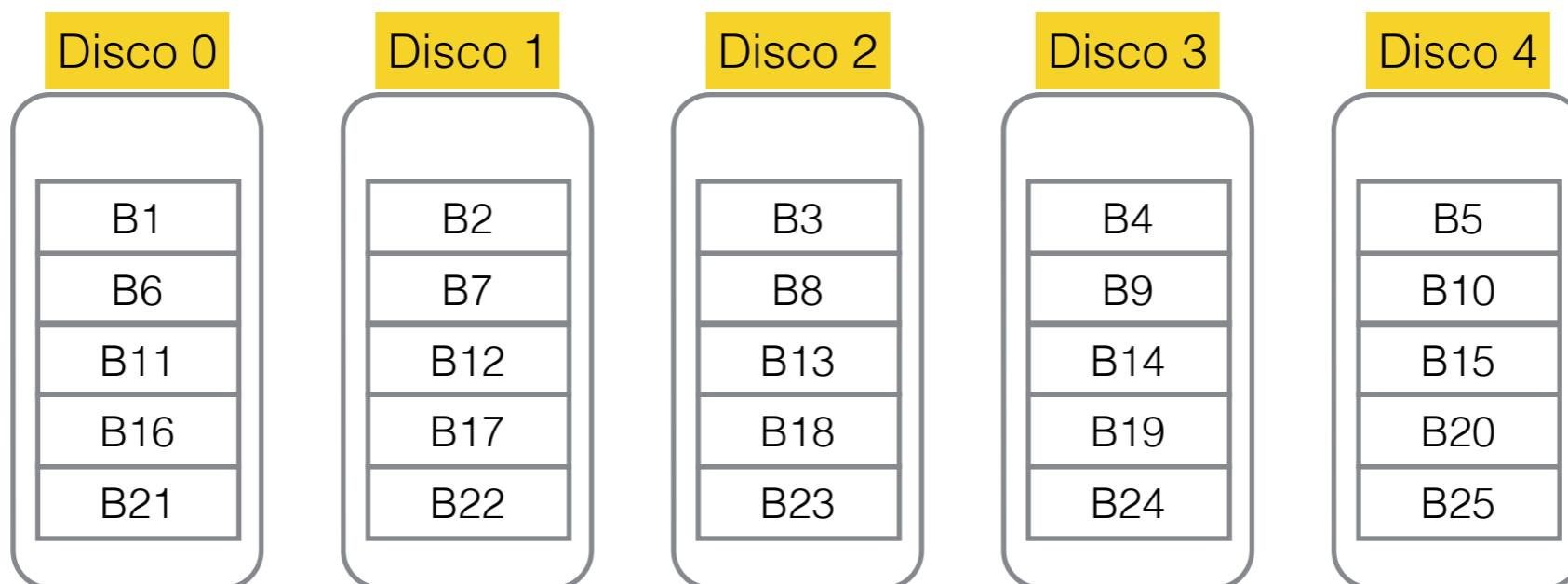
- En vez de perfeccionar los dispositivos mediante ingeniería, que resulta muy caro, los sistemas de almacenamiento utilizan **R**edundant **A**rrays of **I**nexpensive **D**isks (RAID). Actualmente Redundant Arrays of Independent Disks. De esta forma, una falla no significa la pérdida de datos.
- RAID distribuye los datos en forma redundante a través de múltiples discos para soportar fallas de discos individuales.
- Se aplica tanto a discos rotacionales como flash.

Niveles RAID

- Originalmente se propusieron 6 niveles RAID: RAID 0, RAID 1, RAID 2, RAID 3, RAID 4 y RAID 5.
- Actualmente, sólo 3 están en uso:
 - RAID 0: Llamado también JBOD (Just a Bunch of Disks)
 - RAID 1: Disco espejado
 - RAID 5: Paridad rotada
- Adicionalmente surgen otros que se constituyen en estándares:
 - RAID 6: Redundancia dual. Similar a RAID 5, pero en vez de tener un bloque de paridad por grupo almacena dos.
 - RAID 10 y RAID 50: RAID anidado. Originalmente se llamaron RAID 1+0 y RAID 5+0. Combinan RAID 0 con RAID 1 o RAID 5.

RAID 0

- Esta organización se conoce como JBOD (Just a Bunch of Disks)
- RAID 0 esparce los datos en un conjunto de discos sin redundancia. Si hay falla, hay pérdida de datos.

