Resumen Administración I/O

Tomás Felipe Melli

Noviembre 2024

Índice

1 Introducción

Nos vamos a centrar en **dispositivos de almacenamiento** como **discos rígidos** principalmente, pero también cinta y unidades removibles.
Existen *discos virtuales* que son unidades de almacenamiento que no están físicamente en mi dispositivo, sino que están en la red. Esto permite acceder remotamente y poder compartir info entre muchos usuarios.
Otra alternativa ?
Las **SAN (Storage Area Network)** donde la idea es tener una red especial con protocolos específicos para almacenar información con tiempos de acceso muy rápidos.
Cómo accedo a los discos virtuales ?

2 Protocolos de Almacenamiento

Un protocolo es un conjunto de reglas y estándares que permiten la comunicación y el intercambio de información entre dispositivos de una red

Estos son para gestionar el acceso a archivos y recursos de almacenamiento en red.

- **NFS (Network File System)** : permite que dispositivos UNIX y LINUX compartan archivos.
- **CIFS (Common Internet File System)** : utilizado principalmente en redes Windows.
- **DFS (Distributed File System)** : agrupa varios recursos de almacenamiento en una sóla vista.
- **AFS (Andrew File System)** : soporta la replicación de datos y el acceso fuera de línea.
- **Coda** : permite la desconexión y la sincronización de archivos.

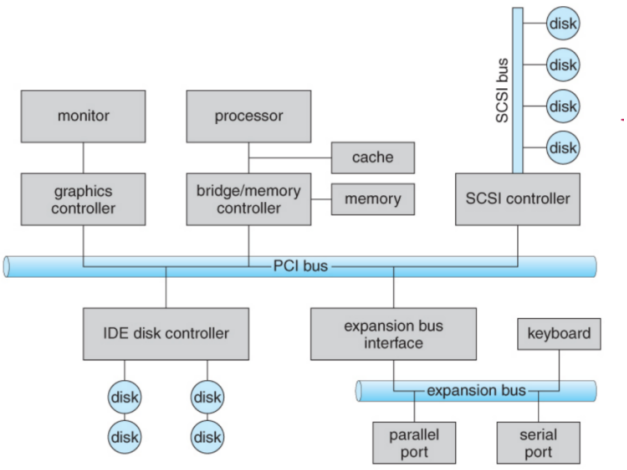
Se menciona **NAS (Network Attached Storage)** a los dispositivos dedicados al almacenamiento en red. Estos dispositivos puede ser configurados para usar uno o varios de los protocolos mencionados, permitiendo que los usuarios accedan a archivos y carpetas desde diferentes dispositivos conectados a la misma red.

3 Dispositivos I/O

Vamos a entender a un dispositivo de entrada y salida como dos partes.

1. **dispositivo físico**.
2. **controlador del dispositivo**(este es el que interactúa con el SO).

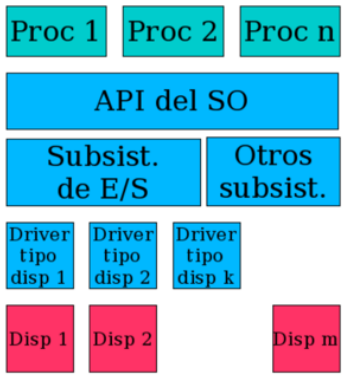
Desde el punto de vista del BUS...



El **SCSI bus** (Small Computer System Interface) es un standard para conectar y comunicar dispositivos de almacenamiento y periféricos a un ordenador.

4 Capas

Mis n-procesos quieren hacer operaciones de I/O :



4.1 Drivers

Esta cadena de interacción es la que hace que unn proceso pueda hacer una operación de I/O. En particular, el programa especial es el **Driver**. Los **drivers son componentes de software específicos que conocen las particularidades del hardware del dispositivo**. Corren con máximo privilegio y son resposables del rendimiento I/O.
Cómo interactúan?

- **Polling** : chequeos periódicos para ver si el dispositivo se comunicó (es sencillo pero consume mucha CPU).
- **Interrupciones** : el dispositivo avisa (eventos asíncronos poco frecuentes, pero muchos context switch impredecibles).
- **DMA (Direct Memory Access)** : la CPU no interviene y la data pasa directo a memoria (requiere controlador DMA, este cuando termina interrumpe a la CPU)

4.2 API del SO

La idea de esta API es brindar acceso consistente a todos los dispositivos *ocultando las particularidades de cada uno* en la medida de lo posible. Para ello, **todo es considerado un archivo**. Se tienen funciones de alto nivel para acceder :

