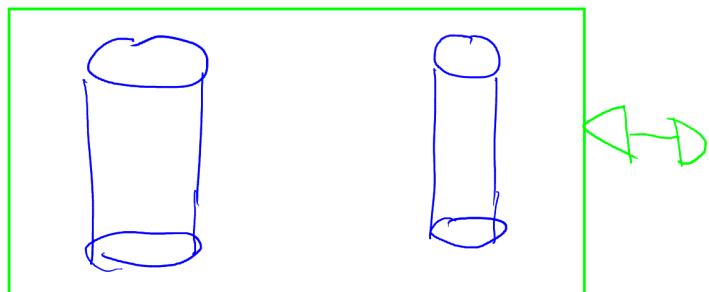


RAID \Rightarrow Virtualización de almacenamiento. Implementados por HW o por el SO

RAID \Rightarrow Redundant Array of Independent Disk



Raid por software es malo y se ralentizan las operaciones de entrada y salida

Por HW se usa una controladora Raid conectada a un puerto PCIe

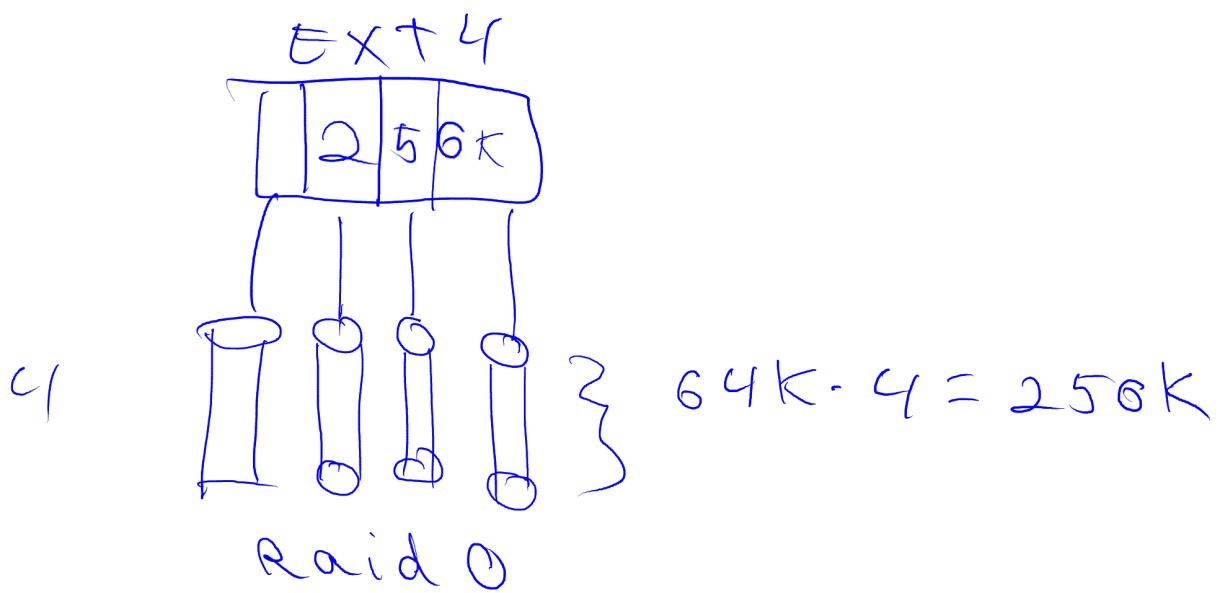
Tipos de RAID

- RAID 0: Mejora de velocidad lectura/escritura sin redundancia.

striping (interleaving) \Rightarrow los bloques contiguos se escriben en paralelo.

Al configurar el RAID se configuran el tamaño de bloque.

La suma de todos los bloques debería de ser múltiplo e igual al número de discos multiplicado por el tamaño de cada bloque del disco



• RAID 1: (Espejo)

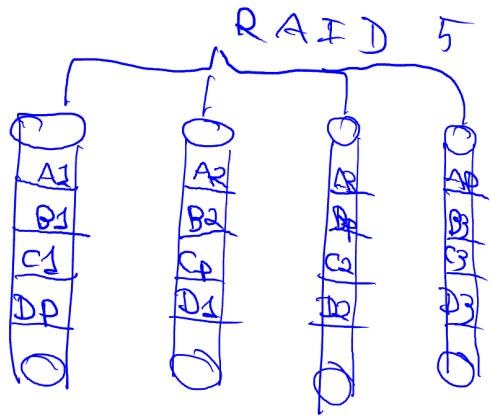
Se hace réplicas de cada uno de los stripes con dos discos. No hay partición de los datos a bajo nivel. No mejora la velocidad de lectura/escritura. Es aconsejable usar discos idénticos

Cuando se usan muchos discos aumenta mucho la probabilidad de fallo. (Paradoja del cumpleaños)

• RAID 4:

Dedica un disco entero a los datos de paridad. Soporta el fallo de un disco (si no es el de paridad)

• RAID 5: Mejora al RAID 4 en cuanto al problema de que un disco se encargue de guardar la paridad, pero necesita 3 discos o más.



- RAID 1 + 0 : Dos raid 1 que se combinan en un raid 0.
nivel inferior Nivel Superior
Es mayor la seguridad que con raid 5
- SAN (Storage Area Network)
Proporciona almacenamiento de bloques a través de la red.
No es un sistema de ficheros en red (NAS)
- Sistemas de ficheros en red
- Nos permite compartir ficheros en red a través de cliente/servidor.
Estos han de hablar un protocolo de sistema de ficheros (NFS, SAMBA, etc.)
- NAS (Network-Attached Storage):
Hardware específico especializado en servir un FS en red.