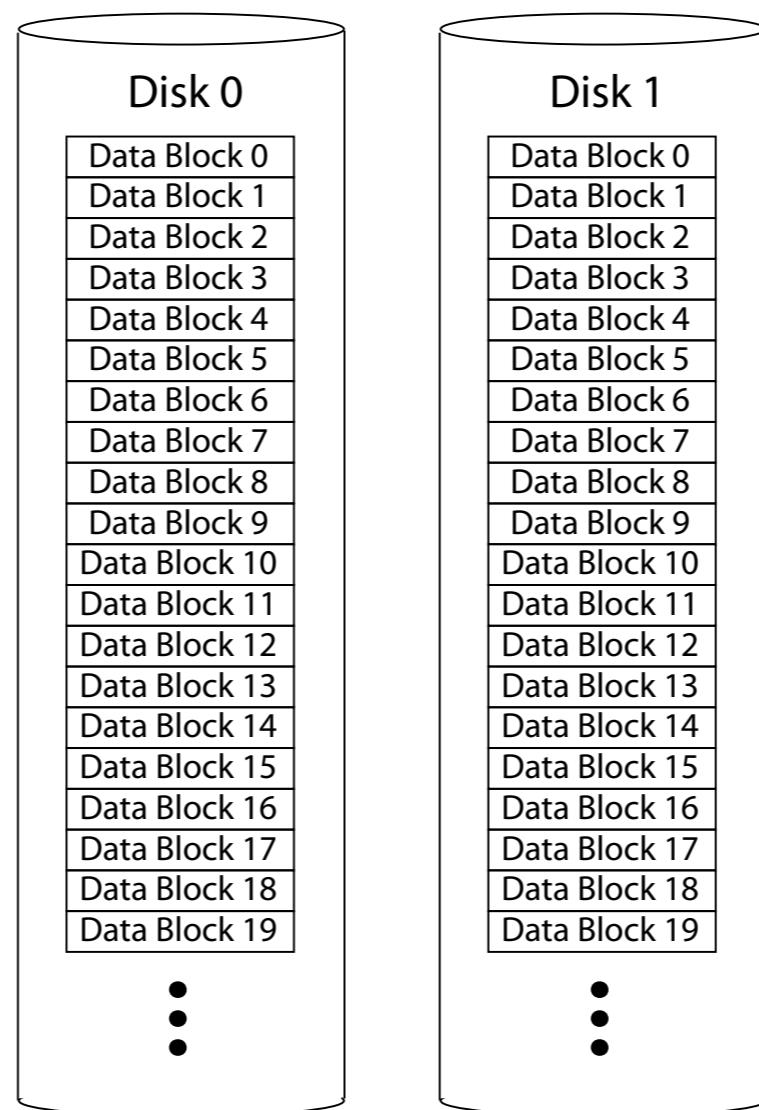


Bi= Bloque i

RAID 1

- RAID 1 corresponde a discos espejados. Se replica la escritura en dos discos. La lectura puede ser a cualquiera de ellos.
- Se utiliza frecuentemente en aplicaciones de base de datos donde la disponibilidad y tiempos de transacción son más importantes que la eficiencia.

RAID 1



RAID 5

- Paridad rotada: Se reducen overheads de replicación almacenando varios bloques de datos en diferentes discos y protegiendo estos bloques con un bloque redundante almacenado en otro disco.
- Se utilizan G discos y se escriben cada uno de los G-1 bloques de datos en diferentes discos.
- Cada bit de un bloque de paridad se genera:

$$\text{paridad} = \text{data}_0 \oplus \text{data}_1 \oplus \dots \oplus \text{data}_{G-1}$$

Paridad

- Si un disco presenta una falla, los bloques perdidos se pueden reconstruir utilizando los datos y paridad de los demás bloques.
- Como el sistema sabe cual disco ha fallado, la paridad es suficiente para la corrección. Por ejemplo, si el disco que contiene el bloque data_0 falla, el bloque se puede reconstruir con:

$$\text{data}_0 = \text{paridad} \oplus \text{data}_1 \oplus \dots \oplus \text{data}_{G-1}$$