- **FOPEN/FCLOSE**
- **FREAD/FWRITE** : lectura/escritura *block mode*
- **FGETC/FPUTC** : lectura/escritura *char mode*
- **FGETS/FPUTS** : lectura/escritura *char stream mode*
- **FSCANF/FPRINTF** : lectura/escritura *char mode con formato*

4.3 Subsistema I/O

Este es el que como programadores vemos al usar el SO. Este nosa da una API sencilla : *open() / close()* para abrir y cerrar archivos, *read() / write()* para leerlos/escribirlos, *seek()* para mover el puntero de lectura/escritura en un archivo. Este subsistema **nos resuelve tener que conocer las características de ese dispositivo**, si es local,....
Tenemos **dos grupos de dispositivos**

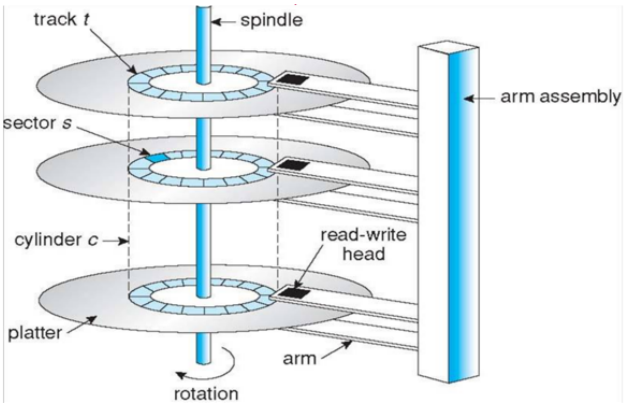
1. **Char device** : transmiten la data **byte a byte**. No soportan acceso aleatorio (no puedo leer en cualquier parte) y utilizan CACHE.
2. **Block device** : transmiten la info en bloques como el *disco rígido*. Permiten acceso aleatorio, puedo leer desde cualquier lugar y en general usan un buffer (cache).

5 Cuál es el rol del Scheduler ?

Tenemos que manejar apropiadamente el **Disco** para el buen rendimiento del I/O. Recordemos que los pedidos I/O a disco son constantes.

5.1 Funcionamiento del Disco

El disco se compone de una **cabeza** que es la que se debe posicionar sobre el sector adecuado sobre el que queremos buscar. También tenemos que conocer sobre el **ancho de banda**, es decir, **la cantidad de bytes que podemos transferir a la vez**. La **latencia rotacional** es *el tiempo necesario para rotar el disco de manera que la cabeza quede en el sector deseado.* Habiendo introducido estos conceptos, lo más importante es el **SEEK TIME** que es **el tiempo necesario para que la cabeza logre ubicarse sobre el CILINDRO que tiene el sector buscado**. Miremos el esquema :



5.2 Políticas de Scheduler de I/O a Disco

Ese SEEKTIME es importante porque *la cola de pedidos de I/O*, en particular cómo se ordena esta, es fundamental para **minimizar el seektime** y obtener el máximo de los pedidos.

- **FCFS (First-Come-First-Served)** es el más simple implementado como una cola FIFO, el tema es que la cabeza del disco va y viene según el pedido y esto es costo en tiempo y también daña el disco tanto movimiento.
- **SSTF (Shortest-Seek-Time-First)** atiende al próximo pedido más cercado a la posición actual de la cabeza. Produce inanición (trivial)
- **SCAN o ELEVATOR** va en un sentido atendiendo los pedidos que encuentra en el camino y luego cambia de sentido.

Dato de color : *ninguno se usa puro : hay prioridades (bajar páginas de cache, swapping de procesos)*

5.3 Volvamos al disco : estado del arte

Actualmente, se suele utilizar SSD (Solid State Drive) que consumen menos energía, son más resistentes, livianos y silenciosos. Tienen dos problemas :

- La **durabilidad** dado que tienen un número limitado de ciclos de escritura por celda. Esto se debe a que la tensión que se le aplica a la celda para cambiar su estado con el tiempo rompe la estructura de la memoria *flash nand*
- **write amplification** ocurre cuando la **cantidad de datos escritos en la memoria física es mayor que la cantidad de datos que el usaurio quiere escribir**. Esto ocurre *porque la estructura del alamacenamiento es por BLOQUES que son varias páginas.* Por la forma en que se implementa, el borrado también es de a bloques. Si modificás un dato en una página, la SSD lo escribe todo el bloque de nuevo en otro lugar, marca el viejo como "invalid" por más pequeña que sea la modificación. Esto produce mucho desgaste, muchas operaciones de lectura y escritura.