Tipos de sistemas de ficheros en red

• Stateless

- Sencillo
- Operaciones autocontenidoas
- Los handles que da el servidor son únicos
- No requiere recuperación tras fallo
- El servidor se puede reemplazar

• Stateful

- Estado permanente
- El servidor puede olvidar los handles después de una sesión
- Si falla el servidor hay que recuperar el estado o abortar.

Semántica de compartición de ficheros

- UNIX (Strong Consistency): cualquier cambio es automáticamente visible al resto.
- Ficheros inmutables: un cambio en un fichero implica una nueva versión del mismo
- Transacciones: Las operaciones se empaquetan en transacciones (ACID)
- Otros: Coherencia débil, Semántica de sesión

NFS

- NFS3 era stateless
 - NFS4 es stateful
 - No hay open ni close
 - Problemas con deadlock
- Tenemos aplicaciones a alto nivel haciendo llamadas al sistema a un Virtual File System (VFS) que proporciona el puente para conectar con el NFS a través de una conexión de red proporciona acceso a los datos.

GP

- stateful: El servidor mantiene el estado de la conexión y posibilita la exportación de ficheros sintéticos.
- Sencillo: Solo 13 RPCs
- Integrado en el SO: el Kernel de GP es multiplexor de IP

Distributed FS

- Los nodos tienen una réplica del arbol de ficheros
- Semantica desconectada: trabajar offline y sincronizar cuando haya conexión
- Cuidado con los conflictos al sincronizar

Distributed version control system

- Mantienen copias de un arbol con control de versiones
- Guarda distintas versiones de los ficheros, gestiona commits, puede crear ramas
- Sirve para coordinar cambios en el código de un proyecto
- Cuidado también con los conflictos a la hora de sincronizar

Cluster Fs

- Bloques de ficheros (chunks) distribuidos en un cluster \Rightarrow File Striping
 - ↳ Google FS

Google FS

- El fichero no puede estar en un único punto porque puede producir un cuello de botella al acceder al recurso compartido
El objetivo es doble:
 - La información esté disponible para la mayor cantidad de gente posible (Alta disponibilidad)
 - Si un archivo es grande, se pueda trocear y al obtener la información en paralelo se pueda obtener rápidamente
- Cuando guardamos datos en un cluster, cada nodo recibe la información de los datos en chunks de 64 Mbytes.
- Los servidores no pueden guardarse archivos completos porque los problemas en los nodos de cluster son bastante recurrentes

Mecanismo de Google FS

- 1-Al guardar el archivo se hacen chunks (ni muy pequeños, ni muy grandes)
- 2-Al tener el tamaño de chunk predeterminado, cada chunk se copia 3 veces en distintos Chunk Servers para que se pueda garantizar la disponibilidad.

- 3.- La aplicación solicita los chunks a los chunks servers. El mapeado de los chunks es llevado a cabo por el Master que contiene los metadatos de donde está la información. El Master no une los chunks, eso lo hace la aplicación.

Características de Google FS

- Tolerancia a fallos
- Optimizado para appends
- Escalable (miles de nodos)
- No muchos ficheros pero sí de gran tamaño
- Coherencia relajada
- Sacrificio de latencia por rendimiento
- Chunks de 64 Mb (con checksum)
- El nodo Master puede ser un problema si se cae, por tanto, tenemos otros servidores Master copiando la información del nodo Master original por si este deja de dar servicio, poder tomar el testigo del control del cluster. Estos son los Shadow Master.

Características del Master en Google FS

- Solo almacena metadatos (memoria + log en disco)
- Actualizaciones atómicas de los metadatos
- Controla el namespace
- Localiza los chunks, rebalancea y controla caídas de Chunk Servers
Ordena la replicación de chunks (3t de replica por chunk)
Crea nuevos chunks
- Controla leases sobre réplicas primarias y los locks sobre los ficheros
- Realiza recolección de basura
- No coordina la escritura en los chunks

Dos tipos de Mutacion

→ Write (poco usado)
→ Atomic Record Append

Mutacion: Atomic Record Append

- 1= El cliente envía los append a todas las réplicas
- 2= El cliente ordena atomic-append al primario
- 3= El primario comprueba si se desborda el último chunk:
 - En caso de que ocurra se le indica al cliente que el cambio lo ha de hacer en un nuevo chunk
 - En caso de que no ocurra, se escribe en el chunk
- 4= si falla alguna réplica, se repite la operación append. Los datos del append aparecen repetidos en las réplicas que no fallaron (at-least-once)

Apache Hadoop

● HDFS

Se basa en Google FS

Optimizado para maximizar el rendimiento

Tamaños de bloque grandes

En las primeras versiones se usaba un nodo para los metadatos (NameNode) y varios nodos para los datos (DataNodes). (versión MR1). En las versiones más modernas (YARN) los datos y metadatos están distribuidos.

● Diferencia entre MR1 y YARN

Sopporta batch, flujo de datos)

Data Operating System (un conjunto de datos, multiples instancias)

Mejor aprovechamiento de los recursos

de computación y almacenamiento

Seguridad y autenticación en el diseño

Retrocompatible

Mecanismo de YARN

- Se aligera el rol del Master en MR\$ a un simple planificador (Resource Manager)
- Cada nodo tiene un Node Manager que gestiona los recursos de computación que están siendo usados por las aplicaciones Y en cambio, los metadatos que contienen la información de la localización de cada chunk están en los Application Master. Estos además están replicados en distintos nodos, al igual que los chunks de datos que se encuentran en los Containers para poder así garantizar la alta disponibilidad y no tener un único punto de fallos (como ocurre en Google FS o en Hadoop MR \$) con el Master.