Bloque de paridad

10001101... bloque1

01101100... bloque2

11000110... bloque3

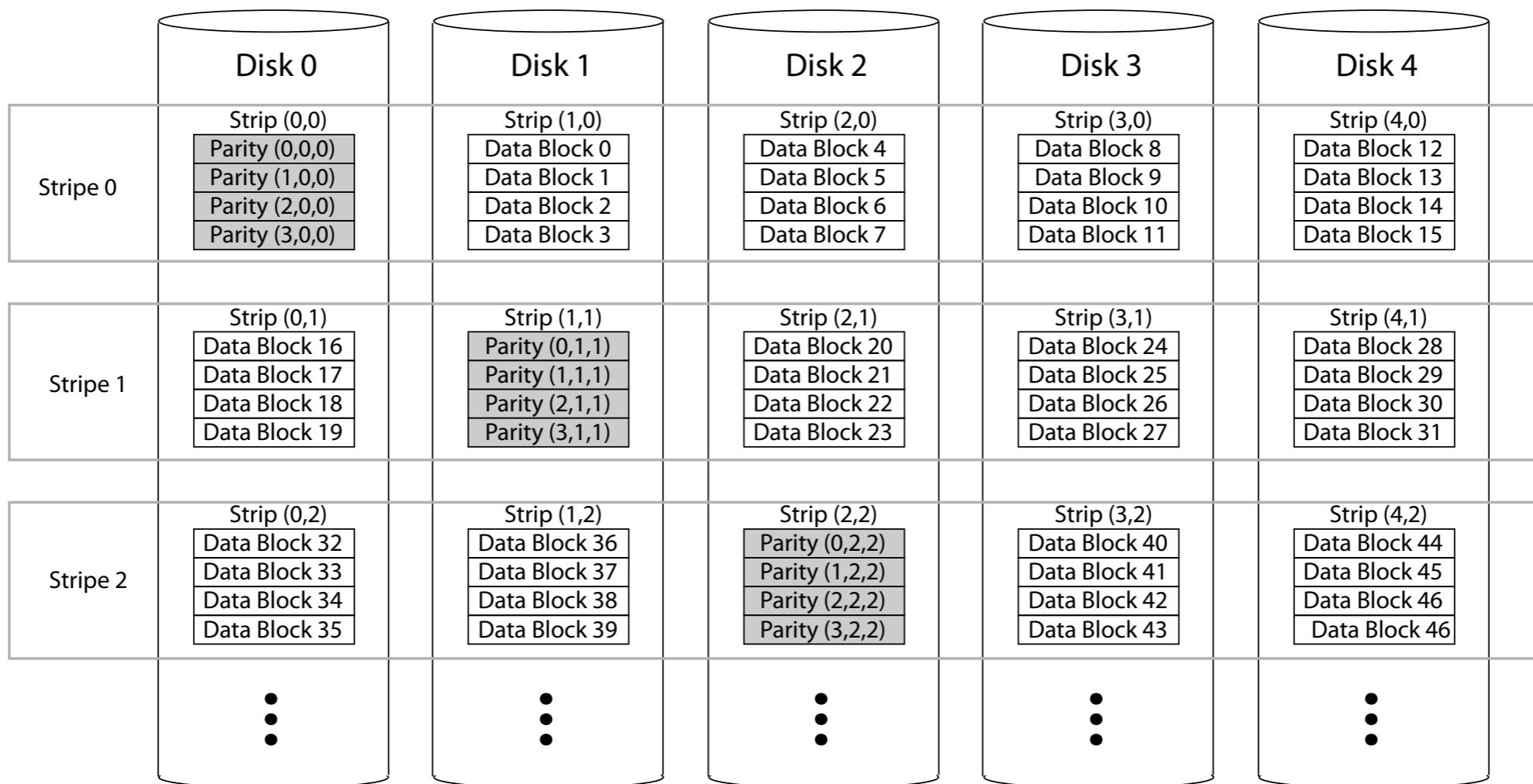
00100111... bloque de paridad

RAID 5: Optimización

- Para maximizar el desempeño, RAID 5 organiza el layout de datos rotando la paridad y segmentando los datos, de forma tal que cada segmento sea almacenado en un disco diferente (data striping).
- **Paridad rotada:** Como la paridad para un conjunto de datos se debe actualizar cada vez que se actualiza un bloque de datos, los bloques de paridad en promedio tienden a ser accesados más frecuentemente que los datos.

... RAID 5: Optimización

- Striping data: Una tira (strip) de datos de varios bloques es ubicada en un disco, la siguiente en otro disco, ...
- Stripe: Conjunto de G-1 tiras de datos y su strip de paridad.
- La siguiente figura muestra el layout de 5 discos en RAID 5.