TRIM permite al SO informar a las SSD qué bloques no están en uso (GARBAGE COLLECTOR)

5.4 Gestión del disco

- **Formateo** : código para detectar si el disco está dañado.
- **Booteo** : al momento de prender la PC la BIOS o UEFI carga de una sección particular (sector 0) que le permite saber de dónde cargar al SO.
- **Bloques Dañados** : en los discos magnéticos si recibe un golpe en la cabeza podría tocar la superficie y dañar bloques, el SO puede implementar mecanismo para detectar estas fallas.

5.5 Spooling (Simultaneous Peripheral Operation)

Es un mecanismo para que muchos procesos puedan hacer uso de un dispositivo I/O. Se implementa como una cola fifo en la que se encolan los pedidos. Pensemos en una impresora : muchos pedidos de impresión se encolan en orden.
Se puede usar I/O para sincronización con open(..., O_create, O_excl) ya que es atómico y crea el archivo si no existe o falla si existe. Se suele usar para hacer LOCKS, no es muy eficiente igual.

6 Protección de la información

6.1 BackUp

La **Copia de seguridad** se puede almacenar en otro disco en casa o en otro lugar como una cinta robotizada (una cinta escrita con un brazo robot que la va moviendo). Esto toma tiempo y si hay modificaciones en el medio se ocasionarían problemas de **consistencia**. Copiar todo suele ser costoso y es por ello que se puede hacer cada t-tiempo una total :

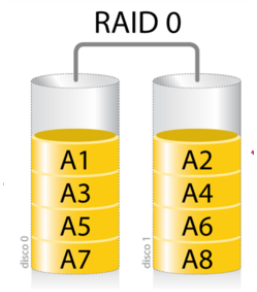
- **Copia Incremental** : sólo los modificados desde la última incremental.
- **Copia Diferencial** : sólo los modificados desde la última total.

Si se rompe el disco, podemos recuperar, el tema es que restaurarlo nos lleva mucho tiempo (el sistema debe estar prendido lo antes posible)

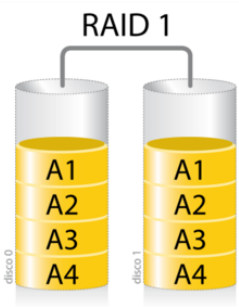
6.2 RAID (Redundant Array of Inexpensive Disks)

La idea es tener mucha data redundante, es decir, tengo el doble del disco, entonces si se cae uno, uso el otro. Vale leer de ambos. Existen una serie de **Niveles**

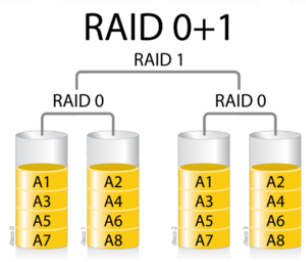
- **RAID 0** : stripping, no aporta redundancia, sólo distribuye los datos entre varios discos



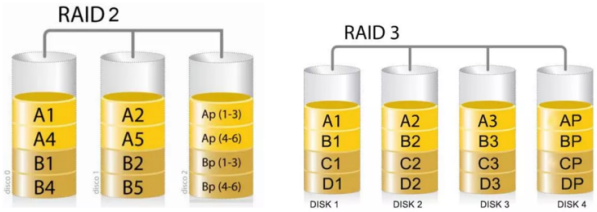
- **RAID 1** :mirroring, duplica los datos en dos discos



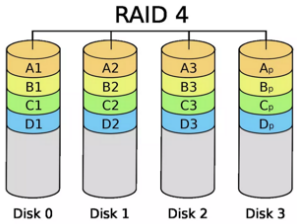
- **RAID 01** : combinación RAID 0 + RAID 1



- **RAID 2 y 3** se utiliza código de corrección para poder reconstruir si se pierde data. Los datos se distribuyen en discos y se requiere uno adicional para almacenar cada **bit de paridad**. Esto es el valor que representa la suma de los bits de datos en un conjunto. Este permite deducir qué bit se rompió a la hora de mirar el daño.
  RAID 2 requiere 3 discos de paridad por cada 4 de datos mientras que RAID 3 sólo 1 por cada 4.



- **RAID 4** como RAID 3, pero usa *stripping* a nivel de bloque (cada bloque por disco)



- **RAID 5** es de los más usados. Usa datos redundantes distribuidos en $N + 1$-discos. Cada bloque de cada archivo va a un disco distinto. Para cada bloque, uno de los discos tiene los DATOS y otro la info de PARIDAD. Ya no hay cuello de botella para las escrituras, hay que mantener la paridad distribuida. Puede soportar la pérdida de 1 disco cualquiera.