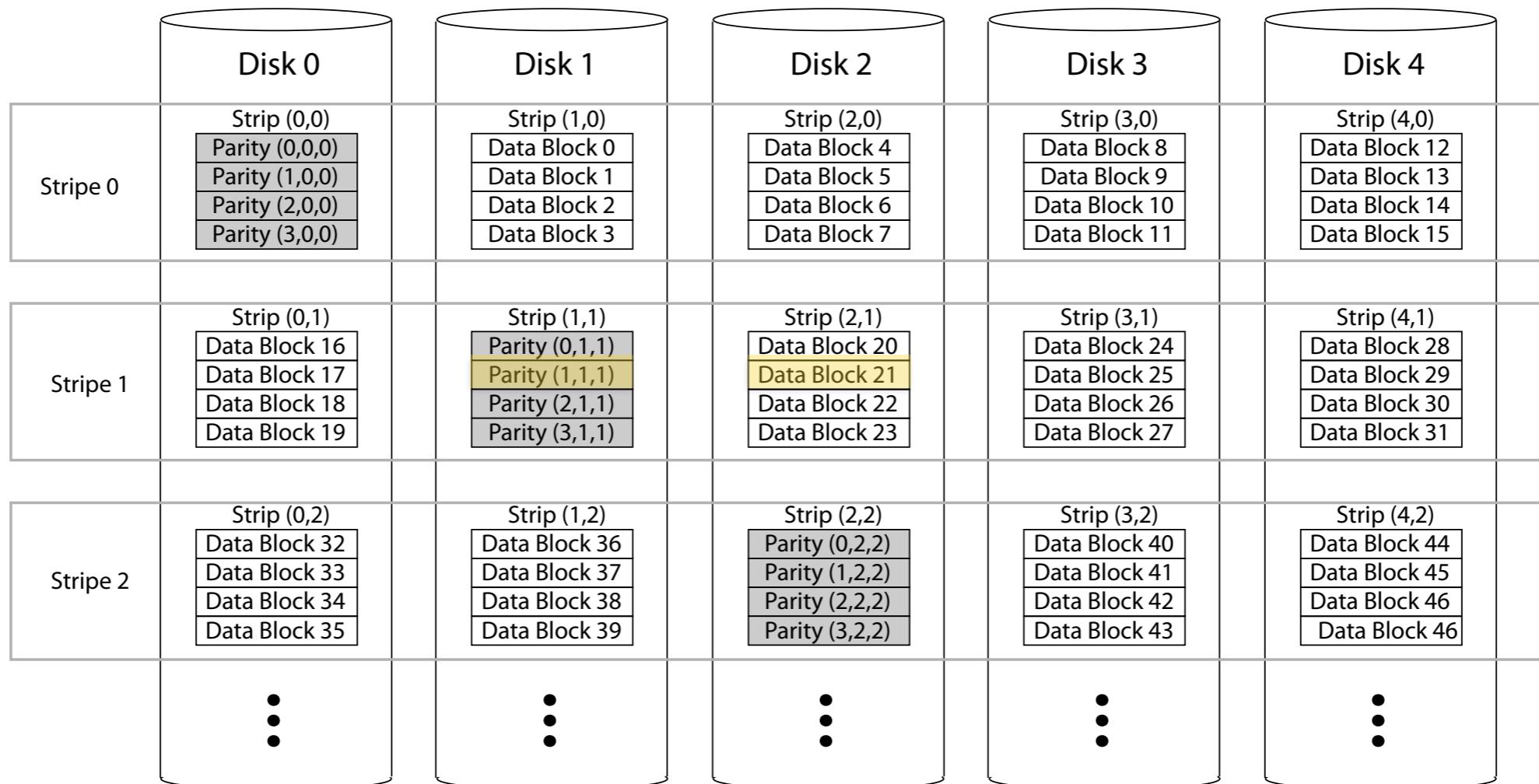


Ejercicio

Actualización en un RAID 5. Para la configuración anterior, se desea actualizar el bloque 21. ¿Qué operaciones de I/O deben ocurrir?.

- ▶ No solo se actualiza el bloque N°21, sino también el bloque de paridad correspondiente.
- ▶ El bloque 21, es el bloque del strip (2,1) y este strip es parte del stripe 1, por lo que se necesita actualizar el bloque de paridad 1 del stripe de paridad (Parity (1,1,1) en la figura).
- ▶ Se requieren **4 I/O**:
 - Leer datos D_{21} y paridad $P_{1,1,1}$.
 - Remover datos anteriores del cálculo $P_{tmp} = P_{1,1,1} \oplus D_{21}$.
 - Calcular la nueva paridad $P'_{1,1,1} = P_{tmp} \oplus D'_{21}$.
 - Escribir el nuevo dato D'_{21} y paridad $P'_{1,1,1}$ en los discos 2 y 1 respectivamente.

4 operaciones de I/O para actualizar un bloque





Sistemas Operativos

Capítulo 10 El Sistema de Archivos

Prof. Javier Cañas R.